INFORMATIKA V PRÍRODNÝCH VEDÁCH A MATEMATIKE – ZOŠIT INFORMATIKA

GABRIELA ANDREJKOVÁ A ZUZANA TKÁČOVÁ

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM ĽUDSKÉ ZDROJE

MINISTERSTVO ŠKOLSTVA, VEDY, VÝSKUMU A ŠPORTU SLOVENSKEJ REPUBLIKY

it akadémia

UNIVERZITA MATEJA BELA

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

Informatika v prírodných vedách a matematike – zošit Informatika

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Bratislava 2020

Informatika v prírodných vedách a matematike – zošit Informatika

Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Autori: Gabriela Andrejková, Zuzana Tkáčová

Recenzenti: Božena Mannová, Veronika Stoffová, Juraj Ťapák

Neprešlo jazykovou úpravou.

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2020

Vydanie: 1. vydanie

ISBN: 978-80-89965-69-4

EAN: 9788089965694

Bratislava 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.

OBSAH

ÚVOD

Stručne charakterizujeme tri oblasti, ktorým je venovaný tento text a pre ich výučbu sú pripravené metodiky. Text je určený pre učiteľov, ktorí budú predmet Informatika v prírodných vedách a matematike vyučovať. Tiež uvedieme predstavu a odporúčanie pre výučbu.

Výučba je orientovaná do troch nasledujúcich oblastí:

1) Vytváranie a spracovanie obrázkov na úrovni pixelov
2) Modelovanie jednoduchých vzťahov
3) Použitie databáz

VYTVÁRANIE A SPRACOVANIE OBRÁZKOV NA ÚROVNI PIXELOV

Pixel Art je umenie, v ktorom sa pri kreslení využívajú pixely.

V súčasnosti existuje mnoho aplikácií, ktoré nám poskytujú možnosti manipulácie a úpravy fotografií a iných obrázkov. Vizualizácia dát kvôli ich lepšiemu pochopeniu sa vykonáva v každej oblasti, preto je dobré porozumieť, ako to prebieha, o čom je potrebné pri ich spracovaní uvažovať. Naším cieľom je, aby žiaci pochopili princípy tvorby a úprav obrázkov, a naučiť ich niektoré základné pojmy, ktoré sa tu používajú. Náš zámer je naučiť sa:

- Vytvárať a zobrazovať rôzne obrázky použitím prostriedkov knižníc v programovacom jazyku Python. Poznámka: Vytvorený pomocný program a obrázky sú k dispozícii, nové obrázky sa žiaci naučia vytvárať a v programe ich môžu modifikovať.
- Rovinný obrázok spracovať na základe farebnosti použitých pixelov. Naučiť sa meniť farby pixelov, pridať pixely do obrázka. Poznámka: Obrázok bude získavaný ako výrez skutočného obrázka. Na spracovanie je pripravený program v jazyku Python.
- Vytvoriť obrázok živočícha v rovine, ktorý rastie na základe daných pravidiel, v programovacom jazyku Python. Poznámka: Pravidlá rastu budú analyzované, pomocný program je k dispozícii. Žiaci budú robiť malé modifikácie pravidiel. Cieľom je predpovedať, ako živočích narastie v nejakom čase.

MODELOVANIE

Tieto úlohy vedú tiež k biologicky motivovaným úlohám o raste nádorov.

Táto časť je zameraná na vytváranie modelov vychádzajúcich z nameraných údajov (alebo inak získaných dát), na základe ktorých je možné predpovedať ďalší vývoj v dátach. Modelovanie by sme mohli brať ako istý spôsob dôležitý pre naše rozhodovanie pre budúcnosť. Modelovanie by malo robiť abstraktnú matematiku zmysluplnejšou a relevantnejšou pre stredoškoláka. Preto budeme riešiť nasledujúce jednoduché úlohy, na ktorých vysvetlíme metódu vytvárania modelov:

- Údaje o športových rekordoch získané z rôznych zdrojov informácií (napríklad z webových stránok) budú vizualizované a budú v diskusii analyzované trendy vývoja v budúcnosti.
- Lineárne modelovanie bude použité pri sledovaní trendu predchádzajúcich dát a na základe modelu bude urobená predikcia roku, kedy sa očakáva nový rekord. Poznámka: Pomocný program v jazyku Python je pripravený. Šikovnejší žiaci budú môcť experimentovať s inými dátami sami.
- Na modelovaní pomocou polynómov ukážeme, že môže lepšie vystihnúť trend v dátach. Úloha je pripravená pre dáta získané v experimentoch pre analýzu priestorového sluchu.

PRÁCA S DATABÁZAMI

Práca s existujúcimi databázami nám umožňuje prístup k najnovším informáciám.

Vďaka digitálnym technológiám je možné v súčasnosti spracovať a uchovať obrovské množstvo rozmanitých dát aj v prírodných vedách, čo dynamicky ovplyvňuje výskum a napredovanie v tejto oblasti. Prostredníctvom internetu sme získali jednoduchý prístup k najnovším biologickým a chemickým dátam, ktoré sa stali verejne dostupnými – nie je preto problém zistiť potrebné informácie o génoch a genetických poruchách, či preskúmať chemickú štruktúru zlúčenín, ich využitie v praxi, či prípadnú toxicitu a bezpečnostné riziká. Náš zámer je naučiť žiakov:

- Navrhnúť a vytvoriť vlastnú štruktúrovanú reprezentáciu dát pomocou dátového typu slovník v programovacom jazyku Python, vkladať, modifikovať a vyhľadávať v nej informácie.
- Vyhľadávať biologické, chemické a geografické informácie v dostupných online databázach.
- Vyhľadávať na internete geografické dáta z meraní a pozorovaní a vykonať ich analýzu a vizualizáciu pomocou dostupných nástrojov.

V textovej prílohe A sa nachádzajú texty programov v jazyku Python pripravené pre učiteľa, ktoré predstavujú jedno z možných riešení k odpovedajúcemu pracovnému listu. Na uľahčenie práce k tomuto zošitu patria aj súbory s pripravenými programami v jazyku Python (7 programov pre učiteľov a 6 programov pre žiakov k pracovným listom) a súbory pracovných listov (vo worde) pre každú metodiku, teda 9 súborov s pracovnými listami.

1 PRÁCA S OBRÁZKAMI, KORYTNAČIA GRAFIKA

<table>
<tr><td>Tematický celok / Téma</td><td>ISCED / Odporúčaný ročník</td></tr>
<tr><td>Informatika/

Vykreslenie pixelov obrázku pomocou korytnačej grafiky</td><td>ISCED3/3.-4. ročník GY
Voliteľný predmet
Informatika v prírodných vedách
2 vyučovacie hodiny</td></tr>
<tr><td colspan="2">Ciele</td></tr>
<tr><td>Žiakom osvojované vedomosti a zručnosti</td><td>Žiakom rozvíjané spôsobilosti</td></tr>
<tr><td>Rovinný obrázok vykresliť na základe farebnosti použitých pixelov. Naučiť sa meniť farby pixelov, pridať pixely do obrázku, odstrániť pixely z obrázku.

