

Programovanie v jazyku Python

Pracovný zošit

Meno:

Trieda:

Programovanie v jazyku Python, Pracovný zošit

Autori: Ján Guniš, Ľubomír Šnajder, Valentína Gunišová, Zuzana Tkáčová

Recenzenti: PaedDr. Milan Trnka, Mgr. Anton Belan, PhD.

Ministerstvo školstva, vedy, výskumu a športu Slovenskej republiky schválilo pod č. 2022/13319:7-A2201 edukačnú publikáciu v elektronickej podobe Programovanie v jazyku Python, Pracovný zošit. Doložka nadobúda účinnosť 31. mája 2022 a má platnosť do 31. augusta 2027.

Vydavateľ: Univerzita Pavla Jozefa Šafárika v Košiciach

Rok vydania: 2022

Vydanie: 1. vydanie

Košice 2022

Obsah podlieha licenci Creative Commons CC BY SA 4.0.

Dielo je odvodeninou publikácie: Guniš, J., Šnajder, Ľ., Gunišová, V. & Tkáčová, Z. (2021). Programovanie v Pythone: Zbierka inovatívnych metodík pre stredné školy. Bratislava: Centrum vedecko-technických informácií SR. Dostupné na Internete: <https://registracia.itakademia.sk/admin/theme/download/zim-python.pdf>.

Dielo sa môže rozmnožovať, rozširovať, vystavovať dielo a odvodené diela za podmienky uvedenia autora.

Je možné rozširovať odvodené diela len za podmienky použitia identickej licencie pre odvodené diela.

Ako citovať:

Guniš, J., Šnajder, Ľ., Gunišová, V. & Tkáčová, Z. (2022). Programovanie v jazyku Python: Pracovný zošit. Košice: Univerzita Pavla Jozefa Šafárika v Košiciach. Dostupné na Internete: [https://di.ics.upjs.sk/publikacie/Programovanie v jazyku Python - Pracovný zošit.pdf](https://di.ics.upjs.sk/publikacie/Programovanie_v_jazyku_Python_-_Pracovny_zosit.pdf)

OBSAH

Obsah	3
Predslov.....	4
01 Úvod do programovania, výpočty v konzole	5
02 Korytnačia grafika	12
03 Vlastné funkcie bez parametrov a bez návratovej hodnoty	19
04 Cyklus s pevným počtom opakovaní.....	28
05 Opakovanie I. + didaktický test	36
06: Vlastné funkcie s parametrami – kresliace úlohy	40
07 Vlastné funkcie s parametrami a výstupom – výpočtové úlohy	48
08 Chyby vo výpočtoch, ich rozpoznávanie a odstraňovanie	55
09 Podmienený príkaz.....	62
10 Opakovanie II. + didaktický test	73
11 Reťazce	78
12 Reťazcové metódy, zložené a vnorené podmienky	84
13 Algoritmy s reťazcami	92
14 Odchytávanie výnimiek	97
15 Generovanie výnimiek	104
16 Opakovanie III. + didaktický test	111
17 Zoznamy a metódy zoznamov.....	117
18 Vytváranie a modifikácia zoznamov, použitie náhody	125
19 Algoritmy so zoznamami	133
20 Cyklus s podmienkou	139
21 Vnorené riadiace štruktúry	148
22 Opakovanie IV. + didaktický test	154
23 Grafické používateľské rozhranie – tlačidlá, textové polia, popisy	159
24 Grafické používateľské rozhranie – plátno	164
25 Komplexný projekt – výber problému, analýza a návrh riešenia problému.....	172
26 Komplexný projekt – implementácia riešenia problému.....	177
27 Komplexný projekt – finalizácia projektu + prezentácia a diskusia	179
Prílohy:	184

PREDSLOV

Milí žiaci,

do rúk sa Vám dostáva pracovný zošit k výučbe programovania v jazyku Python.

Našou snahou je ukázať Vám, že počítače nemusíte používať len ako konzumenti, ale aj ako autori vlastných programov. Spoločne sa naučíme vytvárať zaujímavé a užitočné softvérové aplikácie, ktoré poslúžia nie len Vám. Pri programovaní úloh z tohto pracovného zošita sa naučíte nielen nové príkazy, ale tiež rôzne prístupy k riešeniu problémov, ktoré využijete aj mimo vyučovania informatiky. Učiť sa programovať nie je jednoduché, ale vedieť programovať je zábavné.

Pracovný zošit je rozdelený do 27 kapitol. Vo väčšine z nich budete s pomocou svojho učiteľa skúmať a objavovať nové poznatky z programovania. Postupným riešením úloh si osvojíte nové učivo a úroveň jeho zvládnutia si môžete overiť v **Sebahodnotiacom teste**. Sumarizáciu nového učiva nájdete v časti **Vedomosti v kocke** a precvičiť si ho môžete riešením úloh z časti **Ďalšie úlohy na precvičenie**. Kapitoly 5, 10, 16, 22 sú venované opakovaniu a systemizácii učiva. Kapitoly 25 až 27 sú venované návrhu, tvorbe a prezentácii záverečných projektov. Súčasťou pracovného zošita je elektronická príloha obsahujúca priečinky 01 až 27 s pracovnými súbormi. Odporúčame, aby ste si do týchto priečinkov ukladali riešenia úloh z jednotlivých kapitol.

Ďakujeme recenzentom tejto publikácie a tiež učiteľom stredných škôl zapojených do **národného projektu IT Akadémia – vzdelávanie pre 21. storočie**, ktorí overovali či používali tento pracovný zošit za poskytnutie cenných pripomienok a návrhov, ktoré veľkou mierou prispeli k požadovanej úrovni finálnej verzie tohto pracovného zošita.

01 ÚVOD DO PROGRAMOVANIA, VÝPOČTY V KONZOLE

ZAPOJENIE

Diskusia o programovaní.

Čo je to programovanie? Aké činnosti v sebe zahŕňa? Aký je význam programovania z pohľadu riešenia problémov a z pohľadu uplatnenie človeka v živote?

SKÚMANIE

Úloha 1 Skúmajte ako Python vyhodnotí nasledovné výrazy. Najskôr si do tabuľky zapíšte svoju predpoveď a potom si ju overte v konzole jazyka Python. Vyhodnotenie každého z výrazov spustíte klávesom Enter. Dodržte poradie výrazov.

	výraz	moja predpoveď	skutočnosť
1.	$1 + 2$		
2.	$3 + 4 * 5.0$		
3.	$1 / 0$		
4.	$10 + * 20$		
5.	$x = 10$		
6.	x		
7.	$x + 5$		
8.	$sucet = x + 8$		
9.	$sucet$		

Úloha 2 Skúmajte ako Python vyhodnotí nasledovné výrazy. Najskôr si do tabuľky zapíšte svoju predpoveď a potom si ju overte v konzole jazyka Python. Vyhodnotenie každého z výrazov spustíte klávesom Enter. Dodržte poradie výrazov.

Riešte podľa pokynov učiteľa

	výraz	moja predpoveď	skutočnosť
10.	z		
11.	$z = 15$		
12.	z		
13.	$z = z + 5$		
14.	z		

VYSVETLENIE

Vysvetlite výsledky z predchádzajúcich úloh. Ktoré z vašich predpovedí boli nesprávne a prečo?

ROZPRACOVANIE

Úloha 3 V pokladnici kina sme kúpili 33 vstupeniek. Jedna vstupenka stojí 3€ a 50 centov. Koľko € sme zaplatili za vstupenky?

Riešenie:

>>>
>>>
>>>

Úloha 4 V pokladnici kina sme kúpili 23 vstupeniek pre spolužiakov, ktorí mali záujem zúčastniť sa predstavenia. Jedna vstupenka stojí 3€ a 50 centov. Neskôr sa rozhodlo pre návštevu kina ďalších 9 spolužiakov, ktorým sme vstupenky dodatočne dokúpili. Koľko € sme zaplatili za vstupenky celkom?

Riešenie:

>>>
>>>
>>>

Úloha 5 Pomocou meracieho pásma sme odmerali rozmery telocvične: šírka 15 m a 32 cm, dĺžka 22 m a 12 cm. Koľko plechoviek farby budeme potrebovať na premaľovanie podlahy telocvične, ak farba z jednej plechovky vystačí na 14 m²?

Riešte podľa pokynov učiteľa

Riešenie:

>>>
>>>
>>>

Úloha 6 V stánku rýchleho občerstvenia predávajú: hotdog za 55 centov, hamburger za 1 € a 20 centov a hranolčeky za 70 centov. Predavač je síce dobrý kuchár, ale zlý počtár. Pomôžme mu, aby vedel cenu nákupu rýchlejšie vypočítať.

Koľko stojí nákup: 2 × hotdog, 5 × hamburger a 3 × hranolčeky?

Pomôcka: Ak si niektoré hodnoty pomenujete, pomôže vám to pri výpočtoch.

Riešenie:

>>>
>>>
>>>
>>>
>>>

Úloha 7 Táto úloha je pokračovaním úlohy 6.

Dodávateľ surovín do stánku zvýšil cenu hranolčekov o 5 %.

Koľko bude stáť nákup: 1 × hotdog, 3 × hamburger a 4 × hranolčeky?

Riešenie:

>>>
>>>
>>>
>>>

Úloha 8 Táto úloha je pokračovaním úloh 6. a 7.

Riešte Koľko zaplatí zákazník za 3 hamburgery?

podľa
pokynov
učiteľa

Riešenie:

>>>
>>>
>>>
>>>

Úloha 9 Na čerpacej stanici predávajú benzín, naftu a LPG. Ceny pre dnešný deň boli stanovené nasledovne:

benzín: 1,234 €/liter,

nafta: 1,109 €/liter,

LPG: 0,520 €/liter.

Ak je kupujúci stálym zákazníkom, dostane zľavu 5%.

1. Koľko má zaplatiť nový zákazník, ktorý natankoval 48,9 litrov benzínu?
2. Koľko má zaplatiť stály zákazník, ktorý natankoval 55 litrov nafty?
3. Koľko má zaplatiť stály zákazník, ktorý natankoval 45,9 litrov benzínu a 41,2 litrov LPG?

4. **Riešenie:**

5. >>>
 6. >>>
 7. >>>
 8. >>>
 9. >>>
 10. >>>
 11. >>>
- >>>

Úloha 10 Zvuk sa vo vzduchu šíri rýchlosťou približne 340 m·s⁻¹. Svetlo sa šíri rýchlosťou približne 299 792 458 m·s⁻¹.

Riešte Ako ďaleko od nás udrel blesk, ak sme jeho zvuk začuli 14,2 s po tom, ako sme ho videli?

podľa
pokynov
učiteľa

Riešenie:

>>>
>>>

```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	Ktoré z uvedených výrazov spôsobia pri vyhodnocovaní chybu ? a) $20 / 15 - 15$ b) $20 / (15 + / 10)$ c) $20 / -5$ d) $20 - / 5$
2.	Aký je výsledok vyhodnotenia posledného z nasledujúcich výrazov? <pre>>>>premenna = 15 >>>premenna = premenna - 10 >>>premenna + 15</pre>
3.	Dnes majú v kaviarni 10 % zľavu na všetko. Ak sme si pomenovali cenu kávy a cenu hotdogu, ako vypočítame cenu nákupu jedného hotdogu a jednej kávy?: a) $(kava + hotdog) * 10$ b) $kava + hotdog * 0.9$ c) $(kava + hotdog) * 0.9$ d) $(kava + hotdog) - (kava + hotdog) * 0.1$

VEDOMOSTI V KOCKE

Programovanie je nástroj, ktorý nám pomáha pri riešení problémov.

V konzole jazyka Python vieme realizovať jednoduché výpočty a vyhodnocovať matematické výrazy:

```
sčítanie:      >>>10 + 2.5          #12.5
odčítanie:     >>>13 - 20         #-7
násobenie:    >>>15 * 4          #60
delenie:       >>>16 / 3         #5.333333333333333
```

Vybranú hodnotu môžeme pomenovať:

```
>>>cena_listka = 2.5             #cena lístka
```

Neskôr ju použiť:

```
>>>vstupne_spolu = cena_listka * 5 #vstupné pre 5-člennú rodinu
```

Alebo ju zmeniť:

```
>>>cena_listka = cena_listka * 0.9 #cena lístka sa znížila o 10%
```

Pri výpočtoch môžeme urobiť rôzne chyby:

```
>>>/2                               #Python nevie čo chceme, nerozumie nám
                                     #SYNTAKTICKÁ chyba
```

```
File "<input>", line 1
  /2
  ^
SyntaxError: invalid syntax
```

```
>>>1 / 0                             #Python vie čo chceme, ale nevie to spraviť
                                     #BEHOVÁ chyba
```

```
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ZeroDivisionError: division by zero
```

```
>>>strana = 10                       #Python spravil čo sme mu zadali
>>>obsah_stvorca = 4 * strana
>>>obsah_stvorca                       #Výsledok však nie je správny
40                                     #Chybu sme spravili my
                                     #LOGICKÁ chyba
```

ĎALŠIE ÚLOHY NA PRECVIČENIE

V niektorých zadaniach nie sú uvedené všetky potrebné informácie pre výpočet. Potrebné chýbajúce údaje si môžete vyhľadať, napr. na internete.

Úloha 11 Index telesnej hmotnosti (angl. Body Mass Index – BMI) patrí medzi najviac používané metódy merania obezity. Počíta sa ako hmotnosť v kilogramoch delená druhou mocninou výšky v metroch. Vypočítajte vaše BMI. Zodpovedajúce zdravotné riziko zistíte v nasledujúcej tabuľke:

BMI	kategória	zdravotné riziko
(0; 18,5)	podváha	stredné až vysoké
<18,5; 25)	normálna hmotnosť	nízke
<25; 30)	nadváha	zvýšené
<30; 35)	obezita 1. stupňa	stredné
<35; 40)	obezita 2. stupňa	vysoké
<40; ∞)	obezita 3. stupňa	životu nebezpečné

Môžeme na základe BMI prisúdiť zdravotné riziko každému človeku? Ak nie, nájdite príklady.

Riešenie:

>>>
>>>
>>>
>>>
>>>

Poznámka:

BMI je len jeden z ukazovateľov zdravotného stavu. Športovci, kulturisti, malé deti môžu mať BMI v rizikových hodnotách. Ich zdravotný stav však môže byť veľmi dobrý.

Úloha 12 Akú približnú hmotnosť má sklenené akvárium s rozmermi:

dĺžka = 1 m,

šírka = 0,3 m,

výška = 0,5 m,

ak sme na jeho výrobu použili sklo s hrúbkou 1 cm?

Riešenie:

>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>

Poznámka:

Hustota skla sa vo fyzikálnych tabuľkách udáva medzi 2400 až 2800 kg · m⁻³. Hmotnosť lepidla môžeme zanedbať.

Úloha 13 Radar (z angl. radio detection and ranging) alebo rádiolokátor je zariadenie, ktoré vysiela elektromagnetické vlny a následne sníma ich odraz od objektu, schopného tieto vlny odrážať. Smerová anténa sa otočí o 360 ° za 5 s. Akou rýchlosťou sa približuje lietadlo smerujúce k radaru, ak prvý odraz radar zaznamenal za $5,33333 \cdot 10^{-4}$ s a druhý odraz za $5,25 \cdot 10^{-4}$ s?

Riešenie:

>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>

Poznámka:

Náročnejšiu verziu tejto úlohy dostaneme, ak lietadlo neletí smerom k radaru a smer k lietadlu sa medzi dvoma odrazmi zmenil o nenulový uhol. Ten je potrebný zadať.

Úloha 14 Predstavte si, že by sme okolo rovníka natiahli drôt tak, aby tesne obopínal Zem. Ak by sme ho potom na jednom mieste presekli, predĺžili o 50 cm a rovnomerne rozmiestnili okolo Zeme, mohla by popod tento drôt prejsť mačka?

Riešenie:

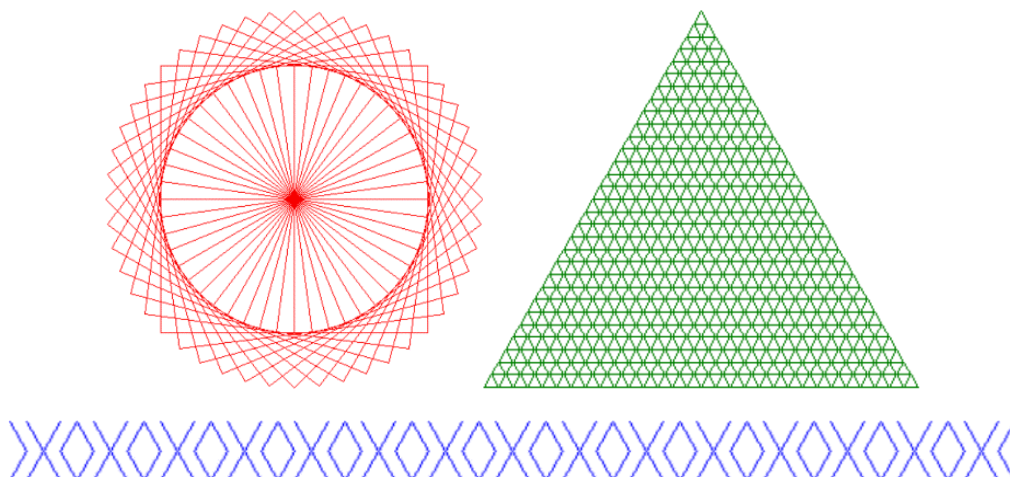
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>

02 KORYTNAČIA GRAFIKA

ZAPOJENIE

Diskusia k nástrojom na kreslenie obrázkov

1. Pomocou ktorých nástrojov viete kresliť obrázky?
2. Pomocou ktorého nástroja by ste vykreslili uvedené tri obrázky?

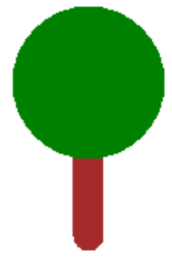


3. Závisí výber kresliaceho nástroja od podoby obrázka?

SKÚMANIE

Úloha 1 Preskúmajte, čo sa stane, ak do konzoly postupne zadáte príkazy jazyka Python uvedené v tabuľke. Vedľa príkazu napíšte svoju predpoveď aj zistenú skutočnosť. (Zadávať každý príkaz do konzoly samostatne (aj keď sú v jednej bunke tabuľky 2 riadky) a na potvrdenie príkazov stlačte kláves ENTER).

#	Príkaz(y)	Aký výsledok predpovedáte?	Čo ste zistili po spustení príkazov?
1.	<code>forward(100)</code>		
2.	<code>import turtle</code> <code>pero = turtle.Turtle()</code>		
3.	<code>pero.forward(100)</code>		
4.	<code>pero.left(60)</code>		
5.	<code>pero.penup()</code> <code>pero.forward(100)</code>		
6.	<code>pero.pendown()</code> <code>pero.backward(100)</code>		
7.	<code>pero.clear()</code>		



Úloha 2 Janko chce v editore kódu naprogramovať strom uvedený na obrázku. Jeho program **strom.py** však nevykresľuje strom podľa predlohy. Pomôžte mu upraviť jeho program, aby správne vykreslil uvedený strom.

Odporúčame, aby ste najprv otvorili program **strom.py** a spustili jeho kód kliknutím na uvedené oblasti v dolnom obrázku. Program vieme spustiť aj stlačením klávesov **SHIFT + F10**. Následne by ste mali preskúmať vysvetľujúce jednoriadkové komentáre vpravo od príkazov a upraviť program tak, aby vykreslil strom podľa predlohy na obrázku vpravo.

```
1 import turtle # rozšírenie Pythonu o príkazy korytnačej grafiky
2
3 tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
4 pero = turtle.Turtle() # vytvorí sa grafické pero s menom 'pero'
5
6 pero.pensize(5) # nastaví sa hrúbka pera na 5 bodov
7 pero.pencolor('brown') # nastaví sa farba pera na hnedú
8 pero.forward(100) # pero sa posunie o 100 bodov dopredu
9 pero.dot(100, 'green') # vykreslí zelený kruh s priemerom 100 bodov
10 pero.forward(-100) # pero sa posunie o 100 bodov vzad
11
12 tabula.mainloop() # ponechá otvorené okno s grafickou plochou
```

Úloha 3 Podľa návodu v súbore **I_SS_02_Python_Navod_PyCharm.pdf** (resp. **I_SS_02_Python_Navod_IDLE.pdf**) vytvorte vo vývojovom prostredí JetBrains PyCharm Edu projekt (resp. pre IDLE príčinok) **kreslenie** a program na vykreslenie veľkého tlačeneho písmena **L**. Vytvorený program uložte v Python súbore s menom **elko.py**.

a) Najprv považujte a uveďte, ktoré grafické príkazy použijete na vykreslenie písmena **L**:

.....

b) Po vytvorení programu, uveďte adresu, kde ste uložili súbor **elko.py** na vašom lokálnom disku:

.....

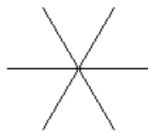
VYSVETLENIE

Prediskutujte a vysvetlite:

- Ktoré príkazy korytnačej grafiky poznáte v jazyku Python a aký je ich význam?
- Akú štruktúru má program v jazyku Python na vykreslenie obrázkov pomocou korytnačej grafiky?
- Ako postupujete pri programovaní vykreslenia obrázkov?
- Ako pomenúvate svoje programy, ktoré vytvárate?
- Kde na disku sú uložené vaše Python programy?

ROZPRACOVANIE

Úloha 4 Vytvorte program **vlocka.py** na vykreslenie vložky uvedenej na obrázku a časti kódu okomentujte:



Úloha 5 Najprv slovne alebo graficky uveďte aký obrázok vykreslí dole uvedený program:

Potom otvorte program **neznamy.py** a overte, či ste uviedli správny obrázok.

```
import turtle # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle() # vytvorí sa grafické pero s menom 'pero'
tabula.bgcolor('lightyellow')

pero.forward(100)
pero.left(45)
pero.forward(-20)
pero.forward(20)
pero.left(-90)
pero.forward(-20)
pero.forward(20)
pero.left(45)
pero.forward(-100)

tabula.mainloop() # ponechá otvorené okno s grafickou plochou
```

Úloha 6 Vytvorte program **hodiny.py** na vykreslenie hodín ukazujúcich čas 10:00 a program **trikolora.py** na vykreslenie trojfarebnej trikolóry podľa predlohy na uvedených obrázkoch:

Riešte
podľa
pokynov
učiteľa

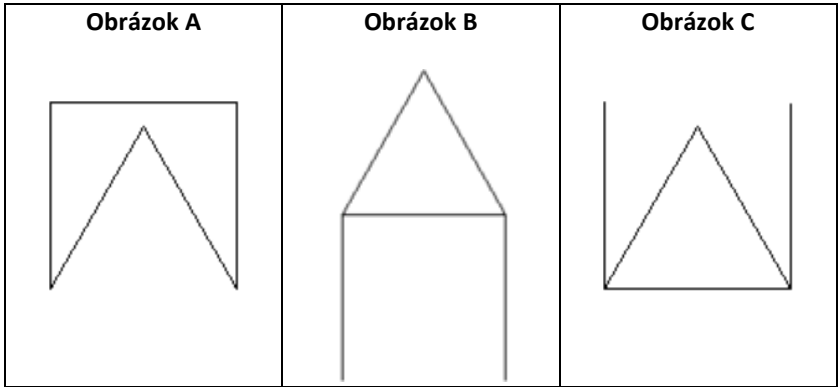


Uveďte, ako by ste postupovali v prípade vykreslenia slovenskej trikolóry s bielou farbou na okraji:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>Zakrúžkujte, ktorý z obrázkov A, B alebo C sa vykreslí pomocou uvedenej postupností príkazov.</p> <pre>pero.forward(100) pero.forward(-100) pero.right(90) pero.forward(100) pero.forward(-100) pero.left(150) pero.forward(100) pero.right(120) pero.forward(100) pero.right(30) pero.forward(100)</pre> <p>Vo vybranom obrázku vyznačte štartovaciu a cieľovú pozíciu a natočenie grafického pera.</p> <p>Uveďte, koľkokrát (.....) bol použitý príkaz <code>forward()</code> a koľko (.....) úsečiek je vykreslených na obrázku.</p> <p>Uveďte o aký celkový uhol sa natočilo grafické pero v cieľovej pozícii oproti štartovacej pozícii:</p>
----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



VEDOMOSTI V KOCKE

Príkazy korytnačej grafiky nie sú základnou súčasťou jazyka Python, ale sú dostupné importovaním modulu **turtle**.

Program na vykreslenie obrázkov pomocou príkazov korytnačej grafiky má nasledovnú štruktúru:

```
import turtle # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle() # vytvorí sa grafické pero s menom 'pero'

# postupnosť príkazov na vykreslenie obrázka

tabula.mainloop() # ponechá otvorené okno s grafickou plochou
```

Základné príkazy korytnačej grafiky sú:

Zápis príkazu	Význam príkazu
<code>pero.forward(100)</code>	Grafické pero sa posunie dopredu o 100 bodov
<code>pero.backward(50)</code>	Grafické pero sa posunie vzad o 50 bodov
<code>pero.left(90)</code>	Grafické pero sa otočí o 90 stupňov vľavo
<code>pero.right(60)</code>	Grafické pero sa otočí o 60 stupňov vpravo
<code>pero.penup()</code>	Grafické pero sa zdvihne (pri pohybe nekreslí)
<code>pero.pendown()</code>	Grafické pero sa položí (pri pohybe kreslí)
<code>pero.pencolor('red')</code>	Grafické pero nastaví svoju farbu na červenú
<code>pero.pensize(5)</code>	Grafické pero nastaví svoju hrúbku na 5 bodov
<code>pero.dot(100, 'green')</code>	Grafické pero vykreslí zelený kruh s priemerom 100 bodov
<code>pero.clear()</code>	Zmaže sa všetko z plátna, čo vykreslilo dané grafické pero
<code>tabula.bgcolor('yellow')</code>	Nastaví farbu plátna na žltú
<code>pero.setpos(-100, 100)</code>	Nastaví grafické pero na súradnicu (-100, 100)

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 7 Zakrúžkovaním vyberte skupinu príkazov (A, B alebo C), pomocou ktorej sa vykreslí rovnoramenný lichobežník s dĺžkou ramien 50 bodov a dĺžkou základní 100 a 50 bodov.



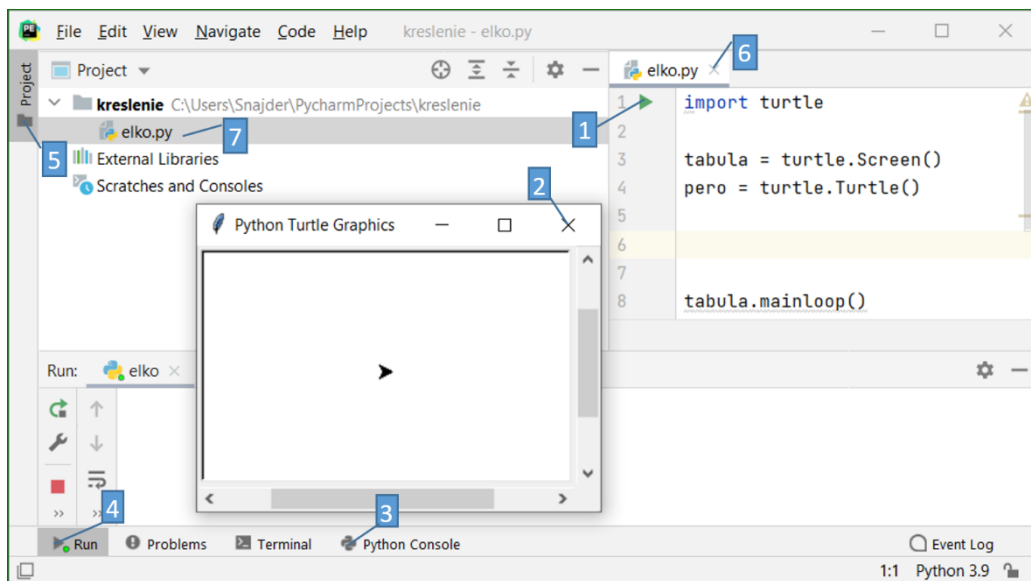
Skupina príkazov A	Skupina príkazov B	Skupina príkazov C
pero.forward(100) pero.left(60) pero.forward(50) pero.left(120) pero.forward(50) pero.left(120) pero.forward(50)	pero.forward(100) pero.left(120) pero.forward(50) pero.left(60) pero.forward(50) pero.left(60) pero.forward(50)	pero.forward(100) pero.left(120) pero.forward(50) pero.left(60) pero.forward(50) pero.left(60) pero.forward(50) pero.left(120)

Po výbere správnej skupiny príkazov prediskutujte so spolužiakmi v čom sa navzájom odlišujú:

- a) skupiny príkazov A a B:
- b) skupiny príkazov B a C:

Úloha 8

Riešte podľa pokynov učiteľa



Okno s vývojovým prostredím JetBrains PyCharm Edu prekryté grafickým oknom.

Klikaním preskúmajte prvky vývojového prostredia označené na obrázku číslami 1 až 7. Svoje zistenia zapíšte do tabuľky. (Poznámka: Miesto 2 nie je v okne prostredia JetBrains PyCharm Edu, ale v samostatnom okne)

(Akcia) Po kliknutí na miesto:	(Odpoveď) Udialo sa:
1	
2	
3	
4	
5	
6	

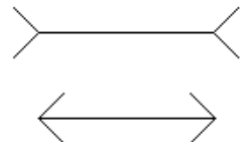
Úloha 9 Pomocou webovej stránky <https://docs.python.org/3/library/turtle.html> nájdite skratky príkazov, ktoré ste použili v predchádzajúcich úlohách a uveďte ich do tabuľky. Do posledného stĺpca tabuľky uveďte rovnocenný zápis pôvodného príkazu pomocou iného príkazu, ak existuje.

#	Príkaz	Skratka	Zápis pomocou iného príkazu
1.	<code>turtle.forward(100)</code>		
2.	<code>turtle.backward(100)</code>		
3.	<code>turtle.left(60)</code>		
4.	<code>turtle.right(120)</code>		
5.	<code>turtle.penup()</code>		
6.	<code>turtle.pendown()</code>		
7.	<code>turtle.pencolor('red')</code>		
8.	<code>turtle.pensize(5)</code>		
9.	<code>turtle.clear()</code>		

Zdôvodnite, prečo pri niektorých príkazoch sa do zátvorky neuvádzajú žiadne hodnoty, inde je číslo alebo text:

.....

Úloha 10 Na obrázku sú vykreslené dve vodorovné úsečky. Na hornej vodorovnej úsečke z jej krajných bodov vychádzajú ďalšie šikmo orientované úsečky smerom von. A na dolnej úsečke z jej krajných bodov vychádzajú ďalšie šikmo orientované úsečky smerom dovnútra. Ktorá z vodorovných úsečiek je dlhšia?



Neuveríte, ale horná úsečka je kratšia (ma dĺžku 99 bodov, dolná úsečka má dĺžku 100 bodov). Ide o známy optický klam. Na jeho preskúmanie vytvorte program **opticky_klam.py**.

(Poznámka: Najprv nakreslite rovnako dlhé úsečky, potom vykresľujte hornú úsečku stále kratšiu a kratšiu ako dolnú, kým nebudete mať pocit, že sú obe úsečky rovnako dlhé)

Úloha 11 Vytvorte program na vykreslenie loga vlastnej firmy (hudobnej skupiny, ornamentu, rodinného erbu, vtipnej dopravnej značky) pozostávajúceho z farebných úsečiek a kruhov.

03 VLASTNÉ FUNKCIE BEZ PARAMETROV A BEZ NÁVRATOVEJ HODNOTY

ZAPOJENIE

Úloha 1 Po viacnásobnom otočení maliarskeho valčeka sa na stenu odtlačil uvedený obrázok s ľudovým motívom:

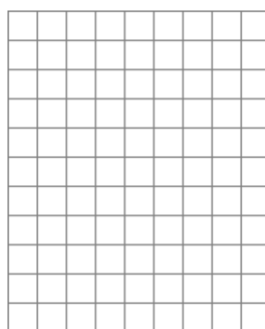
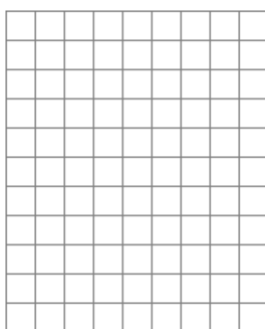
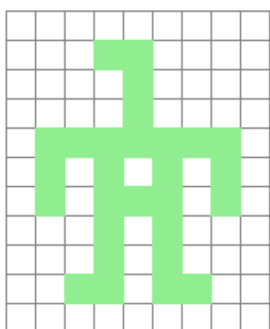


Vyznačte v tomto obrázku vzor, ktorý je nanesený na valčeku. Prípadne tento vzor nakreslite vedľa obrázku.

Uveďte koľkokrát sa otočil valček so vzorom, ktorý na stenu postupne odtlačil uvedený obrázok:krát

Úloha 2 Navrhnite vzor pre pečiatku (pečiatky), pomocou ktorej (ktorých) opečiatkujete celý tvar panáka na obrázku.

Riešte
podľa
pokynov
učiteľa



Uveďte, či stačí na opečiatkovanie celého tvaru panáka pečiatka s jedným vzorom: áno – nie

Ak ste prišli aj iné riešenia tejto úlohy, zakreslite ich do prázdnych štvorcových mriežok vedľa tej s panákom.

Prediskutujte so spolužiakmi svoje riešenia z pohľadu veľkosti a zložitosti pečiatky a počtu jej opečiatkovaní pri tvorbe panáka.

SKÚMANIE

Úloha 3 Na uvedenom obrázku nájdite a (orámčekovaním) vyznačte vzor, pomocou ktorého vieme zostaviť daný obrázok.



V programe na vykreslenie daného obrázka (orámčekovaním) vyznačte časti, ktoré vykresľujú nájdený vzor.

```
import turtle

pero = turtle.Turtle()
tabula = turtle.Screen()
pero.penup()

pero.dot(50, 'black')
pero.forward(50)
pero.dot(50, 'lightgray')
pero.forward(50)
pero.dot(50, 'black')
pero.forward(50)
pero.dot(50, 'lightgray')
pero.forward(50)
pero.right(90)
pero.dot(50, 'black')
pero.forward(50)
pero.dot(50, 'lightgray')
pero.forward(50)

tabula.mainloop()
```

Úloha 4 Preskúmajte uvedený program **corobim.py**. (Poznámka: Na konci riadku 04 s novým príkazom `vzor()` je uvedená dvojbodka. Riadky 05 až 09 sú odsadené dohodnutým počtom medzier, napr. 4.)

```
01 import turtle
02
03
04 def vzor():
05     # zápis postupu vykreslenia vzoru = definovanie funkcie vzor()
06     pero.dot(50, 'black')
07     pero.forward(50)
08     pero.dot(50, 'lightgray')
09     pero.forward(50)
10
11
12 tabula = turtle.Screen()
13 pero = turtle.Turtle()
14 pero.penup()
15
16 vzor()           # vykonanie postupu = volanie funkcie vzor()
17 pero.right(90)  # otočenie sa o 90 stupňov vpravo
18 vzor()           # vykonanie postupu = volanie funkcie vzor()
19
20 tabula.mainloop()
```

- a) Popíšte alebo nakreslite obrázok, ktorý vykreslí tento program:
- b) Vysvetlite, aký je rozdiel medzi príkazom `def vzor()` : na riadkoch 04 až 09 a príkazom `vzor()` na riadkoch 16 a 18:
- c) Uveďte, akú zmenu treba urobiť v uvedenom programe, aby vykreslil rovnaký obrázok ako v úlohe 3:

VYSVETLENIE

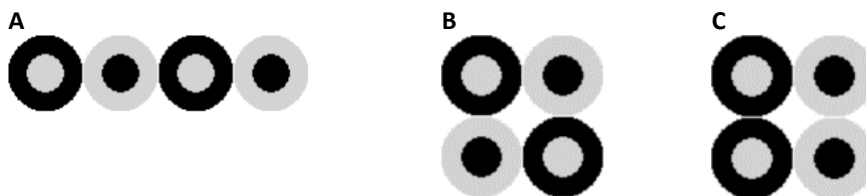
Prediskutujte a vysvetlite:

1. V čom je upravené riešenie úlohy 4 využívajúce vlastné funkcie lepšie, či horšie ako riešenie úlohy 3 bez vlastných funkcií?
2. V ktorej časti programu vytvárame vlastnú funkciu a v ktorej ju používame? Mohli by sme najprv použiť funkciu a až potom ju vytvoriť?
3. Koľkokrát vytvárame vlastnú funkciu a koľkokrát ju môžeme použiť?
4. Z akých častí pozostáva zápis vytvorenia vlastnej funkcie?
5. Kde končí zápis vytvorenia vlastnej funkcie?
6. Ako sa zmení vykonávanie programu, ak v ňom neuvedieme komentáre?
7. Aký význam pre programátora majú komentáre? Nie je to zbytočná strata času pri ich uvádzaní?

ROZPRACOVANIE

Úloha 5 Vytvorte program **stvorgulka2.py** na vykreslenie obrázkov A až C s použitím funkcie na vykreslenie vzoru.

Riešte podľa pokynov učiteľa

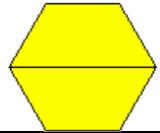


Úloha 6 Preskúmajte, čo robí uvedený program **krabicka.py**.

```

01 import turtle
02
03
04 def obrazok():
05     pero.fillcolor('green')
06     pero.begin_fill()
07     pero.forward(100)
08     pero.left(120)
09     pero.forward(50)
10     pero.left(60)
11     pero.forward(50)
12     pero.left(60)
13     pero.forward(50)
14     pero.left(120)
15     pero.end_fill()
16

```



```
17
18 tabula = turtle.Screen()
19 pero = turtle.Turtle()
20
21 obrazok()
22
23 tabula.mainloop()
```

a) Vysvetlite význam príkazov `fillcolor()`, `begin_fill()`, `end_fill()`:

.....

b) Uvedený program upravte tak, aby vykreslil nasledovný obrázok:

Úloha 7 Vytvorte program **stromy.py** na vykreslenie uvedeného obrázku.

Riešte
podľa
pokynov
učiteľa



VYHODNOTENIE

SEBAHODNOTIACI TEST

1.

Zakrúžkovaním vyberte skupinu (skupiny) príkazov, pomocou ktorej (ktorých) sa vykreslí uvedený obrázok.



Skupina príkazov A

Skupina príkazov B

Skupina príkazov C

<pre>def vzor(): pero.dot(40, 'black') pero.forward(40) pero.dot(40, 'yellow') pero.forward(40) pero.dot(40, 'red') pero.forward(40) pero.dot(40, 'blue') pero.forward(40) pero.dot(40, 'yellow') pero.forward(40) pero.dot(40, 'red') pero.forward(40) vzor() vzor() vzor()</pre>	<pre>def belgicko(): pero.dot(40, 'black') pero.forward(40) pero.dot(40, 'yellow') pero.forward(40) pero.dot(40, 'red') pero.forward(40) def rumunsko(): pero.dot(40, 'blue') pero.forward(40) pero.dot(40, 'yellow') pero.forward(40) pero.dot(40, 'red') pero.forward(40)</pre>	<pre>def dvojgulka(): pero.dot(40, 'yellow') pero.forward(40) pero.dot(40, 'red') pero.forward(40) def sestgulka(): pero.dot(40, 'black') pero.forward(40) dvojgulka() pero.dot(40, 'blue') pero.forward(40) dvojgulka()</pre>
	<pre>belgicko() rumunsko() belgicko() rumunsko()</pre>	<pre>sestgulka() sestgulka()</pre>

Skupinu (skupiny) príkazov s chybným riešením opravte. Stručne okomentujte uvedené riešenia úlohy:

.....

.....

.....

VEDOMOSTI V KOCKE

Pri riešení problémov môžeme využívať rôzne stratégie, napr. **dekompozíciu** (rozdelíme problém na menšie podproblémy), **hľadanie vzoru** (nájdeme a využijeme opakujúceho sa vzoru), **vykreslenie si náčrtu obrázku**, atď. Pri programovaní riešení problémov využívajúcich tieto stratégie sa používajú **vlastné funkcie**.

Pri maľovaní obrázkov s opakujúcim sa vzorom (vzormi) môžeme využiť pečiatku (pečiatky) s daným vzorom (danými vzormi). Postupujeme pri tom tak, že najprv si vyrobíme pečiatky a potom ich namočené odtlačíme na maľovaciu plochu. Pri programovaní je to takmer rovnaké. Najprv vytvoríme (=definujeme) vlastnú funkciu, ktorá popisuje riešenie podproblému a potom ju použijeme (=voláme) pri riešení problému.

V nasledovnej tabuľke uvádzame schematické zápisy dvoch príkladov vlastných funkcií – na vykreslenie obrázku s jedným, resp. dvoma vzormi (pečiatkami).

Program na vykreslenie obrázku s 1 vzorom	Program na vykreslenie obrázku s 2 vzormi
<pre>def pečiatka(): ''' príkazy na vykreslenie vzoru, z ktorého pozostáva obrázok ''' def presun1(): # príkazy na 1. presun grafického pera def presun2(): # príkazy na 2. presun grafického pera pečiatka() presun1() pečiatka() presun2() pečiatka()</pre>	<pre>def pečiatka1(): # príkazy na vykreslenie vzoru 1 def pečiatka2(): # príkazy na vykreslenie vzoru 2 def presun1(): # príkazy na 1. presun grafického pera def presun2(): # príkazy na 2. presun grafického pera pečiatka1() presun1() pečiatka2() presun2() pečiatka1()</pre>

Syntax **definície funkcie** obsahuje kľúčové slovo `def` s priliehavým názvom funkcie, za ktorým je dvojica okrúhlych zátvoriek nasledovaná dvojbodkou. Tento prvý riadok funkcie označujeme aj ako hlavička definície funkcie. Po nej nasleduje telo definície funkcie, ktoré obsahuje postupnosť príkazov, ktoré sú v porovnaní s hlavičkou odsadené od ľavého okraja rovnakým počtom medzier (štandardne 4). Ak na niektorom z riadkov nie sú príkazy už odsadené, tieto príkazy nepatria už do tela definície funkcie.

Syntax **volania funkcie** obsahuje samotný názov funkcie nasledovaný dvojicou okrúhlych zátvoriek.

V niektorých prípadoch môžu vzory obsahovať v sebe ešte ďalšie vzory. Napr. text `LISTLIPALIST`, obsahuje dva vzory `LIST` a `LIPA`, pričom každý z nich obsahuje ďalší vzor `LI`. Program na vypísanie textu by mohol mať nasledovný schematický zápis:

Program na vypísanie textu s vnorenými vzormi

```
def li():
    l() # vypíše L
    i() # vypíše I

def list():
    li() # vypíše LI
    s() # vypíše S
    t() # vypíše T

def lipa():
    li() # vypíše LI
    p() # vypíše P
    a() # vypíše A

list() # vypíše LIST
lipa() # vypíše LIPA
list() # vypíše LIST
```

Typickými chybami začínajúcich programátorov v jazyku Python sú:

- zabúdanie písania okrúhlych zátvoriek za názvom funkcie pri jej definovaní či volaní;
- zabúdanie písania dvojbodky na konci hlavičky definície funkcie;
- nedôsledné odsadzovanie príkazov v tele definície funkcie;
- považovanie prázdneho riadka za ukončenie tela funkcie;
- používanie nepriliehavých názvov funkcií; nepoužívanie komentárov.

Používanie komentárov má význam pre sprehľadnenie zápisov funkcií programu:

- v procese návrhu funkcie (ako poznámky, čo treba urobiť – „to do“),
- na zdokumentovanie funkcionality funkcie (ako komentáru, čo robí daná funkcia),
- na prípadné znefunkčnenie časti kódu.

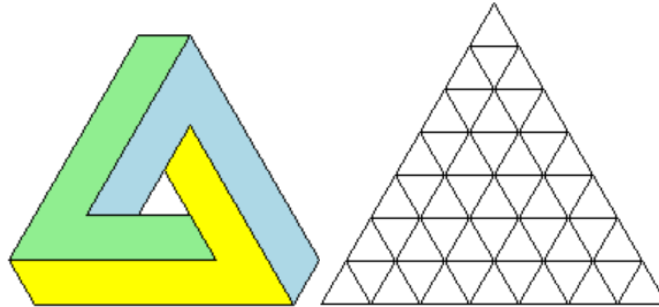
Použitie vlastných funkcií prináša nasledovne **výhody**:

- písanie prehľadnejšieho kódu odpovedajúceho riešenému problému (riešenie každého podproblému bude reprezentovať jedna funkcia),
- lepšia lokalizácia a opravovanie chýb v programe,
- skrátenie kódu viacnásobnou znovupoužiteľnosťou už raz napísaného programového kódu,
- delba práce medzi viacerých programátorov (každý z nich rieši len vybrané podproblémy).

ĎALŠIE ÚLOHY NA PRECVIČENIE

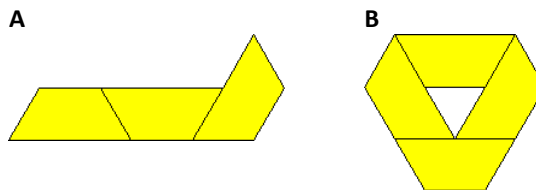
Úloha 8 Poznáte optický klam – Penroseov trojuholník? Zaujímavé je, že si ho v priestore ťažko vieme predstaviť, ale v rovine ho vieme vykresliť. (Zdroj: https://sk.wikipedia.org/wiki/Penroseov_trojuholn%C3%ADk)

Do zobrazenej trojuholníkovej mriežky zakreslite jeden z farebných dielov uvedeného Penroseovho trojuholníka a popíšte jeho presné rozmery (dĺžky strán a uhly nimi zvierané).

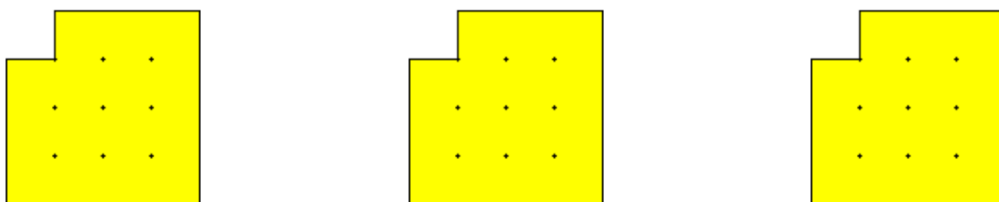


Zakreslite tiež počítačové pozície vykreslenia všetkých troch dielov Penroseovho trojuholníka.

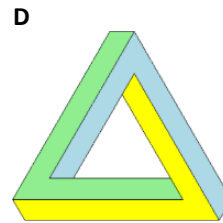
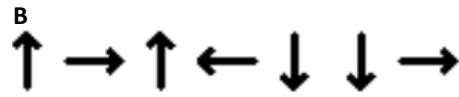
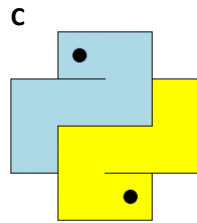
Úloha 9 Vytvorte programy na vykreslenie uvedených obrázkov, v ktorých použijete vlastnú funkciu pre vykreslenie vzoru.



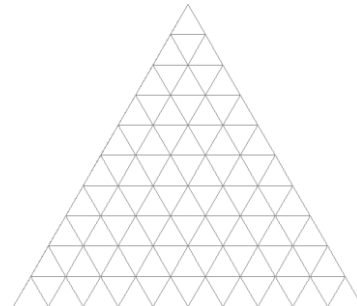
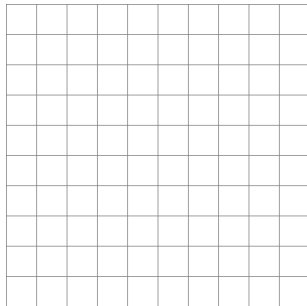
Úloha 10 Máme obrázok s veľkosťou 15 štvorcových jednotiek. Chceme vyrobiť špeciálny puzzle, ktorý obsahuje len rovnaké dieliky. Zakreslite aspoň jeden spôsob pokrytia celého obrázka rovnakými dielikmi.



Úloha 11 Vytvorte programy `cicmiansky_vzor.py`, `sipkovy_jazyk.py`, `logo_pythonu.py`, `penroseov_trojuholnik.py`, ktoré vykreslia uvedené obrázky.



Úloha 12 Navrhните a naprogramujte vlastný zaujímavý obrázok (napr. puzzle, hrací plán stolovej hry, hlavolam, optický klam, logo kapely, čipku či ozdobný ornament), ktorý vznikne opečiatkovaním jedného či dvoch vzorov vo štvorcovej alebo trojuholníkovej mriežke.



04 CYKLUS S PEVNÝM POČTOM OPAKOVANÍ

ZAPOJENIE

Úloha 1 Opíšte *stručne* a čo *najpresnejšie* (pre kamaráta na telefóne), čo vidíte na uvedených obrázkoch **A, B, C, D**:

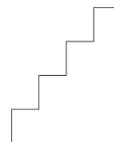
Obrázok A



Obrázok B



Obrázok C



Obrázok D



A

B

C

D

Uvedte, ktoré z obrázkov sa vám opisovali ľahšie a ktoré ťažšie

Uvedte, čo majú spoločné vaše popisy ľahšie opísateľných obrázkov:

.....

SKÚMANIE

Úloha 2 Preskúmajte obidva uvedené programy **A** a **B**. Najprv len na základe prečítania uvedeného programového kódu, potom po spustení ich kódov uložených v súboroch **corobim1.py** a **corobim2.py**.

Program A

```
import turtle

tabula = turtle.Screen()
pero = turtle.Turtle()
pero.penup()

pero.dot(40, 'orange')
pero.forward(40)
pero.dot(40, 'orange')
pero.forward(40)
pero.dot(40, 'orange')
pero.forward(40)
pero.dot(40, 'orange')
pero.forward(40)
pero.dot(40, 'orange')
pero.forward(40)
```

Program B

```
import turtle

tabula = turtle.Screen()
pero = turtle.Turtle()
pero.penup()

for i in range(5):
    pero.dot(40, 'orange')
    pero.forward(40)

tabula.mainloop()
```

- d) *Opište obrázky, ktoré vykreslia uvedené programy A a B:*
- e) *Ktorý zo zápisov pokladáte za lepší a prečo:*
- f) *Ako by sa zmenil výsledok programu B, ak by sme v riadku 7 namiesto `range(5)` uviedli `range(10)`:*

VYSVETLENIE

Prediskutujte a vysvetlite:

1. Na čo je dobrý príkaz `for`?
2. Z akých častí sa skladá zápis príkazu `for`?
3. Ako funguje príkaz `for`, aký význam majú jeho hlavička a telo?
4. Musíme použiť v príkaze `for` vlastné funkcie?

ROZPRACOVANIE

Úloha 3 V uvedenom programe **schody.py** vyznačte opakujúce sa časti a upravte ho tak, aby ste pomocou príkazu `for` skrátili jeho zápis.

Pôvodný program

```
import turtle

def schod():
    pero.forward(20)
    pero.left(-90)
    pero.forward(50)
    pero.left(90)

tabula = turtle.Screen()
pero = turtle.Turtle()
pero.left(90)

schod()
schod()
schod()
schod()
schod()

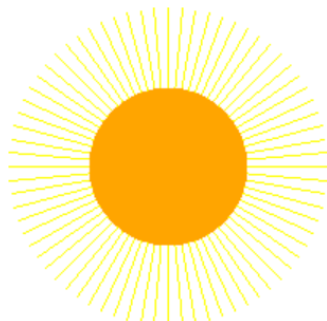
tabula.mainloop()
```

Upravená časť programu

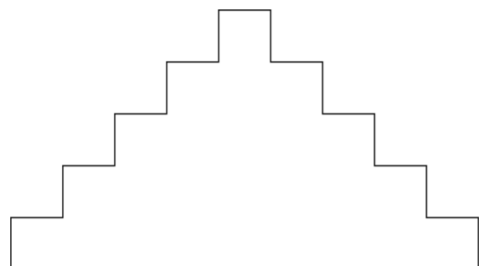
Úloha 4 Vytvorte programy **slnko.py** a **stupne_vitazov.py** vykresľujúce uvedené obrázky **A** a **B** tak, aby bolo grafické pero na konci vykreslenia v počiatočnej pozícii a počiatočnom natočení.

Riešte
podľa
pokynov
učiteľa

Obrázok A

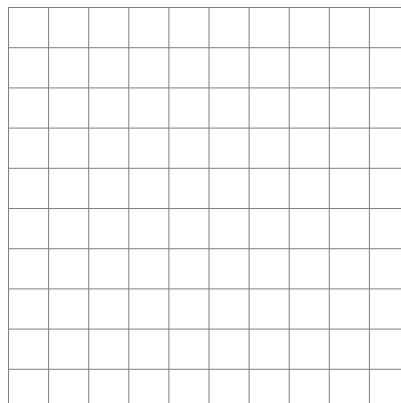
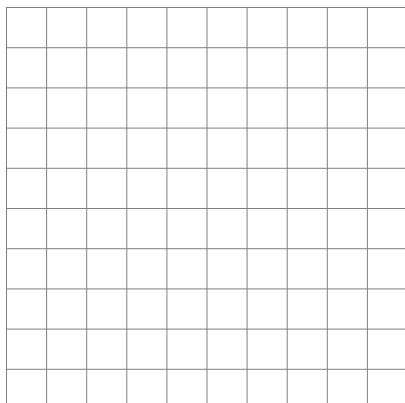


Obrázok B



Úloha 5 Potrebujeme vytvoriť hrací plán pre hru Piškvorky v tvare štvorcovej mriežky 10×10 . Do uvedených obrázkov načrtnite jeden, prípadne dva spôsoby vykreslenia tohto **hracieho plánu**. Vo svojom návrhu vyznačte **počiatočný a koncový bod** vykresľovania **s natočením** grafického pera a **vzor**, ktorý sa pravidelne opakuje v obrázku.

Riešte podľa pokynov učiteľa



Nakoniec vytvorte program **mriezka.py** na vykreslenie štvorcovej mriežky pomocou niektorého z navrhnutých spôsobov riešenia.

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>Uvedte koľkokrát sa vykoná vlastná funkcia <code>prikaz()</code> v programoch A, B a C.</p> <table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th style="width: 33%;">Program A</th> <th style="width: 33%;">Program B</th> <th style="width: 33%;">Program C</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;"> <pre>for i in range(3): prikaz() prikaz() prikaz()</pre> </td> <td style="text-align: left;"> <pre>for i in range(3): prikaz() prikaz()</pre> </td> <td style="text-align: left;"> <pre>for i in range(3): prikaz() prikaz() prikaz()</pre> </td> </tr> <tr> <td>Riešenie: ... krát</td> <td>Riešenie: ... krát</td> <td>Riešenie: ... krát</td> </tr> </tbody> </table>	Program A	Program B	Program C	<pre>for i in range(3): prikaz() prikaz() prikaz()</pre>	<pre>for i in range(3): prikaz() prikaz()</pre>	<pre>for i in range(3): prikaz() prikaz() prikaz()</pre>	Riešenie: ... krát	Riešenie: ... krát	Riešenie: ... krát
Program A	Program B	Program C								
<pre>for i in range(3): prikaz() prikaz() prikaz()</pre>	<pre>for i in range(3): prikaz() prikaz()</pre>	<pre>for i in range(3): prikaz() prikaz() prikaz()</pre>								
Riešenie: ... krát	Riešenie: ... krát	Riešenie: ... krát								
2.	<p>Upravte uvedený program test.py, aby vykreslil obrázok časti notovej osnovy tak, aby na konci vykreslenia bolo grafické pero v počiatočnej pozícii a v počiatočnom natočení. Dĺžka čiar notovej osnovy je 200 bodov a vzdialenosť prvej čiary od piatej je 100 bodov.</p> <pre>import turtle tabula = turtle.Screen() pero = turtle.Turtle() for i in range(5): pero.pendown() pero.forward(200) pero.forward(-200) pero.penup() pero.left(90) pero.forward(100 / 5) pero.left(-90) tabula.mainloop()</pre> <div style="text-align: right; margin-top: 10px;"> </div>									

VEDOMOSTI V KOCKE

Príkaz `for` sa využíva pri riešení problémov, ktoré pozostávajú z viacerých rovnakých (alebo podobných) podproblémov. Dá sa prirovnať k maliarskemu valčeku, ktorý pri otáčaní opakovane odtláča farbu so zadaným vzorom na stenu. Príkaz `for` sa tiež označuje ako **príkaz s pevným (známym) počtom opakovaní** alebo aj skrátene **cyklus** `for`.

#	Zápis kódu bez príkazu cyklu <code>for</code>	Zápis kódu s príkazom cyklu <code>for</code>
1	<code>príkazy()</code>	<code>for i in range(4):</code> <code>príkazy()</code>
2	<code>príkazy()</code>	
3	<code>príkazy()</code>	
4	<code>príkazy()</code>	
1	<code>príkazy()</code>	<code>for i in range(počet):</code> <code>príkazy()</code>
2	<code>príkazy()</code>	
... <code>počet</code>	... <code>príkazy()</code>	
1	<code>a()</code>	<code>for i in range(3):</code> <code>a()</code> <code>b()</code> <code>c()</code>
2	<code>b()</code>	
3	<code>a()</code>	
4	<code>b()</code>	
5	<code>a()</code>	
6	<code>b()</code>	
7	<code>c()</code>	
1	<code>a()</code>	<code>for i in range(3):</code> <code>a()</code> <code>b()</code> <code>c()</code>
2	<code>a()</code>	
3	<code>a()</code>	
4	<code>b()</code>	
5	<code>c()</code>	

Cyklus `for` pozostáva z **hlavičky cyklu** (prvého riadku začínajúcim slovom `for` a končiacim dvojbodkou, na ktorú často začiatovníci zabúdajú) a **tela cyklu** (riadkami s odsadenými príkazmi). Príkazy v tele cyklu sa vykonajú toľkokrát, ako je to uvedené v parametri funkcie `range()` v hlavičke cyklu. Telo cyklu sa neukončí zaradením voľného riadku, ale zrušením odsadenia príkazov v tele cyklu vzhľadom k hlavičke cyklu.

Pri kreslení obrázkov pomocou príkazu cyklu `for` je dôležité, aby v tele cyklu boli uvedené príkazy, po vykonaní ktorých sa dostane grafické pero **do určitej významnej pozície**, napr. počiatkový bod celého kreslenia, alebo počiatkový bod kreslenia nasledovnej časti obrázka.

Cyklus `for` aj vlastná funkcia umožňujú **skrátiť** a **sprehľadniť** programový kód. Pomocou vlastných funkcií sme riešenie problému mohli rozdeliť na riešenie rôznych podproblémov, napr. kreslenie obrázkov s rôznymi vzormi alebo s rovnakými vzormi aj na nepravidelných pozíciách. Pomocou cyklu `for` rozdelíme problém do rovnakých alebo podľa určitého pravidla podobných podproblémov, napr. kreslenie pravidelných obrázkov.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Uveďte spoločné a odlišné črty útvarov znázornených na obrázkoch A až D:

Obrázok A



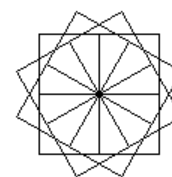
Obrázok B



Obrázok C



Obrázok D

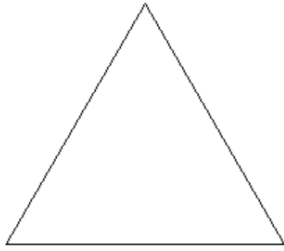


Spoločné črty vybraných útvarov	Odlišné črty vybraných útvarov

Vytvorte programy `nahrдельник1.py`, `nahrдельник2.py`, `stvorce1.py`, `stvorce2.py` na vykreslenie útvarov znázornených na obrázkoch A až D.

Úloha 7 V programe **rovinne_utvary.py** nahradte namiesto bodiek v 6. a 8. riadku programový kód tak, aby vykreslil: rovnostranný trojuholník, štvorec a päťcípú hviezd, ktoré sú uvedené na obrázkoch **A, B, C**.

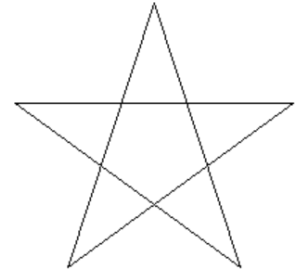
Obrázok A



Obrázok B



Obrázok C



```
import turtle

tabula = turtle.Screen()
pero = turtle.Turtle()

for i in range(...):
    pero.forward(50)
    pero.left(...)
```

```
import turtle

tabula = turtle.Screen()
pero = turtle.Turtle()

for i in range(...):
    pero.forward(50)
    pero.left(...)
```

```
import turtle

tabula = turtle.Screen()
pero = turtle.Turtle()

for i in range(...):
    pero.forward(50)
    pero.left(...)
```

(Poznámka: Pri návrhu riešenia sa zamyslite, v ktorej polohe je grafické pero na začiatku a na konci kreslenia, koľkými čiarami sa vykreslí obrázok, koľkokrát okolo vlastnej osi sa otočí grafické pero a v akom smere)

Úloha 8 Vyznačte vzor, ktorý sa opakuje v oboch obrázkoch **A** a **B**. Uvedte vlastnú funkciu na vykreslenie tohto vzoru:

Obrázok A



Obrázok B



Vytvorte programy **cipka.py** a **optoklam.py** vykresľujúce obrázky **A** a **B**.

Úloha 9 Vytvorte program **domceky.py** na vykreslenie radu 10 domov podľa nasledovného obrázka:



- Vypočítajte, koľko spolu čiar vykreslí váš program:
- Vypočítajte, aký najmenší počet čiar sa dá použiť na vykreslenie obrázka bez výplní:
- Prediskutujte, akú zmenu v programe treba urobiť, ak chcete vykresliť viacero radov domov nad sebou.

Úloha 10 Vytvorte program na vykreslenie vlastného zaujímavého obrázku (napr. hrací plán stolovej hry, ornament, obrázkov na vlastné hracie karty, ozdobu na privesok vytlačiteľnú na 3D tlačiarni). Obrázok pozostáva z aspoň jedného opakujúceho sa vzoru na pravidelne rozmiestnených pozíciách.

05 OPAKOVANIE I. + DIDAKTICKÝ TEST

ÚVOD

Úloha 1 Prepravná spoločnosť nás požiadala o návrh objednávkového systému, preto je potrebné vytvoriť program, ktorý by vykresľoval rozloženie miest v ich autobusoch. Načrtnite, ako by mohol vyzerat grafický výstup programu a navrhnite preň potrebné funkcie.



PRECVIČOVANIE - SAMOSTATNÁ PRÁCA

Úloha 2 Naprogramujte navrhnuté funkcie (z úlohy 1). Vytvorte program **autobus1.py**, ktorý pomocou naprogramovaných funkcií vykreslí rozloženie miest v autobuse podľa úlohy 1.

Úloha 3 Upravte program z predošlej úlohy tak, aby vykresľoval rozloženie miest v inom type autobusov (podľa obrázku vpravo) a uložte ho ako **autobus2.py**.



Úloha 4 Dopĺíte k pripravenej funkcii `sestuholnik()` v súbore `geometria.py` ďalšie funkcie, ktoré funkciu `sestuholnik()` volajú, pomocou ktorých je možné vykresliť nasledovný ornament určený na potlač obrusov:

Riešte podľa pokynov učiteľa

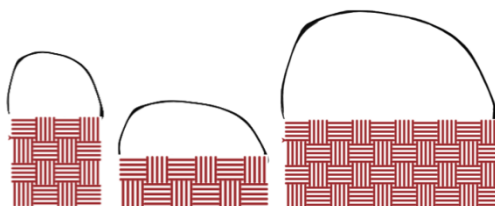


Tvorivé rozšírenie úlohy: Navrhните pomocou Vašich funkcií nový, zaujímavý a netradičný vzor potlače na obrus.



Úloha 5 Navrhните vhodné funkcie pre vykresľovanie jednoduchého výpletového vzoru pre košíky. Funkcie (resp. ich kombinácie) by mali umožňovať vykresľovanie vzorov pre všetky tri výrobné typy košíkov (zobrazené na obrázkoch nižšie). Riešenie uložte do súboru `kosik.py`.

Riešte podľa pokynov učiteľa

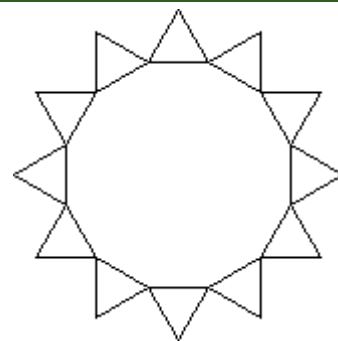


ZHRNUTIE

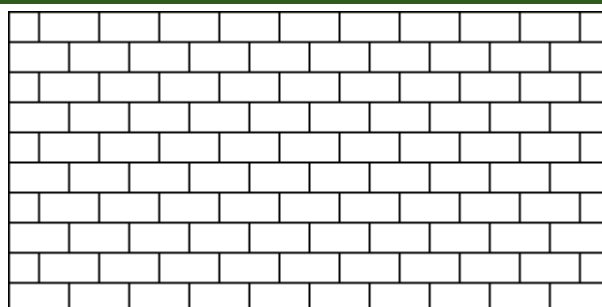
Diskusia k riešeniam úloh

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Vytvorte funkciu `ozubene_koliesko()` pre vykreslenie ozubeného kolieska podľa nasledovného vzoru. Riešenie uložte do súboru **koliesko.py**.



Úloha 7 Vytvorte program **mur.py** pre vykreslenie múru z tehál podľa vzoru:



Úloha 8 Vytvorte program **stromceky.py** pre vykreslenie radu stromov podľa obrázka:



06: VLASTNÉ FUNKCIE S PARAMETRAMI – KRESLIACE ÚLOHY

ZAPOJENIE

Diskusia o zovšeobecnení riešení úloh – parametrizácia funkcií

SKÚMANIE

Úloha 1 V súbore **sestuholnik.py** je program na kreslenie pravidelných šesťuholníkov. Vyznačte v jeho kóde v pracovnom liste **farebne** IBA tie parametre, ktoré spôsobia zmenu veľkosti šesťuholníka:

```
import turtle

def sestuholnik():
    pero.pendown()
    for i in range(6):
        pero.forward(20)
        pero.left(360 / 6)
    pero.penup()

tabula = turtle.Screen()
pero = turtle.Turtle()
pero.width(2)

sestuholnik()

tabula.mainloop()
```

Úloha 2 Doplňte do súboru **sestuholnik.py** podobným spôsobom programy na kreslenie rovnostranných trojuholníkov a štvorcov (t.j. pravidelných štvoruholníkov). Zapište svoje kódy:

```
def trojuholnik():
```

```
def stvoruholnik():
```


Úloha 3 Porovnajete kódy funkcií pre kreslenie trojuholníka, štvoruholníka a šesťuholníka a doplňte podľa nich tabuľku:

Objekt (pravidelný z hľadiska veľkosti uhlov)	Počet uhlov	Uhol otočenia korytnačky (= veľkosť vonkajších uhlov objektu)
trojuholník		
štvoruholník		
päťuholník		
šesťuholník		
sedemuholník		
n-uholník		

Úloha 4 Porovnajete kódy funkcií pre kreslenie trojuholníka, štvoruholníka a šesťuholníka. V ktorých častiach sa líšia?

Navrhnite, ako by mala vyzeráť funkcia `uholnik(n)`, kde `n` je parameter vyjadrujúci počet uhlov (napr. `uholnik(3)` nakreslí trojuholník, `uholnik(6)` nakreslí šesťuholník a pod.):

```
def uholnik(n):
```

Overte správnosť Vášho programu vykreslením trojuholníka a šesťuholníka. Funguje Váš program správne?

ÁNO – NIE

VYSVETLENIE

Diskusia o parametroch funkcií.

Vytvorili sme funkcie pre kreslenie uholníkov s rôznym počtom strán.

Ktoré časti kódu definujú, aký veľký vykreslený uholník bude?

Ako by bolo potrebné tieto časti kódov upraviť?

Ktoré časti nášho programu je tiež potrebné upraviť? Ako?

ROZPRACOVANIE

Úloha 5 Upravte funkciu `uholnik(n)` tak, aby sme pomocou nej dokázali kresliť rôzne veľké uholníky s rôznym počtom strán.

Pre funkciu vytvorte dokumentačný reťazec.

Riešenie:

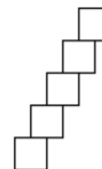
Úloha 6 Vytvorte nový súbor `stvorce.py`.

Časť b)
riešte
podľa
pokynov
učiteľa

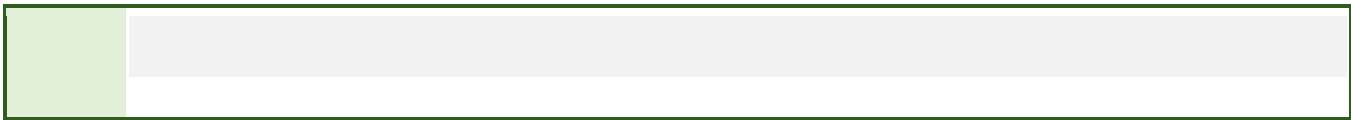
a) Vytvorte v ňom funkciu `veza(pocet, strana)` na vykreslenie veže z požadovaného počtu štvorcov zadanej veľkosti. TIP: Využite funkciu `stvoruholnik(velkost)`, ktorú ste v predošlých úlohách navrhli.
Pre funkcie vytvorte dokumentačné reťazce.



b) Dopĺňte do programu aj funkciu `schody(pocet, strana)` na vykreslenie schodiska z požadovaného počtu štvorcov zadanej veľkosti:
Pre funkcie vytvorte dokumentačné reťazce.



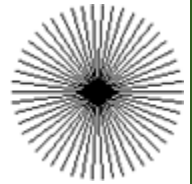
Riešenie:



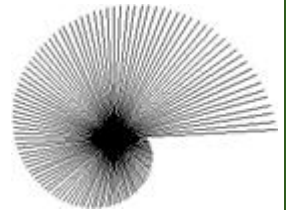
Úloha 7

Časť b)
riešte
podľa
pokynov
učiteľa

a) Vytvorte program **luce.py** a v ňom funkciu `kresli_luce(dlzka, pocet)` tak, aby vedela kresliť rôzne slnká s určeným počtom lúčov a určenou dĺžkou, teda napr. :



b) Pomocou neznámej funkcie `divne_luce()` sme nakreslili nasledujúci obrázok:



V čom sa naša neznáma funkcia líši od Vašej funkcie `luce()`?

Riešenie:

a)

b) .

Úloha 8

Riešte
podľa
pokynov
učiteľa

Vytvorte program **plot.py** na vykreslenie plota. Plot je zložený zo zadaného počtu tyčiek zo zadanou šírkou a výškou.



Spojky spájajúce tyčky plota sú vo výške $1/5$ a $4/5$ výšky tyčky.

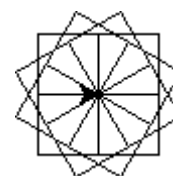
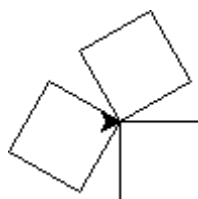
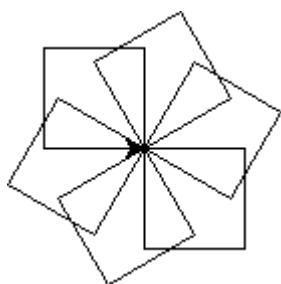
Pre funkcie vytvorte dokumentačné reťazce.

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

Nájdite chyby v nasledujúcom programe, aby sme pomocou neho potom mohli vykresľovať zadaný počet otočených štvorcov zadanej veľkosti, napr.:



```
def stvorec(strana):  
    for i in range(4):  
        pero.forward(50)  
        pero.right(90)  
  
def vzor(pocet):  
    for i in range(pocet):  
        stvorec()  
        pero.right(90)  
  
vzor()
```

VEDOMOSTI V KOCKE

Funkcie pri programovaní môžu používať jeden alebo viacero parametrov. Pomocou parametrov odovzdáme funkcii hodnoty, s ktorými bude ďalej pracovať. Na základe odovzdaných hodnôt môžeme zmeniť výsledné správanie sa funkcií.

Aby sme našim funkciám rozumeli aj neskôr, prípadne aby ich mohli používať aj iní, mali by funkcie obsahovať dokumentačné reťazce. Dokumentačný reťazec je reťaz uzatvorený v trojitých apostrofoch ihneď za hlavičkou funkcie. Stručne v ňom popíšeme čo funkcia robí a aké má parametre.