Korytnačka demonštruje pixely ako štvorce predpísanej veľkosti.</td><td>Konštruovať pole pixelov a manipulovať s ním, pomocou softvéru.
Uvedomiť si možnosť zväčšovania a zmen-šovania veľkosti vykreslených pixelov.</td></tr>
<tr><td colspan="2">Požiadavky na vstupné vedomosti a zručnosti</td></tr>
<tr><td colspan="2">Základná práca s počítačom, práca so súbormi, spustenie hotových programov. Základy práce v Pythone. Základné poznatky o korytnačej grafike.</td></tr>
<tr><td colspan="2">Riešený didaktický problém</td></tr>
<tr><td colspan="2">Systematizácia a prepojenie používateľských zručností a teoretických poznatkov
s programátorskými prístupmi.</td></tr>
<tr><td>Dominantné vyučovacie metódy a formy</td><td>Príprava učiteľa a pomôcky</td></tr>
<tr><td>Bádateľská metóda</td><td>Pomôcky:
1) Inštalácia prostredia pre použitie programovacieho jazyka Python – PyCharmEdu, inštalácia programovacieho jazyka Python.
2) Obrázok, ktorý bude skúmaný v grafickom programe Paint (mtPaint);
3) Pripravený program v jazyku Python – program P1_Pixel1.py
4) Pripravený pracovný list 1 .</td></tr>
<tr><td colspan="2">Diagnostika splnenia vzdelávacích cieľov</td></tr>
<tr><td colspan="2">■ Žiacka diskusia
■ Grafický výstup modifikovaných obrázkov</td></tr>
</table>

1.1 Texty k výučbe

V každodennom živote sa stretávame s rôznymi ilustratívnymi obrázkami a fotografiami, ktoré nám pomáhajú porozumieť tomu, čo je v texte napísané. Samostatný obrázok alebo fotografia (v ďalšom budeme používať názov obrázok pre oboje) je väčšinou uložený v samostatnom súbore a kvalita obrázku závisí od metódy použitej pri jeho uložení, t. j. od grafickej metódy. V počítačovej grafike rozlišujeme dve metódy uloženia obrazu, a to diskrétne a spojito, a podľa nich aj hovoríme o rastrovej alebo vektorovej grafike.

RASTROVÁ GRAFIKA

Obrázok sa skladá z jednotlivých bodov a každý bod je uložený zvlášť pomocou svojho kódu farby a umiestnenia. Takto je v počítači uložená väčšina obrázkov (hlavne fotografie, ...). Obrázky sa nedajú zväčšovať bez straty kvality. Pri ich zväčšovaní dochádza k pokrytiu obrázka malými plôškami, k tvorbe pixelov (deleniu na štvorčeky). Súbory s rastrovými obrázkami sú veľké, a preto sa obvykle ukladajú v komprimovanom tvare, ktorý je vytvorený pomocou kompresného algoritmu. Veľkosť komprimovaného súboru závisí od **veľkosti obrázku, farebnej hĺbky, účinnosti kompresného algoritmu** a u stratovej kompresie (informácia o niektorých pixeloch je stratená) od **zvolenej kvality uloženia obrázku**.

Formáty: bmp, gif, jpg, tif, raw, png
Zdroje obrázkov: digitálny fotoaparát, skenovanie, vytvorenie obrázku v rastrovom grafickom editore (Gimp, Paint, mtPaint ...)

VEKTOROVÁ GRAFIKA

Obrázky vektorovej grafiky sa vytvárajú vo vektorových grafických editoroch (napríklad Corel Draw, modul Gfig v Gimpe, ...). Obrázky sú uložené v tvare rovníc a predpisov, ktoré popisujú jednotlivé objekty, z ktorých sa obrázok skladá (čiary, geometrické tvary, plochy ohraničené krivkami, farebne vyplnené oblasti). Dajú sa plynule zväčšovať, bez straty kvality.

Vo vektorovom formáte nie je možné uložiť fotografiu. Vektorové obrázky sa používajú na vytváranie grafických prvkov, a spolu s rastrovými obrázkami na vytváranie koláží, plagátov, atď. Písmo (jeho font) je navrhované a používané vo vektorovom formáte. Každý vektorový editor si ukladá obrázok vo svojom vlastnom formáte.

Formáty: .eps, .ai, .cdr, .svg.

POUŽITIE FARIEB

Budeme pracovať s Red-Green-Blue, t. j. RGB farbami pixelov, každá farba môže nadobúdať hodnotu z intervalu <0, 255>. Farba pixelu je vyjadrená trojicou čísel, vyjadrujúcou silu každej z uvedených troch farieb. Môžeme predpokladať, že všetky farby v počítači sú namiešané z troch základných farieb: červenej, zelenej a modrej (teda Red, Green, Blue). Farba závisí od toho, ako je v nej zastúpená každá z týchto troch farieb. Zastúpenie jednotlivej farby vyjadrujeme číslom od 0 do 255 (vyčíslenie farby sa zmestí do jedného bajtu, teda sa dá vyjadriť ako 2-ciferné číslo v šestnástkovej sústave). Napr. žltá farba vznikne, ak namiešame 255 červenej, 255 zelenej a 0 modrej. Ak budeme číselné zastúpenie každej farby trochu meniť, napríklad 250 červenej, 240 zelenej a hoci 100 modrej, stále to bude žltá, ale v inom odtieni. Tak vznikajú rôzne odtiene farieb. Uvedieme niektoré farby spolu s ich trojicou RGB hodnôt a zápisom v šestnástkovej číselnej sústave:

Názov farby	RGB	RGB v 16-kovej sústave
Sýte červená	(255, 0, 0)	"#FF0000"
Sýte zelená	(0, 255, 0)	"#00FF00"
Sýte modrá	(0, 0, 255)	"#0000FF"
Čierna	(0, 0, 0)	"#000000"
Biela	(255, 255, 255)	"#FFFFFF"
Tyrkysová	(0, 255, 255)	"#00FFFF"

Viac informácii o počítačovej grafike nájdete v (Ferko, 1995) a o hodnotách pre farby je možné nájsť na webovej stránke

http://www.klikzone.cz/sekce-html/html-barvy.php

INFORMÁCIE K PRIPRAVENÉMU PRACOVNÉMU LISTU, KORYTNAČIA GRAFIKA

Budeme pracovať s korytnačou grafikou. Korytnačka je grafické pero (grafický robot), ktoré si okrem pozície v grafickej ploche (`pos()`) pamätá aj nastavenie smeru (`heading()`)

Príkaz **`t = turtle.Turtle()`** vytvorí grafickú plochu a v jej strede korytnačku `t` s nastaveným smerom na východ.

Volanie **`t.pos()`** vráti momentálnu pozíciu korytnačky (0, 0) a `t.heading()` vráti uhol.

Príkaz **`t.bgcolor(`**`argumenty)` nastaví farbu grafickej (kresliacej) plochy, argumentom je reťazec predstavujúci farbu alebo tri čísla v rozsahu 0..colormode alebo trojica takých čísel. Nastavená hodnota colormode() je 1.

Pre grafickú plochu korytnačej grafiky platí:

- súradnicová sústava má počiatok v strede grafickej plochy
- x-ová súradnica ide vpravo vodorovne od počiatku
- y-ová súradnica ide nahor zvislo od počiatku: smerom nahor sú kladné y-ové hodnoty, smerom nadol idú záporné hodnoty súradníc
- pozícia a smer korytnačky je vizualizovaná malým trojuholníkom - keď sa bude korytnačka hýbať alebo otáčať, bude sa hýbať tento trojuholník
- nastavenie smeru určujeme v stupňoch (nie v radiánoch) a v protismere otáčania hodinových ručičiek: na východ je to 0, na sever je to 90, na západ je to 180, na juh je to 270 stupňov.