```
def n_uholnik(pocet, strana):
    ''' Vykreslí n-uholník so zadaným počtom strán zadanej dĺžky

    :param pocet: počet strán n-uholníka
    :type pocet: int
    :param strana: dĺžka strany n-uholníka
    :type strana: int
    '''
    pero.pendown()
    for i in range(pocet):
        # použitie parametra strana v tele samotnej funkcie
        pero.forward(strana)
        # použitie parametra pocet v tele samotnej funkcie
        pero.left(360 / pocet)
    pero.penup()

# volanie funkcie aj s hodnotami, kreslíme 5-uholník so stranou dĺžky 10
n_uholnik(5, 10)
```

Nezabudnime, že v programe voláme funkciu aj s vhodnými hodnotami parametrov, napr. `stvoruholnik(5, 10)`.

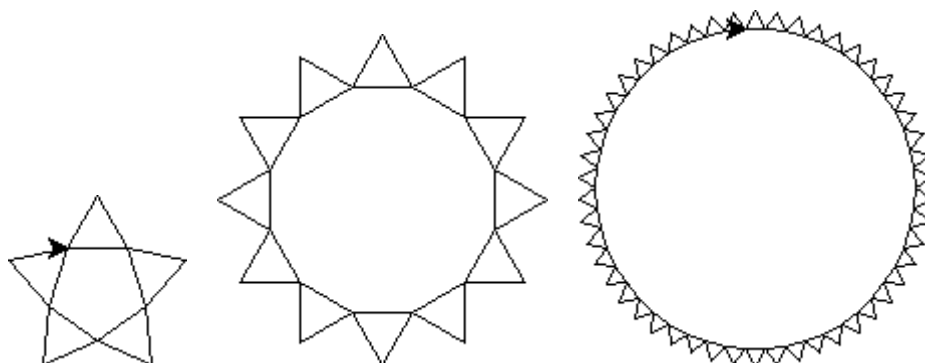
Poradie hodnôt pri volaní funkcie je dôležité:

- `n_uholnik(5, 10)` vykreslí 5-uholník s dĺžkou strany 10,
- `n_uholnik(10, 5)` vykreslí 10-uholník s dĺžkou strany 5.

Parameter funkcie môže byť číselný (napr. dĺžka strany, počet lúčov a pod.) alebo nečíselný (napr. farba objektu).

ĎALŠIE ÚLOHY NA PRECVIČENIE

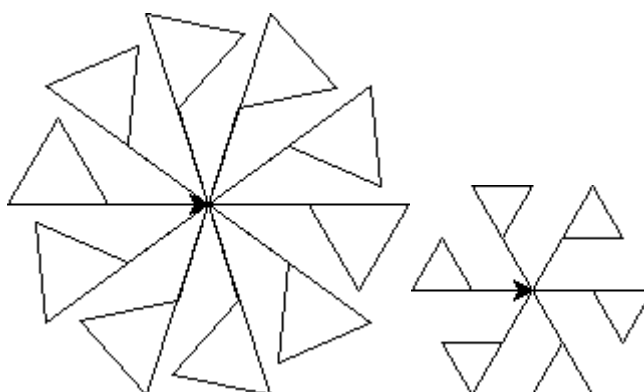
Úloha 9 Vytvorte funkciu `kresli_ozubene_koliesko()` pre vykresľovanie rôznych ozubených koliesok. Riešenie uložte do súboru `ozubene_koliesko.py`.



Poznámka: môžete využiť riešenie úlohy z predchádzajúcej hodiny.

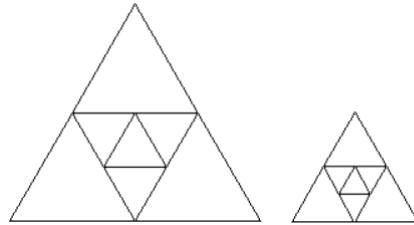
Riešenie:

Úloha 10 Vytvorte funkciu `kresli_vrtulku()` pre kreslenie rôzne veľkých vrtuliek s rôznym počtom listov podľa nasledovného vzoru. Riešenie uložte do súboru `vrtulka.py`.



Riešenie:

Úloha 11 Vytvorte funkciu `kresli_trojuholniky()` pre kreslenie rôzne veľkých trojuholníkových obrázkov nasledovného vzoru. Riešenie uložte do súboru `trojuholniky.py`.



Riešenie:

07 VLASTNÉ FUNKCIE S PARAMETRAMI A VÝSTUPOM – VÝPOČTOVÉ ÚLOHY

ZAPOJENIE

Úloha 1 V reálnom živote využívame množstvo funkcií, ktoré realizujú rôzne výpočty a výsledky ktorých využívame pre plánovanie našich ďalších činností. Napr.:

vstupné parametre		výstupná hodnota
hodinová mzda počet odpracovaných hodín za mesiac daň	→	výška výplaty
veľkosť nádrže auta priemerná spotreba	→	dojazd auta
plocha podlahy výdatnosť plechovky farby cena plechovky farby	→	náklady na vymaľovanie
výška vkladu úroková miera	→	výška budúceho zárobku
dĺžky strán trojuholníka	→	obvod trojuholníka

Diskutujte o funkciách, ktoré sa vyskytujú v našom živote.

Aké sú vstupné hodnoty týchto funkcií?

Čo je výsledkom týchto funkcií?

SKÚMANIE

Úloha 2 Otvorte súbor **kocka.py**, otestujte ho a preskúmajte jeho zdrojový kód.

Čo je výsledkom behu programu kocka.py ?	
V ktorej premennej v hlavnom programe je uložená dĺžka hrany kocky?	
Aká je dĺžka hrany kocky? Zmeňte ju na 50 a spustite program.	
Vo funkcii <code>objem_kocky()</code> nájdite neznámy príkaz: Na čo slúži?	
Nájdite príkaz, ktorým sa vypíše informácia o vypočítanom objeme:	
Vytvorte na vyznačenom mieste v programe funkciu <code>povrch_kocky()</code> a prepíšte si jej definíciu:	<pre>def povrch_kocky(): :</pre>

Doplňte v programe na vhodné miesto aj výpočet a výpis vypočítaného povrchu kocky. Program otestujte a vyskúšajte zmeniť aj veľkosť hrany kocky.	

VYSVETLENIE

Diskusia o nových prvkoch (`return`) v definícií funkcií a o spravovaní návratovej hodnoty funkcie v mieste jej volania.

ROZPRACOVANIE

Úloha 3 Sochár má v úmysle vytvoriť dielo, v ktorom použije niekoľko dutých kociek. Dutú kocku vytvorí tak, že do formy v tvare kocky vloží menšiu kocku a priestor medzi dvoma kockami zaleje betónom. Pomôžte sochárovi vypočítať, koľko m^3 betónu bude potrebovať na dutú kocku a definujte funkciu `objem_dutej_kocky()` na výpočet množstva betónu v dutej kocke.

Pre definovanú funkciu vytvorte dokumentačný reťazec.

Koľko m^3 bude sochár potrebovať, ak väčšia kocka má hranu dĺžky 1,125 m a menšia kocka hranu dĺžky 0,95 m?

Riešenie realizujte v súbore **kocka.py**.

Premyslite si, či a ako sa sa dajú využiť vami definované funkcie.

Riešenie:

Úloha 4 Sochár si pripravil niekoľko foriem pre svoje „kockové“ súsošie, koľko m^3 betónu si musí objednať, ak chce vytvoriť duté kocky nasledovných rozmerov? Riešenie realizujte v súbore **kocka.py**.

	hrana väčšej kocky	hrana menšej kocky
kocka 1	1,125	0,95

kocka2	2,95	2,7
kocka3	0,75	0,5

Riešenie:

Úloha 5 Index telesnej hmotnosti (angl. Body Mass Index – BMI) patrí medzi najviac používané metódy merania obezity. Počíta sa ako hmotnosť v kilogramoch delená druhou mocninou výšky v metroch.

Riešte

podľa Vytvorte funkciu `pocitaj_bmi()` pre výpočet hodnoty indexu BMI. Riešenie uložte do súboru **zdravie.py**.

pokynov

učiteľa

Riešenie:

Úloha 6 Akú hmotnosť má snehuliak?

Riešte

podľa

pokynov

učiteľa

Vytvorte funkciu `hmotnost_snehuliaka()`, ktorá pre zadané priemery troch gúľ, z ktorých je snehuliak postavený, vráti jeho hmotnosť. Riešenie uložte do súboru **snehuliak.py**.

Pomôcka 1: Môžete predpokladať, že 1 m^3 stlačeného snehu v guli má hmotnosť asi 300 kg.

Pomôcka 2: Objem gule V s polomerom r vypočítame podľa vzorca:

$$V = \frac{4}{3}\pi r^3$$

Pomôcka 3: Niektoré výpočty budete robiť opakovane. Premyslite si, pre ktoré časti výpočtu je výhodné definovať samostatné funkcie.

Riešenie:



VYHODNOTENIE

SEBAHODNOTIACI TEST

V programe na prevod hodín na minúty vypadli niektoré časti zdrojového kódu. Doplňte chýbajúce časti tak, aby sme dostali nasledovný výstup: 5 hodín je 300 minút

```
def hodiny_na_minuty(hodiny):  
    minuty =   
    return   
  
cas_v_hodinach = 5  
cas_v_minutach = (cas_v_hodinach)  
print(cas_v_hodinach, 'hodín je', , 'minút')
```

VEDOMOSTI V KOCKE

Vlastné funkcie vieme definovať aj pre výpočty, napr.:

```
def dojazd_auta(objem_nadrze, priemerna_spotreba):  
    ''' Vráti približný dojazd auta s natankovanou plnou nádržou  
  
    :param objem_nadrze: objem nádrže v litroch  
    :type objem_nadrze: float  
    :param priemerna_spotreba: priemerná spotreba v l/100 km  
    :type priemerna_spotreba: float  
    :return: približný dojazd auta v km  
    :rtype: float  
    '''  
    dojazd = objem_nadrze / priemerna_spotreba * 100  
    return dojazd
```

Vypočítanú hodnotu funkcia vráti pomocou príkazu `return`. Ak sa vo funkcii vykoná príkaz `return`, vykonávanie funkcie sa ukončí.

Výslednú hodnotu funkcie by sme na mieste jej volania mali spracovať.

Môžeme ju pomenovať novým menom, použiť v ďalších výpočtoch alebo priamo vypísať.

```
nadrz = 50  
spotreba = 7.3  
  
dojazd = dojazd_auta(nadrz, spotreba)  
print('Auto má približný dojazd', dojazd, 'km')
```

V uvedenom príklade sme návratovú hodnotu funkcie pomenovali `dojazd` a následne vypísali ako súčasť výstupu programu.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 7 V niektorých krajinách sa pri meraní teploty používa Fahrenheitova teplotná stupnica. Medzi teplotou vyjadrenou v °C a v °F platí nasledovný vzťah:

$$\text{teplota_fahrenheit} = \text{teplota_celzius} * 9 / 5 + 32.$$

Definujte funkcie:

`celzius_do_fahrenheit()`, ktorá prevedie zadanú hodnotu teploty zo °C do °F

`fahrenheit_do_celzius()`, ktorá prevedie zadanú hodnotu teploty zo °F do °C



Riešenie uložte do súboru **teplota.py**.

Riešenie:

Úloha 8 Dláždíči, ktorí pokladajú dlažbu si musia vopred vypočítať, koľko kusov dlaždíc budú potrebovať, aby si ich mohli vopred objednať. Pri plánovaní musia zohľadniť nasledovné skutočnosti:

- dlaždice sú štvorcové,
- ak sa z dlaždice časť odreže, táto časť sa už ďalej nepoužije,
- občas nejaká dlaždica pri rezaní praskne, dláždiči preto objednávajú o 5 % dlaždíc viac, ako by potrebovali, keby žiadna dlaždica nepraskla.

a) Vytvorte funkciu `pocet_dlazdic_plocha()`, ktorá pre zadané rozmery dláždenej plochy a veľkosť štvorcovej dlaždice vráti, koľko dlaždíc by mal dláždič objednať.

b) Vytvorte funkciu `pocet_dlazdic_bazen()`, ktorá pre zadané rozmery bazéna a veľkosť štvorcovej dlaždice vráti, koľko dlaždíc by mal dláždič objednať.

Riešenie uložte do súboru **dlazdice.py**.

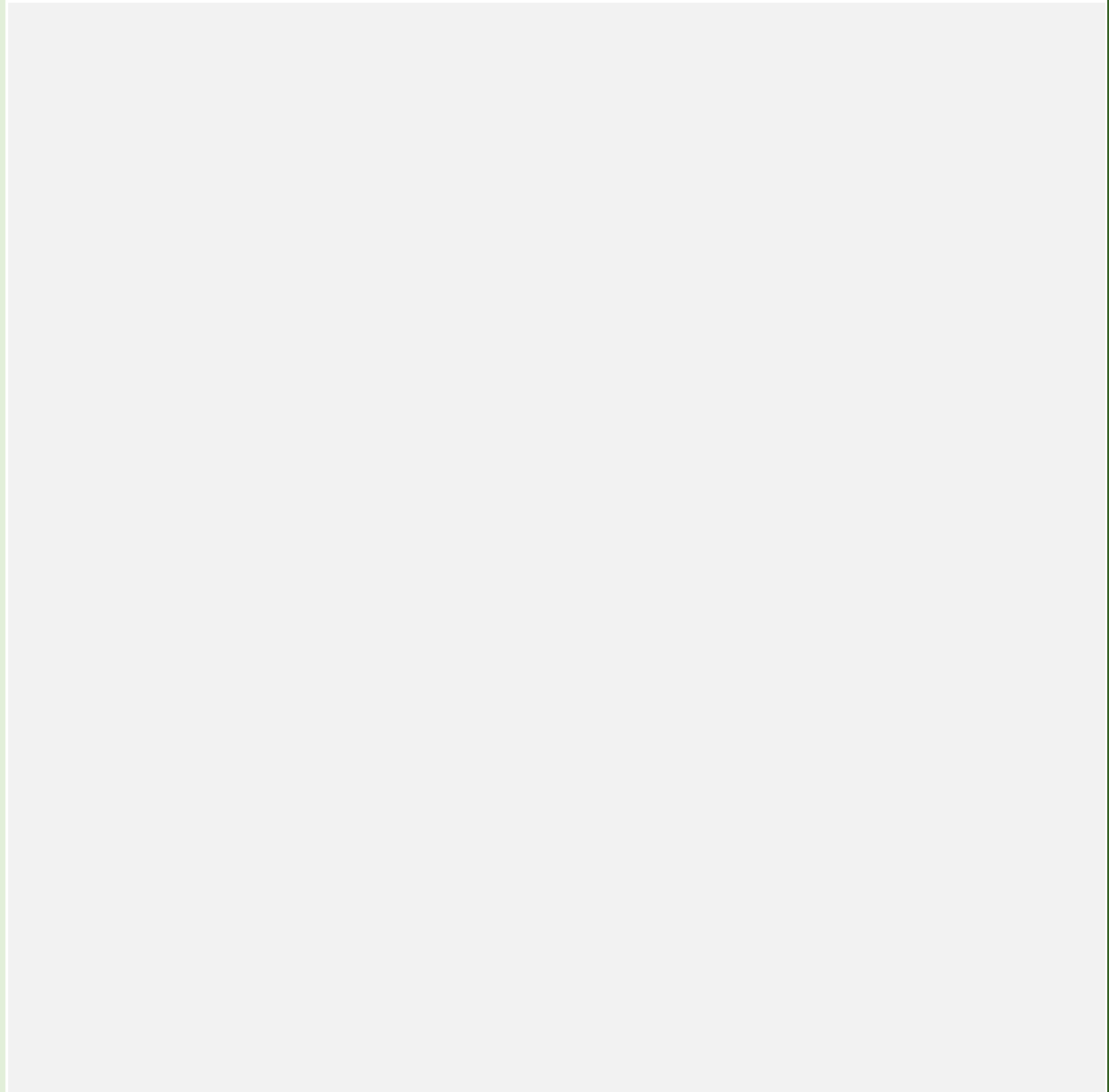
Pomôcka: ak potrebujeme zaokrúhliť číslo smerom hore, využiť môžeme funkciu `ceil()` z modulu `math`. Napr.

```
import math
```

```
cislo = 3.1
```

```
zaokruhlene_hore = math.ceil(cislo)
print(zaokruhlene_hore) # 4
```

Riešenie:



08 CHYBY VO VÝPOČTOCH, ICH ROZPOZNÁVANIE A ODSTRANOVANIE

ZAPOJENIE

Diskutujte na tému chyby pri programovaní. Diskutujte na nasledujúce otázky:

- Robil som chyby počas predchádzajúceho programovania?
- Ako som prišiel na to, že v programe je chyba?
- Hľadal som chybu v programe aj keď to vyzeralo, že program funguje správne?
- Robia chyby aj iní programátori?
- Sú v programoch (textový procesor, webový prehliadač ...), ktoré používame, chyby? Ak áno, prečo a ako sa s tým vysporiadame?

SKÚMANIE

Úloha 1 Žiaci v teste z programovania dostali za úlohu naprogramovať tri funkcie:

- `obsah_kruhu(polomer)`
- `objem_gule(polomer)`
- `povrch_kvadra(a, b, c)`

a pre nejaké konkrétne hodnoty si overiť správnosť riešenia. Karolove riešenie (uvedené v súbore **karol.py**) aj s overením správnosti je nasledovné:

```
def obsah_kruhu(polomer):  
    vysledok = 3.14 * polomer * 2  
    return vysledok  
  
def objem_gule(polomer):  
    vysledok = 4 * 3.14 * polomer * polomer ** 2 / 3  
    return vysledok  
  
def povrch_kvadra(a, b, c):  
    vysledok = 2 * (a * b + b * c + b * a)  
    return vysledok  
  
print(obsah_kruhu(2)) # má byť približne 12.56, výsledok je ok  
print(objem_gule(3)) # má byť približne 113.04, výsledok je ok  
print(povrch_kvadra(20, 10, 10)) # má byť 1000, výsledok je ok
```

Preskúmajte Karolove riešenie. Môže byť Karol spokojný a očakávať dobrú známku? Ak nie, akých chýb sa dopustil a prečo ich neodhalil, keď si overoval správnosť svojho programu? Zdôvodnite svoju odpoveď.

Riešenie:

Úloha 2 Čo robí nasledujúci program? Ak sú v ňom nejaké, pre vás neznáme príkazy, preskúmajte čo robia. Svoje zistenia si zaznamenajte. Program nájdete v súbore **neznamy.py**.

```
meno = input('Ako sa voláš? Napíš svoje meno: ')
hodiny = input('Zadaj, koľko hodín si už venoval programovaniu: ')
hodiny = int(hodiny)

minuty = hodiny * 60

print('Ahoj', meno)
print('Celkovo si programovaniu venoval už', минуты, 'minút.')
```

Riešenie:

VYSVETLENIE

Vysvetlite zistenia a nové príkazy z predchádzajúcich úloh.

ROZPRACOVANIE

Úloha 3 Nasledujúci program by mal pomôcť taxikárom vypočítať sumu, ktorú by mal zákazník zaplatiť. Taxikár má takéto pravidlá pre výpočet výslednej sumy:

- nástupné: 1,5 EUR
- čakanie: 30 centov / minúta
- jazda: 70 centov / km

Navrhňte vhodné testovacie hodnoty a overte správnosť nasledujúceho programu.
Zdôvodnite, prečo ste navrhli práve tieto testovacie hodnoty.
Ak ste v programe našli chyby, opravte ich.

Program je uložený v súbore **taxi.py**.

```
def vysledna_suma(cakanie_min, prejdene_km):  
    nastupne = 1.5  
    cena_cakanie = 0.3 * cakanie_min  
    cena_jazda = 0.7 * prejdene_km  
    suma = nastupne + cena_cakanie + cena_jazda  
    return suma  
  
vzdialenost = input('Zadaj prejdenú vzdialenosť v km: ')  
vzdialenost = int(vzdialenost)  
  
cakanie = input('Zadaj dobu čakania v min: ')  
cakanie = int(cakanie)  
  
zaplatit = vysledna_suma(vzdialenost, cakanie)  
print('Celková suma na zaplataenie:', zaplatit)
```

Riešenie:

Úloha 4 V predchádzajúcich úlohách bolo pomerne jednoduché opraviť chyby, ak sme prišli na to, že programy sú chybné. Niekedy chybu, aj keď o nej vieme, nie je jednoduché lokalizovať len na základe nesprávneho výsledku programu. V týchto prípadoch by nám pomohlo vedieť, ako výpočet prebieha v jednotlivých krokoch programu.

Pre výpočet faktoriálu prirodzeného čísla sme si vytvorili program **vypocet_faktorialu.py**. Otvorte daný program a overte jeho správnosť. Ak je program chybný, postupným krokováním výpočtu nájdite a následne opravte chyby.

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>Vytvorili sme funkciu <code>NSD()</code>, ktorá pre dve zadané prirodzené čísla vráti ich najväčšieho spoločného deliteľa. Ktoré z uvedených hodnôt sú vhodné pre testovanie správnosti funkcie vzhľadom na vzájomné vzťahy medzi zadávanými hodnotami?</p> <p>a) <code>NSD(12, 16)</code> b) <code>NSD(12, 16)</code> c) <code>NSD(12, 16)</code> d) <code>NSD(17, 19)</code> <code>NSD(45, 75)</code> <code>NSD(19, 12)</code> <code>NSD(45, 75)</code> <code>NSD(1, 12)</code> <code>NSD(20, 50)</code> <code>NSD(23, 23)</code> <code>NSD(17, 19)</code> <code>NSD(17, 17)</code> <code>NSD(60, 150)</code> <code>NSD(1, 12)</code> <code>NSD(13, 11)</code> <code>NSD(13, 17)</code></p>
2.	<p>Aký je výstup nasledovného programu, ak používateľ zadá na vstupe hodnoty 5 a 12?</p> <pre> dlzka_dovolenky = input('Zadaj koľko dní budeš na dovolenke: ') dlzka_dovolenky = int(dlzka_dovolenky) rozpocet_den = input('Zadaj rozpočet na jeden deň: ') rozpocet_dovolenka = dlzka_dovolenky * rozpocet_den print('Rozpočet na dovolenku:', rozpocet_dovolenka) </pre> <p>a) Rozpočet na dovolenku: 60 b) Rozpočet na dovolenku: 555555555555 c) Rozpočet na dovolenku: 1212121212 d) Program skončí predčasne s chybou.</p>

VEDOMOSTI V KOCKE

Príkaz `input()` slúži na získanie **vstupu** od používateľa.

Používateľove vstupy sú typu reťazec – `str`. Ak s nimi potrebujeme pracovať ako s číslami, musíme ich **pretypovať** pomocou funkcie `int()` alebo `float()`.

Funkciu `range()` môžeme používať aj s viac ako jedným parametrom:

- `for i in range(5)` – premenná `i` nadobúda hodnoty 0, 1, 2, 3, 4 (t.j. 5 hodnôt od 0 až po $5-1 = 4$)
- `for i in range(2, 5)` – premenná `i` nadobúda hodnoty 2, 3, 4 (t.j. štartovacia hodnota je 2, hraničná hodnota nepatriaca do rozsahu je 5)
- `for i in range(2, 10, 3)` – premenná `i` nadobúda hodnoty 2, 5, 8 (t.j. štartovacia hodnota je 2, hraničná hodnota nepatriaca do rozsahu je 10, krok rozsahu je 3)

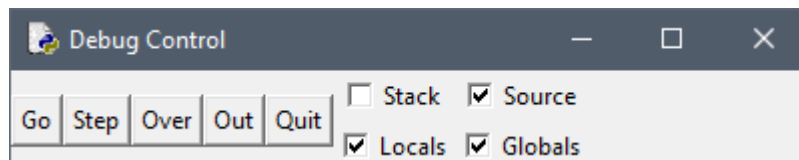
Pre overovanie správnosti programu môžeme program spustiť pre **testovacie hodnoty**. Pri ich návrhu dodržme niekoľko pravidiel:

- hodnoty navrhujeme tak, aby sme vedeli, aký výsledok očakávame a ľahko ho mohli porovnať s výsledkom programu,
- hodnoty navrhujeme tak, aby sme odhalili prípadné zámeny premenných v programe,
- nenavrhujeme také hodnoty, ktoré sa v programoch používajú ako konštanty,
- hodnoty navrhujeme tak, aby sme odhalili chyby, ktorých sme sa pri programovaní dopustili.

Pre postupné krokovanie a sledovanie výpočtu môžeme využiť **ladiace nástroje** dostupné vo vývojových prostrediach. Tieto nástroje nám umožňujú postupne, **krok za krokom** sledovať priebeh výpočtu, zmeny hodnôt premenných, volania funkcií a pod.

V prostredí IDLE spustíme ladiaci režim nasledovne:

- Otvoríme program, ktorý chceme krokovať.
- V okne Shell spustíme Debugger (menu Debug|Debugger). V okne Shell sa objaví výpis „[DEBUG ON]“ a otvorí sa nové okno „Debug Control“. Pred začatím ladenia programu je vhodné všetky tri okná („Debug Control“, „Shell“, editor kódu) usporiadať prehľadne na obrazovke tak, aby sa vzájomne neprekrývali.
- Po spustení programu je možné program krokovať. Slúžia na to tlačidlá v okne „Debug Control“.



Go – Spustí vykonávanie programu. Vykonávanie sa zastaví, ak riadenie narazí na „BreakPoint“ alebo po vykonaní celého programu. Použiť Breakpoint je výhodné najmä vtedy, ak sa chceme sústrediť na vykonávanie konkrétnej časti programu a nechceme strácať čas postupným krokováním predchádzajúcich príkazov. Breakpoint do programu vložíme kliknutím pravým tlačidlom myši na príslušný riadok programu a výberom „Set Breakpoint“ z kontextovej ponuky.

Step – Vykoná sa príkaz v aktuálnom riadku programu. Ak by riadok obsahoval volanie nejakej funkcie, prejde vykonávanie do funkcie.

Over - Vykoná sa príkaz v aktuálnom riadku programu. Ak by riadok obsahoval volanie nejakej funkcie, príkazy v jej tele sa vykonajú na pozadí a vykonávanie pokračuje v nasledujúcom riadku.

Out - Ak sa aktuálny riadok nachádza vo funkcii, zvyšok príkazov funkcie sa vykoná a vykonávanie pokračuje na mieste, odkiaľ bola funkcia volaná.

Quit - Ukončí vykonávanie programu.

Okrem toho je možné zobraziť:

Stack – zásobník, v ktorom je vidieť strom (postupnosť, keď z jednej funkcie voláme ďalšiu) volania funkcií,

Source – zvýrazní práve vykonávaný riadok v programe,


Locals – zobrazí hodnoty lokálnych premenných,


Global – zobrazí hodnoty globálnych premenných.


V prostredí PyCharm spustíme ladiaci režim nasledovne:


- Klikneme do kódu programu pravým tlačidlom myši a z kontextovej ponuky vyberieme „Step through ..“, resp. „More Run/Debug| Step through ..“.
- Následne sa objaví v spodnej časti okna záložka „Debug:“ s ovládacími prvkami.




 Vykoná sa príkaz v aktuálnom riadku programu. Ak by riadok obsahoval volanie nejakej funkcie, príkazy v jej tele sa vykonajú na pozadí a vykonávanie pokračuje v nasledujúcom riadku.

 Vykoná sa príkaz v aktuálnom riadku programu. Ak by riadok obsahoval volanie nejakej funkcie, prejde vykonávanie do funkcie.

 Vykoná sa príkaz v aktuálnom riadku programu. Ak by riadok obsahoval volanie nejakej nami definovanej funkcie, prejde vykonávanie do funkcie. Ak by riadok obsahoval volanie nejakej „cudzej“ funkcie, príkazy v jej tele sa vykonajú na pozadí a vykonávanie pokračuje v nasledujúcom riadku.

 Ak sa aktuálny riadok nachádza vo funkcii, zvyšok príkazov funkcie sa vykoná a vykonávanie pokračuje na mieste, odkiaľ bola funkcia volaná.

 Vykonajú sa všetky zostávajúce príkazy až po pozíciu kurzora v programe.

Niekedy nie je potrebné kontrolovať program krok po kroku, pretože nás zaujíma správanie sa konkrétnej časti programu. Skúmaný riadok môžeme označiť značkou „Breakpoint“. Klikneme myšou do šedej oblasti vľavo od skúmaného riadku (na tomto mieste sa objaví červený krúžok). Ak po spustení ladiaceho režimu klikneme na ikonku . Vykonávanie programu sa zastaví až na riadku označenom značkou „Breakpoint“.

V časti „Variables“ okna „Debug:“ môžeme sledovať hodnoty premenných a ich zmeny počas výpočtu. Hodnoty premenných sa zobrazujú aj šedou farbou priamo v kóde programu.

V časti „Console“ okna „Debug:“ sa realizuje vstup a výstup programu.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 5 Pre obchodníkov sme vytvorili program, ktorý im vypočíta cenu tovaru s DPH aj cenu tovaru bez DPH. Príslušné funkcie sú uložené v súbore **cen.py**.

Otestujte, či sú navrhnuté funkcie správne.

Navrhnite spôsob, ako by sa dala čiastočne otestovať správnosť funkcií, ktoré sú vzájomne inverzné.

Riešenie:

Úloha 6 V obchode predávajú nádrže na dažďovú vodu. Výrobca na štítku uvádza výšku nádoby v metroch a priemer podstavy v centimetroch. Nedopatrením sa na štítok nedostala informácia o objeme.

Vytvorte funkciu `objem_nadrze()` s vhodnými parametrami, ktorá pre zadané údaje zo štítku vráti objem nádrže v litroch.

Navrhnite testovacie hodnoty a overte si správnosť navrhnutej funkcie. Riešenie uložte do súboru **nadrz.py**.

Riešenie:

09 PODMIENENÝ PRÍKAZ

ZAPOJENIE

Úloha 1 Meteorológom zamestnávateľ poskytol pracovné oblečenie: nie veľmi teplý plášť do dažďa a premokavý kabát do chladného počasia. Ak zamestnanec nemá vhodné oblečenie do daného počasia, môže pracovať z domu. Ak počasie nevyžaduje ochranné oblečenie, môže zamestnanec relaxovať v prírode.

Sformulujte jednoduché pravidlá pre meteorológa Karola, aby sa vedel ľahko rozhodnúť, ako bude tráviť deň.

Riešenie:

SKÚMANIE

Úloha 2 Preskúmajte program v súbore **podmienky1.py**. Nájdite v ňom neznáme príkazy a odhadnite ich funkciu.

```
print('Test')
pocet = input('Koľko mesiacov má jeden rok: ')
pocet = int(pocet)

if pocet == 12:
    print('správna odpoveď')
else:
    print('nesprávna odpoveď')

print('Koniec testu')
```

Riešenie:

Úloha 3 Preskúmajte program v súbore **podmienky2.py**. Nájdite v ňom neznáme príkazy a odhadnite ich funkciu.

```
print('Vekové kategórie')
vek = input('Zadaj vek človeka rokoch: ')
vek = int(vek)
```

```
if vek == 0:
    print('Novorodenec alebo dojča')
elif vek <= 3:
    print('Batoľa')
elif vek <= 6:
    print('Predškolský vek')
elif vek <= 12:
    print('Mladší školský vek')
elif vek <= 15:
    print('Mladší školský vek')
elif vek <= 18:
    print('Puberta')
else:
    print('Dospelý')
```

Riešenie:

VYSVETLENIE

Vysvetlite zistenia o nových príkazoch v úlohách 2 a 3.

ROZPRACOVANIE

Úloha 4 V úlohe o vekových kategóriach (súbor **podmienky2.py**) ste skúmali, ako funguje príkaz `if`.

Pracuje program správne, ak zadáme záporný vek alebo vek ako desatinné číslo? Ak nie, upravte program tak, aby akceptoval na vstupe aj desatinné číslo (napr. ak človek má 3,5 roka) a v prípade záporného čísla program výpisom upozornil na chybné zadanú hodnotu.

Riešenie:

Úloha 5 Doplňte do programu **podmienky2.py** aj kategóriu staroba, do ktorej sa človek dostane v 70. roku života.

Riešenie

Úloha 6 V letnom tábore organizujú pravidelné súťažné popoludnia. Aby si deti vyskúšali rôzne športy, rozdeľujú ich podľa rôznych kritérií. Pre dnešné popoludnie sa deti rozdeľujú podľa pohlavia a výšky do jednotlivých športov podľa nasledujúcej tabuľky:

Pohlavie\výška	< 160 cm	<= 170 cm	<= 180 cm	> 180 cm
Chlapec	futbal	vodné pólo		basketbal
Dievča	akvabely	tenis	volejbal	

Vytvorte program **sportove_popoludnie.py**, ktorý pre zadané pohlavie a výšku dieťaťa vypíše, v akom športe bude dieťa súťažiť.

Riešenie:

Úloha 7 Využite programový kód z predchádzajúcej úlohy (rozdelenie detí do jednotlivých športov) a vytvorte funkciu `urci_sport()`, ktorá pre zadané pohlavie a výšku zistí a vráti názov športu, v ktorom bude dieťa súťažiť.

Riešte
podľa
pokynov
učiteľa

Riešenie:

Úloha 8 V pracovnom liste číslo 7 ste v súbore **zdravie.py** vytvorili funkciu `pocitaj_bmi()`, ktorá pre zadanú hmotnosť a výšku človeka vypočítala a vrátila hodnotu BMI. Využite vytvorenú funkciu a vytvorte funkciu `kategoria_bmi()`, ktorá pre zadanú hmotnosť a výšku vráti kategóriu BMI, do ktorej človek patrí. Riešenie uložte do súboru **bmi_kategorie.py**.

Riešte
podľa
pokynov
učiteľa

Pomôcka: Pre jednotlivé kategórie môžete využiť nasledujúce hraničné hodnoty:

<code>bmi < 18,5</code>	<i>podvýživa</i>
<code>18,5 <= bmi < 25</code>	<i>ideálna, zdravá hmotnosť</i>
<code>25 <= bmi < 30</code>	<i>nadváha</i>

```
30 <= bmi < 40      obezita
40 <= bmi            ťažká obezita
```

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>Predajca predáva vianočné stromčeky a cenu stanovil podľa výšky stromčeka. Vytvoril si program, ktorý by mal pre zadanú výšku vypísať cenu stromčeka.</p> <pre>vyska = input('Zadaj výšku stromčeka v cm: ') vyska = int(vyska) if vyska < 150: print('cena: 8 EUR') elif vyska < 180: print('cena: 10 EUR') elif vyska < 200: print('cena: 12 EUR') else: print('cena: 15 EUR')</pre> <p>Aký bude výpis programu, ak zadáme výšku stromčeka 165 cm?</p> <table border="0"><tr><td>a)</td><td>b)</td><td>c)</td><td>d)</td></tr><tr><td>cena: 15 EUR</td><td>cena: 10 EUR</td><td>cena: 10 EUR</td><td>cena: 10 EUR</td></tr><tr><td></td><td></td><td>cena: 12 EUR</td><td>cena: 12 EUR</td></tr><tr><td></td><td></td><td></td><td>cena: 15 EUR</td></tr></table>	a)	b)	c)	d)	cena: 15 EUR	cena: 10 EUR	cena: 10 EUR	cena: 10 EUR			cena: 12 EUR	cena: 12 EUR				cena: 15 EUR
a)	b)	c)	d)														
cena: 15 EUR	cena: 10 EUR	cena: 10 EUR	cena: 10 EUR														
		cena: 12 EUR	cena: 12 EUR														
			cena: 15 EUR														
2.	<p>Čo je výstupom nasledovného programu?</p> <pre>koeficient_a = 5 koeficient_b = 7 if koeficient_a < 3 : if koeficient_b <= 7: print('Kategória A')</pre>																

```

else:
    print('Kategoria A')
else:
    if koeficient_b > 7:
        print('Kategoria C')
    else:
        print('Kategoria D')
print('Koniec')

```

a)

Kategoria A

Kategoria D

Koniec

b)

Kategoria D

Koniec

c)

Kategoria D

d)

Kategoria A

Kategoria D

VEDOMOSTI V KOCKE

Okrem už známych aritmetických operátorov (+, -, *, /, **) existujú aj ďalšie **aritmetické operátory a operátory porovnania**:

% zvyšok po delení, napr. `7.8 % 3 # 1.8`

// celá časť podielu, napr. `7.8 // 3 # 2`

== rovná sa, napr. `2 == 2`

!= nerovná sa, napr. `2 != 3`

< menší, napr. `1 < 2`

<= menší alebo rovný, napr. `1 <= 2`

> väčší, napr. `2 > 1`

>= väčší alebo rovný, napr. `2 >= 1`

Pri programovaní môžeme vytvárať **logické výrazy**. Ich vyhodnotením dostaneme jednu z dvoch logických hodnôt:

- `True` - pravda alebo áno, napr.:

```
1 < 2
3 <= 3 < 5
heslo = 'tajne' # pomenovanie hodnoty
heslo == 'tajne' # test rovnosti hodnoty
```

- `False` - nepravda alebo nie, napr.:

```
1 == 2
3 < 3 < 5
5 != 2 + 3
heslo = 'tajne' # pomenovanie hodnoty
heslo != 'tajne' # test rovnosti hodnoty
```

Logické výrazy môžeme využiť pri vetvení výpočtu do niekoľkých vetiev. Na samotné **podmienené vetvenie** slúži príkaz `if-elif-else`, ktorý môže mať niekoľko variácií, prípadne príkazy môžeme vzájomne vnárať:

```
znamka = 5
if znamka == 5:
    print('žiak neprospel')
print('výpočet pokračuje')
```

žiak neprospel
výpočet pokračuje

```
znamka = 4
if znamka == 5:
    print('žiak neprospel')
else:
    print('žiak prospel')
print('výpočet pokračuje')
```

žiak prospel
výpočet pokračuje

```
znamka = 4
if znamka <= 2:
    print('dobrá známka')
elif znamka <= 4:
    print('zlá známka')
else:
    print('žiak neprospel')
print('výpočet pokračuje')
```

zlá známka
výpočet pokračuje

```
znamka = 4
if znamka == 5:
    print('žiak neprospel')
else:
    print('žiak prospel')
    if znamka <= 2:
        print('dobrá známka')
    elif znamka <= 4:
        print('zlá známka')
print('výpočet pokračuje')
```

žiak prospel
zlá známka
výpočet pokračuje

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 9 John zo Spojených štátov a Kamil zo Slovenska sú dvaja kamaráti, ktorí sa stretli v tábore a vzájomne si vymieňajú kartičky športovcov. Kým John pozná hodnotu svojich kartičiek v amerických dolároch, tak Karol vie ohodnotiť svoje kartičky v eurách. Vytvor pre nich funkciu `zmen()`, ktorá pre zadanú menu z ktorej prevádzame (`usd` alebo `eur`), zadanú menu do ktorej prevádzame (`usd` alebo `eur`) a sumu, vypočíta a vráti hodnotu peňazí v novej mene. Riešenie uložte do súboru `prevod_meny.py`.

Riešenie:

```
def zmen(mena_z, mena_do, suma):
    kurz_usd = 1.2064
    if mena_z == mena_do:
        return suma
    if mena_z == 'usd':
        suma = kurz_usd * suma
    else:
        suma = suma / kurz_usd
    return suma

print(zmen('usd', 'eur', 1))
print(zmen('eur', 'usd', 1.2064))
```

Úloha 10 Upravte riešenie predchádzajúcej úlohy tak, aby v prípade, že niektorá zo zadaných mien nie je `eur` ani `usd`, funkcia vrátila hodnotu `'Chyba! Neplatná mena'`. Riešenie uložte do súboru `prevod_meny_chyba.py`.

Riešenie:

```
def zmen(mena_z, mena_do, suma):
    kurz_usd = 1.2064
    if mena_z != 'usd':
        if mena_z != 'eur':
            return 'Chyba! Neplatná mena'

    if mena_do != 'usd':
        if mena_do != 'eur':
            return 'Chyba! Neplatná mena'

    kurz_usd = 1.2064
    if mena_z == mena_do:
        return suma
    if mena_z == 'usd':
        suma = kurz_usd * suma
    else:
        suma = suma / kurz_usd
    return suma

print(zmen('usd', 'eur', 1))
print(zmen('eur', 'usd', 1.2064))
```

Testovanie, či sme funkcii zadali povolený kód meny, je možné realizovať aj vyhodnotením zloženej podmienky:

```
if mena_z != 'usd' and mena_z != 'eur':  
    return 'Chyba! Neplatná mena'
```

Úloha 11 Vytvorte funkciu `korene_kvadratickej_rovnice()`, ktorá pre zadané reálne koeficienty (a , b , c) kvadratickej rovnice vypočíta a vráti jej korene. Riešenie uložte do súboru `kvadraticka_rovnica.py`.

Riešenie:

```
def korene_kvadratickej_rovnice(a, b, c):  
    diskriminant = b ** 2 - 4 * a * c  
    if diskriminant > 0:  
        x1 = (-b + diskriminant ** 0.5) / (2 * a)  
        x2 = (-b - diskriminant ** 0.5) / (2 * a)  
        return x1, x2  
    elif diskriminant == 0:  
        x = -b / (2 * a)  
        return x  
    else:  
        return
```

Úloha 12 Súradnicové osi rozdeľujú rovinu na štyri kvadranty (označujeme ich I, II, III a IV). Vytvorte funkciu `umiestnenie_bodu()`, ktorá pre zadané súradnice x a y zistí a vráti, v ktorom kvadrante, prípadne na ktorej osi alebo v počiatku súradnicovej sústavy bod leží. Navrhňte vhodné testovacie hodnoty a svoj program otestujte. Riešenie uložte do súboru `kvadranty.py`.

```
def umiestnenie_bodu(x, y):  
    if x > 0:  
        if y > 0:  
            return 'kvadrant I'  
        elif y == 0:  
            return 'os x'  
        else:  
            return 'kvadrant IV'  
    elif x == 0:  
        if y == 0:  
            return 'počiatok súradnicovej sústavy'  
        else:  
            return 'os y'  
    else:  
        if y > 0:  
            return 'kvadrant II'  
        elif y == 0:  
            return 'os x'  
        else:  
            return 'kvadrant III'
```

Testovacie hodnoty by mali byť navrhnuté tak, aby sme vybrali bod z každej uvažovanej časti roviny:

		x		
		-	0	+
y	-	III	os y	IV
	0	os x	stred	os x

	+		os y	
--	---	--	------	--

Úloha 13 Z matematiky vieme, že prvočíslo je každé prirodzené číslo väčšie ako 1, ktoré je deliteľné len jedničkou a sebou samým. Riešenie uložte do súboru **prvocislo.py**.

a) Vytvorte funkciu `je_prvocislo()`, ktorá pre zadané prirodzené číslo zistí a vráti, či číslo je alebo nie je prvočíslo.

b) Upravte definíciu funkcie `je_prvocislo()` tak, aby funkcia vyhodnotila ľubovoľné číslo.

Riešenie:

a)

```
def je_prvocislo(cislo):  
    if cislo < 2:  
        return False  
    for delitel in range(2, cislo):  
        if cislo % delitel == 0:  
            return False  
    return True
```

b)

```
def je_prvocislo(cislo):  
    if cislo < 2:  
        return False  
    if cislo % 1 != 0:  
        return False  
    for delitel in range(2, cislo):  
        if cislo % delitel == 0:  
            return False  
    return True
```


10 OPAKOVANIE II. + DIDAKTICKÝ TEST

RÝCHLA DIAGNOSTIKA VEDOMOSTÍ A ZRUČNOSTÍ

Úloha 1 V inteligentnej budove sa osvetlenie zapína automaticky, ak sa deteguje prítomnosť človeka a intenzita svietenia sa nastaví tak, aby boli splnené minimálne požiadavky na úroveň osvetlenia daného priestoru. Pre najnáročnejšie práce je potrebné osvetlenie aspoň 500 luxov.

Každá miestnosť obsahuje snímač úrovne osvetlenia. Na základe hodnoty zo snímača lampa v prípade potreby rozsvieti niekoľko zo svojich štyroch LED diód. Každá z diód generuje svetlo o intenzite 150 luxov. V súbore **inteligentna_bodova.py**:

- Vytvorte funkciu `pocet_diod()`, ktorá pre zadanú minimálnu požadovanú úroveň osvetlenia vráti, koľko diód je potrebných rozsvietiť. Využite simulátor snímača vo funkcii `intenzita_osvetlenia()` v programe **inteligentna_bodova.py**.
- Navrhňte vhodné testovacie dáta a otestujte svoju funkciu. V prípade potreby upravte časť programu.

```
def intenzita_osvetlenia():  
    import random  
    intenzita = random.randint(0, 600)  
    return intenzita
```

Riešenie:

a)

b)

PRECVIČOVANIE – SAMOSTATNÁ PRÁCA

Úloha 2 Pri objednávke autobusu na školský výlet sú stanovené nasledovné sadzby:

Sadzba 1: 0-50 km 0,50 € / km

Sadzba 2: 51-100 km 0,45 € / km

Sadzba 3: 101 km a viac 0,40 € / km

- Vytvorte program **autobus.py**, ktorý načíta prejdenú vzdialenosť a vypíše cenu za prenájom autobusu. Na výpočet prenájmu definujte funkciu `cena_za_prenajom()`, ktorá pre zadaný počet km vráti cenu prenájmu.
- Upravte funkciu `cena_za_prenajom()` tak, aby v prípade zápornej vzdialenosti funkcia vrátila „Chybne zadaná vzdialenosť“.
- Upravte definíciu funkcie `cena_za_prenajom()` tak, aby sa v prípade sadzby 2 účtovalo navyše štartovné vo výške 2 € a v prípade sadzby 3 štartovné vo výške 6 €.
- Vytvorte funkciu `cennik()`, ktorá vypíše ceny za prenájom postupne od 1 km až po 110 km.

Riešenie:

Úloha 3 Vytvorte funkciu `pocet_delitelov()`, ktorá pre zadané prirodzené číslo zistí a vráti počet jeho deliteľov. Riešenie uložte do súboru **delite.py**.

Riešenie:

Úloha 4 Dopravný podnik stanovil nasledovné ceny časových lístkov:

typ\čas	< 30 min	>= 30 min
obyčajný	0,4 €	0,7 €
zľavnený	0,25 €	0,4 €

- Vytvorte program **dopravny_podnik.py** a definujte funkciu `cena_dopravy()`, ktorá pre zadaný typ lístka a čas dopravy vráti cenu dopravy.
- Upravte funkciu `cena_dopravy()` tak, aby v prípade nesprávne zadaného typu lístka vrátila „Chybne zadaný typ lístka“.
- Upravte funkciu `cena_dopravy()` tak, aby v prípade záporného času vrátila „Chybne zadaný čas“.
- Navrhňte testovacie dáta pre otestovanie správnosti funkcie.

Riešenie:

ZHRNUTIE

Ak ste pri riešení úloh narazili na problém, ktorý ste nevedeli vyriešiť, prekonzultujte ho s učiteľom.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 5 V školskej jedálni nakupujú mliekarenské výrobky z miestnej mliekarne. Mliekareň poskytuje množstevnú zľavu. Prvých 20 kusov sa predáva za plnú cenu. Na zvyšné kusy je zľava 10 %.

Vytvorte program **jedalen_nakup.py**, ktorý si od používateľa vypýta jednotkovú cenu produktu a množstvo objednávaných kusov.

Vytvorte funkciu `cena_nakupu()`, ktorá pre zadanú jednotkovú cenu produktu a množstvo objednávaných kusov vráti celkovú cenu nákupu.

Riešenie:

Úloha 6 V školskej jedálni nakupujú mliekarenské výrobky z miestnej mliekarne. Mliekareň poskytuje množstevnú zľavu. Prvých 20 kusov sa predáva za plnú cenu. Na ďalších 50 kusov poskytuje mliekareň zľavu 10 %. Na zvyšné kusy je zľava 20 %.

Vytvorte (resp. upravte) program **jedalen_nakup.py**, ktorý si od používateľa vypýta jednotkovú cenu produktu a množstvo objednávaných kusov.

Vytvorte funkciu `cena_nakupu2()`, ktorá pre zadanú jednotkovú cenu produktu a množstvo objednávaných kusov vráti celkovú cenu nákupu.

Riešenie:

Úloha 7 *V trojuholníku platí trojuholníková nerovnosť: súčet dĺžok dvoch strán je väčší ako dĺžka tretej strany. Vytvorte funkciu `existuje_trojuholnik()`, ktorá pre zadané dĺžky zistí, či takýto trojuholník existuje. Riešenie uložte do súboru **trojuholnik.py**.*

Riešenie:

Úloha 8 *Pri konštrukcii javiskovej scény sa využívajú kocky zostrojené zo železných tyčí. Kvôli pevnosti sa každá stena ešte spevní uhlopriečkami a celá kocka sa spevní telesovými uhlopriečkami. Kvôli bezpečnosti je potrebné vopred vedieť hmotnosť jednotlivých kociek. Vytvorte funkciu `hmotnost_kocky()`, ktorá pre zadanú dĺžku hrany kocky a hmotnosť 1 m tyče vypočíta a vráti hmotnosť celej kocky. Riešenie uložte do súboru **zelezna_kocka.py**.*

Riešenie:

11 REŤAZCE

ZAPOJENIE

Diskusia o typoch dát

SKÚMANIE

Úloha 1 Otvorte program *retazec.py*.

```
1  meno = 'Juraj'
2  priezvisko = 'Bok'
3  spolu = meno + ' ' + priezvisko
4  print(spolu)
5  print(len(spolu))
6  print(spolu[4])
7  print(spolu[2:5])
8  print(spolu[6:])
9  if 'raj' in spolu:
10     print('áno')
11 else:
12     print('nie')
13 for znak in spolu:
14     print(znak)
```

Program spustíte viackrát. Skúšajte meniť reťazce/znaky, čísla, operácie s reťazcami a sledujte výpisy programu do konzoly. Na základe svojich pokusov vyplňte prázdny stĺpec tabuľky, v ktorom vysvetlíte, čo je výsledkom jednotlivých príkazov.

VYSVETLENIE

Úloha 2 Zuzka sa odsťahovala s celou rodinou do Kanady. So svojou najlepšou kamarátkou Katkou komunikujú písomne a na utajenie svojich správ si dohodli šifru: do textu správy vložia za každý znak ľubovoľný znak. Takto upravená správa vyzerá ako motanica nezmyselných slov, napr. text Ahoj Zuzka! po zašifrovaní vyzerá takto: A*huoXjj QZ8uyzKk+a,!{.

Aby sa im správy ľahšie dešifrovali, obidve vytvorili vlastnú funkciu `desifruj()`, ktorej návratovou hodnotou je dešifrovaná správa. Každá z funkcií však vyzerá odlišne, dievčence sa nevedia dohodnúť, ktorá je správna. Pomôžte im pri rozhodovaní – určte, ktorá z funkcií plní danú úlohu.

Zuzkina funkcia	Katkina funkcia
<pre>def desifruj(s): vysledok = '' for i in range(0, len(s), 2): vysledok = vysledok + s[i] return vysledok</pre>	<pre>def desifruj(s): vysledok = s[::2] return vysledok</pre>

Riešenie:

Úloha 3 Vytvorte program **palindrom.py**, ktorý zistí, či slovo zadané na vstupe je palindróm. Slovo zadávame malými písmenami, nepoužívame medzery a diakritiku.

Poznámka: Palindróm je slovo, veta, číslo (všeobecne akákoľvek postupnosť symbolov), ktorá má tú vlastnosť, že pri čítaní zľava doprava alebo sprava doľava znie rovnako.

Riešenie:

ROZPRACOVANIE

Úloha 4 Tajomstvo komunikácie Zuzky a Katky odhalil Katkin brat Miško. Preto sa dievčatá rozhodli, že budú komunikovať po anglicky a zároveň budú používať šifru Pig Latin – jazykovú hru, ktorá slúži na pobavenie, aj na utajenie komunikácie pred nepovolanými osobami. Princíp hry spočíva v úprave slov podľa týchto pravidiel

- Ak slovo začína spoluhláskou, táto sa presunie na koniec slova a za ňu sa pridá prípona –ay, napr. door => oorday, pen => enpay.
- Ak slovo začína samohláskou, pridá sa len prípona –way, napr. apple =>appleway, old => oldway.

Vytvorte program **pig_latin.py**, ktorý na vstupe dostane slovo (zapísané malými písmenami anglickej abecedy) a do konzoly vypíše toto slovo upravené podľa pravidiel jazykovej hry Pig Latin.

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>Nasledujúci program dešifruje vstupnú správu, ktorá vznikla podľa tohto pravidla šifrovania – pred a za každý znak správy bol vložený jeden náhodný znak (napr. správa „TAJNÉ“ je zašifrovaná v tvare „aTčýA7iJ8žN89Éí“). Doplňte chýbajúcu časť kódu tak, aby bol program funkčný pre ľubovoľnú správne zašifrovanú správu.</p> <pre>zasifrovana_sprava = input('Zašifrovaná správa: ') odsifrovana_sprava = zasifrovana_sprava[<input type="text"/>:<input type="text"/>:<input type="text"/>] print(f'{zasifrovana_sprava} => {odsifrovana_sprava}')</pre>
2.	<p>Aký výstup bude mať nasledujúci program, ak mu na vstupe zadáme slovo 'Zima'?</p> <pre>retazec = input('Vstupný retazec: ') podretazec = retazec[-1] + retazec[::-1] + retazec[0] print(podretazec)</pre> <p>Výstup programu pre vstupné slovo 'Zima' je: '<input type="text"/>'</p>

VEDOMOSTI V KOCKE

Reťazec je iterovateľná štruktúra pozostávajúca zo znakov. Reťazec uvádzame ako postupnosť znakov uzatvorenú v apostrofoch, napr.: `'reťazec'`.

Ak potrebujeme vytvoriť reťazec zložený z iných hodnôt, výrazov alebo reťazcov, vytvoríme ho pomocou formátu f-string. Premenné a výrazy uvedené v zátvorkách {} nahradí Python ich hodnotou.

```
premenna = 10
print(f'Hodnota premennej: {premenna}') # Hodnota premennej: 10
```

Pri práci s reťazcami vieme používať:

- Operáciu +, ktorá umožňuje spojiť (zreťaziť) viaceré reťazce do nového v poradí, v akom ich uvedieme (tento súčet nie je komutatívny).

```
'Ahoj' + ' ' + 'kamarát' # 'Ahoj kamarát'
```

- Relačné operátory <, >, >=, <=, ==, !=, ktoré umožňujú porovnávať reťazce; napr. 'auto' > 'astma', heslo == 'informatika' a pod.
- Operátor príslušnosti in, ktorý umožňuje testovať, či sa reťazec nachádza v inom reťazci ako jeho podreťazec. Reťazec je iterovateľná štruktúra, môžeme teda prechádzať v cykle po znakoch reťazca:

```
retazec = '1. retaz3c z c1slami'
for znak in retazec: # prechádzame znakmi reťazca
    if znak in '0123456789': # testujeme, či znak je číslicou
        print(znak) # vypíšeme číslice 131 z reťazca
```

- Funkciu len(), ktorej návratovou hodnotou je dĺžka reťazca.
- Indexovanie – k jednotlivým znakom daného reťazca môžeme pristupovať pomocou indexov (nekladných alebo záporných), pričom prvý znak má index 0.

Ak zvolíme napr. `retazec = 'Informatika'`, potom platí:

I	n	f	o	r	m	a	t	i	k	a
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- `retazec[4] => r`
 - `retazec[-1] => a`
 - `retazec[20] => chybové hlásenie`
- Výrezy – k podreťazcom daného reťazca môžeme pristupovať pomocou výrezov, resp. výrezov s krokom, pričom formálny zápis tejto operácie má tvar `retazec[zaciatok:konec]`, kde hodnota `zaciatok` určuje prvý znak výrezu, hodnota `konec` určuje posledný znak výrezu – znak s indexom `konec - 1`.
- Ak nevedieme prvý index (`zaciatok`), výrez začne od začiatku reťazca. Ak nevedieme druhý index (`konec`), výrez skončí posledným znakom reťazca. Ak nevedieme ani jeden index, budeme pracovať s celým reťazcom; napr. použijeme reťazec z predchádzajúceho bodu:

- `retazec[:]` => Informatika
- `retazec[5:]` => matika
- `retazec[:4]` => Info
- `retazec[5:10]` => matik
- `retazec[2::3]` => fmi
- `retazec[-4:]` => tika
- `slovo[::2]` => Ifraia
- `slovo[:::-1]` => akitamrofni

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 5 Pani učiteľka v rámci záverečného opakovania pripravuje na každú hodinu slovenského jazyka krátky diktát. Jeho náročnosť posudzuje najmä podľa počtu písmen *i, í, I, Í, y, ý, Y, Ý* v diktáte.

Vytvorte pre pani učiteľku program **diktat.py**, ktorému zadá text diktátu a program spočíta a vypíše počet sledovaných písmen *i, í, I, Í, y, ý, Y, Ý* v zadanom texte.

Napr. pri vstupnom texte „V našej peci myši pištia. Asi nie sú sýte.“ bude odpoveďou číslo 8.

Riešenie:

Úloha 6 Cézarova šifra je historický spôsob šifrovania správy. Cézarova šifra je druh šifry, pri ktorej je každé písmeno správy posunuté o n pozícií ďalej v abecede, pričom n môže byť 1 až $m - 1$, kde m je počet znakov príslušnej abecedy.

Definujte funkciu `cezar_sifruj()`, ktorá zašifruje zadaný text Cézarovou šifrou. Riešenie uložte do súboru **cezar.py**.

Poznámka: Môžete predpokladať to, že na vstupe je reťazec, ktorý obsahuje len abecedné znaky 'a' až 'z'.

Pomôcka: Vieme, že všetky znaky si počítač kóduje pomocou znakovej sady. V nej má každý znak priradený jednoznačný kód, napr.:

znak	a	b	...	y	z
kód <code>ord(znak)</code>	97	98		121	122

Pri práci so znakom môžeme použiť funkciu `ord()`, ktorá nám vráti kód znaku. Podobne existuje funkcia `chr()`, ktorá k zadanému kódu znaku vráti znak so zodpovedajúcim kódom.

Riešenie:

Úloha 7 V programe **cezar.py** definujte funkciu `cezar_desifruj()`, ktorá dešifruje text zašifrovaný Cézarovou šifrou. Navrhnite vhodné testovacie dáta pre overenie správnosti oboch funkcií.

Riešenie:

Pri testovaní by sme mali otestovať situácie:

12 REŤAZCOVÉ METÓDY, ZLOŽENÉ A VNORENÉ PODMIENKY

ZAPOJENIE

Úloha 1 Analyzujte nasledovné problémy a navrhnite postup ich riešenia.

1. Pri analýze vstupných údajov aplikácie pre zadávanie sťažností sa zistilo, že používatelia často omylom stlačia kláves CapsLock. Výsledný text teda obsahuje namiesto malých písmen veľké a namiesto veľkých písmen malé. Navrhnite program, ktorý v zadanom vstupnom reťazci zamení malé písmená za veľké a naopak.
2. Jedným zo vstupných údajov programu je meno používateľa v tvare Meno (prvé písmeno mena je veľké, nasledujúce písmená sú malé, reťazec obsahuje len znaky anglickej abecedy). Navrhnite postup, ktorým by ste overili, či vstupný reťazec spĺňa dané podmienky. Ako by ste vstupný reťazec upravili tak, aby spĺňal prvé dve podmienky?

Má vaše riešenie nejaké úskalia alebo nedostatky?

Uvedte iné problémy, pri ktorých je potrebné manipulovať s reťazcami podľa istých kritérií.

Riešenie:

SKÚMANIE

Úloha 2 Otvorte program `metody1.py`. Program spustte viackrát pre rôzne reťazce obsahujúce rôzne znaky.

```
vstup = 'vOLÁM SA jOŽKO mRKVIČKA, MÁM 15 ROKOV'
```

```
vystup = vstup.swapcase()  
print(f'{vstup} => {vystup}')
```

Čo je výsledkom reťazcovej metódy `reťazec.swapcase()`?

Riešenie:

Úloha 3 Otvorte program **metody2.py**. Program spustíte viackrát pre rôzne reťazce.

```
meno = input('Zadajte svoje meno: ')

if meno.isalpha() and meno[0].isupper():
    print(f'Zadané meno je v poriadku.')
else:
    print(f'Zadané meno nie je v poriadku.')
```

- Pre ktoré reťazce program vypíše vetu 'Zadané meno je v poriadku.'?
- Pre ktoré reťazce program vypíše vetu 'Zadané meno nie je v poriadku.'?
- Čo je výsledkom príkazu `retazec.isalpha()`?
- Čo je výsledkom príkazu `retazec.isupper()`?

VYSVETLENIE

Úloha 4 Otvorte program **pismena.py**. Doplníte tento program tak, aby ku každému znaku vstupného reťazca vypísal veľké písmeno ekvivalentné danému znaku. Vhodnú metódu použijete namiesto zápisu `'...'` v príkaze `print()`. Môžete predpokladať, že na vstupe je reťazec pozostávajúci len z malých a veľkých písmen.

Riešenie:

Úloha 5 S využitím skúseností z predchádzajúcej úlohy riešite nasledujúcu úlohu:

Vytvorte program **uprava.py**, ktorý dostane na vstupe meno používateľa. Podmienkou je, aby vstupný reťazec obsahoval len písmená. Program skontroluje, či zadaný vstup obsahuje len písmená – ak nie, vypíše oznam o chybnom vstupe. Následne upraví vstupný reťazec tak, aby prvé písmeno bolo veľké a všetky ostatné malé.

Riešenie:

ROZPRACOVANIE

Úloha 6 Škola v rámci svojho informačného systému ponúka žiakom vlastný komunikačný kanál. Žiak pri registrácii zadá svoju prezývku a vek. Ak ich zadal správne, získa prístup a môže komunikovať s ostatnými žiakmi školy. Vek je prirodzené číslo v rozsahu od 10 do 20, pre prezývku platia podmienky:

- na začiatku prezývky je podreťazec 'ZIAK',
- za podreťazcom 'ZIAK' nasledujú už len číslice, pričom číselnú časť tvoria aspoň 3, najviac však 8 číslic.

Vytvorte program **registracia.py**, ktorý otestuje vstupné hodnoty podľa uvedených podmienok.

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

<p>1.</p>	<p>Pani učiteľka v rámci záverečného opakovania pripravuje na každú hodinu slovenského jazyka krátky diktát vo forme dopĺňovačky: namiesto i/y napíše znak _ a žiaci vpisujú na tento znak správne písmeno.</p> <p>Vyberte správny kód, ktorý upraví vstupný text diktátu tak, že nahradí všetky výskyty písmen i, í, y, ý, l, í, Y, Ý znakom _, napr. pri vstupnom texte „V našej peci myši pištia. Asi nie sú sýte.“ získame výstupný text „V našej pec_ m_š_ p_št_a. As_n_e sú s_te.“</p> <p>Pomôcka: Viac o metóde <code>replace()</code> sa dozviete v časti „Vedomosti v kočke“.</p> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid gray; padding: 5px; width: 45%;"> <p>Kód 1</p> <pre>text = 'V koryte spí milá myš.' pismena = 'iiyyIIYY' for znak in pismena: text = text.replace(znak, '_') print(text)</pre> </div> <div style="border: 1px solid gray; padding: 5px; width: 45%;"> <p>Kód 2</p> <pre>text = 'V koryte spí milá myš.' pismena = 'iiyyIIYY' for znak in pismena: text.replace(znak, '_') print(text)</pre> </div> </div> <p>Správny je kód číslo _____.</p>
<p>2.</p>	<p>Preštudujte si nasledujúci program:</p> <pre>vstup = input('Zadajte vstupný reťazec: ') if vstup[0] == 'L' and vstup[-3:] == '035': if len(vstup) < 8: print('Výpis1') else: print('Výpis2') else: print('Výpis3')</pre> <p>A. Aký bude výstup nasledujúceho programu, ak na vstupe zadáme reťazec 'Letisko035'? <input style="width: 80px;" type="text"/></p> <p>B. Vytvorte taký vstupný reťazec, aby výstupom programu bol Výpis1. <input style="width: 80px;" type="text"/></p>

VEDOMOSTI V KOCKE

Použitie reťazcových metód nám môže uľahčiť riešenie daného problému. Namiesto vytvárania vlastného algoritmu riešiaceho daný podproblém zvolíme vhodnú metódu.

Dôležité je zapamätať si, že **reťazcová metóda nemení reťazec**, na ktorý ju použijeme – **vytvára jeho upravenú kópiu** (podľa zvolenej metódy) a my túto **kópiu musíme spracovať**. Môžeme ju prepojiť s novou alebo pôvodnou premennou, prípadne ju použiť ako parameter niektorej funkcie:

```
s = 'kaTKa'  
nove = s[0].upper() + s[1].lower()
```

```
s = 'kaTKa'  
print(s[0].upper() + s[1].lower())
```

Pri úpravách reťazcov často využívame podmienky – zložené alebo vnorené.

Zložené podmienky vytvárame pomocou logických operátorov *and*, *or* a *not*:

Logický operátor	Popis
<code>and</code>	Vráti <code>True</code> , ak sú všetky logické výrazy pravdivé.
<code>or</code>	Vráti <code>True</code> , ak je pravdivý aspoň jeden z logických výrazov.
<code>not</code>	Neguje výsledok, vracia <code>False</code> , ak bol výsledok <code>True</code> a naopak.

Častokrát je vhodnejšie použiť namiesto zložených podmienok **vnorené podmienky** (nahrádzame nimi najmä viacnásobné použitie logického operátora `and`). Napr. ak chceme od používateľa získať jeho vek a výšku, budeme pravdepodobne vyžadovať, aby obe čísla boli kladné. Je pre používateľa pohodlnejšie, ak mu oznámime hneď po načítaní jeho veku, že zadal nekorektné číslo, a len v prípade správne zadaného veku budeme žiadať zadanie výšky:

Riešenie úlohy pomocou zložených podmienok	Riešenie úlohy pomocou vnorených podmienok
<pre>vek = input('Vek: ') vek = int(vek) vyska = input('Výška (cm): ') vyska = int(vyska) if vek > 0 and vyska > 0: print('Údaje sú OK.')else: print('Chybný vstup.')</pre>	<pre>vek = input('Vek: ') vek = int(vek) if vek > 0: vyska = input('Výška (cm): ') vyska = int(vyska) if vyska > 0: print('Údaje sú OK.') else: print('Chybná výška.')else: print('Chybný vek.')</pre>
Riešenie je stručnejšie, núti však používateľa zadať obidve vstupné hodnoty, aj keď hneď prvá je chybná. Navyše používateľ nevie, pri ktorom vstupe zadal chybný vstup.	Použitie vnorených podmienok predpokladá, že programátor nad úlohou premýšľal aj z pohľadu používateľa.

Vybrané reťazcové metódy:

`reťazec.capitalize()` Vráti kópiu reťazca, v ktorej je prvý znak prevedený na veľké písmeno a ostatné znaky na malé písmená.

`reťazec.count(podreťazec)` Vráti počet neprekrývajúcich sa výskytov podreťazca v reťazci.

	<p>Napr. <code>'tralala'.count('la') => 2</code> <code>'traLala'.count('la') => 1</code></p>
<code>retazec.find(podretazec)</code>	<p>Vráti najnižší index výskytu podreťazca v reťazci. Ak sa v reťazci podreťazec nevyskytuje, vráti -1.</p> <p>Napr. <code>'traLala'.find('la') => 5</code>.</p>
<code>retazec.index(podretazec)</code>	<p>Podobne ako <code>find()</code>, ale ak sa v reťazci podreťazec nevyskytuje, vráti chybu <code>ValueError</code>.</p>
<code>retazec.isalnum()</code>	<p>Vráti <code>True</code>, ak reťazec obsahuje len alfanumerické znaky (písmená a číslice) a obsahuje aspoň jeden znak. Inak vráti <code>False</code>.</p> <p>Napr. <code>'Horná 15'.isalnum() => False</code></p>
<code>retazec.isalpha()</code>	<p>Vráti <code>True</code>, ak reťazec obsahuje len alfa znaky (písmená) a obsahuje aspoň jeden znak. Inak vráti <code>False</code>.</p> <p>Napr. <code>'B2c35'.isalpha() => False</code> <code>'Horná'.isalpha() => True</code></p>
<code>retazec.isupper()</code>	<p>Vráti <code>True</code>, ak reťazec obsahuje len veľké znaky (písmená) a obsahuje aspoň jeden znak. Inak vráti <code>False</code>.</p> <p>Napr. <code>'B2c35'.isupper() => False</code> <code>'HORNÁ'.isupper() => True</code></p>
<code>retazec.islower()</code>	<p>Vráti <code>True</code>, ak reťazec obsahuje len malé znaky (písmená) a obsahuje aspoň jeden znak. Inak vráti <code>False</code>.</p> <p>Napr. <code>'B2c35'.islower() => False</code> <code>'horná'.islower() => True</code></p>
<code>retazec.isnumeric()</code>	<p>Vráti <code>True</code>, ak reťazec obsahuje len číselné znaky. Inak vráti <code>False</code>.</p> <p>Napr. <code>'12.35'.isnumeric() => False</code> <code>'1235'.isnumeric() => True</code></p>
<code>retazec.upper()</code>	<p>Vráti kópiu reťazca, v ktorej sú všetky alfa znaky prevedené na veľké písmená.</p> <p>Napr. <code>'B2č35'.upper() => 'B2Č35'</code></p>
<code>retazec.replace(čo, čím)</code>	<p>Vráti kópiu reťazca, v ktorej sú všetky výskyty reťazca „čo“ nahradené reťazcom „čím“.</p> <p>Napr. <code>'ja som jana.'.replace('ja', 'Ja') => 'Ja som Jana.'</code></p>
<code>retazec.strip()</code>	<p>Vráti kópiu reťazca, v ktorej sú odstránené všetky biele znaky (medzera, znak konca riadku, tabulátor) zo začiatku a z konca reťazca.</p> <p>Napr. <code>' s1 s2 '.strip() => 's1 s2'</code></p>
<code>retazec.swapcase()</code>	<p>Vráti kópiu reťazca, v ktorej sú všetky výskyty malých písmen nahradené veľkými písmenami a naopak.</p> <p>Napr. <code>'aNNA vESELÁ'.swapcase() => Anna Veselá</code></p>

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 7 Na hodinách matematiky žiaci radi riešia príklady, v ktorých dopĺňajú operátory súčtu, rozdielu, súčinu a podielu medzi dané čísla tak, aby dosiahli daný výsledok, napr. príklad `45_9_1=6` doplnia takto: `45/9+1=6`.

Vytvorte pre pani učiteľku matematiky program **priklady.py**, ktorému na vstupe zadá vyriešený príklad uvedeného typu v tvare `číslo operátor číslo operátor číslo ... = číslo` a program v príklade **nahradí** všetky operátory znakom `_`.

Napr. po zadaní reťazca `45 / 9 + 1 = 6` program vypíše upravený reťazec `45 _ 9 _ 1 = 6`.

Riešenie:

Úloha 8 Vo výskumnom oddelení firmy **Water Locator** vyvíjajú nový typ robota, ktorý vyhľadáva pomocou špeciálnych senzorov zdroj podzemnej vody (a pomáha tak určiť miesto, kde kopať studňu). Žiaľ, nemajú ešte zvládnutý algoritmus pohybu robota po pozemku – pohybuje sa rovnako dlhými krokmi, pričom pred každým krokom si náhodne zvolí jeden zo štyroch smerov: sever (S), juh (J), západ (Z) a východ (V). Kód aktuálneho smeru pohybu ukladá robot do reťazca, napr. reťazec `'SSVSJJZZ'` reprezentuje pohyb sever – sever – východ – sever – juh – juh – západ - západ. Keď robot nájde podzemný vodný zdroj, ukončí zápis do reťazca a vráti sa na svoju štartovaciu pozíciu.

Problémom je, že reťazec reprezentujúci pohyb robota je často krát dlhý a je neefektívne prechádzať celú trasu podľa reťazca. Napr. namiesto pôvodné záznamu trasy `'SSVSJJZZ'` môžeme použiť skrátenejší tvar: `'SZ'`.

Definujte funkciu `skratka()`, ktorá pre záznam pôvodnej trasy vráti najkratšiu cestu so štartovacej pozície k nájdenému podzemnému zdroju. Riešenie uložte do súboru **robot.py**.

Riešenie:



13 ALGORITMY S REŽAZCAMI

MOTIVÁCIA

Diskusia o vlastnostiach programov

EXPOZÍCIA

Diskusia o manipulácii s reťazcami

FIXÁCIA

Úloha 1

Katka sa chystá na študijný pobyt do USA. Naštudovala si, že okrem inej meny a dĺžkových jednotiek používajú obyvatelia USA aj inú jednotku teploty – stupeň Fahrenheita. Zaskočilo ju to, lebo zistila, že prepočet medzi stupňami Fahrenheita a Celzia je dosť náročný.

Definujte funkciu `prevod_teploty()`, ktorá pre zadaný teplotný údaj (reťazec reprezentujúci číslo a jednotku teploty) v jednej stupnici, vráti zodpovedajúci teplotný údaj v druhej stupnici. Riešenie uložte do súboru **teplota.py**.

Vstup funkcie:	Výstup funkcie:
'45 F'	'7.22 C'
'37 C'	'98.60 F'

Pomôcka – Vzorec pre prevod teploty zo stupňov Celzia na stupne Fahrenheita: $F = C * 1,8 + 32$.

Riešenie:

Úloha 2

Peter poprosil svoju sestru Betku, aby prepísala jeho rukou písaný referát na počítači. Keď mu sestra odovzdala prepísaný referát, zistil, že pri písaní mala zamenené klávesy y a z.

Definujte funkciu `vymen_yz()`, ktorá pre zadaný text vráti upravený text, v ktorom budú vzájomne vymenené všetky výskyty znakov 'y' a 'z'. Riešenie uložte do súboru **referat.py**.

Pomôcka: overte si, aké znaky sa do dokumentu vložia, ak sa pokúsite napísať i s mäkčeňom alebo z s dlžňom.

Riešenie:

Úloha 3

Pani učiteľka slovenského jazyka posudzuje náročnosť diktátu podľa počtu písmen i, l, í, í, y, Y, ý a Ý v diktovanom texte. Definujte funkciu `pocet_iy()`, ktorá pre zadaný reťazec vypíše, koľko písmen i, l, í, í, y, Y, ý a Ý reťazec obsahuje. Riešenie uložte do súboru **diktat.py**.

Riešenie:

Úloha 4

Vo vstupnom formulári pre evidenciu nových zamestnancov programátori vytvorili textové pole pre zadanie mena a priezviska zamestnanca. Neskoro si uvedomili, že pre potreby databázy zamestnancov potrebujú samostatne meno a samostatne priezvisko.

Definujte funkcie `meno()` a `priezvisko()`, ktorých vstupným parametrom je reťazec v tvare 'menomedzerapriezvisko'. Funkcia `meno()` nech vráti meno zamestnanca (prvé písmeno veľké, ostatné

malé). Funkcia `priezvisko()` nech vráti priezvisko zamestnanca (všetky písmená veľké). Riešenie uložte do súboru `zamestnanec.py`.

Riešenie:

DIAGNOSTIKA

Diskusia o riešeniach úloh

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 5

Definujte funkciu `prevod_na_m()`, ktorá pre zadaný reťazec reprezentujúci hodnotu a jednotku dĺžky, vráti hodnotu dĺžky v metroch a jednotku *m*. Uvažujte nasledovné jednotky dĺžky: *mm*, *cm*, *dm*, *m*, *km*. Riešenie uložte do súboru `prevod_dlzky.py`.

Vstup	Výstup
'45 cm '	'0.45 m '
'3 dm '	'0.3 m '

Riešenie:

Úloha 6

Pani učiteľka v rámci záverečného opakovania pripravuje na každú hodinu slovenského jazyka krátky diktát. Má k dispozícii množstvo zaujímavých diktátov, potrebuje ich však upraviť tak, aby do vytlačeného textu diktátu žiaci len dopisovali chýbajúce písmená *i*, *í*, *y*, *ý*, *l*, *í*, *Y*, *Ý* (tie budú v texte nahradené znakom `'_'`).

Vytvorte funkciu `vytvor_diktat()`, ktorá pre zadaný text diktátu vráti text s vynechanými písmenami *i/y* a ich variantmi. Riešenie uložte do súboru `opakovanie_diktat.py`.

Vytvorte pre pani učiteľku program `diktat.py`, ktorému zadá text diktátu a program nahradí všetky výskyty písmen *i*, *í*, *y*, *ý* v zadanom texte znakom `'_'`.

Napr. pri vstupnom texte `'V našej peci myši pištia.'`

vypíše `'V našej pec_ m_ š_ p_ št_ a.'`

Riešenie:

Úloha 7

Juraj je básnik, no píše básne v anglickom jazyku. Rozhodol sa, že niektoré z nich pošle do časopisu na zverejnenie. Má ich však napísané rukou kade-tade po papieroch, zošitoch, účtenkách,... Sadol si teda k počítaču a pustil sa do ich prepisovania. Až v polovici práce si všimol, že na jeho klávesnici je stlačený kláves CapsLock, a teda v jeho básňach sú pôvodne malé písmená veľké a pôvodne veľké písmená malé.

Vytvorte funkciu `zmena_velkosti()`, ktorá upraví zadaný text tak, že veľké písmená zmení na malé a naopak.

Riešenie uložte do súboru **zmena.py**.

Riešenie:

14 ODCHYTÁVANIE VÝNIMIEK

ZAPOJENIE

Úloha 1 Diskutujte o chybách vo vašich doterajších programoch.

- S akými chybami ste sa pri spúšťaní programov doteraz stretli?
- Ako na chyby zareagoval Python?
- Je vôbec potrebné „preložiť“ chybové hlásenia programu do zrozumiteľnej reči?

SKÚMANIE

Úloha 2 Nasledujúci program počíta počet fliaš potrebných na uskladnenie kečupu.

Otvorte program **kecup.py** a preštudujte si ho.