Útvary, ktoré nakreslí korytnačka sa dajú aj vyfarbiť. Farbu výplne meníme príkazom **`fillcolor(farba)`**.

Korytnačkám môžeme meniť ich tvar - momentálne je to malý trojuholník. Príkaz `shape()` zmení tvar na jeden s preddefinovaných tvarov: `'arrow', 'turtle', 'circle', 'square', 'triangle', 'classic'` (default je `'classic'`)

Príkaz `shapesize()` nastavuje zväčšenie tvaru a hrúbku obrysu tvaru: `shapesize(sirka,vyska,hrubka)`

UŽITOČNÉ PRÍKAZY KORYTNAČEJ GRAFIKY:

Tabuľka 1 ZOZNAM NIEKTORÝCH PRÍKAZOV PRE PRÁCU KPORYTNAČKY

Metóda	**Význam**	**Príklad použitia**
`forward(d)`	Choď dopredu	`t.fd(200); t.fd(-20)`
`back(u)`	Cúvaj	`t.bk(55); t.bk(-25)`
`right(u)`	Otoč sa vpravo	`t.rt(90); t.rt(-45)`
`left(u)`	Otoč sa vľavo	`t.lt(90); t.lt(-45)`
`penup()`	Zdvihni pero	`t.pu()`
`pendown()`	Spusti pero	`t.pd()`
`setpos(x,y)`	Choď na pozíciu	`t.setpos(20,80)`
`pos()`	Zisti pozíciu korytnačky	`t.pos()`
`ycor()`	Zisti y-ovú súradnicu	`t.ycor()`
`xcor()`	Zisti x-ovú súradnicu	`t.xcor()`
`heading()`	Zisti uhol korytnačky	`t.heading()`
`setheading(u)`	Nastav uhol korytnačky	`t.seth(100)`
`pensize(h)`	Nastav hrúbku pera	`t.pensize(4)`
`pencolor(f)`	Nastav farbu pera	`t.pencolor(`blue`)`
`pencolor()`	Zisti farbu pera	`t.pencolor()`
`fillcolor(f)`	Nastav farbu výplne	`t.fillcolor(`green`)`
`fillcolor()`	Zisti farbu výplne	`t.fillcolor()`
`color(f1,f2)`	Nastav farbu pera a	`t.color(`cyan`,`blue`)`
`color()`	Zisti farbu pera a výplne	`t.color()`
`reset()`	Zmaž kresbu a inicializuj	`t.reset()`
`clear()`	Zmaž kresbu	`t.clear()`
`begin_fill()`	Začiatok vyfarbovania	`t.begin_fill()`
`end_fill()`	Koniec vyfarbovania	`t.end_fill()`

`turtle.bgcolor(farba)` - zmení farbu pozadia grafickej plochy, pričom všetky kresby v ploche ostávajú bez zmeny.

ZOZNAM ZOZNAMOV

Vieme, že jazyk Python podporuje aj tzv. zložené dátové typy, ktoré slúžia na uloženie iných hodnôt. Najuniverzálnejší z nich je zoznam. Pri práci s obrázkami budeme používať práve túto dátovú štruktúru. Zoznam vytvoríme tak, že jednotlivé jeho položky postupne vymenujeme. V programe ich napíšeme do hranatých zátvoriek a položky oddelíme čiarkami. Zoznam pomenujeme nejakým identifikátorom. Napríklad, `zoz=['Adam','muž', 2019, 1234];`

Do zoznamu je možné uložiť položky rôznych typov, a teda je možné uložiť aj zoznamy. Špeciálnym tvarom je zoznam zoznamov, napríklad
`Z=[[0, 5, 6, 8], [1, 1, 1, 1], [2, 0, 3, 0], [4, 4, 4, 3]]`.

`Z[2]` potom predstavuje zoznam `[2, 0, 3, 0]` a `Z[2][3]` predstavuje hodnotu `0`.

Viac informácií k programovaciemu jazyku Python nájdete v (Blaho, 2016), (Belan, 2013), (Švec, 2002).
O práci s pixelami existuje informácia v (CoBaLa, 2018).

1.2 Priebeh výučby

Učiteľ môže začať výučbu motiváciou skúmania odtlačkov prstov, kde sa ide do detailov a hľadajú sa určité tvary v odtlačkoch. Je dobre odtlačky prstov zväčšiť a tvary sledovať. Iná motivácia môže pochádzať z medicíny, keď sa sleduje a modeluje rast nádorov (pridávajú sa ku skupine buniek ďalšie bunky.

Učiteľ oboznámi žiakov s rastrovou a vektorovou grafikou a pripravený obrázok odtlačku je možné skúmať v programe Paint (mtPaint). Program Paint je v základnom softvérovom vybavení (žiaci ho poznajú) a poslúži na zobrazenie obrázkov, ich zväčšovanie a zmenšovanie, aby si žiaci uvedomili pixely v obrázku, ktoré pri dostatočnom zmenšení neuvidia. mtPaint je voľne dostupný a je veľmi podobný programu Paint, avšak má viacej možnosti pre prácu s obrázkami.

K pracovnému listu je potrebné vysvetliť dátovú štruktúru `zoznam zoznamov` a použitie farieb pri kreslení v korytnačej grafike.

ZAPOJENIE (CCA 10 MIN.)

CIEĽ

- oboznámiť žiakov s metódami uloženia grafických súborov podľa uvedeného textu;
- jednoduché spracovanie a úprava obrázkov modifikovaním pripraveného programu v programovacom jazyku Python;

Úloha s bádateľským prístupom:

Obrázok 1. ODTLAČKY PRSTA

Obrázok 1 je možné nájsť na stránke: Credit line: **© Marcos2006, http://www.stockphotos.sk/image.php?img_id=6154275&img_type=1**

Na priloženom obrázku 1 sú dva odtlačky prstov a my budeme skúmať raster obrázku v rastrovej grafike. Vľavo je čiernobiely obrázok, vpravo je umelecký obrázok vo farbách dúhy. Je možné použiť program Paint (mtPaint) na zobrazenie týchto odtlačkov. Odtlačky budeme zväčšovať percentuálne, pripadne zväčšovať a zmenšovať počet pixelov. Keď odtlačok dostatočne zväčšíme, uvidíme, že sa skladá zo štvorčekov. Jeden štvorček reprezentuje jeden pixel obrázku.

OTÁZKA DO DISKUSIE

Obrázok predstavuje časť odtlačku prsta. Kde sa stretávame s takými obrázkami? Ako sa zisťuje, že daný odtlačok patrí danej osobe?

Nechajte žiakom možnosť vyjadriť svoj názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

Metodická poznámka:
O daktyloskopii nájdete viac na: https://magazin.centrum.sk/spektrum/co-ste-nevedeli-o-odtlackoch-prstov/748929.html

SKÚMANIE (CCA 15 MIN.):

Zámer: Preskúmať, čo by sa dalo zistiť a urobiť pri skúmaní odtlačkov programe Paint (mtPaint).

Program Paint (mtPaint) je pomôckou k tomu, aby sme ukázali, že pixely obrázku je možné zobrazovať rôzne veľké.

VÝSKUMNÉ OTÁZKY 1

- Ako by sme vedeli pozrieť detailnejšie tvary tmavých a svetlých (farebných) kriviek na vyššie uvedenom obrázku?
- Je možné zistiť koľko tmavých a svetlých pixelov je v danom obrázku, prípadne koľko pixelov má červenú farbu vo farebnom obrázku? Vieme to urobiť pomocou programu Paint (mtPaint)?
- Je možné nejakú časť obrázka zmeniť na svetlú farbu? Vieme to urobiť pomocou programu Paint (mtPaint)?
- Keby sme mali obrázok v takom digitálnom tvare, že by sme presne vedeli identifikovať riadok a stĺpec pixelu, bolo by to jednoduchšie?
- Ak obrázok vhodne otvoríme v programe v Pythone, budeme vedieť pracovať s pixelmi lepšie?

Poznámka:

Nech žiaci opíšu vlastnými slovami, ako by postupovali pri riešení vyššie uvedených úloh. Otázkami podporujte diskusiu

VYSVETLENIE (CCA 10 MIN.):

Zámer: Formou dialógu vysvetlíme, ako by sme mohli robiť zásahy do obrázku. Predpokladáme, že obrázok je uložený v dvojrozmernom poli, v ktorom každé políčko predstavuje pixel. Pixel je charakterizovaný tromi hodnotami farieb RGB.

CVIČENIE – ANALYZUJTE!

- Ako by sme vedeli meniť farby pixelov, ako miešať farby?
- Ako by sme zistili číselné hodnoty RGB farieb?

VYKRESLENIE JEDNOFAREBNÉHO OBRÁZKA POMOCOU POZÍCIÍ OZNAČENÝCH PIXELOV (CCA 30 MIN.):

K tejto časti je pripravený pracovný list 1.

Inšpirácia a tvar funkcie `kresli` je z webovej stránky Pixel Art - http://www.101computing.net/pixel-art-in-python/

ÚLOHA 1

- Spustiť používané prostredie (pyCharm) pre programovací jazyk Python. Pripraviť prácu s korytnačou grafikou.
- Vytvoriť funkciu **`stvorcek`**, ktorá vykresľuje štvorček vyplnený určenou farbou. Ako sa dá meniť veľkosť štvorčeka?
- Pripraviť zoznam zoznamov s označenými pixelmi. Neoznačeným pixelom priradíme hodnotu 0 a označeným hodnotu 1. Meno zoznamu si možno zvoliť, v texte a v pripravenom programe to je `pixels_obrazok`.
- Vytvoriť funkciu **`kresli`**, v ktorej korytnačka vykresľuje obrázok daný pozíciou označených pixelov a určiť, aké parametre potrebuje.

Poznámky:

- V tejto časti niektorí žiaci môžu mať problém s vytváraním funkcie, preto kontrolujeme ich postup. Je dôležité, aby žiaci pochopili, ako funkcie pracujú.
- Nechajte žiakom čas na prácu podľa pracovného listu. Môžu pracovať aj vo dvojiciach.

ÚLOHA 2

- Zmeniť hodnoty farieb vo funkcii `stvorcek`.
- Zaviesť nejaké podmienky pre zmenu farieb pixelov. Pokračujeme v práci podľa pracovného listu.

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

ROZŠÍRENIE (CCA 15 MIN.):

CIEĽ

Naučiť žiakov vytvárať farebné obrázky na základe určených čísel v šestnástkovej číselnej sústave použitých pre vyjadrenie farieb.

ÚLOHA 3

Zaviesť nejaké podmienky pre zmenu farieb pixelov. Vykresliť pripravený zoznam zoznamov tak, aby každý pixel bol zafarbený inou farbou. Pokračujeme v práci podľa pracovného listu.

ÚLOHA 4

- Vytvoriť zoznam v zozname so zafarbenými pixelmi a vykresliť ho. Na zoznam v zozname je možné klásť rôzne požiadavky (symetria, umelecký dojem, postavička, atď.)
- Vytvoriť funkciu `kresliF`, ktorá vykreslí obrázok so zadanými farbami pixelov.

Nechajte žiakom čas na prácu na programe.

Veďte so žiakmi diskusiu, v ktorej si uvedomia význam ovládania práce s pixelmi.

HODNOTENIE A SEBAHODNOTENIE (CCA 10 MIN.):

Zámer: Žiaci majú hodnotiť, ako a čím im pomohol skúmaný problém. Hlavne, či budú vedieť samostatne postupovať pri riešení podobných úloh.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

1.3 Pracovný list 1

Vykreslenie obrázku pomocou korytnačej grafiky

Pracovný list obsahuje úlohy, ktoré je potrebné urobiť písomne, ale tiež naprogramovať v programovacom jazyku Python. Preto na začiatku je potrebné otvoriť prostredie, v ktorom budete pracovať, pripraviť projekt (nazvime ho P1), v ktorom budete vytvárať 4 programy pre prácu s grafikou a nakoniec vytvoriť prvý program - súbor s vybraným menom a koncovkou .py. Nový prvý program v rámci projektu P1 nazvite **`P1_Pixel1.py`**

Pomôcka: Na konci pracovného listu je uvedených niekoľko príkazov pre prácu s korytnačkou.

1. Ak chceme pracovať s korytnačou grafikou programe vytváranom v jazyku Python, potrebujeme použiť príkaz

```
import ......
```

Doplňte písomne a príkaz použite v programe.

2. Príprava grafického okna na kreslenie v programe – okno pomenujte okno, nastavte jeho farbu pozadia a ku grafickému oknu priraďte titulok (tento návrh príkazov je možné prebrať)

```
okno = turtle.Screen()
okno.bgcolor("yellow")
okno.title("Vykreslenie obsahu 0/1 pixelového poľa")
```

Príprava pera, rýchlosti a farby (v tvare čísla v šestnástkovej sústave) pera korytnačky

```
myPen = turtle.Turtle()
myPen.speed(0)
myPen.color("#000000")
```

Na konci programu použite príkaz

```
okno.exitonclick()
```

ktorý zabezpečí, že grafické okno zostane viditeľné do kliknutia myšou.

3. Pripravte v programe funkciu `stvorcek`, pomocou ktorej korytnačka vykreslí vyplnený štvorček nejakou farbou. Jedna z možností:

```
def stvorcek(fvelkost):
    myPen.begin_fill()
    # 0 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 90 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 180 deg.
    myPen.forward(fvelkost)
```

```
    myPen.left(90)
    # 270 deg.
    myPen.forward(fvelkost)
    myPen.end_fill()
    myPen.setheading(0)
```

Funkciu vyskúšajte spustením s konkrétnym parametrom pre veľkosť štvorčeka.
Napíšte vedľa príkazov, čo postupne urobí uvedená funkcia, ak `fvelkost` sa bude rovnať 7. Napíšte, čo robí funkcia `myPen.setheading()`.

4. Zapíšte vyššie uvedenú funkciu pomocou cyklu. Vidíme, že sa vo funkcii niektoré časti opakujú.

5. Nastavte korytnačku približne do stredu ľavého horného kvadrantu grafického okna (pri rozdelení grafického okna na štyri rovnaké kvadranty) a zavolaním funkcie vykreslite jeden vyplnený štvorček.
Experimentujte s veľkosťou štvorčeka. Zvoľte hodnotu 5, 10, ...
Popíšte svoje experimentovanie.