```
objem_kecupu = input('Objem kečupu (v Litroch): ')
objem_kecupu = float(objem_kecupu)
objem_flase = input('Objem fľaše (v Litroch): ')
objem_flase = float(objem_flase)
print(f'Počet fliaš: {objem_kecupu // objem_flase}')
```

Spustte viackrát program pre rôzne prirodzené čísla. Čo je výsledkom operácie //?

Pokúste sa zadávať také vstupné hodnoty, ktoré spôsobia zastavenie behu programu alebo nelogický výsledok. Zistené poznatky zapíšte do tabuľky, pokúste sa vysvetliť, čo bolo príčinou správania sa programu.

skúmané hodnoty	moja očakávanie (prečo som zvolil/a práve tieto skúmané hodnoty)	výstup programu (v prípade chybového hlásenia zapíšte len názov chyby)
<code>objem_kecupu =</code> <code>objem_flase =</code>		
<code>objem_kecupu =</code> <code>objem_flase =</code>		
<code>objem_kecupu =</code> <code>objem_flase =</code>		
<code>objem_kecupu =</code> <code>objem_flase =</code>		

Riešenie:

Zastavenie programu:

Nelogický výsledok:

Úloha 3 Predchádzajúci program sme vylepšili. Spustte program `kecup_uprava.py` a otestujte jeho správne fungovanie pre problematické vstupy, ktoré ste objavili v úlohe 2. Následne porovnajte kódy oboch programov.

```
objem_kecupu = input('Objem kečupu (v litroch): ')
objem_flase = input('Objem fľaše (v litroch): ')
try:
    objem_kecupu = float(objem_kecupu)
    objem_flase = float(objem_flase)
    pocet_flias = objem_kecupu // objem_flase
except ValueError:
    print('Na vstupe nebol číselný údaj.')
except ZeroDivisionError:
    print('Objem fľaše nemôže byť nulový.')
else:
    print(f'Počet fliaš: {pocet_flias}')
```

Odpovedzte na nasledujúce otázky:

- a) Pri akých vstupoch sa upravený program správa rovnako ako neupravený?
Odpoveď:
- b) Kedy sa upravený program správa inak ako neupravený? Prečo?
Odpoveď:
- c) Ktorú časť programového kódu umiestňujeme do časti `try`?
Odpoveď:
- d) Vysvetlite, v akých situáciách sa vykonávajú príkazy v častiach `except`.
Odpoveď:
- e) Vysvetlite, v akej situácii sa vykoná blok príkazov v časti `else`.
Odpoveď:

VYSVETLENIE

Čo sú to behové chyby a ako ich vieme programovo ošetriť.

Úloha 4 V záhradkárskej osade „Štvorčekovo“ si záujemca môže prenajať len pozemok v tvare štvorca. Stačí, ak oznámi, na akej ploche chce záhradkáriť a geodet mu v osade určí štvorcový pozemok s požadovanou plochou. K tomu však geodet potrebuje poznať rozmery tohto pozemku. Pokúsil sa vytvoriť program **pozemok.py**, ktorý mu pomôže vypočítať rozmery pozemku s požadovanou plochou. Nedokázal ho však dokončiť tak, aby v prípade nekorektného vstupu zrozumiteľne oznámil túto chybu.

Vstupom tohto programu je plocha pozemku (v metroch štvorcových), výstupom rozmery pozemku (v metroch). Doplňte program **pozemok.py** tak, aby pre korektný vstup vypísal rozmery pozemku, v opačnom prípade vypísal správu, že vstupný údaj nebol správny.

Riešenie:

ROZPRACOVANIE

Úloha 5 Peter je fanúšik adrenalínových zážitkov. Najnovšie ho láka bungee jumping (skok strmhlav z výšky). Našiel ponuky rôznych firiem, líšia sa najmä dĺžkou lana, s ktorým sa skáče (práve od dĺžky lana závisí doba trvania voľného pádu). Peter preto vytvoril program **pad.py**, ktorý na základe hodnoty dĺžky lana vypočíta dobu trvania voľného pádu skokana. Upravte program **pad.py** tak, aby ste odchytili všetky možné výnimky.

Pomôcka: Čas t , za ktorý dopadne teleso z výšky h nad zemským povrchom, získame pomocou vzorca $t = \sqrt{\frac{2 \cdot h}{g}}$.

Samozrejme, Peter na zem nedopadne, zachytí ho lano – Petrov voľný pád trvá, kým mu to dovoľí dĺžka lana.

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.

Program na výpočet obvodu trojuholníka sme ošetrili konštrukciou `try - except`.

```
1 def obvod(a, b, c):
2     return a + b + c
3
4
5 strana_a = input('a = ')
6 strana_b = input('b = ')
7 strana_c = input('c = ')
8 try:
9     print('Program počíta obvod trojuholníka.')
10    print('Zadaj postupne dĺžky strán trojuholníka (a, b, c)')
11    strana_a = float(strana_a)
12    strana_b = float(strana_b)
13    strana_c = float(strana_c)
14    print(f'Obvod trojuholníka je {obvod(strana_a, strana_b, strana_c)}.')
15 except ValueError:
16    print('Na vstupe sa vyskytli nenumerické hodnoty.')
17 except ZeroDivisionError:
18    print('Počas výpočtu došlo k deleniu nulou.')
19 else:
```

Ktoré riadky môžeme presunúť pred konštrukciu `try - except`?

Ktoré riadky presunieme do časti `else` konštrukcie `try - except`?

Sú všetky ošetrenia chýb (v časti `except`) potrebné? Ak nie, ktoré riadky by sme mohli vypustiť?

VEDOMOSTI V KOCKE

Pre korektné riešenie daného problému je dôležité uvedomiť si **podmienky pre vstupné údaje** (formát, rozsah, obmedzenia) a **podmienky pre korektné výpočty**. Vyžaduje to dôkladnú analýzu problému, napr.:

Úloha: Vytvorte program, ktorý pre zadané hodnoty a , b vyčíslí hodnotu výrazu $\frac{\sqrt{a}}{b}$. Čísla a , b sú celé.

Analýza problému: V riešení úlohy použijeme pre výpočet výstupnej hodnoty vzorec $\frac{\sqrt{a}}{b}$, hodnoty a , b musia byť celé čísla. Ku chybám teda môže dôjsť v týchto situáciách:

- na vstupe nebude celé číslo (chyba nastane pri pretypovaní) – používateľ zadá nečíselný vstup alebo nezadá celé číslo,
- vstupná hodnota a bude záporné číslo (chyba nastane pri výpočte odmocniny zo záporného čísla),
- vstupná hodnota b bude 0 (chyba nastane pri výpočte podielu, kde nastane delenie 0)

Po analýze **zapišeme program**, zatiaľ bez odchytenia výnimiek:

```
import math

a = input('a = ')
b = input('b = ')
a = int(a)
b = int(b)
vysledok = math.sqrt(a) / b
print(f'Výsledok výpočtu: {vysledok}')
```

Program doplníme tak, aby sme odchytili možné chyby. Z analýzy daného problému už vieme, **ktoré príkazy sú možným zdrojom chýb**. Tie príkazy umiestnime do časti `try`. V časti `except` uvedieme, čo sa má udiť, ak dôjde pri realizácii príkazov v časti `try` ku chybe. Ak ku chybe nedošlo, v časti `else` realizujeme požadovaný výpočet:

```
import math

a = input('a = ')
b = input('b = ')
try:
    a = int(a)
    b = int(b)
    vysledok = math.sqrt(a) / b
except ValueError:
    print('Na vstupe nebolo celé číslo.')
except ZeroDivisionError:
    print('Počas výpočtu došlo k deleniu nulou.')
else:
    print(f'Výsledok výpočtu: {vysledok}')
```

Poznámka: Pre vstupné hodnoty $a = 3,5$, $b = 0$ bude ako prvá odchytená chyba `ValueError`, vypíše sa príslušný oznam a program skončí. Ak by sme vymenili časti `except`, prvá by bola odchytená chyba `ZeroDivisionError`.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Žiaci sa pripravujú na vianočný koncert. Súčasťou kulís je aj plátno so žltými kruhmi. Žltá farba na textil sa predáva v tubách s objemom 50 ml a farba má výdatnosť 0,5 ml / cm². Žiaci chcú mať predstavu o spotrebe farby pri rôznych polomeroch žltých kruhov. Vytvorte pre tento účel program, ktorému na vstupe zadáme rozsah polomerov kruhov `od` a `do` a ktorý vypíše pre každé prirodzené číslo (polomer) z tohto intervalu obsah žltého kruhu a množstvo farby potrebné na nakreslenie tohto kruhu.

Polomer (cm) Obsah (cm²) Spotreba farby (ml)

`od`

`od + 1`

`...`

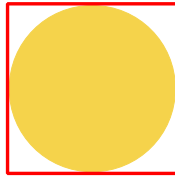
`do - 1`

`do`

Program ošetríte pre možné chybové hlásenia, ktoré môžu spôsobiť jeho ukončenie. Riešenie uložte do súboru **farba1.py**.

Riešenie:

Úloha 7 Žlté kruhy z predchádzajúcej úlohy chcú žiaci orámoviť štvorcom vytvoreným pomocou červenej stuhy:



Stuha sa kruhu dotýka, neprekrýva ho. Vytvorte program ktorý vypočíta, koľko centimetrov stuhy budeme potrebovať na olemovanie jedného kruhu, ak na vstupe zadáme jeho plochu v centimetroch štvorcových.

Program ošetríte pre možné chybové hlásenia, ktoré môžu spôsobiť jeho ukončenie. Riešenie uložte do súboru **farba2.py**.

Riešenie:

)

Úloha 8 Prevádzkovatelia atrakcie bungee jumping sa rozhodli vytvoriť nový cenník. Cena zoskoku bude závislá od dĺžky lana, s ktorým bude zákazník skákať (a teda aj od dĺžky trvania voľného pádu). Stanovili cenu za najkratšiu možnú dĺžku lana a rozhodli sa, že za každý ďalší meter zákazník doplatí konštantnú sumu (v EUR). Minimálna dĺžka lana je prirodzené číslo.

Vytvorte program, ktorému zadáme vstupné hodnoty: minimálna dĺžka lana, cena za zoskok s minimálnym lanom, maximálna dĺžka lana a doplatok za každý ďalší meter lana, a program vypíše cenník v tvare tabuľky. V prvom stĺpci bude dĺžka lana (počínajúc minimálnou dĺžkou a každá ďalšia bude o 1 meter väčšia), v druhom stĺpci cena prislúchajúca cene zoskoku s týmto lanom. Riešenie uložte do súboru **bungeejumping.py**.

Riešenie:

15 GENEROVANIE VÝNIMIEK

ZAPOJENIE

Úloha 1 Diskutujte o chybách v programoch.

- Odhalí Python všetky nekorektné vstupy? Teda, ak program skončil bez chybového hlásenia, môžeme si byť istí, že sme dostali správny výsledok?
- Skúsili ste niekedy „nachytať“ nejakú aplikáciu/program (online kalkulačky na internete, aplikáciu v mobile) tak, že ste experimentovali so vstupnými hodnotami?
- Aké typy chýb sme nedokázali odchytiť pomocou konštrukcie try – except?
- Vo firmách, ktoré vyrábajú počítačové aplikácie, existuje špeciálna pracovná pozícia „tester“. Jeho úlohou je okrem iného objavovanie a testovanie chýb. Stačí, ak tento pracovník len sedí pri počítači a zadáva náhodné dáta?

SKÚMANIE

Úloha 2 Otvorte program **stvorec1.py**.

```
def obsah_stvorca(a):  
    return a ** 2  
  
strana = input('Dĺžka strany štvorca: ')  
  
try:  
    strana = float(strana)  
except ValueError:  
    print('Nečíselná hodnota pre dĺžku strany štvorca.')else:  
    print(f'Obsah štvorca je {obsah_stvorca(strana)}')
```

Program spustíte viackrát. Nájdite také vstupné hodnoty, ktoré spôsobia nezmyselný výsledok.

skúmané vstupné hodnoty	dôvod ich voľby	skutočný výstup programu	aký by mal byť správny výstup programu

Dokážete popísať v akých situáciách program skončil bez chybového hlásenia, ale dal nezmyselný výsledok?

Úloha 3 Otvorte program **stvorec2.py**. Skôr, než program spustíte, zapíšte do stĺpca „predpokladaný výstup programu“ tabuľky výstup, ktorý predpokladáte pre uvedené vstupné hodnoty.

```
def obsah_stvorca(a):  
    try:  
        a = float(a)  
    except ValueError:  
        raise ValueError('Nečíselná hodnota pre dĺžku strany štvorca.')
```



```

if a <= 0:
    raise ValueError('Záporná alebo nulová dĺžka strany štvorca.')
return a ** 2

```

```

strana = input('Dĺžka strany štvorca: ')
try:
    print(f'Obsah štvorca je {obsah_stvorca(strana)}.')
except ValueError as chyba:
    print(chyba)

```

Spustíte tento program viackrát, otestujte hodnoty z tabuľky, do tabuľky doplňte skutočné výstupné hodnoty.

vstupné hodnoty	predpokladaný výstup programu	skutočný výstup programu	výstup	Aký je dôvod rozdielneho výstupu v porovnaní s predchádzajúcim programom?
3				
-2				
xyz				
3,1				

VYSVETLENIE

Úloha 4 Kamil naprogramoval online hru. Časť programu, ktorá umožní hráčom registrovať sa, však ešte nie je dokončená. K registrácii je potrebná prezývka, pod ktorou bude hráč v hre vystupovať. Kamil však chce, aby si hráči volili prezývku, ktoré spĺňajú nasledovné pravidlá:

- prvý znak musí byť abecedný,
- prezývka musí mať minimálne 5 a maximálne 15 znakov,
- prezývka môže obsahovať len alfanumerické znaky.

Ak prezývka spĺňa uvedené podmienky, prezývka sa upraví tak, aby prvý znak bol veľký a ostatné malé.

Definujte funkciu `spracuj_prezývku()`, ktorá zadanú prezývku upraví do požadovaného tvaru. V prípade, ak prezývka nespĺňa požiadavky, funkcia nech vygeneruje zmysluplnú výnimku. Riešenie uložte do súboru **registracia.py**.

Otestujte správnosť funkcie v hlavnom programe.

Riešenie:

ROZPRACOVANIE

Úloha 5 Škola cestovateľov často organizuje pre svojich žiakov výlety do blízkeho aj vzdialenejšieho okolia. Podľa počtu záujemcov o výlet objednávajú potrebný počet autobusov vo firme „Cestujte s nami“.

Táto firma má k dispozícii postačujúci počet autobusov. Škola firme nahlási počet účastníkov výletu, firma pripraví potrebný počet autobusov v závislosti od počtu účastníkov.

Na tento výpočet majú vo firme špeciálny program **autobus.py**, v ktorom však nie sú ošetrené nekorektné vstupy.

Upravte program **autobus.py** tak, aby ste ošetrili všetky chybné vstupy a vygenerovali k nim zodpovedajúce výnimky. Otestujte správnosť riešenia v hlavnom programe

```
import math

def pocet_autobusov(pocet_ucastnikov, kapacita_autobusu):
    autobusy = math.ceil(pocet_ucastnikov / kapacita_autobusu)
    return autobusy

# načítanie vstupných hodnôt: počet účastníkov a kapacita vybraného
autobusu
pocet_ucastnikov = input('Počet účastníkov výletu: ')
pocet_ucastnikov = float(pocet_ucastnikov)
kapacita_autobusu = input('Kapacita autobusu: ')
kapacita_autobusu = float(kapacita_autobusu)

print(f'Počet autobusov: {pocet_autobusov(pocet_ucastnikov,
kapacita_autobusu)}')
```

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1. V predajni má každý druh tovaru pridelený kód. Kód je tvorený aspoň štvormiestnym prirodzeným číslom. Majiteľ predajne sa rozhodol všetky kódy skrátiť tak, že odstráni poslednú cifru kódu. Pre tento účel vytvoril program, ktorý neskôr doplní o zápis nového kódu do súboru. Súbor následne vytlačí na špeciálny etiketovací papier a kódy umiestni na svoj tovar.

```
1 def uprav_kod(vstup):
2     try:
3         vstup = int(vstup)
4     except ValueError:
5         raise ValueError('Chyba1')
6     if vstup < 1000:
7         raise ValueError('Chyba2')
8     return vstup // 10
9
10
11
12 kod = input('Zadaj kód tovaru: ')
13
14 try:
15     novy_kod = uprav_kod(kod)
16 except ValueError as chyba:
17     print(chyba)
18 else:
19     print(f'{kod} => {novy_kod}')
```

Určte výstup tohto programu pre vstupnú hodnotu:

- | | | | |
|----------|-------|----------|-------|
| a. 3 | _____ | d. 68974 | _____ |
| b. Xy.k | _____ | e. 0 | _____ |
| c. -4567 | _____ | f. 3,456 | _____ |

VEDOMOSTI V KOCKE

Doposiaľ sme vedeli odchytiť výnimky, ktoré generuje Python – chyba pri pretypovaní vstupnej hodnoty, delenie nulou, druhá odmocnina zo záporného čísla.

Niekedy však potrebujeme prípustnú vstupnú hodnotu viac špecifikovať, napr. len kladné číslo, len reťazec istej dĺžky, len párne číslo a podobne. Vtedy musíme generovať vlastné výnimky. Použijeme na to príkaz `raise`, ktorý odovzdá hlavnému programu chybové hlásenie v závislosti od chyby, ktorá nastala. Teda, už neoznamujeme hlavnému programu len to, že chyba nastala, ale špecifikujeme, aká chyba nastala.

Najprv analyzujeme vstupné hodnoty, ktoré nie sú prípustné pre riešenie danej úlohy. Následne ich rozdelíme na tie, ktoré vieme odchytiť (generuje ich priamo Python, napr. `ValueError`, `ZeroDivisionError`), a na tie, ktoré vygenerujeme sami (na základe podmienok zadania úlohy).

Samotné odchyťovanie alebo generovanie výnimiek realizujeme v nami definovanej funkcii.

Pri odchyťovaní výnimiek použijeme už známe príkazy `try - except`, avšak časť `except` doplníme o príkaz `raise`, aby sme hlavnému programu poslali konkrétnejší popis chyby.

```
try:
    vykonaj operácie so vstupom
except ValueError:
    raise ValueError(...popis chyby, ktorá nestala...)
```

K takýmto potenciálne problémovým situáciám patrí pretypovanie (ak chceme pracovať s číslami (`float`), prípadne len s celými číslami (`int`), počítame druhú odmocninu alebo vykonávame operáciu podielu a hrozí, že by mohlo dôjsť k deleniu nulou (vtedy napíšeme v časti `except` chybu `ZeroDivisionError`).

Pri generovaní výnimiek použijeme príkaz `if` s vhodne formulovanou podmienkou. Ak táto podmienka platí, príkaz `raise` odošle hlavnému programu popis chyby, ku ktorej pri práci so vstupom došlo.

```
if testovaný výraz:
    raise ValueError(...popis chyby, ktorá nestala...)
```

Ak žiadna z chýb nenastane, funkcia vráti hlavnému programu príkazom `return` informáciu, že je všetko v poriadku (napr. `return True`) alebo vráti očakávaný výsledok.

V hlavnom programe načítame vstupné hodnoty a funkciu voláme v časti `try - except`:

```
try:
    # volanie funkcie s príslušnými parametrami
except ValueError as chyba:
    print(chyba)
else:
    # ďalšie spracovanie korektného vstupu
```

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Rodina Nováková sa rozhodla, že časť svojich úspor vloží do banky AK-NAB. Táto banka ponúka zaujímavý produkt: sumu na účte na konci každého roka zúročí úrokovou mierou, ktorá je najvyššia na súčasnom finančnom trhu. Definujte funkciu `buduca_hodnota()`, ktorá pre zadané hodnoty vráti výšku investície na konci sledovaného obdobia (celé roky). Ak údaje nie sú korektné, nech funkcia vygeneruje zmysluplnú výnimku. Správnosť riešenia si overte v hlavnom programe.

Riešenie uložte do súboru **banka.py**.

Riešenie:

Úloha 7 Jožka na hodine matematiky veľmi zaujala tzv. Fibonacciho postupnosť a jej prítomnosť v prírode a v umení. Rozhodol sa, že ju aplikuje aj do športu – konkrétne do svojho tréningového plánu. Každé ráno bude cvičiť kľuky, pričom každý pondelok zvýši ich počet. Pri zvyšovaní počtu kľukov sa bude riadiť práve Fibonacciho postupnosťou (t.j. 1. týždeň urobí každé ráno jeden kľuk, 2. týždeň tiež jeden kľuk, 3. týždeň dva kľuky, 4. týždeň 3 kľuky, 5. týždeň päť kľukov atď., 6. týždeň osem kľukov, 7. týždeň trinásť kľukov atď.).

Definujte funkciu `pocet_klukov()`, ktorá pre zadané číslo týždňa tréningu vráti počet kľukov, ktoré by mal Jožko každý deň v danom týždni spraviť. Správnosť riešenia (aj pre nekorektné vstupy) si overte v hlavnom programe. Riešenie uložte do súboru **kluky.py**.

Riešenie:

Úloha 8 Jožko videl v televízii reklamu finančnej inštitúcie Zbohatni. Tá ponúka zaujímavú možnosť investovania: Na začiatku prvého mesiaca investujete 1 € a následne bude suma na vašom konte na začiatku každého ďalšieho mesiaca zdvojnásobená. (t.j. po 1. mesiaci budete mať na účte 2 €, po 2. mesiaci 4 €, po 3. mesiaci 8 €, po 4. mesiaci 16 € atď.).

Jožko bol zvedavý, ako rýchlo porastie jeho investícia, vytvoril preto program **investicia.py**, ktorý pomocou funkcie `hodnota_na_konci_mesiac()` pre zadaný počet mesiacov investovania vypíše hodnotu investície na konci tohto obdobia.

- Otestujte, či Jožkov program pracuje správne. Ak nájdete logickú chybu, opravte ju.
- Upravte program tak, aby zobrazil vývoj investície počas celej doby investovania (tú určuje zadaný počet mesiacov investovania) vo forme tabuľky. V prvom stĺpci vypíše poradové číslo mesiaca, v druhom stĺpci aktuálnu hodnotu investície.

```
def hodnota_na_konci_mesiac(mesiac):
    try:
        # overujeme celočíselnosť
        mesiac = int(mesiac)
    except ValueError:
        raise ValueError('Počet mesiacov nie celé je číslo.')
    if mesiac <= 0:
        raise ValueError('Počet mesiacov sporenia musí byť kladný.')
    hodnota = mesiac * 2
    return hodnota

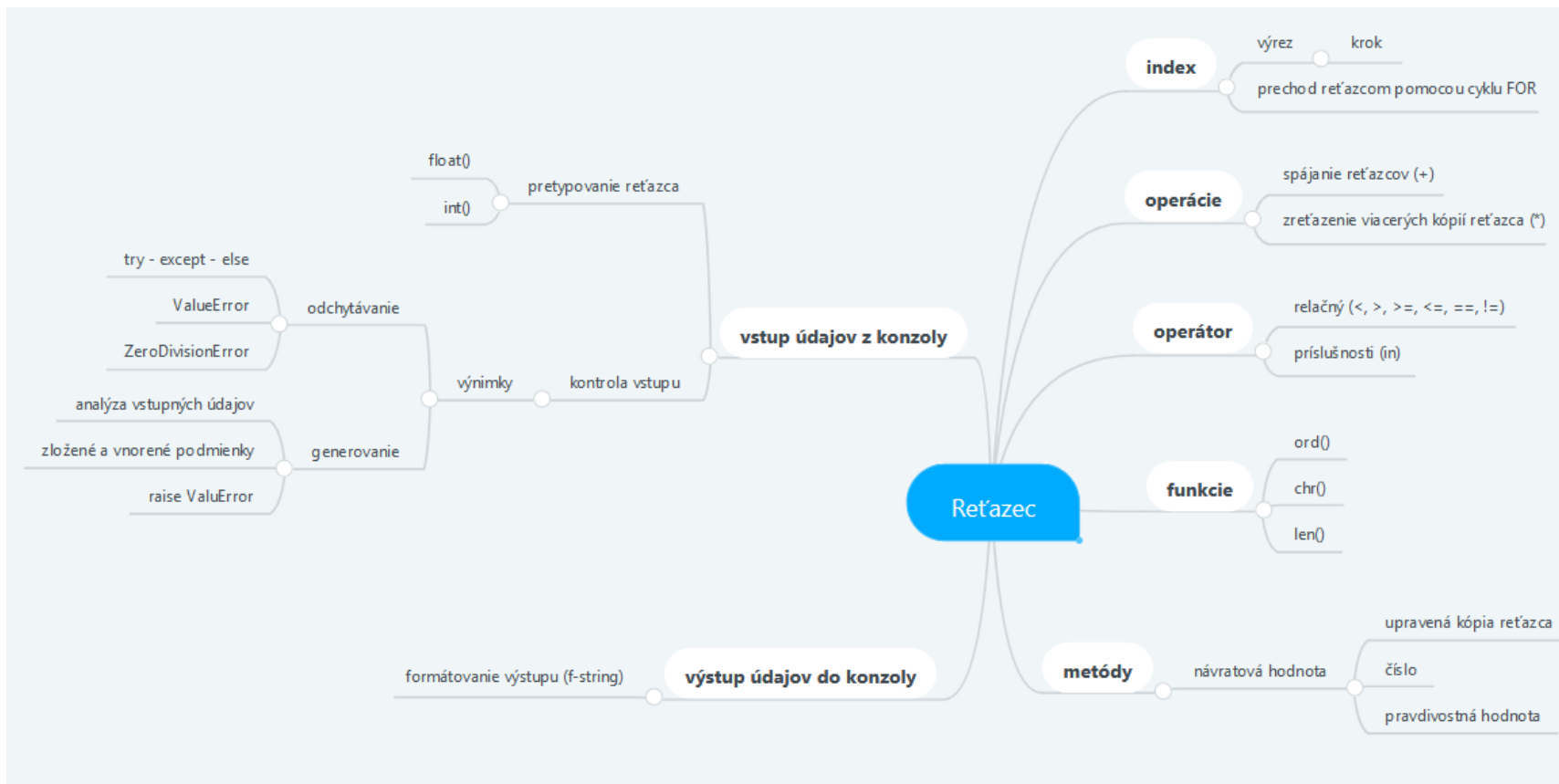
# počet mesiacov investovania
pocet_mesiacov = input('Počet mesiacov: ')
print(f'Na konci {pocet_mesiacov}. mesiaca:
{hodnota_na_konci_mesiac(pocet_mesiacov)} EUR')
```

Riešenie:

16 OPAKOVANIE III. + DIDAKTICKÝ TEST

ÚVOD

Zopakovanie poznatkov o reťazcoch a výnimkách



PRECVIČOVANIE

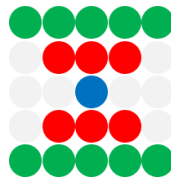
Úloha 1

Členovia školskej firmy sa rozhodli, že ako predmet svojej činnosti budú v škole prevádzkovať automat, ktorý bude pomocou nažehľovacích korálikov vytvárať vzor, ktorý si navolí zákazník. Takto vytvorený vzor si zákazník vezme a doma nažehlí na tričko, tašku alebo čiapku.

Špecifikácia činnosti automatu je takáto:

- k dispozícii sú tri farby nažehľovacích korálikov (červená, zelená a modrá),
- vzor zadaný zákazníkom je v automate reprezentovaný v podobe reťazca, v ktorom majú jednotlivé znaky nasledovný význam:
 - 'r' korálik červenej farby,
 - 'g' korálik zelenej farby,
 - 'b' korálik modrej farby,
 - ' ' prázdne miesto vo vzore,
 - '/' nový riadok vo vzore,
 - 'k' koniec vzoru.

Napríklad reťazec 'ggggg/ rrr/ b/ rrr/ggggk' reprezentuje nasledovný vzor:



- Definujte funkciu `cena_celkom()`, ktorá pre zadaný reťazec reprezentujúci vzor nažehľovacích korálikov vráti výslednú cenu. Za každý korálik červenej farby zaplatí zákazník 1 cent, za zelený korálik 1,5 centa a za modrý korálik 2 centy.
- Definujte funkciu `je_vzor_ok()`, ktorá otestuje zadaný reťazec, či je definovaný podľa pravidiel a vráti zodpovedajúcu logickú hodnotu.
- Upravte funkciu `cena_celkom()`, tak, aby v prípade, že zadaný reťazec je chybný vyhodila funkcia zmysluplnú výnimku.
- Upravte hlavný program tak, aby načítal vzor od používateľa a následne vypísal cenu, resp. chybovú správu.

Riešenie uložte do súboru **koraliky.py**.

Riešenie:

Úloha 2

Časť B
riešte
podľa
pokynov
učiteľa

Určite poznáte mnoho návodov, ako zistiť svoje šťastné číslo. V triede 3.A sa žiaci dohodli, že ho vypočítajú tak, že sčítajú všetky cifry vo svojom rodnom čísle.

Žiaci prišli za správcom informačného systému s prosbou: v systéme sú uložené rodné čísla (v tvare rrrrmmdd/xxxx) každého žiaka školy – mohlo by sa každému žiakovi po prihlásení do systému vypísať jeho šťastné číslo?

Riešenie tejto úlohy realizujte v pripravenom súbore **informacny_system.py**.

- Vytvorte funkciu `stastne_cislo()`, ktorá pre zadané rodné číslo žiaka vráti šťastné číslo žiaka.
- Občas sa stane, že rodné číslo je chybné. Definujte funkciu `je_rc_ok()`, ktorá overí, či rodné číslo je korektné a vráti zodpovedajúcu logickú hodnotu. Rodné číslo je korektné, ak má vyššie uvedený tvar, je deliteľné číslom 11 (po odstránení lomky) a dátumová časť predstavuje korektný dátum (ženám sa k mesiacu pripočíta číslo 50). Pre overenie korektnosti dátumu môžete použiť pripravenú funkciu `je_datum_ok()`.
- Upravte funkciu `stastne_cislo()` tak, aby v prípade chybného rodného čísla vygenerovala zmysluplnú výnimku.
- Upravte hlavný program tak, aby program načítal rodné číslo od používateľa a následne vypísal jeho šťastné číslo.

Riešenie:

ZHRNUTIE

Doplnenie do pojmovej mapy, v ktorej úlohe sa ktoré poznatky využili.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 3 V záhradkárskej osade Ovožel má každý pozemok tvar trojuholníka. Nájomca pozemku platí raz do roka poplatok za nájom pôdy vo výške 0,75 € za každý m^2 plochy svojho pozemku.

Definujte funkciu `vyska_najmu()`, ktorá pre zadané dĺžky strán trojuholníkového pozemku vráti cenu nájmu.

Ošetríte prípadné chybné vstupy, spracujte ich pomocou chybového hlásenia do konzoly.

Riešenie uložte do súboru **ovoze1.py**.

Pomôcka:

- Obsah trojuholníka, ktorý je zadaný dĺžkami (a , b , c) svojich strán sa vypočíta nasledovne:
$$S = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$$
, kde $s = \frac{a+b+c}{2}$
- Premyslite si vhodnú dekompozíciu problému. Pre jednotlivé podproblémy definujte zodpovedajúce funkcie.

Riešenie:

Úloha 4 V súbore **gravitacia.py** je definovaná funkcia `gravitacna_sila()`, ktorá pre zadané hmotnosti dvoch telies a ich vzájomnú vzdialenosť vráti veľkosť gravitačnej sily, ktorou na seba telesá pôsobia.

Na výpočet výslednej sily sme použili vzorec: $F = G \cdot \frac{m_1 \cdot m_2}{R^2}$, kde G je gravitačná konštanta ($6,67 \cdot 10^{-11}$), m_1 a m_2 sú hmotnosti telies v kg a R je vzájomná vzdialenosť telies v metroch.

Skontrolujte správnosť riešenia a upravte program tak, aby program pre korektné vstupy vrátil správne výsledky a pre nekorektné vstupy zobrazil zrozumiteľnú chybovú správu.