6. V programe zavolajte funkciu `stvorcek` 4 krát. Zapíšte, čo sa udialo. Kde sú vykreslené štvorčeky?

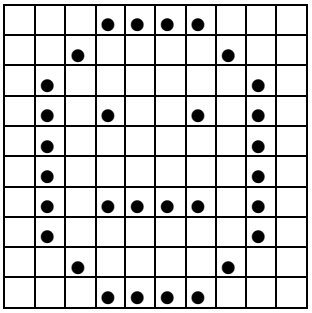
7. Pripravte v programe zoznam
`pixels_riadok=[1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,1]`

8. Zavolajte funkciu `stvorcek` pre výpis všetkých prvkov tohto zoznamu do jedného riadku. Použite cyklus

```
velkost=10
for j in range(0, len(......)):
    if pixels_riadok[j] == 1:
        myPen.color("#00FF00")
        stvorcek(velkost)
    myPen.penup()
    myPen.forward(velkost)
    myPen.pendown()
```

Napíšte, čo je potrebné uviesť namiesto bodiek vo funkcii `len(......)`?
Napíšte, čo zisťuje príkaz `if`? Čo sa stane, keď platí `pixels1[j] == 1`?

9.



Okienka s bodkou nech predstavujú označené pixely, s priradenou hodnotou 1, neoznačené majú hodnotu 0. Už vieme vytvoriť zoznam pre každý riadok. Zoznam pre prvé dva riadky vyzerá takto:
`[[0,0,0,1,1,1,1,0,0,0],[0,0,1,0,0,0,0,1,0,0]]`.
Zapíšte zoznam pre posledné tri riadky.
K danému obrázku vytvorte v programe zoznam zoznamov `pixels_obrazok` (po riadkoch).

10.

Vytvorte v programe funkciu `kresli`, v ktorej korytnačka vykreslí tento obrázok približne do stredu grafického okna. Funkcia bude použiteľná pre kreslenie ľubovoľného obrázku zapísaného pomocou zoznamu zoznamov z núl a jednotiek.
Je zrejmé, že je možné použiť cyklus z úlohy 8. Tento cyklus je potrebné vnoriť do cyklu pre riadky. Tučným písmom je vyznačené vykreslenie jedného riadku, ktoré už poznáme

```
def kresli(pixs, fvelkost):
    farba="#00FF00"
    for i in range(0, len(pixs)):
        for j in range(0, len(pixs[i])):
            if pixs[i][j] == 1:
                myPen.color(farba)
                stvorcek(fvelkost)
            myPen.penup()
            myPen.forward(fvelkost)
            myPen.pendown()
        myPen.setheading(270)
        myPen.penup()
        myPen.forward(fvelkost)
        myPen.setheading(180)
        myPen.forward(fvelkost * len(pixs[i]))
        myPen.setheading(0)
        myPen.pendown()
```

Napíšte, čo vykonávajú príkazy, ktoré sú v cykle pre i a nie sú zapísané tučným písmom.

11.	Napíšte, čo sa stane, keď vo funkcii kresli zmeníme farbu `"#00FF00" na "#000000"`? Experimentujte s inými farbami vyjadrenými v šestnástkovej číselnej sústave.
12.	Ako by sme mohli modifikovať funkciu `kresli`, aby sa pixely vykresľovali rôznymi farbami? Napíšte, kde by bolo potrebné meniť farbu vykresľovaného pixelu?
	Záverečné hodnotenie a sebahodnotenie
O1	Napíšte, čo je to pixel? Ako ho môžem znázorniť/vykresliť v programe?
	Napíšte ako by sme mohli vyjadriť ružovú farbu zápisom šestnástkovej číselnej sústave?
O2	Viem meniť veľkosť štvorčeka reprezentujúceho pixel. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem
	Viem zmeniť farbu pixelu priradením inej farby vyjadrenej v šestnástkovej číselnej sústave. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem

Užitočné príkazy pre prácu s korytnačkou (súčasť pracovného listu):

Metóda	Význam	Príklad použitia
`forward(d)`	Choď dopredu	`t.fd(200); t.fd(-20)`
`back(u)`	Cúvaj	`t.bk(55); t.bk(-25)`
`right(u)`	Otoč sa vpravo	`t.rt(90); t.rt(-45)`
`left(u)`	Otoč sa vľavo	`t.lt(90); t.lt(-45)`
`penup()`	Zdvihni pero	`t.pu()`
`pendown()`	Spusti pero	`t.pd()`
`setpos(x,y)`	Choď na pozíciu	`t.setpos(20,80)`
`pos()`	Zisti pozíciu korytnačky	`t.pos()`
`ycor()`	Zisti y-ovú súradnicu	`t.ycor()`
`xcor()`	Zisti x-ovú súradnicu	`t.xcor()`
`heading()`	Zisti uhol korytnačky	`t.heading()`
`setheading(u)`	Nastav uhol korytnačky	`t.seth(100)`
`pensize(h)`	Nastav hrúbku pera	`t.pensize(4)`
`pencolor(f)`	Nastav farbu pera	`t.pencolor(`blue`)`
`pencolor()`	Zisti farbu pera	`t.pencolor()`
`fillcolor(f)`	Nastav farbu výplne	`t.fillcolor(`green`)`
`fillcolor()`	Zisti farbu výplne	`t.fillcolor()`
`color(f1,f2)`	Nastav farbu pera a	`t.color(`cyan`,`blue`)`
`color()`	Zisti farbu pera a výplne	`t.color()`
`reset()`	Zmaž kresbu	`t.reset()`
`clear()`	Zmaž kresbu	`t.clear()`
`begin_fill()`	Začiatok vyfarbovania	`t.begin_fill()`
`end_fill()`	Koniec vyfarbovania	`t.end_fill()`

2 PRÁCA S OBRÁZKAMI, SPRACOVANIE REÁLNEHO OBRÁZKA

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Spracovanie reálneho obrázka, úprava farieb pixelov	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	*Žiakom rozvíjané spôsobilosti*
Rovinný obrázok vyhodnotiť na základe farebnosti použitých pixelov. Naučiť sa meniť farby pixelov v reálnych obrázkoch vytvorených v inom prostredí.	Konštruovať model a manipulovať s ním pomocou softvéru. Vysvetľovať a upravovať modelovacie postupy Porovnať dáta získané z modelu s reálnymi dátami. Aplikovať experimentálne postupy na nové problémy.
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca so súbormi, spustenie hotových programov. Základy práce v Pythone. Základné poznatky o rastrovej a o vektorovej grafike.	
Riešený didaktický problém	
Systematizácia a prepojenie používateľských zručností a teoretických poznatkov s programátorskými prístupmi.	
Dominantné vyučovacie metódy a formy	*Príprava učiteľa a pomôcky*
Bádateľská metóda	Pomôcky: 1) Pripravený program v jazyku Python –program `P1_Pixel2_ucitel.py` 2) Grafický program Paint (mtPaint) 3) Pracovný list 2 a súbor obsahujúci kostru programu pre žiakov `P1_Pixel2.py`
Diagnostika splnenia vzdelávacích cieľov	
■ Žiacka diskusia ■ Grafický výstup modifikovaných obrázkov	

2.1 Texty k výučbe

V počítačovej grafike rozlišujeme dve metódy uloženia obrazu, a to diskrétne a spojito, a podľa nich aj hovoríme o rastrovej alebo vektorovej grafike. Budeme sa venovať rastrovej grafike, a síce dozvieme sa viac o tom, ako prebieha rasterizácia obrázku a v obrázku, ktorý máme k dispozícii budeme meniť farbu niektorých pixelov.