```
def gravitacna_sila(m1, m2, r):
    m1 = int(m1)
    m2 = int(m2)
    r = int(r)
    g = 6.67 * 10 ** -11
    sila = g * m1 * m2 / r ** 2
    return sila

hmotnost1 = input('Zadaj hmotnosť 1. telesa (kg): ')
hmotnost2 = input('Zadaj hmotnosť 2. telesa (kg): ')
vzdialenost = input('Zadaj vzájomnú vzdialenosť telies (m): ')

try:
    print(f'Telesá sa navzájom priťahujú silou veľkosti
    {gravitacna_sila(hmotnost1, hmotnost2, vzdialenost)} N.')
except ValueError as chyba:
    print(chyba)
```

Riešenie:

17 ZOZNAMY A METÓDY ZOZNAMOV

ZAPOJENIE

Úloha 1 Čo majú spoločné a čo rozdielne nasledovné typy informácií? Uveďte ďalšie podobné príklady.

Nákupný lístok:	Ranná teplota pacienta (°C):	Trasa výletu:
jogurt, 3	38,9	Parkovisko Kežmarská Biela voda
mlieko, 2	38,9	Nad Matliarmi
maslo, 1	38,6	Šalviový prameň
chlieb, 1	37,8	Kovalčíková poľana
vrecúško jablák, 1	37,8	Veľké Biele pleso
		Dolina Zeleného plesa

Spoločné vlastnosti:

Rozdielne vlastnosti:

Ďalšie podobné príklady:

SKÚMANIE

Úloha 2 V konzolovom režime realizujte nasledovné príkazy. Najskôr odhadnite, čo bude výsledkom príkazu, potom si svoj predpoklad overte.

Pomôcka: všimnite si, že niektoré zápisy sa podobajú na zápisy, ktoré ste používali pri reťazcoch. Táto podobnosť nie je náhodná a môžete ju využiť.

```
znamky = [2, 1, 3, 2, 3, 1, 2]
znamky
```

predpoveď:

skutočnosť:

príkaz	predpoveď:	skutočnosť:
<code>len (znamky)</code>		
<code>znamky [2]</code>		
<code>znamky [-2]</code>		

<code>znamky.count(3)</code>		
<code>znamky.index(2)</code>		
<code>5 in znamky</code>		
<code>sum(znamky)</code>		
<code>for znamka in znamky: print(znamka)</code>		

VYSVETLENIE

Diskusia o dátovom type zoznam, jeho vlastnostiach a manipulácii s nim.

ROZPRACOVANIE

Úloha 3 Riešenie úlohy uložte do súboru **znamky.py**.

Časti
b) až c)
riešte
podľa
pokynov
učiteľa.

- Jeden zo spôsobov ako zistiť výslednú známku na vysvedčení je vypočítať a zaokrúhliť priemer všetkých známok, ktoré žiak za polrok z predmetu získal. Vytvorte funkciu `vysledna_znamka()`, ktorá pre zadaný zoznam známok vypočíta a vráti výslednú známku na vysvedčení.
- Preskúmajte vami navrhnutú funkciu. Otestujte funkciu `vysledna_znamka()`, či pre všetky vstupy pracuje korektne. Pre ktoré vstupy funkcia nepracuje správne?
- Na základe poznatkov z bodu b) upravte funkciu `vysledna_znamka()` tak, aby v prípade nekorektného vstupu funkcia vyhodila relevantnú výnimku.
- Pri hodnotení nemajú všetky známky rovnakú váhu (vplyv na výslednú známku). Posledná známka je známka zo záverečného testu a jej váha je trojnásobná. Upravte funkciu `vysledna_znamka()` tak, aby posledná známka mala trojnásobnú váhu. Môžete predpokladať, že záverečný test písal každý žiak.

Riešenie A:

Riešenie B:

Problematické vstupy:

Riešenie C:

Riešenie D:

Úloha 4 *Riešenie úlohy uložte do súboru **projekty.py**.*

Časť b)
riešte
podľa
pokynov
učiteľa.

- a) V súťaži hodnotenia žiackych projektov hodnotia projekt jednotliví členovia komisie. Každý člen komisie prideli projektu nejaký počet bodov. Celkové hodnotenie projektu sa vypočíta tak, že z hodnotení sa vyhodí jedno najlepšie a jedno najhoršie hodnotenie. Zo zvyšných hodnotení sa vypočíta a zaokrúhli priemer. Vytvorte funkciu `hodnotenie_projektu()`, ktorá pre zadaný zoznam hodnotení projektu členmi komisie vypočíta celkové hodnotenie zaokrúhlené na jednodesatinné miesto..
- b) Počet členov komisie je každý rok iný. Upravte funkciu `hodnotenie_projektu()` tak, aby sa v prípade malého počtu čiastkových hodnotí počítal priemer zo všetkých hodnotení. Ak by sa celkové hodnotenie ani tak nedalo vypočítať, nech funkcia vyhodí výnimku.

Riešenie A:

Riešenie B:

Úloha 5 *Riešenie úlohy uložte do súboru **pacienti.py**.*

Časti
b) a c)
riešte

- a) Sestrička zaznamenáva mená pacientov do zoznamu tak, ako sa príbežne telefonicky objednávajú. Občas sa stane, že pacient zavolá a chce vedieť, kedy sa dostane na rad. Aj keď výpočet nie je

podľa pokynov učiteľa.

- komplikovaný, sestričku to zbytočne zdržiava. Vytvorte funkciu `zisti_cas()`, ktorá pre zadané meno a zoznam pacientov vráti čas (ako reťazec `h:m`), kedy sa pacient dostane na vyšetrenie.
- b) Občas zavolá pacient, ktorý si pomýli deň a v zozname pacientov jeho meno nie je. Upravte funkciu `zisti_cas()` tak, aby v takomto prípade vrátila hodnotu `pacient neobjednaný`.
- c) Sestrička si do zoznamu značí len priezviská pacientov. Občas sa stane, že sa v jeden deň objednávajú menovci. Upravte funkciu `zisti_cas()` tak, aby v takomto prípade vrátila všetky časy, kedy pacienti s týmto menom prídu na rad.

Predpokladajte, že lekár začína ordinovať o 8:00, na každého pacienta si rezervuje 20 min a v čase 13:00 – 13:30 má obedňajšiu prestávku.

V súbore **pacienti.py** je pripravený zoznam pacientov na testovanie.

Riešenie A:

Riešenie B:

Riešenie C:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	Tinka dostala z informatiky nasledovné známky: 1, 2, 1, 3, 2, 2. Zoznam známok Tinky z informatiky vytvoríme nasledovne: Vyberte správne možnosti.
a)	b)

	<code>znamky = [1, 2, 1, 3, 2, 2]</code>	<code>znamky = (1, 2, 1, 3, 2, 2)</code>	
	c)	d)	
	<code>znamky = [1, 1, 2, 2, 2, 3]</code>	<code>znamky = 1, 2, 1, 3, 2, 2</code>	
2.	Čo je výstupom nasledovného programu?		
	<pre>data = [1, 2, 3, 2, 1] for hodnota in data: print(hodnota, data.index(hodnota))</pre>		
	a)	b)	c)
	1 1	1 0	1 0
	2 2	2 1	2 1
	3 3	3 2	3 2
	2 4	2 1	2 3
	1 5	1 0	1 4
			d)
			1 1
			2 2
			3 3
			2 2
			1 1

VEDOMOSTI V KOCKE

Zoznam je dátová štruktúra a definujeme ju ako postupnosť hodnôt uzatvorených v hranatých zátvorkách a oddelených čiarkou, napr.:

```
[11, 21, 22, 12]
['tri', 'dva', 'jeden', 'štart']
```

K prvkom zoznamu prístupujeme cez ich indexy. Index je pozícia prvku v zozname.

```
data = ['tri', 'dva', 'jeden', 'štart']
print(data[1]) # dva
```

Indexy prvkov môžeme číslovať zľava od 0 alebo sprava od -1:

```
index:      0          1          2          3
data = [  'tri'    ,  'dva'    ,  'jeden'  ,  'štart'  ]
index:     -4        -3        -2        -1
```

Zoznam je iterovateľná štruktúra. Cez jeho prvky môžeme prechádzať cyklom, priamo alebo cez indexy.

data = ['tri', 'dva', 'jeden', 'štart']		
<pre>for hodnota in data: print(hodnota)</pre>	<pre>for index in range(len(data)): print(data[index])</pre>	<pre>for index in range(len(data)): print(index, data[index])</pre>
tri	tri	0 tri
dva	dva	1 dva
jeden	jeden	2 jeden
štart	štart	3 štart

Ak nepotrebujeme pracovať s indexom, využime prvú možnosť.

Python ponúka funkcie a operácie, pomocou ktorých vieme zistiť niektoré vlastnosti zoznamu:

<code>len(zoznam)</code>	vráti počet prvkov zoznamu,
<code>max(zoznam)</code>	vráti najväčší prvok zo zoznamu (prvky musia byť vzájomne porovnateľné),
<code>min(zoznam)</code>	vráti najmenší prvok zo zoznamu (prvky musia byť vzájomne porovnateľné),
<code>sum(zoznam)</code>	vráti súčet prvkov zoznamu (prvky musia byť čísla),
<code>hodnota in zoznam</code>	výsledkom je logická hodnota, či sa hodnota nachádza v zozname.

Samotné zoznamy ponúkajú vlastné funkcie:

`zoznam.index(hodnota)` – vráti pozíciu prvého výskytu hodnoty v zozname, ak hodnota v zozname nie je, vyhodí výnimku, funkcia má ďalšie dva varianty:

`zoznam.index(hodnota, start)` – vyhľadáva od indexu `start` až do konca

`zoznam.index(hodnota, start, stop)` – vyhľadáva od indexu `start` po `stop`

`zoznam.count(hodnota)` – vráti počet výskytov hodnoty v zozname

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Systém v banke vždy na konci mesiaca vygeneruje zoznam pohybov na účte. Prvé číslo v zozname je zostatok z predchádzajúceho mesiaca. Ďalšie čísla predstavujú pohyby na účte. Záporné číslo výber, kladné číslo vklad.

Vytvorte nasledovné funkcie (riešenie uložte do súboru **banka.py**):

- `zostatok()` – pre zadaný zoznam pohybov na účte vráti nový zostatok na konci mesiaca,
- `vkklady_celkom()` - pre zadaný zoznam pohybov na účte vráti celkovú sumu vkladov za daný mesiac,
- `vybery_celkom()` - pre zadaný zoznam pohybov na účte vráti celkovú sumu výberov za daný mesiac,
- `nastalo_prečerpanie()` - pre zadaný zoznam pohybov na účte zistí, či sa hodnota na účte počas mesiaca dostala do mínusu.

Riešenie:

Úloha 7 *Riešenie úlohy uložte do súboru **pocasio.py**.*

- Prístroj v meteorologickej stanici zaznamenáva teplotu v °C (zoznam `teplota`) a rýchlosť vetra v m/s (zoznam `rychlost`). Pre lepší prehľad by bolo výhodnejšie mať údaje z jedného času merania zobrazené vedľa seba. Vytvorte funkciu `stav_pocasia()`, ktorá pre zadané zoznamy teplôt a rýchlostí vetra vypíše tieto údaje spárované vedľa seba.*
- Z vlastnej skúsenosti isto viete, že aj keď sa teplota nemení ale zvyšuje sa rýchlosť vetra, človek pocitovo vníma, že je chladnejšie. Túto skutočnosť si uvedomujeme najmä v zime. Tento jav sa odborne nazýva pocitová teplota. Pre výpočet pocitovej teploty sa dá použiť vzorec:
$$T_{poc} = 13,12 + 0,6215 * T - 11,37 * V^{0,16} + 0,3965 * T * V^{0,16}$$
kde T je teplota ovzdušia v °C a V je rýchlosť vetra v km/h*

Upravte funkciu `stav_pocasia()` tak, aby okrem teploty a rýchlosti vetra zobrazila aj pocitovú teplotu.

Riešenie A:

Riešenie B:

Úloha 8 Súboj dvoch hráčov pozostáva z niekoľkých hier. Herný systém postupne zaznamenáva mená hráčov podľa toho, kto v hre vyhral. Súboj končí, ak niektorý z hráčov vyhrá 3 krát za sebou.

V súboji vyhráva ten hráč, ktorý vyhral viac hier.

Vytvorte funkciu `vitaz_suboja()`, ktorá pre zadaný zoznam mien hráčov vráti meno víťaza alebo v prípade remízy text `remíza`.

Riešenie úlohy uložte do súboru **suboj.py**.

Pomôcka: môžete predpokladať, že záznam hier je korektný.

Riešenie:

18 VYTVÁRANIE A MODIFIKÁCIA ZOZNAMOV, POUŽITIE NÁHODY

ZAPOJENIE

Úloha 1 Nižšie sú popísané konkrétne situácie, kde využívame zoznamy. Popíšte činnosti, ktoré by sa mali s uvedenými zoznamami realizovať v konkrétnej situácii.

Nákupný zoznam: zubná niť, mlieko, cibuľa, biely chlieb, mydlo, prací prášok.

Minuli sa nám aj špagety, je potrebné ich kúpiť.

Kúpme namiesto bieleho chleba radšej čierny chlieb, je zdravší.

Mydlo nekupuj, ešte ho máme!

Do tabuľky si kvôli dlhodobému prehľadu značíme všetky nákupy. Kvôli jednoduchšej orientácii sú položky v tabuľke uvedené podľa abecedy. Ako upraviť nákupný zoznam, aby sme položky z neho čo najrýchlejšie zaznamenali do tabuľky.

Susedka nás požiadala, aby sme jej nakúpili suroviny na koláč podľa jej zoznamu.

Aby sme nákup realizovali čo najrýchlejšie, rozdelíme sa!

SKÚMANIE

Úloha 2 Otvorte súbor **zoznamy.py**. V súbore je niekoľko blokov príkazov, ktoré pracujú so zoznamami. Bloky postupne okomentujte a identifikujte v každom bloku nový, pre vás neznámy príkaz. Spustite uvedený kód a poznačte si, názov nového príkazu a jeho funkcionality.

```
nakup = ['zubná niť', 'mlieko', 'cibuľa', 'biely chlieb', 'mydlo', 'prací prášok']  
print(nakup)
```

```
print('Blok 1')  
pozicia = nakup.index('biely chlieb')  
nakup[pozicia] = 'čierny chlieb'  
print(nakup)
```

```
print('Blok 2')  
nakup.append('špagety')  
print(nakup)
```

```
print('Blok 3')  
nakup.sort()  
print(nakup)
```

```
print('Blok 4')
nakup.remove('mydlo')
print(nakup)
```

```
print('Blok 5')
stred = len(nakup) // 2
nakup1 = nakup[:stred]
nakup2 = nakup[stred:]
print(nakup)
print(nakup1)
print(nakup2)
```

```
print('Blok 6')
susedka_nakup = ['múka', 'kvasnice']
nakup.extend(susedka_nakup)
print(nakup)
```

VYSVETLENIE

Diskusia o funkciách zoznamov, možnostiach ich modifikácie a vytváraní nových zoznamov.

ROZPRACOVANIE

Úloha 3 Priemerná denná teplota sa vypočíta na základe troch meraní teploty počas dňa (7:00, 14:00 a 21:00.) nasledovne: $t_{denná} = (t_7 + t_{14} + 2 * t_{21}) / 4$. Meteorologická stanica zaznamenáva teplotu v dané hodiny a ukladá ich do príslušných zoznamov: `teploty7`, `teploty14` a `teploty21`.

- Vytvorte funkciu `denne_teploty()`, ktorá pre zadané zoznamy teplôt z meraní vráti zoznam priemerných denných teplôt za sledované obdobie. Riešenie uložte do súboru **teploty.py**. Predpokladajte, že záznamy teplôt sú rovnako dlhé.
- Upravte funkciu `denne_teploty()` tak, aby v prípade rozdielnych dĺžok vstupných zoznamov funkcia vyhodila zmysluplnú výnimku. Riešenie si overte.

Riešenie:

Úloha 4 Žiaci na hodine matematiky preberali tému štatistika. Janku zaujala problematika náhody a rozhodla sa, že na riešenie domácej úlohy využije počítač.

Zadanie domácej úlohy z matematiky: Urči si postupnosť troch čísiel z intervalu 1 .. 6. Hod' hracou kockou 100 krát. Zaznamenaj si jednotlivé čísla, ktoré na kocke padli. Zisti nasledovné informácie:

- 1 Padli čísla z tvojej trojice ihneď za sebou a v rovnakom poradí?
- 2 Padli čísla z tvojej trojice ihneď za sebou, ale v opačnom poradí?

Vytvorte funkcie, ktoré môže Janka pri riešení domácej úlohy využiť.

- a) Vytvorte funkciu `generuj_hody()`, ktorá vráti zoznam 100 náhodných hodov kockou.
- b) Vytvorte funkciu `padla_trojica()`, ktorá pre zadaný zoznam hodov a určenú trojicu zistí, či táto trojica padla.

Riešenie uložte do súboru **kocky.py**.

Na generovanie hodu môžete využiť modul `random` a funkciu `randrange()`. Funkcia `randrange()` sa používa nasledovne:

```
import random

# vygeneruje náhodné celé číslo z intervalu <1, 6>
cislo = random.randrange(1, 7)
print(cislo)
```

Riešenie:

Úloha 5

Riešte
podľa
pokynov
učiteľa

Na tanečnom krúžku učiteľka tanca páruje dvojice „chlapec, dievča“ podľa toho, v akom poradí žiaci na hodinu tanca prišli. Chlapci sa pri príchode zapisujú do jedného a dievčatá do druhého zoznamu. Vytvorte program `tanec.py` a v ňom funkciu `paruj()` podľa nasledovných požiadaviek:

- Funkcia `paruj()` pre zadané zoznamy chlapcov a dievčat vráti zoznam dvojíc „chlapec, dievča“, ktoré budú spolu tancovať. Predpokladajte rovnaký počet chlapcov a dievčat.
- Upravte funkciu `paruj()` tak, že nebudete predpokladať rovnaké počty chlapcov a dievčat. Žiaci, ktorým sa pár nenájde, nebudú tancovať.
- Upravte funkciu `paruj()` tak, že žiaci, ktorí nemajú pár opačného pohlavia sa budú párovať medzi sebou „chlapec, chlapec“ alebo „dievča, dievča“.
- Upravte funkciu `paruj()` tak, aby sa žiaci do dvojíc vybrali náhodne. Najskôr dvojice rozdielneho pohlavia, potom dvojice rovnakého pohlavia.
- Upravte funkciu `paruj()` tak, aby sa žiaci do dvojíc vybrali v poradí podľa abecedy. Najskôr dvojice rozdielneho pohlavia, potom dvojice rovnakého pohlavia.

Na náhodné preusporiadanie prvkov zoznamu môžete využiť funkciu `shuffle()` v module `random`.

```
import random

zoznam = [1, 2, 3]
# náhodne preusporiadame prvky zoznamu
random.shuffle(zoznam)
print(zoznam)
```

Riešenie:

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	<p>V zozname <code>mena</code> sú uvedené mená žiakov triedy. Zmenu poradia mien tak, aby boli uvedené v opačnom poradí zrealizujeme nasledovne:</p> <table border="0" data-bbox="210 280 1364 443"><tr><td data-bbox="210 280 730 353">a) <code>mena.reverse()</code></td><td data-bbox="849 280 1364 353">b) <code>mena = mena.reverse()</code></td></tr><tr><td data-bbox="210 362 730 443">c) <code>mena[::-1]</code></td><td data-bbox="849 362 1364 443">d) <code>mena = mena[::-1]</code></td></tr></table>	a) <code>mena.reverse()</code>	b) <code>mena = mena.reverse()</code>	c) <code>mena[::-1]</code>	d) <code>mena = mena[::-1]</code>
a) <code>mena.reverse()</code>	b) <code>mena = mena.reverse()</code>				
c) <code>mena[::-1]</code>	d) <code>mena = mena[::-1]</code>				
2.	<p>Aký je výstup nasledovného kódu:</p> <pre data-bbox="210 526 782 593">z = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] print(z[1:7:2])</pre> <div data-bbox="715 622 952 667" style="border: 1px solid black; width: 149px; height: 20px; margin: 0 auto;"></div>				

VEDOMOSTI V KOCKE

Dátová štruktúra zoznam je meniteľná – obsah zoznamu je možné meniť. Na zmenu obsahu zoznamu môžeme využiť **funkcie zoznamu**. Zmena zoznamu sa udeje na mieste. Zmenený zoznam nie je potrebné nanovo pomenovať.

```
zoznam = [1, 2, 3, 2]
```

```
zoznam.append(4)
```

pridá hodnotu na koniec zoznam
zoznam: [1, 2, 3, 2, 4]

```
zoznam.insert(2, 10)
```

vloží pred hodnotu na zadanom indexe zadanú hodnotu
zoznam: [1, 2, 10, 3, 2]

```
zoznam.extend([4, 5, 6])
```

rozšíri zoznam o hodnoty zdaného zoznamu
zoznam: [1, 2, 3, 2, 4, 5, 6]

```
zoznam.reverse()
```

otočí poradie hodnôt v zozname
zoznam: [2, 3, 2, 1]

```
zoznam.remove(2)
```

odstráni prvý výskyt hodnoty zo zoznamu
ak sa hodnota v zozname nenachádza, generuje výnimku
zoznam: [1, 2, 3]

```
zoznam.sort()
```

usporiada prvky zoznamu
zoznam: [1, 2, 2, 3]

```
zoznam.clear()
```

odstráni všetky prvky zo zoznamu
zoznam: []

```
hod = zoznam.pop(2)
```

odstráni a vráti hodnotu na danom indexe (ak index nezadáme, tak posledný),
ak index neexistuje, generuje výnimku
zoznam: [1, 2, 2]
hod: 3

Zo zoznamov vieme vytvárať ďalšie zoznamy pomocou **výrezov**, podobne ako pri reťazcoch. Výrez nenechá pôvodný zoznam. Výsledkom je nový zoznam obsahujúci niektoré z prvkov pôvodného zoznamu.

```
zoznam = [1, 2, 3, 4, 5, 6, 7]
```

```
zoznam[start:stop:krok]
```

vytvorí výrez obsahujúci prvky z pôvodného zoznamu od pozície start, do pozície stop s krokom krok, konkrétne hodnoty môžeme vynechať, štandardne sú nastavené:
start: 0, stop: dĺžka zoznamu, krok: 1

```
vyrez = zoznam[:]
```

vyrez: [1, 2, 3, 4, 5, 6, 7]

```
vyrez = zoznam[2:5]
```

vyrez: [3, 4, 5]

```
vyrez = zoznam[::-1]
```

vyrez: [7, 6, 5, 4, 3, 2, 1]

```
vyrez = zoznam[1:6:2]
```

vyrez: [2, 4, 6]

```
vyrez = zoznam[6::-2]
```

vyrez: [7, 5, 3, 1]

Modul **random** ponúka funkcie pre prácu s **pseudonáhodnými** (~vypočítanými počítačom) **číslami**.

```
import random
```

```
random.randrange(start, stop)
```

pseudonáhodné celé číslo z intervalu <start, stop)

```
random.uniform(start, stop)
```

pseudonáhodné desatinné číslo z intervalu <start, stop)

```
random.choice(zoznam)
```

vráti náhodný prvok zo zoznamu

```
random.shuffle(zoznam)
```

náhodne premieša prvky zoznamu

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Náhodu môžeme využiť na generovanie jednoduchých vtipných viet. Napr. vetu typu „[Kto] [s kým] [čo robil].“ Stačí mať pripravené zoznamy slov, ktoré náhodne poskladáme do vety. Vytvorte funkciu `vytvor_vetu()`, ktorá z pripravených zoznamov slov náhodne zostaví a vráti vetu vyššie uvedeného typu. Riešenie uložte do súboru **vety.py**.

Riešenie:

Úloha 7 V konštruktárskej dielni testujú jednotlivé systémy auta. Aby nemuseli s autom reálne jazdiť, používajú testovacie dáta. Pre test jedného zo systémov potrebujú záznamy o rýchlosti auta merané v pravidelných intervaloch.

Vytvorte funkciu `generuj_rychlosti()`, ktorá vygeneruje zoznam zaznamenaných rýchlostí podľa nasledovných pravidiel:

- Zoznam má zadaný počet (`pocet`) prvkov a prvá hodnota je 0,
- Maximálny rozdiel dvoch po sebe idúcich hodnôt rýchlosti je zadaný (`rozdiel`),
- Pokiaľ rýchlosť nedosiahne interval pre testovaciu rýchlosť (`test_rychlost ± rozdiel`), tak rýchlosť postupne rastie.
- Ak rýchlosť dosiahne interval pre testovaciu rýchlosť, udržiava sa rýchlosť v tomto intervale.

Riešenie uložte do súboru **test_auta.py**.

Riešenie:

19 ALGORITMY SO ZOZNAMAMI

ZAPOJENIE

Úloha 1 Nižšie sú popísané konkrétne situácie, kde využívame zoznamy. Popíšte činnosti, ktoré by sme mali realizovať v konkrétnej situácii.

Situácia:	Potrebné činnosti na vyriešenie situácie:
Zuzka má zoznam slov, pomocou ktorých vytvára krížovky. Zistila však, že omnoho jednoduchšie použije kratšie slová než tie dlhé. Ako by mala postupovať, ak by si chcela vytvoriť len zoznam krátkych slov?	
Predškôlkačka Hanka dostala pri zápise do základnej školy zoznam pomôcok, ktoré bude potrebovať. Jej škôlkarsky spolužiak Jakubko, ktorý bol v tom čase chorý, takýto zoznam nemá. Ako docieľiť, aby aj Jakubko mal zoznam rovnakých pomôcok?	
Andrej si na internetovom e-shope vyhlíadal niekoľko hračiek a ich ceny si poznačil do zoznamu. Až neskôr si všimol, že ceny boli uvedené bez DPH. Ako by mal vytvoriť nový zoznam cien, v ktorom by boli ceny aj s DPH (20 %)?	

SKÚMANIE

Úloha 2 Otvorte súbor **zoznamy.py**. V súbore je niekoľko blokov príkazov, ktoré pracujú so zoznamami. Bloky postupne okomentujte a identifikujte v každom bloku nový, pre vás neznámy príkaz. Spustite uvedený kód a zistite, čo je výsledkom neznámeho príkazu. V popise výsledku identifikujte vzťah medzi novým a pôvodným zoznamom.

```
print('Blok 1')
janka_ulohy = ['umyt riad', 'vyniest smeti', 'utriet prach']
danka_ulohy = [uloha for uloha in janka_ulohy]
print(danka_ulohy)
```

```
print('Blok 2')
hmotnosti_t = [1.1, 2, 5]
hmotnosti_kg = [hmotnost * 1000 for hmotnost in hmotnosti_t]
print(hmotnosti_kg)
```

```
print('Blok 3')
slova = ['aby', 'beh', 'cez', 'les', 'eso', 'dal', 'iba', 'raz']
nove = [slovo for slovo in slova if slovo[0] in 'aeiuoy']
print(nove)
```

```
print('Blok 4')
cisla = [cislo for cislo in range(20) if cislo % 5 != 0]
print(cisla)
```

```
print('Blok 5')
pismena = [znak for znak in 'toto je text' if znak not in 'aeiuoy']
print(pismena)
```

VYSVETLENIE

Diskusia o vytváraní nových zoznamom z iných štruktúr.

ROZPRACOVANIE

Úloha 3 Na jednej z predchádzajúcich hodín ste riešili úlohu, vytvoriť funkciu `generuj_hody()`, ktorá vráti zoznam 100 náhodných hodov kockou. Upravte vytvorenú funkciu tak, aby na vytvorenie zoznamu využila generátorovú notáciu. Pôvodné riešenie je v súbore `kocka.py`.

Porovnajzte obidve riešenia.

Riešenie:

Úloha 4 Škola každý rok organizuje dobročinný ples. Každý účastník plesu si môže kúpiť lístok do tomboly. Lístky sú očíslované od 100 do 999. Tento rok sa škola rozhodla, že výherné lístky bude žrebovať počítač. Vytvorte funkciu `zrebujuj()`, ktorá vráti 10 náhodne vybraných čísiel lístkov. Riešenie uložte do súboru **tombola.py**.

Riešite podľa pokynov učiteľa

Riešenie:

--

Úloha 5 V pracovnom liste č. 9 ste vytvorili funkciu `je_prvocislo()`. Vytvorte funkciu `prvocisla_z_intervalu()`, ktorá pre zadané krajné hodnoty z číselného intervalu vráti zoznam všetkých prvočísiel z tohto intervalu. Časť riešenia je už pripravená v súbore **prvocisla.py**.

Riešenie:

--

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	Doplňte nasledujúci kód tak, aby výstupom bol zoznam [2, 4, 6, 8]
	<pre>zoznam = [1, 2, 3, 4, 5, <input type="text"/>, <input type="text"/>, <input type="text"/>, <input type="text"/>, <input type="text"/>] vystup = [<input type="text"/> for cislo in zoznam if cislo % <input type="text"/> == 0] print(vystup)</pre>
2.	Čo je výstupom nasledujúceho programu?
	<pre>cislo = input('Zadaj prirodzené číslo: ') cislo = int(cislo) vystup = [x ** 2 for x in range(cislo)] print(vystup)</pre>

VEDOMOSTI V KOCKE

Zoznamy môžeme vytvárať využitím prvkov iných sekvencií (zoznam, reťazec, `range()`). Namiesto cyklu, prípadne cyklu s vnorenou podmienkou využijeme skrátenejší zápis, **generátorovú notáciu zoznamu**.

```
[vyraz for hodnota in sekvencia]
```

alebo

```
[vyraz for hodnota in sekvencia if podmienka]
```

Výsledkom generátorovej notácie zoznamu je nový zoznam, pričom obsah použitej sekvencie sa nezmení.

```
zoznam = [1, 2, 3, 4, 5, 6]
```

```
novy_zoznam = [hodnota for hodnota in zoznam]
```

```
novy_zoznam: [1, 2, 3, 4, 5, 6]
```

```
novy_zoznam = [2 * hodnota for hodnota in zoznam]
```

```
novy_zoznam: [2, 4, 6, 8, 10, 12]
```

```
novy_zoznam = [hodnota for hodnota in zoznam if hodnota > 3]
```

```
novy_zoznam: [4, 5, 6]
```

```
novy_zoznam = [hodnota ** 2 for hodnota in zoznam if hodnota > 3]
```

```
novy_zoznam: [16, 25, 36]
```

```
novy_zoznam = [znak.upper() for znak in 'ahoj']
```

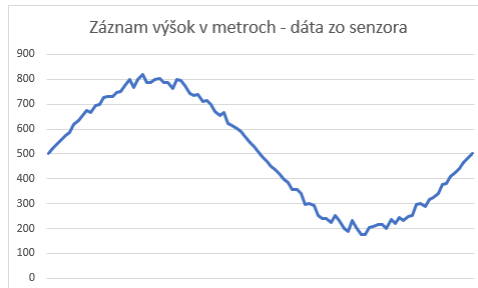
```
novy_zoznam: ['A', 'H', 'O', 'J']
```

```
novy_zoznam = [hodnota for hodnota in range(10)]
```

```
novy_zoznam: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

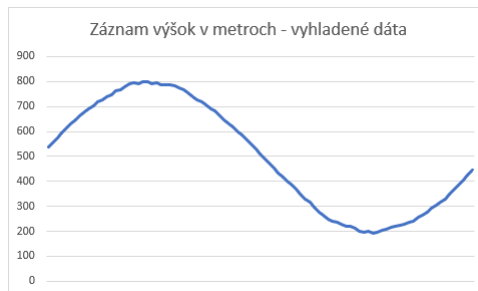

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Senzor výšky zaznamenáva výšku, v ktorej sa nachádza. Meranie je však ovplyvnené rušivými vplyvmi prostredia. Aj keď sa výška senzora menila plynulo, záznam je trochu „roztrasený“. Napr.:



V takýchto prípadoch sa záznam hodnôt vyhladzuje. Vyhladenie spočíva v tom, že určený počet hodnôt idúcich za sebou nahradíme ich priemerom.

Napr.: Ak predchádzajúci záznam vyhladíme tak, že každú päťicu hodnôt nahradíme jej priemerom, dostaneme nasledovný záznam:



Vytvorte funkciu `vyhlad_data()`, ktorá vyhladí dáta v zadanom zozname podľa zadaného počtu hodnôt. Riešenie uložte do súboru **vyhladzovanie.py**.

Poznámka: Vyhladzovanie môžeme v prípade potreby opakovať a dosiahnuť tak ešte lepšie vyhladenie. Musíme si však uvedomiť, že po každom vyhladení sa zmenší počet záznamov.

Riešenie:

Úloha 7 Vernostný systém pre zákazníkov istého obchodu je nastavený tak, že každý mesiac sa vyhodnotia nákupy zákazníka a podľa ich počtu a hodnôt nákupov dostane zákazník kupóny v nejakej hodnote. Systém má nasledovné pravidlá:

- Zákazník má nárok na zľavu, ak v mesiaci uskutočnil aspoň 5 nákupov,
- Za každý nákup od 10 € do 20 € vrátane, dostane zákazník kupón v hodnote 1 % z ceny nákupu,
- Za každý nákup nad 20 € dostane zákazník kupón v hodnote 3 % z ceny nákupu.

Vytvorte funkciu `cena_kuponov()`, ktorá pre zadaný zoznam hodnôt nákupov vráti, v akej cene dostane zákazník kupóny. Riešenie uložte do súboru **obchod.py**.

Riešenie:

Úloha 8 V liečebnom zariadení registrujú pacientov podľa rodných čísiel. Na niektoré procedúry pacientov vyberajú podľa pohlavia. Vytvorte funkciu `pacienti_pohlavie()`, ktorá pre zadaný zoznam rodných čísiel a pohlavie vráti zoznam rodných čísiel pacientov s určeným pohlavím. Riešenie uložte do súboru **liecebna.py**.

Riešenie:

20 CYKLUS S PODMIENKOU

ZAPOJENIE

Úloha 1 Nižšie sú popísané konkrétne situácie, v ktorých sa môžu opakovane vykonávať nejaké činnosti. Popíšte slovne, ako sa rozhodovať, či dané činnosti opakovať. Aký je najmenší a aký najväčší počet opakovaní?