RASTERIZÁCIA OBRAZU, T. J. VYTVÁRANIE RASTRA OBRÁZKU

Základné grafické prvky sú úsečky, kružnice, krivky, oblasti a textového reťazca. Reálny obraz je možné charakterizovať nejakou spojitou funkciou (napr. jasu). Obraz v počítači je však diskrétny (digitálny). Obraz, ktorý je snímaný, má (teoreticky) nekonečný rozsah obrazových hodnôt. Bude však zobrazený konečným počtom pixelov v konečnom počte farieb, jeden pixel má jednu farbu. Prevod vektorového obrazu na rastrový tvar sa nazýva rasterizácia obrazu. **Pixel** je vlastne bod rastrového obrazu. Rasterizácia obrazu prebieha v dvoch krokoch: kvantovanie a vzorkovanie. Podstatou kvantovania je diskretizácia hodnôt obrazovej funkcie. Vzorkovaním rozumieme zaznamenávanie hodnôt - vzoriek, vo vopred daných intervaloch. Výsledkom rasterizácie je obrázok uložený v rastrovej grafike, teda sú to body určenej farby (pixely).

Je potrebné ešte zdôrazniť, že pojem **pixel** má odlišný význam v prípade, keď hovoríme o hardvéri a iný, keď hovoríme o softvéri:

1) V prípade hardvéru sa jedná o skutočný fyzický prvok zobrazovacích zariadení (monitory, displeje smartfóny, atď.). Väčšinou sa skladá z troch svietiacich bodov (červený, zelený, modrý), ktorých kombináciou vzniká príslušná farba. Počet pixelov na ploche zariadenia určuje, koľko bodov je zariadenie schopné zobraziť maximálne, teda jeho rozlíšenie. Väčšinou sa zapisuje ako počet stĺpcov a riadkov, v ktorých sa pixely nachádzajú. Napríklad Full HD monitor má rozlíšenie 1920x1080, teda 1920 stĺpcov a 1080 riadkov pixelov. Fyzická veľkosť pixelu závisí od veľkosti plochy zobrazovacieho zariadenia a rozlíšenia. Pri súčasných monitoroch sa väčšinou pohybuje okolo 0.3 mm.
2) V prípade softvéru a teda spracovania a zobrazenia obrázkov sa taktiež určuje rozlíšenie, tentoraz ale konkrétneho obrázku. Zapisuje sa tiež ako počet stĺpcov a riadkov. Rozdiely sú ale v tom, že pixel sa už ďalej nedelí na menšie body, ktorých kombináciou vzniká farba, ale farba sa zapisuje priamo do informácie pixelu. Takýto pixel je bezrozmerný, nemá žiadnu veľkosť vyjadriteľnú v milimetroch alebo iných fyzických jednotkách. Obraz je možné priblížiť a oddialiť, a teda je možné zobraziť jeden pixel skoro na celú plochu monitora a takisto aj celý obrázok s miliónmi pixelov naraz.

PROGRAMY PRE PRÁCU S GRAFIKOU

1) Prehliadače obrázkov – slúžia na zobrazovanie obrázkov rôznych grafických formátov; dá sa v nich urobiť zmena veľkosti, formátu, orientácie obrázku, výrez obrázku, zmena farebnej hĺbky, prevod do stupňov šedi, zmena jasu a kontrastu (napríklad IrfanView)
2) Editory:
   - Rastrové (poznáme už Paint, mtPaint)
   - Vektorové (napríklad Gimp)

ČÍSLA V ŠESTNÁSTKOVEJ ČÍSELNEJ SÚSTAVE

Na vyjadrenie farebnosti obrázka sa používajú čísla vyjadrené v šestnástkovej číselnej sústave. Šestnástková číselná sústava (v ďalšom budeme hovoriť skrátene šestnástková sústava) pracuje s číslicami 0-9 a písmenami A-F. A predstavuje 10, B -11, C-12, D-13, E-14, F-15. Do jedného bytu je možné uložiť najväčšie číslo FF v šestnástkovej sústave, čo je 11111111 v dvojkovej sústave a 255 v desiatkovej číselnej sústave. Pomocou troch bytov sa dá vyjadriť zastúpenie troch farieb RGB v namiešanej farbe. Budeme experimentovať s miešaním farieb pre pixely, ktoré bude vykresľovať korytnačka a budeme tiež meniť veľkosť štvorčeka reprezentujúceho pixel.

Zápis `"#7A05FE"` predstavuje hodnotu

- 7A pre červenú farbu, čo v desiatkovej sústave predstavuje číslo 82,
- 05 pre zelenú farbu, čo je v desiatkovej sústave tiež 5,
- FE pre modrú farbu, čo je v desiatkovej sústave 254.

Funkcia `hex()` predstavuje prevod celého čísla do reťazca, ktorý reprezentuje toto číslo v šestnástkovej sústave, avšak číslo môže mať aj menší počet znakov ako 6, čo je potrebný počet pre vyjadrenie 3 farieb RGB.

Prevod čísla uloženého vo `farba1` v desiatkovej sústave do reťazca predstavujúceho farbu je možné urobiť takto:

```
farba='#'+'{:06x}'.format(farba1)
```

Formátovací reťazec `'{:06x}'` obsahuje písmeno x, ktoré vyjadruje, že ide o číslo v šestnástkovej sústave, 6 vyjadruje počet miest pre číslice a 0, čo znamená, že zľava majú byť doplnené nuly.

FORMÁTOVANIE REŤAZCOV PRE ČÍSLA V ŠESTNÁSTKOVEJ SÚSTAVE

Formátovanie reťazcov sa dá využiť aj na výpis celých čísel v šestnástkovej sústave. Čísla budeme potrebovať pri stanovení farieb pixelov. Formát '{:5}' označuje, že na príslušnú hodnotu je rezervovaných 5 miest. Za dvojbodku v zátvorkách '{}' okrem šírky môžeme písať aj typ výpisu. Tu sa nám hodí typ výpisu 'x', ktorý označuje výpis v šestnástkovej sústave, lebo budeme potrebovať zapisovať farby v šestnástkovej sústave, napr.

```
>>> '{:x}'.format(121)
'79'
>>> '{:x}'.format(12122)
'2f5a'
'{:8x}'.format(12122)
>>>'    2f5a'
```

2.2 Priebeh výučby

VYSVETLENIE (CCA 10 MIN.):

Učiteľ vysvetlí v čom spočíva rasterizácia obrázku, ktorá umožní prístup k najjednoduchším objektom obrázkov – pixelom a venuje pozornosť prevodom čísel z desiatkovej číselnej sústavy do šestnástkovej sústavy a opačne. Prevody sú dôležité pre miešanie farieb. Po vysvetlení žiaci môžu pracovať s pracovným listom 2, ktorého cieľom je vytvorenie vlastného programu (vzorový program je v súbore P1_Pixel1.py).

ZAPOJENIE (CCA 25 MIN.):

CIEĽ

- oboznámiť žiakov s metódami uloženia grafických súborov podľa uvedeného textu;
- jednoduché spracovanie a úprava obrázkov pomocou vytváraného programu v programovacom jazyku Python

Úloha s bádateľským prístupom:

Obrázok 2. ODTLAČKY PRSTOV S FAREBNÝM ROZLÍŠENÍM PIXELOV

Na priloženom obrázku 1 odtlačkov prsta v rastrovej grafike sme už skúmali raster obrázku. Je možné použiť program Paint (mtPaint). Obrázok sme zväčšovali percentuálne, pripadne zväčšovali sme a zmenšovali počet pixelov. Z farebného obrázku v strede urobíme transformáciu na približne 15x20 pixelov, tak ako to je na treťom obrázku, kde sú viditeľné pixely. Tento obrázok uložíme do nášho pracovného adresára pod názvom *obrPixely.jpg.* Tento obrázok budeme používať pri práci s pripraveným pracovným listom 2.