Situácia:	Koľkokrát je potrebné činnosti opakovať? Koľko najmenej a koľko najviac?
Sestrička u lekára volá na vyšetrenie len objednaných pacientov. Pred sebou má zoznam pacientov a pacientov volá podľa tohto zoznamu. Koľkokrát musí sestrička otvoriť dvere a zavolať pacienta z čakárne na vyšetrenie?	
Marienka si na konci leta zamkla bicykel zámkom s číselným kódom. Na jar si však nevedela na číselnú kombináciu spomenúť. Koľko kombinácií bude musieť skúsiť, aby si bicykel odomkla?	
Mamka upiekla buchty a ponúkla ich Kubkovi. Kubko má buchty rád, ale nerád sa prejedá. Koľkokrát si Kubko zoberie a zje buchtu z misky?	

SKÚMANIE

Úloha 2 Otvorte súbor **opakuj.py**. V súbore sú dva bloky príkazov. Bloky postupne odkomentujte a identifikujte v každom bloku nový, pre vás neznámy príkaz. Spúšťajte uvedený kód a popíšte činnosť neznámeho príkazu. V popise výsledku identifikujte vzťah medzi vašim vstupom a výsledkom bloku príkazov.

```
print('Blok 1')
heslo = input('Zadaj heslo: ')
while heslo != 'abrakadabra':
    heslo = input('Zadaj heslo: ')
print('Výborne!')
```

```
print('Blok 2')
pocet = input('Koľko prvočísiel potrebuješ? ')
```

```
pocet = int(pocet)
prvocisla = []
cislo = 1
while len(prvocisla) < pocet:
    if je_prvocislo(cislo):
        prvocisla.append(cislo)
        cislo = cislo + 1
print(prvocisla)
```

VYSVETLENIE

Úloha 3 Vráťte sa k predchádzajúcej ukážke kódu. Čím je zabezpečené, že príkazy sa nebudú opakovať do nekonečna?

Riešenie:

Diskusia príkaze cyklu s podmienkou behu.

ROZPRACOVANIE

Úloha 4 V matematike pri úprave zlomkov na základný tvar hľadáme spoločného deliteľa čitateľa aj menovateľa. Jedným z postupov ako nájsť spoločného deliteľa, dokonca toho najväčšieho, je Euklidov algoritmus. V matematike ho zapíšeme nasledovne:

Časť c)
riešte
podľa
pokynov
učiteľa.

$$NSD(x, y) = x, \Leftrightarrow x = y$$

$$NSD(x - y, y) \Leftrightarrow x > y$$

$$NSD(x, y - x) \Leftrightarrow x < y$$

- Zvoľte si dve prirodzené čísla a a b a overte, či podľa vyššie uvedeného postupu nájdete ich najväčšieho spoločného deliteľa. Ako by ste slovami popísali uvedený algoritmus.
- Vytvorte funkciu `nsd()`, ktorá pre dve zadané prirodzené čísla vráti ich najväčšieho spoločného deliteľa.
- Upravte funkciu `nsd()` tak, aby v prípade, že zadané čísla nie sú prirodzené, funkcia vyhodila výnimku.

Riešenie uložte do súboru **euklides.py**.

Riešenie a):

Riešenie b):

Riešenie c):

Úloha 5 Preskúmajte Euklidov algoritmus a identifikujte situácie, v ktorých tento postup nie je veľmi efektívny. Vedeli by ste sa k výsledku, ktorý dosiahnete postupným odpočítavaním menšieho čísla od väčšieho, dopracovať

Riešte
podľa
pokynov
učiteľa

aj rýchlejšie? Ak áno, implementuje efektívnejší postup do funkcie `nsd_vylepsene()`. Riešenie uložte do súboru **euklides.py**.

Porovnajete čas výpočtu pôvodným a vylepšeným spôsobom. Spustíte každú z funkcií pre rovnakú dvojicu čísiel a porovnajete čas behu.

Riešenie:

Úloha 6 Hru „Hádaj číslo!“ asi pozná každý z nás. Jeden z dvojice si vymyslí číslo a druhý sa ho pokúša na čo najmenší počet pokusov uhádnuť. Hádajúci pri hádaní dostáva odpovede „viac“, „menej“ alebo „uhádol si“.

Časť b)
riešte
podľa
pokynov
učiteľa

- Vytvorte program **hadaj_cislo.py**, kde úlohu toho, kto si vymyslí číslo, bude simulovať počítač. Používateľ bude ten, ktorý sa bude pokúšať, počítačom myslené číslo uhádnuť.
Poznámka: Je rozumné určiť interval pre vymyslené číslo.
- Upravte program tak, aby vymyslené číslo bolo z intervalu 1..16 a hádajúci mal maximálne 4 pokusy. Podarí sa vám na maximálne 4 pokusy číslo vždy uhádnuť? Zdôvodnite!

Riešenie a):

Riešenie b):

VYHODNOTENIE

SEBAHODNOTIACI TEST

1.	Čo bude výstupom nasledujúceho programu? Vyberte jednu z uvedených možností a) až f).
----	---------------------------------------------------------------------------------------

```

odpocet = 3

while odpocet > 1:
    print(odpocet)
    odpocet = odpocet - 1
print('Štart')

```

a)	b)	c)	d)	e)	f)
3	3	3	3	3	3
2	2	Štart	2	2	Štart
	Štart	2	1	1	2
		Štart		Štart	Štart
					1
					Štart

2. Doplňte prázdne miesta v nasledujúcom programe tak, aby výstupom bolo 5 písmen X.

```

pocitadlo = 
print('X')
print('X')
while pocitadlo != 5:
    print('X')
    pocitadlo = pocitadlo + 

```

VEDOMOSTI V KOCKE

Príkaz cyklu `while` použijeme v situáciách, v ktorých potrebujeme opakovane vykonávať nejakú postupnosť príkazov, ale vopred nevieme určiť (ani vypočítať) koľkokrát.

Opakovanie sa riadi platnosťou podmienky behu cyklu za slovom `while`. Platnosť podmienky behu cyklu sa testuje pred každým opakovaním. Ak podmienka behu cyklu neplatí, cyklus skončí. Ak pred prvým opakovaním podmienka behu cyklu neplatí, cyklus sa nevykoná ani raz. Ak potrebujeme dosiahnuť, aby sa uvažovaná postupnosť príkazov vykonala vždy **aspoň 1x**, umiestnime ju aj pred cyklus:

```
while podmienka:  
    postupnost_prikazov
```

Ak sa `postupnost_prikazov` nemusí vykonať **ani raz**.

```
postupnost_prikazov  
while podmienka:  
    postupnost_prikazov
```

Ak sa `postupnost_prikazov` musí vykonať **aspoň raz**.

Pri konštrukcii cyklu `while` musíme zabezpečiť, aby podmienka behu cyklu raz prestala platiť. Inak vytvoríme nekonečný cyklus. Najjednoduchšie to spravíme tak, že vykonávanie príkazov v tele cyklu bude ovplyvňovať platnosť podmienky behu cyklu tak, aby raz podmienka prestala platiť.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 7 Váhavý Kamil má problém s rozhodovaním. Ak sa má rozhodnúť pre niektorú z možností, je to pre neho skoro neriešiteľná úloha. Vymyslel si preto pomôcku, podľa čoho sa rozhodovať. Opakovane hádže mincou.

- Keď nastane situácia, že počet padnutí čísla je o 5 viac ako počet padnutí znaku, vyberie si prvú možnosť.
- Keď nastane situácia, že počet padnutí znaku je o 5 viac ako počet padnutí čísla, vyberie si druhú možnosť.

a) Vytvorte funkciu `rozhodni()`, ktorá bude simulovať hádzanie mincou a vráti, pre ktorú možnosť by sa mal Kamil rozhodnúť. Riešenie uložte do súboru **kamil.py**.

b) Môžeme Kamilov postup zjednodušiť bez vplyvu na výsledok?

Riešenie a):

Riešenie b):

Úloha 8 Slimáky sa šplhajú po tyči smerom hore. Ich cieľom je dosiahnuť vrchol tyče, ktorý sa dotýka skleníka, v ktorom rastie chutná zelenina. Cez deň slimák vylezie niekoľko centimetrov smerom hore. V noci keď spí, sklzáne sa o niekoľko centimetrov smerom dole. Ak má slimák dostatok slizu, podarí sa mu cez deň vyliezť vyššie. Veľa slizu však spôsobí, že v noci sklzáne viac.

- a) Vytvorte funkciu `pocet_dni_na_vrchol()`, ktorá bude simulovať pohyb slimáka po tyči a vráti, na ktorý deň slimák dosiahne vrchol. Predpokladajte, že slimák vylezie každý deň rovnakú vzdialenosť. V noci sklzáne od 10 % do 90 % z toho, čo cez deň vylezie. Riešenie uložte do súboru **slimak.py**. Rozmyslite si, aké parametre by funkcia mala mať.
- b) Upravte funkciu `pocet_dni_na_vrchol()` tak, aby funkcia dostala maximálnu dennú vzdialenosť o ktorú môže slimák vyliezť. V skutočnosti slimák denne vylezie o vzdialenosť 0 až zadané maximum.
- c) Upravte funkciu `pocet_dni_na_vrchol()` tak, aby nočné sklzánutie slimáka bolo od 10 % do 90 % z maximálnej dennej vzdialenosti.

Riešenie



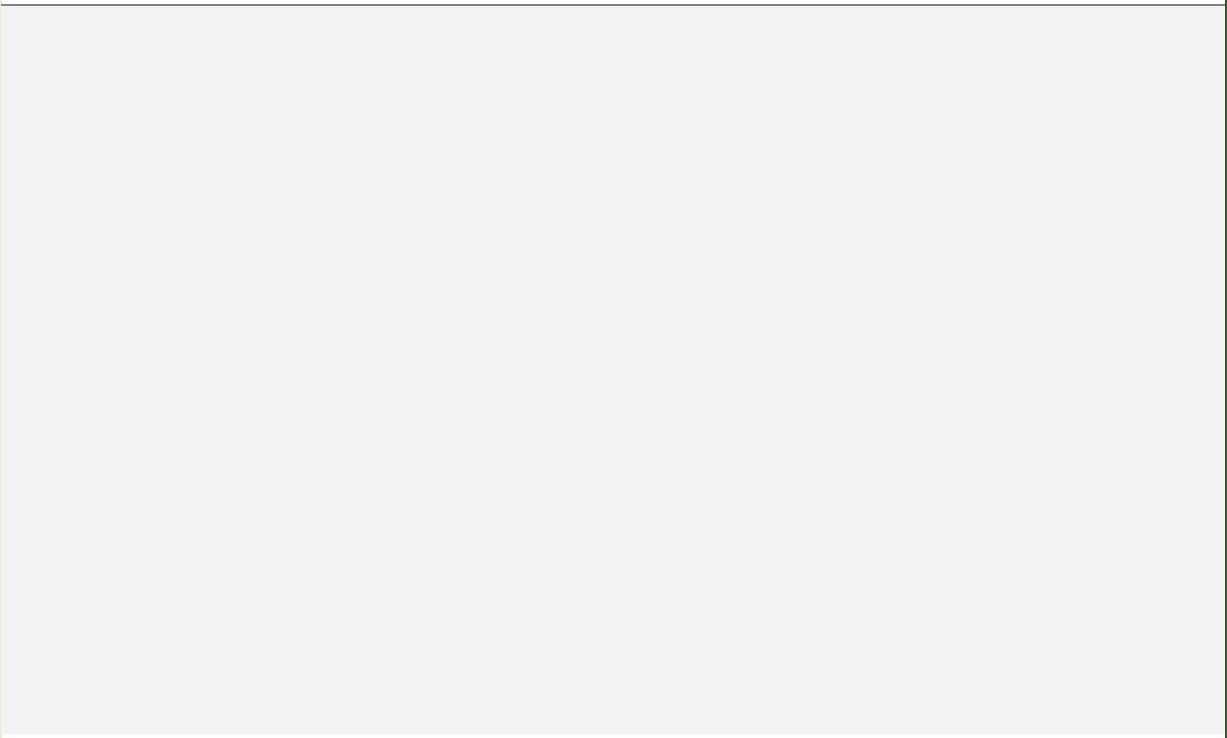
Úloha 9 Automatický žeriav je schopný prekladať tovary z nakladacej rampe do prepravného vozíka, ktorý má určenú nosnosť. Obslužný systém žeriava má k dispozícii informácie o hmotnostiach tovaroch na rampe. Inžinieri navrhli niekoľko stratégií, ako vozíky plniť:

- a) Tovary sa budú nakladať v takom poradí v akom sú umiestnené na rampe. Ak by sa naložením tovaru, ktorý je v poradí, prekročila nosnosť vozíka, nakládka vozíka končí.
- b) Tovary sa budú nakladať v takom poradí v akom sú umiestnené na rampe. Ak by sa naložením tovaru ktorý je v poradí prekročila nosnosť vozíka, žeriav tento tovar vynechá a pokračuje nasledujúcim tovarom na rampe.
- c) Tovary sa budú nakladať na striedačku. Najskôr sa naloží najťažší tovar, potom najľahší a toto sa opakuje až dovtedy, keď už nie je žiaden tovar, ktorý by sa dal naložiť.

Implementujte jednotlivé stratégie. Funkcia, ktorá implementuje niektorú zo stratégií, dostane zoznam hmotností tovarov (v poradí v akom sú tovary uložené na rampe) a nosnosť vozíka a vráti zoznam hmotností tovarov, ktoré žeriav na vozík naloží.

Riešenie uložte do súboru **zeriav.py**.

Riešenie:



Úloha 10 Hra „Vojna“ je jednoduchá kartová hra pre dvoch hráčov. Hrávajú ju aj súrodenci Viktor a Oskar, keď ich mamka chce, aby sa trochu zabavili. Pravidlá sú jednoduché.

- 1 Zamiešame sadu kariet a každý dostane polovicu z nich.
- 2 Každý vyloží kartu, ktorú má na vrchu svojej kopy. Ak niektorý z hráčov nemá kartu na vyloženie, pokračuje sa bodom 4.
- 3 Ten, kto má na svojej vyloženej kope na vrchu vyššiu kartu, vloží všetky vyložené karty (najskôr svoju, potom súperovu vyloženú kopy) na spodok svojej kopy. Pokračuje sa bodom 2. Ak by na vrchu oboch vyložených kôp boli rovnaké karty, každý vyloží tri karty zo svojej kopy a opakuje sa bod 3. Ak niektorý z hráčov nemá tri karty na vyloženie, pokračuje sa bodom 4.
- 4 Víťazom je hráč, ktorý má dostatok kariet na to, aby ich vyložil. Ak nastala situácia, že obaja hráči už nemajú dostatok kariet na vyloženie, hra končí remízou.

Súrodencom napadlo, že hru nemusia hrať, ale nechajú počítač aby ju zahral namiesto nich. Vytvorte funkciu `hraj_vojnu()`, ktorá pre dva zadané zoznamy kariet hráčov bude priebeh hry Vojna simulovať. Výsledkom funkcie je informácia, ktorí z hráčov vyhral alebo či hra skončila remízou.

Riešenie uložte do súboru **vojna.py**.

Riešenie:

21 VNORENÉ RIADIACE ŠTRUKTÚRY

ÚVOD

Úloha 1 Zopakujme si, ktoré riadiace štruktúry už poznáme.

Riešenie:

cyklus for

príkaz if

cyklus while

výnimky

PRECVIČOVANIE, SAMOSTATNÁ PRÁCA

Úloha 2 Na čerpacej stanici majú uvedený nasledovný systém zliav:

odber v litroch	ZĽAVA	
	zákaznícka karta – NIE	zákaznícka karta – ÁNO
< 20	0 %	1 %
< 40	1 %	2 %
>= 40	2 %	4 %

Vytvorte funkciu `platba()`, ktorá pre zadaný objem natankovaného paliva, jednotkovú cenu paliva a informáciu, či zákazník má zákaznícku kartu vráti sumu, ktorú má zákazník zaplatiť. Riešenie uložte do súboru `cerpacia_stanica.py`

Riešenie:

Úloha 3 Obeh Zeme okolo Slnka netrvá celý počet dní. Presne je to 365 dní, 5 hodín, 48 minút a 45,4 sekundy. Aby sme kalendárny rok zosúladiť s obehom Zeme okolo Slnka, musíme jeho dĺžku pravidelne korigovať. Táto korekcia spočíva v zavedení priestupných rokov, ktoré majú o jeden deň (29. február) viac, ako ostatné roky. Priestupné sú tie roky, ktoré sú deliteľné 4. Ak je rok zároveň deliteľný 100, tak priestupný nie je. Ak by však bol deliteľný aj 400, priestupný bude. Vytvorte funkciu `je_priestupny_rok()`, ktorá pre zadaný rok zistí a vráti, či je priestupný alebo nie. Riešenie uložte do súboru **priestupne_roky.py**.

Riešenie:

Úloha 4 V súťaži v strojopise sa porovnáva originálny text s jeho prepisom. Ak súťažiaci navzájom vymení dva po sebe idúce znaky, dostane jeden mínusový bod. Ak súťažiaci namiesto originálneho znaku napísal nejaký iný znak, dostáva dva mínusové body.

Časť b) riešite podľa pokynov učiteľa.

a) Vytvorte funkciu `vyhodnot()`, ktorá porovná originálny text s jeho prepisom a vráti počet mínusových bodov. Ak porovnávané texty nemajú rovnakú dĺžku, nech funkcia vyhodí zmysluplnú výnimku. Riešenie uložte do súboru **strojopis.py**.

b) Upravte funkciu `vyhodnot()` tak, aby dokázala porovnať aj rôzne dlhé texty. Texty porovnávejte od začiatku a chýbajúce znaky považujte za chybné prepísané znaky.

Riešenie a):

Riešenie b):

Úloha 5 V laboratóriu skúmajú zaujímavý kmeň buniek. Zistili, že ak ich umiestnia do kruhu, populácia buniek sa každú hodinu zmení podľa nasledovných pravidiel:

- ak bunka má dvoch susedov, do ďalšej generácie neprežije a umiera na nedostatok potravy,
- ak bunka má jedného suseda, do ďalšej generácie bunka prežije,
- ak bunka nemá žiadneho suseda, do ďalšej generácie neprežije a umiera na osamotenie,
- ak prázdne políčko má dvoch susedov, v ďalšej generácii sa na tomto mieste objaví živá bunka,
- v ostatných prípadoch sa situácia na danom mieste nezmení.

- a) Vytvorte funkciu `dalsia_generacia()`, ktorá zmení zadanú generáciu buniek na ďalšiu generáciu.
b) Vytvorte funkciu `zobraz_generacie()`, ktorá pre zadaný počet hodín a počiatočnú generáciu buniek postupne zobrazí, ako sa populácia buniek vyvíjala.

Pomôcka: Premyslite si, ako vhodne reprezentovať stav populácie buniek.

Riešenie uložte do súboru **zivot.py**.

Riešenie:

ZHRNUTIE

Diskusia o problematických úlohách a ich riešeniach.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 V pretekoch „Veteráni na doraz“ je úlohou vodičov prejsť trasu medzi dvoma miestami čo najrýchlejšie a pritom neprekročiť maximálnu povolenú rýchlosť. Preteky vždy začínajú v obci, mimo diaľnice alebo cesty pre motorové vozidlá. Aby organizátori vylúčili podvod, pri ktorom auto prekročí rýchlosť, namontovali do každého auta nezávislý snímací systém. Systém v pravidelných intervaloch zaznamenáva rýchlosť auta. Okrem toho je schopný registrovať dopravné značky informujúce o začiatku alebo konci obce, diaľnice a cesty pre motorové vozidlá ('zo', 'ko', 'zd', 'kd', 'zcmv', 'kcmv').

Na Slovensku platia nasledovné rýchlostné obmedzenia:

- obec – max 50 km/h
- mimo obce – max 90 km/h
- cesta pre motorové vozidlá v obci – max 50 km/h
- cesta pre motorové vozidlá mimo obce – max 90 km/h
- diaľnica v obci – max 90 km/h
- diaľnica mimo obce – max 130 km/h

- a) Vytvorte funkciu `prekrocil_rychlost()`, ktorá pre zadaný záznam monitorovacieho prístroja zistí, či vodič prekročil rýchlosť.
- b) Občas sa stane, že systém nejakú značku nezaregistruje. Ak takú situáciu dokážeme detegovať, záznam nemá zmysel vyhodnocovať. Upravte funkciu `prekrocil_rychlost()` tak, aby v prípade detekcie nekorektného záznamu funkcia vyhodila zmyslupnú výnimku.

Záznam systému môže vyzeráť nasledovne: ['20', '25', '49', 'zd', '70', '120', '135', 'ko', '125', 'kd', '48']

Pomôcka: Ak snímací systém zaregistruje dve dopravné značky na jednom stĺpiku, do záznamu najskôr uloží údaj o ukončení až potom údaj o začatí. Predpokladajte, že iné obmedzenia rýchlosti neexistujú.

Riešenie uložte do súboru **veteran.py**.

Riešenie a):

Riešenie b):



22 OPAKOVANIE IV. + DIDAKTICKÝ TEST

PREDSTAVENIE CIEĽOV VYUČOVANIA

Precvičenie a systematizácia poznatkov z predchádzajúcich hodín. Pri riešení úloh budeme pracovať so zoznamami, s cyklami, podmienenými príkazmi, s vnorenými riadiacimi štruktúrami a s výnimkami.

PRECVIČOVANIE, PRÁCA ŽIAKOV

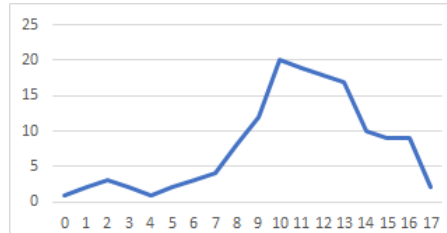
Úvodný text

Tento text je spoločný pre všetky nasledujúce úlohy.

Turisti si pomocou výškomera priebežne zaznamenávajú nadmorskú výšku miest, ktorými prechádzajú počas turistických výletov. Používajú výškomer, ktorý v pravidelných jednominútových intervaloch zaznamenáva nadmorskú výšku. Zaznamenávanie spustia pri štarte a zastavia ho, keď prídu na koniec trasy. V ročenke potom pri prejdenej trasách uvádzajú aj ich základné charakteristiky.

Spracovanie záznamov do ročenky by si radi čo najviac uľahčili. Hodili by sa im programy alebo funkcie, ktoré by dokázali záznamy výškomera vyhodnocovať.

Riešenia nasledujúcich úloh ukladajte do súboru **turistika.py**.



Úloha 1 Vytvorte funkcie:

- `trvanie_trasy()`, ktorá pre zadaný záznam trasy vráti trvanie trasy v minútach,
- `najvyšší_bod()`, ktorá pre zadaný záznam trasy vráti najvyšší bod (v metroch) trasy.

Riešenie:

Úloha 2 Vytvorte funkcie:

Časť b) riešte podľa pokynov učiteľa

- `celkove_stupanie()`, ktorá pre zadaný záznam trasy vráti jej celkové stúpanie (súčet všetkých stúpaní na trase),
- `celkove_klesanie()`, ktorá pre zadaný záznam trasy vráti jej celkové klesanie (súčet všetkých klesaní na trase).

Riešenie:

Úloha 3 Vytvorte funkcie:

Časť b)
riešte
podľa
pokynov
učiteľa

- a) `najdlhsie_stupanie()`, ktorá pre zadaný záznam trasy vráti trvanie úseku, v ktorom sa najdlhšie stúpalo,
b) `najdlhsie_klesanie()`, ktorá pre zadaný záznam trasy vráti trvanie úseku, v ktorom sa najdlhšie klesalo.

Riešenie:

Úloha 4 Občas sa stane, že turista pri štarte zabudne kalibrovať výškomer. Všetky záznamy výškomera sú potom posunuté smerom hore alebo smerom dole podľa toho, aká bola počiatočná odchýlka výškomera.

- Vytvorte funkciu `posun_vysky()`, ktorá pre zadaný záznam trasy a odchýlku výškomera pri štarte vráti upravený záznam.

Riešenie:

Úloha 5 Niektoré trasy sú typu „vrcholovka“. Priebeh týchto trás je charakteristický tým, že od začiatku trasy až na vrchol trasa nikde neklesá a z vrcholu až na koniec trasa nikde nestúpa.

- Vytvorte funkciu `je_vrcholovka()`, ktorá pre zadaný záznam trasy vráti informáciu, či trasa je typu „vrcholovka“.

Riešenie:

Úloha 6 V niektorých prípadoch si turisti prácu rozdelia a každý prejde a zaznamená len časť trasy. Posledný zaznamenaný údaj prvého turistu predstavuje ten istý bod trasy ako prvý zaznamenaný údaj druhého turistu. Pred vyhodnotením musia záznamy spojiť.

Časť b)
riešte
podľa
pokynov
učiteľa

- a) Vytvorte funkciu `pripoj_zaznam()`, ktorá do prvého zadaného záznamu pridá údaje z druhého záznamu.
- b) Upravte funkciu `pripoj_zaznam()` tak, aby pridávané výšky z druhého záznamu boli upravené (posunuté) tak, aby prvá výška v druhom zázname bola rovnaká, ako posledná výška v prvom zázname.

Riešenie a):

Riešenie b):

Úloha 7 V ročenke sa pri popise trasy uvádza aj jej profil. Výškomer je citlivý prístroj a niektoré faktory prostredia spôsobia, že zaznamenaná výška sa mierne líši od skutočnosti a výsledný profil je trochu „roztrasený“.

- a) Vytvorte funkciu `vyhlad_zaznam()`, ktorá vráti vyhladený záznam trasy. Záznam trasy vyhladí tak, že každú hodnotu nahradí priemerom troch hodnôt v jej okolí (hodnotu na pozícii `idx` nahradí

priemerom hodnôt na pozíciách $idx-1$, idx a $idx+1$. Prvý, resp. posledný bod trasy nahradí priemerom prvých, resp. posledných dvoch hodnôt.

- b) Upravte funkciu `vyhlad_zaznam()` tak, aby v prípade, že sa záznam nedá pre malý počet údajov vyhodnotiť, funkcia vyhodila zmysluplnú výnimku.

Riešenie:

ZHRNUTIE

Diskusia o problematických úlohách a ich riešeniach.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 8 Hanka a Zuzka hrávajú hru, v ktorej reťazia slová tak, aby niektoré písmenká jedného slova boli súčasťou aj iného slova. Na začiatku premiešajú spoločnú kopy kartičiek so slovami a následne si z neho každá zoberie rovnaký, vopred dohodnutý počet kariet na svoju kopy.

Počas hry striedavo ukladajú na hraciu plochu karičky so slovami zo svojich kôp tak, aby sa aspoň v jednom písmene prekrývali s nejakým už vyloženým slovom. Ak niektorá nevie vyložiť žiadne slovo, prenechá ťah súperke. Vyhráva tá, ktorej hodnota kopy nevyložených slov je na konci hry najnižšia. Ak k takejto situácii dôjde a na hracom pláne je ešte veľa voľných miest dohodnú sa, že si zo spoločnej kopy každá ešte nejaké karty zoberie.

			o						
			d	i	a				
			o		b				
			b	y	l	a	d		
			á		a	e			
			b	e	h	u	s		

Aj keď používajú len trojpísmenkové slová, slov je veľa. Pomohlo by im, keby mali nejaký program, ktorý by im manipuláciu s nimi uľahčil.

Vytvorte funkcie:

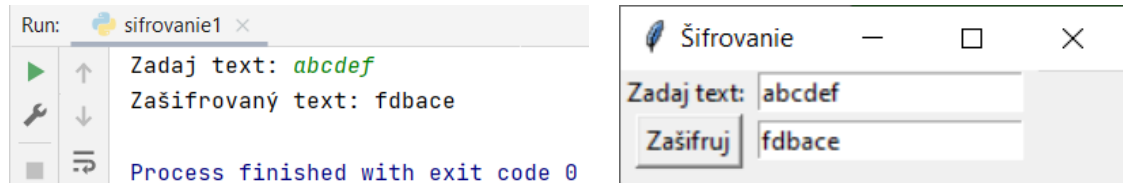
- `rozdel_slova()` – zamieša spoločný zoznam kariet a presunie určený počet kariet zo spoločného zoznamu do dvoch zoznamov, ak by celkový počet slov bol nedostatočný, nech funkcia vyhodí zmysluplnú výnimku,
- `zoznam_slov_zacinajucich_pismenom()` – pre zadaný zoznam slov a písmeno, vráti zoznam slov zo zoznamu začínajúcich daným písmenom,
- `zoznam_slov_konciacich_pismenom()` – pre zadaný zoznam slov a písmeno, vráti zoznam slov zo zoznamu končiacich daným písmenom,
- `nahodne_slovo_zacinajuca_na_pismeno()` – vráti náhodne slovo zo zadaného zoznamu, ktoré začína daným písmenom, ak také slovo neexistuje, nech funkcia vyhodí zmysluplnú výnimku,
- `nahodne_slovo_konciace_na_pismeno()` – vráti náhodne slovo zo zadaného zoznamu, ktoré končí daným písmenom, ak také slovo neexistuje, nech funkcia vyhodí zmysluplnú výnimku,
- `zoznam_slov_so_zaciatkom_a_koncom()` – pre zadaný zoznam slov a začiatkové a koncové písmeno, vráti zoznam slov zo zoznamu, ktoré začínajú a končia uvedenými písmenami,
- `usporiadaj_abecedne()` – zadaný zoznam slov usporiada abecedne,
- `usporiadaj_abecedne_naopak()` – zadaný zoznam slov usporiada abecedne odzadu,
- `odstran_slova_zo_zoznamu()` – zo zadaného zoznamu slov odstráni všetky slová so zadaného zoznamu slov na odstránenie, pri odstraňovaní sa odstránia aj zrkadlové obrazy slov,
- `hodnota_zoznamu()` – vráti súhrnnú hodnotu slov v zozname,
 - každé slovo má hodnotu minimálne 1b,
 - za každý znak s diakritikou, sa hodnota slova zvýši o 1 bod,
 - ak je slovo zrkadlové, jeho hodnota sa zvýši o 1 bod,
- `pridaj_karty()` – presunie karty zo spoločného zoznamu do zadaného zoznamu. Aby si dievčatá hru trochu ozvláštnili dohodli sa, že každá povie nejaké písmenko a karty zo spoločnej kopy si pridáva do svojej kopy dovtedy, kým nenarazí na kartu so slovom, v ktorom sa písmenko nachádza. Túto kartu si do svojho zoznamu už nevloží. Táto karta sa vloží na koniec spoločného zoznamu.

Riešenie uložte do súboru `hra_slov.py`. V súbore `slova.txt` je už pripravený zoznam slovenských trojpísmenkových slov. Funkcia `nacitaj_zo_suboru()` v súbore `hra_slov.py` ich načíta a vráti ich zoznam.

23 GRAFICKÉ POUŽÍVATEĽSKÉ ROZHRAŇIE – TLAČIDLÁ, TEXTOVÉ POLIA, POPISY

ZAPOJENIE

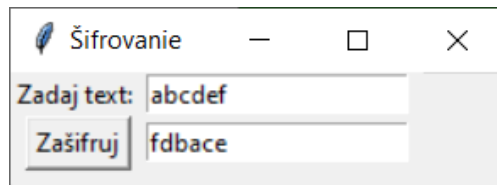
Úloha 1 Otvorte programy *sifrovanie1.py* a *sifrovanie2.py* a spustite ich s rovnakými vstupnými dátami.



- a) Uveďte, v ktorom z týchto programov sa Vám lepšie pracovalo a zdôvodnite aj prečo.
- b) Uveďte, ktoré prvky grafického rozhrania sú použité v programe *sifrovanie2.py*. Ktoré z nich slúžia ako vstupy resp. výstupy?

SKÚMANIE

Úloha 2 Otvorte program *sifrovanie2.py*. Spustite ho a preskúmajte jednotlivé prvky grafického rozhrania.



Do mriežky s grafickým rozhraním doplňte na mieste červených bodiek použité prvky grafického rozhrania a tiež čísla riadkov (angl. rows) a stĺpcov (angl. columns), kde sú umiestnené (číslovanie začíname od 0).

	stĺpec ...	stĺpec ...
riadok
riadok

Do tabuľky doplňte význam a použitie uvedených prvkov grafického rozhrania.

Grafický prvok	Význam a použitie grafického prvku	Grafický prvok	Význam a použitie grafického prvku
Label		Entry	
Button		Entry	

Úloha 3 Otvorte súbor *sifrovanie2.py*. Spustite súbor a preskúmajte ako je v ňom realizovaný výpočet.

```
import tkinter

def vypocet():
    ''' Vráti zašifrovaný text zo zadaného otvoreného textu

    :param text_otvoreny: otvorený text
    :type text_otvoreny: StringVar
    '''
    vstup = text_otvoreny.get()
    if len(vstup) % 2 == 0:
        vystup = vstup[-1::-2] + vstup[0::2]
    else:
        vystup = vstup[-2::-2] + vstup[0::2]
    text_sifrovany.set(vystup)

okno = tkinter.Tk()
okno.title('Šifrovanie')

# riadok 0: Label, Entry
tkinter.Label(okno, text='Zadaj text: ').grid(row=0, column=0)

text_otvoreny = tkinter.StringVar()
text_otvoreny.set('abcdef')
tkinter.Entry(okno, textvariable=text_otvoreny).grid(row=0, column=1)

# riadok 1: Button, Entry
tkinter.Button(okno, text='Zašifruj', command=vypocet).grid(row=1,
column=0)

text_sifrovany = tkinter.StringVar()
text_sifrovany.set('?')
tkinter.Entry(okno, textvariable=text_sifrovany).grid(row=1, column=1)

okno.mainloop()
```

a) Uveďte názvy premenných, ktoré boli vytvorené ako špeciálne premenné modulu tkinter typu **StringVar**:

b) Uveďte význam metód

premenná.set():

premenná.get():

c) Uveďte význam špeciálnych premenných modulu tkinter typu **StringVar**:

VYSVETLENIE

Diskusia:

1. Aké sú pozitíva a negatíva grafických programov?
2. Aké prvky grafického prostredia poznáte?
3. Ako sa rozmiestnili v okne jednotlivé grafické prvky?
4. Pri ktorých prvkoch grafického rozhrania používame špeciálne premenné modulu tkinter typu StringVar? Na čo slúžia?

Úloha 4 Otvorte program na výpočet BMI **bmi_kategorie1.py** a upravíte ho z konzolovej verzie na grafickú verziu a výsledný programový kód uložte do súboru **bmi_kategorie2.py**.

ROZPRACOVANIE

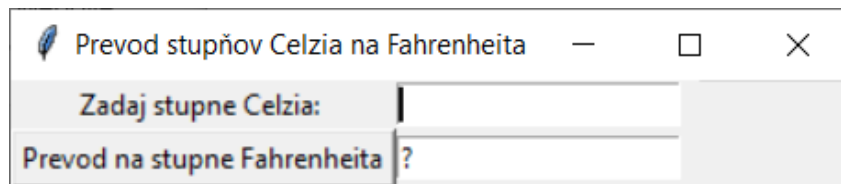
Úloha 5 Vytvorte program na overenie korektnosti zadaného rodného čísla. Prinajmenšom ošetríte zadanie numerických znakov, dĺžku vstupu a deliteľnosť rodného čísla číslom 11. Prípadne môžete ošetriť aj korektnosť dátumov a pohlavia, ktoré je uvedené v úlohe 3 v kapitole 16. Výsledný program uložte do súboru **overenie_rc.py**.

VYHODNOTENIE

SEBAHODNOTIACI TEST

1. Doplňte niektoré z uvedených šiestich textov na označené miesta **1 2 3** v programe na výpočet prevodu stupňov Celzia na stupne Fahrenheita.

- `stupne_Celzia.get()`
- `stupne_Celzia.set(vystup)`
- `stupne_Fahrenheita()`
- `stupne_Fahrenheita.set(vystup)`
- `vypocet()`
- `vypocet`



```
def vypocet():
    vstup = 1
    try:
        vystup = 32 + 1.8 * float(vstup)
    except ValueError:
        zobraz_chybu('Nezadali ste číselný vstup')
    2

tkinter.Button(okno, text='Prevod na stupne Fahrenheita',
command=3).grid(row=1, column=0)

stupne_Fahrenheita = tkinter.StringVar()
stupne_Fahrenheita.set('?')
tkinter.Entry(okno, textvariable=stupne_Fahrenheita).grid(row=1, column=1)
```

VEDOMOSTI V KOCKE

Pri tvorbe grafického programu typu vstup-spracovanie-výstup nám stačí použiť základné tri prvky grafického rozhrania:

- **popis** (angl. **Label**) na doprovodný text k ostatným prvkom grafického rozhrania,
- vstupné a výstupné **políčka** (angl. **Entry**),
- **tlačidlo** (angl. **Button**) na spustenie výpočtu.