Žiaci môžu vytvoriť vlastný obrázok svojho prsta (fotografia z mobilu a z nej urobiť výrez) a porovnať ho s daným obrázkom.

CVIČENIE – ANALYZUJTE!

Je možné pomocou programu Paint (mtPaint) získať odpovede na nasledujúce otázky:

- Je možné zistiť koľko tmavých a svetlých pixelov je v danom obrázku? Vieme to urobiť pomocou programu Paint (mtPaint)?
- Je možné nejakú časť obrázka zmeniť na svetlú farbu? Vieme to urobiť pomocou programu Paint (mtPaint)?
- Vieme zmeniť farbu ľubovoľného pixelu na inú farbu pomocou programu Paint (mtPaint)?

Preto hľadáme riešenie pomocou Pythonu.

OTÁZKA DO DISKUSIE

Keby sme mali obrázok v takom digitálnom tvare, že by sme presne vedeli identifikovať riadok a stĺpec pixelu, bolo by to jednoduchšie?

RGBA je anglická skratka pre farebný model pozostávajúci z červenej, zelenej a modrej farby (angl. **R**ed, **G**reen, **B**lue) a **A**lfa kanálu (priehľadnosti, priesvitnosti). Napríklad #225599ff

Čísla farieb sú uvedené v šestnástkovej sústave červená farba (22), zelená farba (55) a modrá farba (99). Posledný údaj udáva hodnotu priesvitnosti. Tento konkrétny zápis patrí tmavomodrej farbe s nulovou priesvitnosťou.

ÚLOHA 1 – RIEŠTE!

Pomocou programu Paint (mtPaint) pripraviť obrázok na prácu s pixelmi.
K úlohe je pripravený pracovný list 2, úloha 1.

SKÚMANIE A OBJAVOVANIE INFORMÁCIÍ O PIXELOCH (CCA 30 MIN.):

Zámer: Priviesť žiakov k tomu, aby pochopili, že je možné ovládať farbu a priesvitnosť každého pixelu. Nech žiaci opíšu vlastnými slovami, ako by postupovali pri riešení vyššie uvedených úloh. Otázkami podporujte diskusiu.

Formou dialógu vysvetlíme, ako by sme mohli robiť zásahy do obrázku, ktorý je vykreslený pomocou programu. Obrázok je vlastne uložený v dvojrozmernom poli, v ktorom každé políčko predstavuje pixel. Pixel je charakterizovaný tromi hodnotami farieb RGB. Niektoré formáty poskytujú aj zložku A, čo je priesvitnosť. Je potrebné vysvetliť alebo zopakovať prevody z desiatkovej do šestnástkovej sústavy a opačne.

ÚLOHA 2 – RIEŠTE!

- Importovať obrázok vytvorený v programe Paint (mtPaint) do programu v jazyku Python. Zobraziť tento obrázok.
- Vypísať informácie o jednotlivých pixeloch v obrázku predstavujúce číselné hodnoty o farbách.
- Úloha je pripravená v pracovnom liste 2.

Poznámky:

- V tejto časti niektorí žiaci môžu mať problém s prípravou obrázkov na spracovanie v Pythone.
- Nechajte žiakom čas na prácu na programe. Môžu pracovať aj vo dvojiciach.

CVIČENIE – OTÁZKY!

Ako by sme vedeli meniť farby pixelov? Vedeli by sme miešať farby?

ÚLOHA 3 – RIEŠTE!

Miešanie farieb, ak sú dané tri čísla v desiatkovej sústave v intervale <0, 255> predstavujúce červenú, zelenú a modrú farbu. Vytvoriť funkciu na miešanie farieb.

Úloha je pripravená v pracovnom liste 2. K pracovnému listu je pripravený program pre žiakov **`P1_Pixel2.py`**, do ktorého budú doplňovať a je tiež pripravený úplný program **`P1_Pixel2_ucitel.py`**, pre učiteľa.

ROZŠÍRENIE (CCA 15 MIN.):

CIEĽ:

Naučiť žiakov vytvárať zložitejšie modifikácie obrázkov.

ÚLOHA 4 – RIEŠTE!

- Analyzovať hodnoty farieb RGB, meniť ich a pozrieť, ako sa mení obrázok.
- Zaviesť nejaké podmienky pre zmenu farieb pixelov. Napríklad, ak je pixel modrý, zafarbiť ho červenou farbou.

Poznámky:

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

Podobne môžeme modelovať aj vzťah medzi živočíchmi v potravnom reťazci.

Tieto úlohy vedú k biologicky motivovaným úlohám o raste nádorov.

ÚLOHA 5 – RIEŠTE!

- Zameniť v obrázku bielu farbu za čiernu.
- Do stredu obrázka vložiť čiernu škvrnu, t. j. 2-4 čierne pixely.

HODNOTENIE (CCA 10 MIN.):

Zistiť, ako žiaci ovládajú prístup k jednotlivým pixelom v obrázku a či dokážu urobiť ich farebnú úpravu.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

Veďte so žiakmi diskusiu, v ktorej si uvedomia význam ovládania práce s pixelmi.

2.3 Pracovný list 2

Spracovanie reálneho obrázka, úprava pixelov

Pracovný list obsahuje úlohy, ktoré je potrebné urobiť písomne, ale tiež naprogramovať v programovacom jazyku Python. Preto na začiatku je potrebné otvoriť prostredie, v ktorom budete pracovať a otvoriť pripravený program - súbor `P1_Pixel2.py`. Tento program dostanete spolu s pracovným listom a doplňujete doň príkazy podľa pracovného listu.

K práci potrebujete aj súbor **`obrPixely1.png`**

1.	Príprava obrázku na spracovanie. V programe Paint/mtPaint otvoriť obrázok Odtlacky.png. Urobte v obrázku výrez približne 15x20 pixelov, výrez uložte pod novým menom obrPixely1.png do adresára, kde sú uložené .py súbory daného projektu (P1, na ktorom pracujete).
2.	Ak chcete pracovať s korytnačou grafikou, potrebujete použiť v programe príkaz - doplňte `import ......` Príkaz je vložený v programe.
3.	Budete pracovať s obrázkami, preto v programe použijete knižnicu PIL. Knižnicu importujeme pomocou príkazu `from PIL import Image`

4.	Zaveďte v programe premennú pre meno súboru s obrázkom, s ktorým budete pracovať a vypíšte toto meno v konzole
	`meno = 'obrPixely1.png'`
	`print(meno)`

5.	Otvorte obrázok a v programe sa naň budeme odvolávať premennou `img`
	`img = Image.open(meno)`
	
	V konzole vypíšte veľkosť obrázka, vypíše sa počet riadkov a počet stĺpcov pixelov
	`print("Velkost obrazka: ", img.size)`

6.	Obrázok je možné zobraziť pomocou programu Paint/mtPaint príkazom `img.show()`
7.	Budete pracovať s pixelami, preto je potrebné v nejakej premennej v programe uložiť tento tvar obrázka. Nazvime túto premennú `pxl_mapa` `pxl_mapa = img.load()`

8. Zaujímajú nás informácie o jednotlivých pixeloch. V konzole vypíšte informácie o pixeli v 6-riadku a 8-om stĺpci

```
print('R=',pxl_mapa[5,7][0])
print('G=',pxl_mapa[5,7][1])
print('B=',pxl_mapa[5,7][2])
print('A=',pxl_mapa[5,7][3])
```

Napíšte výsledky zápisu z konzoly vedľa príkazov.