Uvedené prvky grafického rozhrania umiestnime na obrazovke aplikácie pomocou manažéra grafického obsahu **grid** do jednotlivých políčok mriežky.

Na prepojenie prvkov grafického rozhrania (v našom prípade vstupných a výstupných políčok) s funkciami pre výpočet použijeme špeciálne **StringVar** premenné, ktorých hodnoty načítavame resp. zapisujeme pomocou metód **get()** resp. **set()**.

Robustnosť programu (jeho odolnosť voči zadaniu chybných vstupov) zabezpečíme generovaním a zachytávaním výnimiek. Na rozdiel od konzolového programu pri grafickom programe chybu nevypisujeme do konzoly, ale ju zobrazíme do nového okna pomocou grafického prvku **vyskakovacie okno** (angl. **messagebox**), ktorý potrebujeme zvlášť importovať.

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 6 Vytvorte program s grafickým používateľským rozhraním na výpočet vzdialenosti dvoch bodov na Zemi so zadanými geolokačnými súradnicami. (Nápoveda: Pre výpočet vzdialeností dvoch miest so zemepisnými šírkami a dĺžkami (\check{s}_1, d_1) a (\check{s}_2, d_2) môžeme použiť vzťah:

$$\text{vzdialenosť} = 6371 \cdot \arccos(\sin \check{s}_1 \cdot \sin \check{s}_2 + \cos \check{s}_1 \cdot \cos \check{s}_2 \cdot \cos(d_2 - d_1))$$

Zdroj: https://en.wikipedia.org/wiki/Great-circle_distance)

Úloha 7 Vytvorte program s grafickým používateľským rozhraním na výpočet zloženého úrokovania. Na vstupe je zadaný počiatočný vklad, ročná úroková miera (v percentách), počet rokov úročenia. Výsledkom programu je výsledná suma, ktorú dostaneme po viacročnom úročení.

24 GRAFICKÉ POUŽÍVATEĽSKÉ ROZHRAŇIE – PLÁTNO

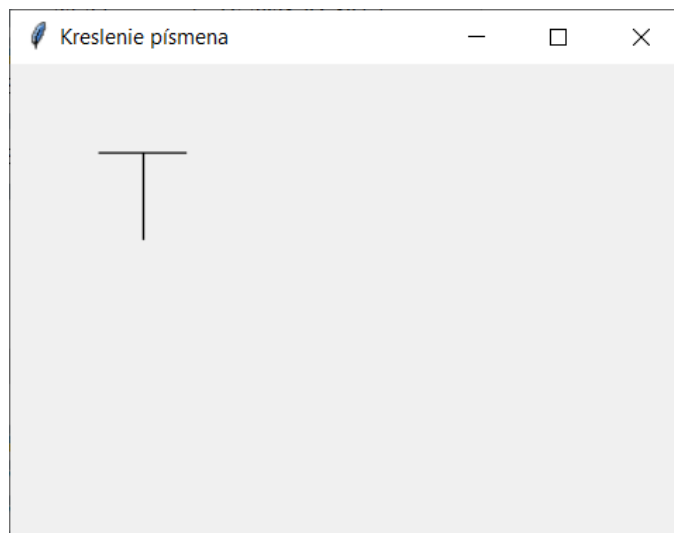
ZAPOJENIE

Úloha 1 Otvorte a spustite súbor **pismo.py**.

```
import tkinter

def pismo():
    platno.create_line(50, 50, 100, 50)
    platno.create_line(75, 50, 75, 100)

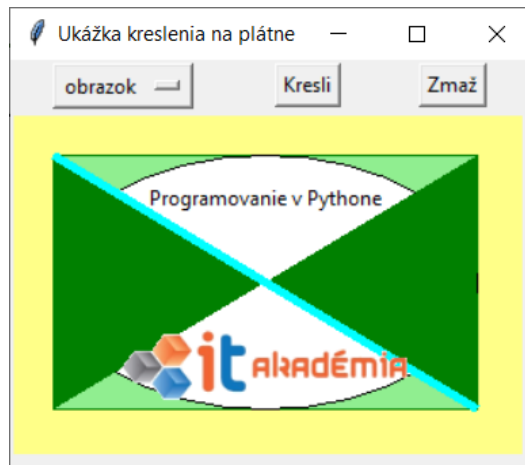
okno = tkinter.Tk()
okno.title('Kreslenie písmena')
platno = tkinter.Canvas(okno)
platno.grid()
pismo()
okno.mainloop()
```



- Do obrázku doplňte súradnice koncových bodov úsečiek vykreslených pomocou funkcie **pismo()**.
- Upravte súradnice jednej z úsečiek vo funkcii **pismo()** tak, aby vykreslila písmeno L.
- V čom sa líši tento súradnicový systém a bežne používaný karteziánsky súradnicový systém?

SKÚMANIE

Úloha 2 Otvorte súbor **platno_ukazka.py**. Spustite ho a preskúmajte jednotlivé grafické prvky a ich metódy.



```
1 import tkinter
2
3
4 def kresli():
5     if vyber.get() == 'obdĺžnik':
6         platno.create_rectangle(25, 25, 275, 175, fill='lightgreen',
7             outline='green')
8         elif vyber.get() == 'elipsa':
9             platno.create_oval(25, 25, 275, 175, fill='white')
10        elif vyber.get() == 'čiara':
11            platno.create_line(25, 25, 275, 175, fill='cyan', width=5)
12        elif vyber.get() == 'lomená čiara':
13            platno.create_polygon(25, 25, 275, 175, 275, 25, 25, 175,
14                fill='green')
15        elif vyber.get() == 'text':
16            platno.create_text(150, 50, text='Programovanie v Pythone',
17                fill='black')
18        elif vyber.get() == 'obrázok':
19            platno.create_image(150, 150, anchor='center', image=obrazok)
20
21
22
23 def zmaz():
24     platno.delete('all')
25
26
27
28 okno = tkinter.Tk()
29 okno.title('Ukážka kreslenia na plátne')
30 platno = tkinter.Canvas(okno, width=300, height=200, background="#FF8")
31 obrazok = tkinter.PhotoImage(file='ita_logo.gif')
32
33 # riadok 0: Label, Button, Button
34 zoznam = ['obdĺžnik', 'elipsa', 'čiara', 'lomená čiara', 'text', 'obrázok']
35 vyber = tkinter.StringVar()
36 vyber.set(zoznam[0])
37 tkinter.OptionMenu(okno, vyber, *zoznam).grid(row=0, column=0)
38
39 tkinter.Button(okno, text='Kresli', command=kresli).grid(row=0, column=1)
40
41 tkinter.Button(okno, text='Zmaž', command=zmaz).grid(row=0, column=2)
42
43 # riadok 2: Canvas
44 platno.grid(row=1, column=0, columnspan=3)
```

40

41 `okno.mainloop()`

- a) Do tabuľky doplňte význam uvedených grafických metód použitých na plátne (v riadkoch 6, 8, 10, 12, 14, 16).

Metóda	Význam metódy
<code>create_rectangle()</code>	
<code>create_oval()</code>	
<code>create_line()</code>	
<code>create_polygon()</code>	
<code>create_text()</code>	
<code>create_image()</code>	

- b) Uveďte, v ktorom stĺpci/stĺpcoch je umiestnené plátno? Ako dosiahneme, že sa natiahne na všetky tri stĺpce? Ktorý parameter na to slúži?

- c) Uveďte na čo slúži prvok grafického rozhrania **OptionMenu**:

- d) Uveďte na čo slúži prvok grafického rozhrania **PhotoImage**:

Úloha 3 Otvorte a spustite súbor *stalaktity.py*.

```

1  import tkinter
2
3
4  def nacistaj_data(meno_suboru):
5      ''' Načítanie dát zo súboru so zadaným menom do zoznamu
6
7      :param meno_suboru: zadané meno súboru s dátami
8      :type meno_suboru: str
9      :return: zoznam riadkov s dátami
10     :rtype: list of str
11     '''
12     with open(meno_suboru, 'r', encoding='utf-8') as f:
13         data = f.readlines()
14         data = [polozka.strip() for polozka in data]
15         return data
16
17
18     def kresli(zoznam):
19         for x in range(len(zoznam)):
20             platno.create_line(x + 5, 5, x + 5, zoznam[x])
21
22
23     okno = tkinter.Tk()
24     okno.title('Vykreslenie stalaktitov')
25     platno = tkinter.Canvas(okno, width=300, height=300, background='yellow')
26     platno.grid()

```

```
27 zoznam = [50, 40, 90, 70, 20, 60, 80, 10, 30, 100]
28 # zoznam = nacistaj_data('stalaktity1.txt')
29 # zoznam = nacistaj_data('stalaktity2.txt')
30 # zoznam = nacistaj_data('stalaktity3.txt')
31 # zoznam = nacistaj_data('stalaktity4.txt')
32 kresli(zoznam)
33
34 okno.mainloop()
35
```

- a) Akú zmenu vo funkcii **kresli()** treba urobiť, aby sme stalaktity nezobrazovali zhora nadol, ale ako úsečky zľava doprava?
- b) Ako sa zmení správanie programu, ak zakomentujeme riadok 28 a odstránime znak komentára z riadku 29 (resp. 30, 31 alebo 32)?
- c) Čo robí funkcia **nacistaj_data()**?
- d) Ktorý zo spôsobov považujete za lepší? Spracovanie hodnôt zoznamu uvedeného v programe, alebo v súbore mimo programu?

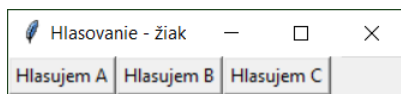
VYSVETLENIE

Diskusia:

5. Ako je orientovaný súradnicový systém na plátne modulu tkinter? Kde sa nachádza bod (0, 0)?
6. Ktoré ďalšie nové prvky grafického prostredia poznáte? Uveďte aj na čo slúžia.
7. Ktoré metódy grafického prvku Canvas poznáte? Uveďte, ktoré útvary vieme nimi vykresliť na plátne.
8. Kedy je vhodné spracovať hodnoty zoznamu uvedené v programe a kedy, keď sú hodnoty uvedené v súbore?

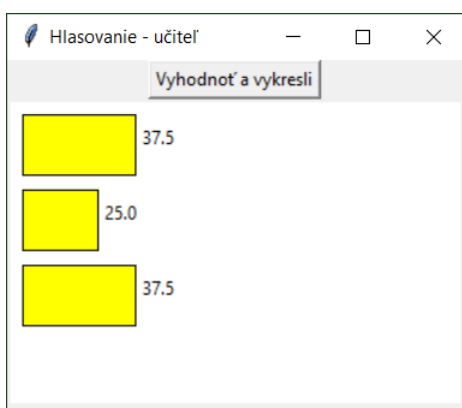
Úloha 4 Vytvorte hlasovací systém pozostávajúci z dvoch programov. V jednom programe budú žiaci hlasovať pomocou troch tlačidiel a v druhom bude učiteľ zobrazovať aktuálny stav hlasovania.

Otvorte a spustite súbor **hlasovanie_ziak.py**. Urobte niekoľko výberov kliknutím na tlačidlá a medzitým kontrolujte obsah súboru **hlasovanie.txt**.



Uveďte čo ste zistili z tohto experimentovania. Ako vyzerá súbor **hlasovanie.txt**?

Otvorte program **hlasovanie_ucitel.py** pre učiteľa, ktorý načítava hlasovanie žiakov zo súboru **hlasovanie.txt**. Naprogramujte v ňom funkciu **vyhodnot()**, ktorá vypočíta percentuálne zastúpenie jednotlivých hlasovacích volieb a vykreslí ich ako vodorovné stĺpcové diagramy na plátne spolu s percentuálnymi hodnotami.



ROZPRACOVANIE

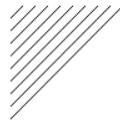
Úloha 5 Otvorte súbor **robot.py**. Doprogramujte v ňom dve funkcie **kresli_mriezku()** a **tah()**, aby program vykresľoval animáciu pohybu robota podľa jednopísmenkových príkazov z množiny {H, D, L, P} uvedených v textovom reťazci vo vstupnom poli. Funkcia **kresli_mriezku()** má vykresliť štvorcovú mriežku, po uzloch ktorej sa bude pohybovať robot. Funkciu **tah()** pre zadané súradnice robota a zadaný príkaz, vráti nové súradnice robota po vykonaní zadaného príkazu. Funkcia by mala ošetriť prípad, ak v reťazci príkazov bol použitý zadaný iný znak ako z množiny {H, D, L, P} a tiež prípad, ak sa robot dostal mimo štvorcovú mriežku.

Úloha 6 Premyslite si námet projektu, ktorý budete vytvárať na ďalších vyučovacích hodinách a zloženie vývojového tímu.

VYHODNOTENIE

SEBAHODNOTIACI TEST

1. Doplňte označené miesta ❶ ❷ ❸ v nasledujúcej funkcii tak, aby vykreslila skupinu úsečiek uvedenú na obrázku.



```
def kresli1():  
    for i in range(0, 101, 10):  
        platno.create_line(0, 1, 2, 3)
```

VEDOMOSTI V KOCKE

Okrem grafických prvkov **popis** (angl. **Label**), vstupné a výstupné **políčka** (angl. **Entry**), **tlačidlo** (angl. **Button**) a **vyskakovacie okno** (angl. **messagebox**), predstavených v predchádzajúcej kapitole, sa pri tvorbe grafických programov využíva aj grafický prvok **plátno** (angl. **Canvas**). Pomocou neho môžeme vizualizovať výpočty, kde použijeme rôzne metódy plátna, napr.:

- **create_rectangle()** – vykreslenie obdĺžnika so zadanými súradnicami ľavého horného a pravého dolného bodu obdĺžnika,
- **create_oval()** – vykreslenie elipsy určenou jej obálkou – obdĺžnikom so zadanými súradnicami ľavého horného a pravého dolného bodu obdĺžnika, ak sú tieto súradnice rovnaké, vykreslí sa bod,
- **create_line()** – vykreslenie úsečky so zadanými súradnicami jej koncových bodov,
- **create_polygon()** – vykreslenie uzavretej lomenej čiary so zadanými súradnicami jej koncových bodov so zadanou farbou jej výplne ,
- **create_text()** – vykreslenie zadaného textu na zadané súradnice plátna,
- **create_image()** – vykreslenie obrázku zo súboru na zadané súradnice plátna, pričom je potrebné grafický prvok **PhotoImage** so zadaným grafickým súborom na disku (s príponou png, gif).

Ďalším užitočným grafickým prvkom je **rozbaľovacia ponuka** (angl. **OptionMenu**) pre výber jednej z vymenovaných možností.

Súradnicový systém v module **tkinter** je odlišný od zaužívaného systému v matematike, t. j. počiatok súradnicového systému *O* bod (0,0) je v ľavom hornom rohu plátna s **osou x** nasmerovanou **doprava** a **osou y** nasmerovanou **nadol**.

Pri vykresľovaní na plátno sa štandardne prekrýva okno a v ňom uložené plátno v šírke dvoch bodov. Ak chceme vykresľovať body na plátno včítane bodu (0, 0), vieme to zabezpečiť nastavením parametra **highlightthickness=0** pri vytvorení plátna.

Pri spracovaní dát sa odporúča oddeliť dáta od programu. Tieto môžu byť uložené v textovom súbore. Na pridanie dát na koniec textového súboru môžeme použiť nasledovnú funkciu:

```
def zapis_data(data, subor):  
    with open(subor, 'a', encoding='utf-8') as f:  
        f.write(f'{data}\n')
```

Na načítanie dát z textového súboru do zoznamu môžeme použiť nasledovnú funkciu:

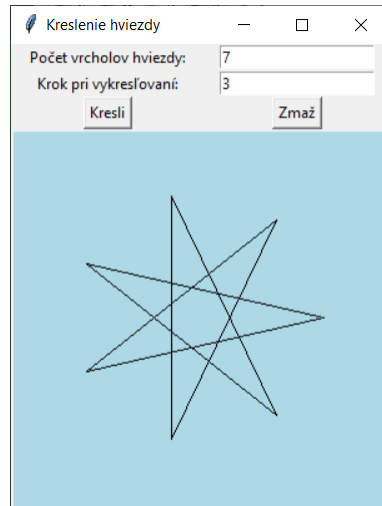
```
def nacitaj_data(meno_suboru):  
    with open(meno_suboru, 'r', encoding='utf-8') as f:  
        data = f.readlines()  
        data = [polozka.strip() for polozka in data]  
        return data
```

Ak nechceme uvádzať meno konkrétneho súboru, ale chceme ho vybrať pomocou dialógu v aktuálnom priečinku môžeme použiť grafický prvok **filedialog** a jeho metódu **askopenfilename()**:

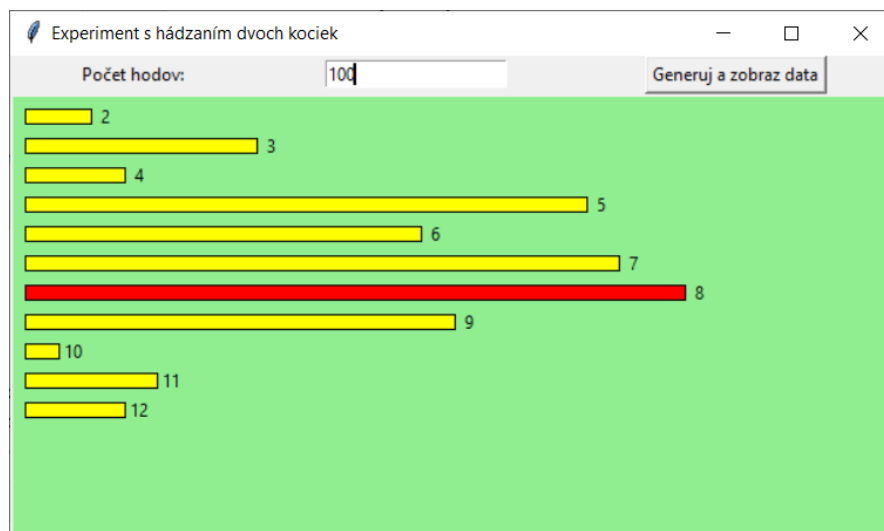
```
import tkinter.filedialog  
  
zoznam = nacitaj_data(tkinter.filedialog.askopenfilename())
```

ĎALŠIE ÚLOHY NA PRECVIČENIE

Úloha 7 Vytvorte program **hviezda.py** na vykreslenie jedným ťahom pravidelnej n -cípej hviezdy z n úsečiek.



Úloha 8 Dvaja kamaráti hrajú hru, v ktorej stokrát hádžu dvomi kockami. Každý z nich si vopred zvolí celé číslo od 2 do 12. Víťazom hry je ten hráč, ktorému viackrát padne jeho zvolený súčet kociek ako zvolený súčet kociek jeho súpera. Nasimulujte hádzanie dvomi kockami pre zadaný počet hodov a vyhodnoťte, či sú všetky súčty čísel rovnocenné alebo či sú niektoré súčty čísel viac pravdepodobné ako ostatné. Výsledky simulácie zobrazte na plátne.



25 KOMPLEXNÝ PROJEKT – VÝBER PROBLÉMU, ANALÝZA A NÁVRH RIEŠENIA PROBLÉMU

ZHRNUTIE DOTERAJŠÍCH VEDOMOSTÍ A ZRUČNOSTÍ Z PROGRAMOVANIA (CCA 6 MIN)

Úloha 1 *Prediskutujte v dvojiciach a zhrňte do tabuľky, čo ste sa naučili v predchádzajúcich hodinách programovania v Pythone.*

Typy premenných	Príkazy (len typovo)	Algoritmy	Oblasti problémov

DISKUSIA K TYPOM PROJEKTOV A NÁVRHOM KONKRÉTNÝCH PROJEKTOV (CCA 8 MIN)

Úloha 2 *Uvedte aké praktické problémy by sa mohli riešiť pomocou programovania?*

.....

.....

.....

.....

.....

.....

ETAPY TVORBY PROJEKTU (CCA 1 MIN)

Pri tvorbe projektu budeme postupovať v nasledovných etapách:

- **Zostavenie tímu. Výber témy projektu. Analýza a návrh riešenia problému** (jeho možné vstupy a výstupy). **Tvorba anotácie projektu.** (1 hodina)
- **Implementácia** riešenia problému. (1 hodina)
- **Finalizácia projektu** (testovanie funkčností programu, kompletizácia dokumentačných reťazcov funkcií, ošetrovanie chýb). **Sebahodnotenie výsledného projektu** (jeho úroveň, zoznam jeho možných vylepšení, úroveň svojej práce a práce ostatných členov tímu). (1 hodina)
- **Prezentácia projektov s diskusiou. Hodnotenie projektov učiteľom a spolužiakmi.** Založenie projektov do žiackeho (aj učiteľského) portfólia. (1 hodina)

POŽIADAVKY NA KOMPLEXNÝ PROJEKT. (SEBA)HODNOTIACA RUBRIKA (CCA 5 MIN)

Mená autorov projektu:		Trieda:		Dátum:	
Úroveň Položka	Úroveň 1 (1 bod)	Úroveň 2 (2 body)	Úroveň 3 (3 body)	Úroveň 4 (4 body)	Spolu
1 Náročnosť riešenia algoritmického problému, pri riešení žiak použil	nanajvyš jednoduchý sekvenčný algoritmus	algoritmus obsahujúci cykly alebo algoritmus obsahujúci vetvenie	algoritmus obsahujúci cykly a vetvenie	algoritmus obsahujúci vzájomné vnorené cykly a vetvenie	
2 Dátové štruktúry použité pri riešení algoritmického problému	žiadne alebo len jednoduché typy int alebo float alebo boolean	štruktúrovaný typ str	štruktúrovaný typ list	kombinácia viacerých štruktúrovaných typov	
3 Ošetrenie chybových stavov	testujú sa nanajvyš požiadavky na vstup	testujú sa operácie s dátami	testujú sa požiadavky na vstup a operácie s dátami		
4 Reakcia na chybové stavy	v prípade problematickeho stavu nie je používateľ informovaný alebo je informovaný len systémovou hláškou	v prípade problematickeho stavu je používateľ informovaný relevantnou hláškou o problematickom stave (napr. aj výpisom do konzoly)	v prípade problematickeho stavu je používateľ informovaný relevantnou hláškou o problematickom stave prostredníctvom grafického rozhrania		
5 Vstup dát	program nepoužíva vstup alebo len vstup z konzoly	vstupné textové okienko/tlačidlo grafickej aplikácie, myš v grafickej ploche	vstupné dáta zo súboru	kombinácia vstupu zo súboru s ďalším interaktívnym vstupom	
6 Výstup dát	program nepoužíva výstup alebo len výstup do konzoly	výstupné textové okienko/ výstup do grafickej plochy	výstupné dáta do súboru	kombinácia výstupu do súboru v kombinácii s ďalším typom výstupu	

7 Dekompozícia problému	žiadna alebo len jedna funkcia riešiaci celý problém	vhodne navrhnutá funkcia s parametrom alebo funkcia s návratovou hodnotou	správne dekomponovaný problém, vhodne navrhnutá funkcie s parametrami a s návratovými hodnotami	správne dekomponovaný problém, jednotlivé časti problému sa riešia v samostatných funkciách, funkcie sa vzájomne volajú a odovzdávajú si výstupné hodnoty	
8 Kód programu	identifikátory sú nejasné, z ich názvu nie je možné dedukovať ich význam	identifikátory (názvy premenných a funkcií) sú popisné, z ich názvu je možné dedukovať význam ich obsahu, resp. použitia	funkcie obsahujú dokumentačné reťazce popisujúce výsledok funkcie, vstupné a výstupné hodnoty	dokumentačné reťazce sú úplné, vstupy a výstupy sú definované podľa štandardu	
9 Používateľské rozhranie	grafické rozhranie nie je k dispozícii alebo je neprehľadné	grafické rozhranie je prehľadné	je zrejmé, ktoré polia sú vstupné a ktoré výstupné, obsah polí je popísaný (napr. pomocou Label)	výstupné polia sú chránené voči používateľským vstupom	
10 Prezentácia projektu žiakom	žiak vie nanajvyš „prerozprávať“ časti kódu	žiak vie popísať logiku algoritmu a identifikovať jednotlivé časti programu (napr. táto funkcia robí to a to)	žiak vie zdôvodniť použitý algoritmus, postup, žiak vie vysvetliť jednotlivé časti programu	žiak vie popísať hraničné (extrémne, krajné) prípady a ako na ne v programe reaguje, príp. v ktorých situáciách program nebude pracovať správne	
11 Využitie projektu v praxi	použiteľný v praxi po veľkých úpravách	použiteľný v praxi po menších úpravách	použiteľný v praxi bez potrebných úprav		
Spolu					

SAMOSTATNÁ PRÁCA V TÍMOCH NA TVORBE ANOTÁCIE PROJEKTU (CCA 20 MIN)

Úloha 3 Zostavte dvoj až trojčlenné projektové tímy. Uvedte **mená a e-maily** členov tímu a doplňte dohodnutého **hlavného zodpovedného člena tímu**:

.....
.....
.....

Prediskutujte a vyberte **tému** vášho tímového projektu:

.....

Napište stručnú **anotáciu** – akého typu je projekt, pre koho je určený, aký je jeho význam/úžitok pre prax, čo konkrétne bude program robiť:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Uložte si túto anotáciu projektu a pošlite ju spolupracovníkom v tíme a tiež svojmu učiteľovi.

26 KOMPLEXNÝ PROJEKT – IMPLEMENTÁCIA RIEŠENIA PROBLÉMU

SPRESNENIE ŠPECIFIKÁCIE PROBLÉMU A NÁVRH FUNKCIONALÍT PROGRAMU (CCA 15 MIN)

Úloha 1 Formulujte problém, ktorého riešenie chcete naprogramovať:

.....

Urobte detailnejšiu analýzu problému. Určte, čo sú vstupy a výstupy:

Vstupy

Výstupy

Uveďte vzťahy medzi vstupmi a výstupmi:

.....

.....

Rozdeľte problém do podproblémov. Pre každý podproblém navrhnite jeho funkcionality:

.....

.....

.....

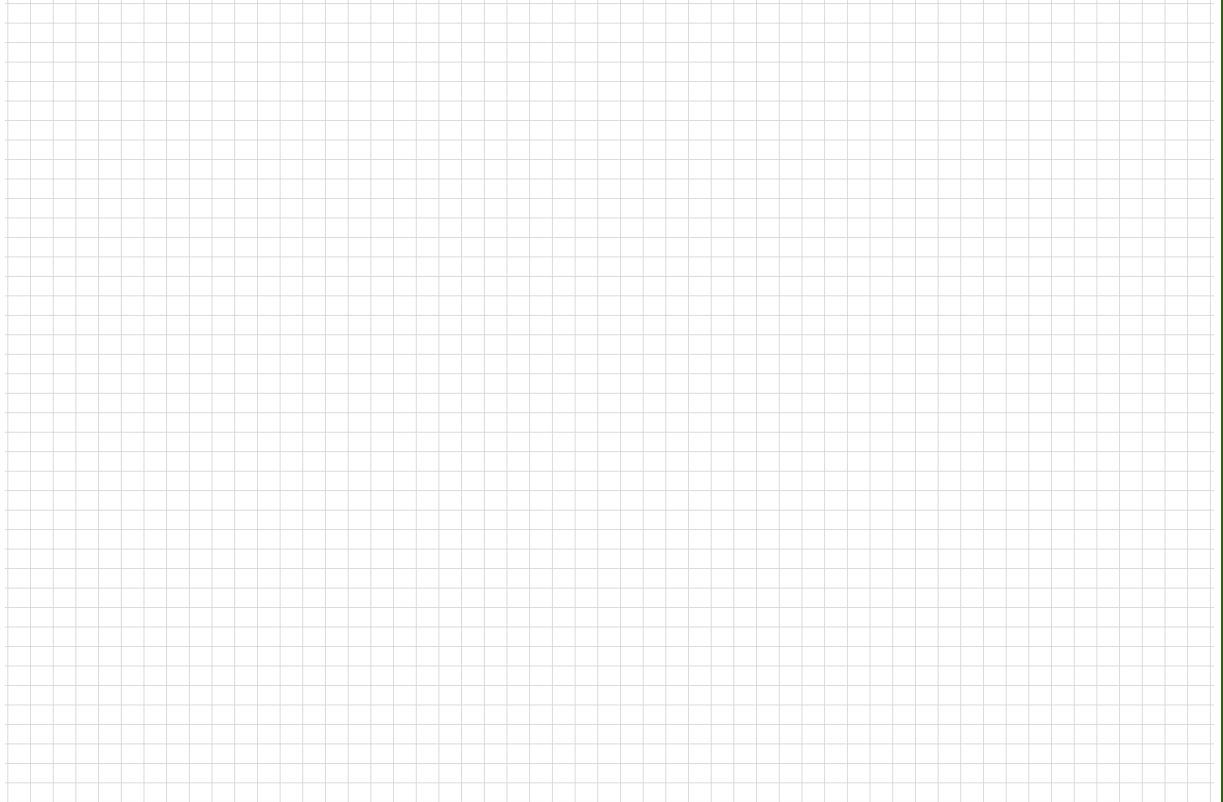
.....

.....

.....

NÁVRH GRAFICKÉHO POUŽÍVATEĽSKÉHO ROZHRAINIA (CCA 5 MIN)

Úloha 2 Navrhnite grafické používateľské rozhranie – vytvorte maketu s prvkami grafického používateľského rozhrania umiestnenými v mriežke so stručným opisom ich použitia.



IMPLEMENTÁCIA RIEŠENIA PROBLÉMU (CCA 20 MIN)

Úloha 3 Pre jednotlivé podproblémy navrhnite funkcie, ktoré ich budú riešiť. Pri každej funkcii určte vstupy, výstupy a vzťah medzi nimi. Odporúčame začať tvorbu funkcie napísaním dokumentačného reťazca a až následne dotvoriť programový kód funkcie.

27 KOMPLEXNÝ PROJEKT – FINALIZÁCIA PROJEKTU + PREZENTÁCIA A DISKUSIA

Mená autorov projektu:			Trieda:	Dátum:	
Úroveň	Úroveň 1 (1 bod)	Úroveň 2 (2 body)	Úroveň 3 (3 body)	Úroveň 4 (4 body)	Spolu
Položka					
1 Náročnosť riešenia algoritmického problému, pri riešení žiak použil	nanajvýš jednoduchý sekvenčný algoritmus	algoritmus obsahujúci cykly alebo algoritmus obsahujúci vetvenie	algoritmus obsahujúci cykly a vetvenie	algoritmus obsahujúci vzájomné vnorené cykly a vetvenie	
2 Dátové štruktúry použité pri riešení algoritmického problému	žiadne alebo len jednoduché typy int alebo float alebo boolean	štruktúrovaný typ str	štruktúrovaný typ list	kombinácia viacerých štruktúrovaných typov	
3 Ošetrovanie chybových stavov	testujú sa nanajvýš požiadavky na vstup	testujú sa operácie s dátami	testujú sa požiadavky na vstup a operácie s dátami		
4 Reakcia na chybové stavy	v prípade problematickeho stavu nie je používateľ informovaný alebo je informovaný len systémovou hláškou	v prípade problematickeho stavu je používateľ informovaný relevantnou hláškou o problematickom stave (napr. aj výpisom do konzoly)	v prípade problematickeho stavu je používateľ informovaný relevantnou hláškou o problematickom stave prostredníctvom grafického rozhrania		
5 Vstup dát	program nepoužíva vstup alebo len vstup z konzoly	vstupné textové okienko/tlačidlo grafickej aplikácie, myš v grafickej ploche	vstupné dáta zo súboru	kombinácia vstupu zo súboru s ďalším interaktívnym vstupom	
6 Výstup dát	program nepoužíva výstup alebo len výstup do konzoly	výstupné textové okienko/ výstup do grafickej plochy	výstupné dáta do súboru	kombinácia výstupu do súboru v kombinácii s ďalším typom výstupu	
7 Dekompozícia problému	žiadna alebo len jedna funkcia riešiaci celý problém	vhodne navrhnutá funkcia s parametrom alebo	správne dekomponovaný problém,	správne dekomponovaný problém,	

		funkcia s návratovou hodnotou	vhodne navrhnutá funkcie s parametrami a s návratovými hodnotami	jednotlivé časti problému sa riešia v samostatných funkciách, funkcie sa vzájomne volajú a odovzdávajú si výstupné hodnoty	
8 Kód programu	identifikátory sú nejasné, z ich názvu nie je možné dedukovať ich význam	identifikátory (názvy premenných a funkcií) sú popisné, z ich názvu je možné dedukovať význam ich obsahu, resp. použitia	funkcie obsahujú dokumentačné reťazce popisujúce výsledok funkcie, vstupné a výstupné hodnoty	dokumentačné reťazce sú úplné, vstupy a výstupy sú definované podľa štandardu	
9 Používateľské rozhranie	grafické rozhranie nie je k dispozícii alebo je neprehľadné	grafické rozhranie je prehľadné	je zrejmé, ktoré polia sú vstupné a ktoré výstupné, obsah polí je popísaný (napr. pomocou Label)	výstupné polia sú chránené voči používateľským vstupom	
10 Prezentácia projektu žiakom	žiak vie nanajvyš „prerozprávať“ časti kódu	žiak vie popísať logiku algoritmu a identifikovať jednotlivé časti programu (napr. táto funkcia robí to a to)	žiak vie zdôvodniť použitý algoritmus, postup, žiak vie vysvetliť jednotlivé časti programu	žiak vie popísať hraničné (extrémne, krajné) prípady a ako na ne v programe reaguje, príp. v ktorých situáciách program nebude pracovať správne	
11 Využitie projektu v praxi	použiteľný v praxi po veľkých úpravách	použiteľný v praxi po menších úpravách	použiteľný v praxi bez potrebných úprav		
Spolu					

SEBAHODNOTIACI DOTAZNÍK

Uvedte na koľko percent hodnotíte **celkovú funkčnosť Vášho programu** vzhľadom k anotácii: %

Uvedte **ďalšie funkcionality programu**, o ktoré by sa dal **doplniť** a **vylepšiť** Váš komplexný projekt:

.....

.....

.....

.....

Uvedte, za ktoré záležitosti pri tvorbe Vášho komplexného projektu ste boli Vy **osobne zodpovedný/á**:

.....

Do akej miery ste **spokojný/á** s **tímovou prácou** a **výsledkom projektu**? Vyberte vhodnú odpoveď pre každú položku:

S/So	som	veľmi spokojný/á	skôr spokojný/á	neviem	skôr nespokojný/á	veľmi nespokojný/á
svojou prácou na projekte						
prácou ostatných členov tímu						
výsledným programom						

Uvedte, čo ste sa pri tvorbe projektu **nové naučili**:

.....

Áké sú Vaše postoje k **programovaniu**? Vyberte vhodnú odpoveď pre každú položku:

Programovanie je	určite áno	skôr áno	neviem	skôr nie	určite nie
náročné					
zaujímavé					
dôležité					

Okrem tohto komplexného projektu ste naprogramovali niečo užitočné **aj mimo vyučovania** programovania, napr. hru, pomôcku pre nejaký školský predmet? áno – nie

Ak áno, stručne uvedte o **aké softvérové aplikácie** išlo:

.....

.....

PRÍLOHY:

Elektronická príloha Programovanie v jazyku Python - Pracovný zošit - príloha.zip dostupná z [https://di.ics.upjs.sk/publikacie/Programovanie v jazyku Python - Pracovný zošit - príloha.zip](https://di.ics.upjs.sk/publikacie/Programovanie_v_jazyku_Python_-_Pracovny_zosit_-_priloha.zip). Archív obsahuje priečinky 01 až 27 obsahujúce pythonovské zdrojové kódy pracovných súborov, prípadne referenčné materiály či inštruktážne listy.