9. V programe vytvorte cyklus, ktorý v konzole vypíše informácie o pixeloch v 3-ťom riadku obrázku. Porovnajte tieto informácie.

10. Napíšte, ako by ste zistili, či dva pixely majú rovnakú farbu?

11. Farby v pixelovej mape `pxl_mapa` sú vyjadrené ako tri čísla v desiatkovej sústave. Korytnačka potrebuje farbu ako číslo v šestnástkovej číselnej sústave. Preto v programe vytvorte funkciu, ktorá namieša farbu pre korytnačku z troch čísel v desiatkovej sústave. Pomocou príkazu `return` funkcia vráti vypočítaný výsledok, v našom prípade farbu vypočítanú v premennej `farba`.

```
def namiesajFarbu(fR,fG,fB):
    farbaR=256*256*fR
    farbaG=256*fG
    farbaB=fB
    farba='#'+'{:06x}'.format(farbaR+farbaG+farbaB)
    return farba
```

12. Príprava grafického okna na kreslenie – okno pomenujte okno, nastavte jeho farbu pozadia a ku grafickému oknu priraďte titulok, pripravte pero a nastavte rýchlosť kreslenia. Príkazy sú v programe pripravené.

```
okno = turtle.Screen()
```

```
okno.bgcolor("yellow")
okno.title("Práca s reálnym obrázkom")
```

```
myPen = turtle.Turtle()
myPen.speed(0)
```

Nastavte a v konzole vypíšte veľkosť štvorčeka pre zobrazovanie pixelu

```
velkost=10
print("Velkost stvorceka pre pixel je", velkost)
```

Na konci programu použite príkaz

```
okno.exitonclick()
```

	ktorý zabezpečí, že grafické okno, ktoré budete používať, zostane viditeľné do kliknutia myšou.

13.	Z predchádzajúceho programu P1_Pixel1.py použite funkciu `stvorcek`,
	`def stvorcek(fvelkost):`
	`  myPen.begin_fill()`
	`  # 0 deg.`
	`  myPen.forward(fvelkost)`
	`  myPen.left(90)`
	`  # 90 deg.`
	`  myPen.forward(fvelkost)`
	`  myPen.left(90)`
	`  # 180 deg.`
	`  myPen.forward(fvelkost)`
	`  myPen.left(90)`
	`  # 270 deg.`
	`  myPen.forward(fvelkost)`
	`  myPen.end_fill()`
	`  myPen.setheading(0)`

14	Namiešajte nejakú farbu pomocou funkcie `namiesajFarbu` a pomocou korytnačky vykreslíte štvorček predstavujúci pixel tejto farby. Doplňte chýbajúce parametre
	`farbaPix=namiesajFarbu(...,...,...)`
	`myPen.color(farbaPix)`
	`stvorcek(velkost)`

15	Na vykreslenie obrázku vytvorte v programe funkciu `kresli2`, ktorá je veľmi podobná funkcii `kresli` (v programe P1_Pixel1.py), má o dva parametre `pocRiadkov`, `pocStlpcov` viac a mieša farbu z číselných hodnôt pixelov.
	
	`def kresli2(pixs, fvelkost, pocRiadkov, pocStlpcov):`
	`  for i in range(0, pocStlpcov-1):`
	`  for j in range(0, pocRiadkov-1):`
	`  farba1 = namiesajFarbu(pixs[j, i][0], pixs[j, i][1], pixs[j, i][1])`
	`  myPen.color(farba1)`
	`  stvorcek(fvelkost)`
	`  myPen.penup()`
	`  myPen.forward(fvelkost)`
	`  myPen.pendown()`
	`  myPen.setheading(270)`
	`  myPen.penup()`
	`  myPen.forward(fvelkost)`
	`  myPen.setheading(180)`
	`  myPen.forward(fvelkost * (pocRiadkov-1))`

	`myPen.setheading(0)` `myPen.pendown()`
16	Vykreslite obrázok pomocou korytnačky približne do stredu grafickej plochy.
17	Nahraďte pixely v treťom riadku červenou farbou a obrázok znovu vykreslite pod predchádzajúci obrázok.
18	Približne do stredu druhého obrázka vložte čiernu škvrnu, t. j. 2-4 čierne pixely.
	Záverečné hodnotenie a sebahodnotenie
01	Sú dané tri čísla v desiatkovej číselnej sústave 160, 25, 200 predstavujúce farby v poradí R, G, B. Aká bude hodnota farby v šestnástkovej číselnej sústave?
	Je daná farba #9235990f v šestnástkovej číselnej sústave. Aké sú hodnoty R, G, B farieb v desiatkovej sústave?
02	Viem, aké informácie sú uložené o každom pixeli pri RGBA formáte. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem
	Viem, ako previesť 3 desiatkové čísla do farby vyjadrenej v šestnástkovej sústave. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem
03	Viem, ako je možné zmeniť hodnotu vybraného pixelu. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem

3 PRÁCA S OBRÁZKAMI, GENEROVANIE NÁHODNÝCH LOŽÍSK

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Modifikácia rovinného obrázku – generovanie infikovaných ložísk	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	*Žiakom rozvíjané spôsobilosti*
Zručnosti pri modifikácii rastrových obrázkov pomocou určených podmienok. Prispieť k vedomostiam v príprave na maturitu.	Programovať v jazyku Python, Zdokonaľovanie práce s obrázkami
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca so súbormi, základné príkazy jazyka Python.	
Riešený didaktický problém	
Systematizácia a prepojenie používateľských zručností a teoretických poznatkov s programátorskými prístupmi.	
Dominantné vyučovacie metódy a formy	*Príprava učiteľa a pomôcky*
Bádateľská metóda	Pomôcky: 1) Pripravený program v jazyku Python – program P1_Pixel3_ucitel.py 2) Pracovný list 3 a súbor obsahujúci kostru programu P1_Pixel3.py pre žiakov
Diagnostika splnenia vzdelávacích cieľov	
■ Žiacka diskusia ■ Grafický výstup modifikovaných obrázkov	

3.1 Texty k výučbe

Pretože práca s korytnačou grafikou je pomalá, budeme používať iné grafické prostredie, a síce tkinter a tiež sa oboznámime detailnejšie so zložitejším prostredím na vykresľovanie obrázkov, a síce s mtPaint.

mtPaint je prostredie na kreslenie obrázkov, kde je možné ovládať každý pixel. Práca v prostredí je veľmi jednoduchá a taká je aj inštalácia tohto prostredia. Avšak nedajú sa v ňom obrázky modifikovať na základe nejakých pravidiel. Preto je potrebné vytvorený obrázok dostať do prostredia, kde je to možné. Budeme používať programovací jazyk Python pri tejto modifikácii.

Modifikovaný obrázok je potom znovu zobraziteľný v prostredí mtPaint, ale postupné modifikácie obrázku budeme sledovať v grafickom prostredí jazyka Python. Výučbový text rozdelíme do nasledujúcich častí:

1) Stručná informácia o prostredí mtPaint
2) Potrebné informácie o programe v jazyku Python na modifikáciu prostredia.
3) Príprava na ďalšie modifikácie programu.

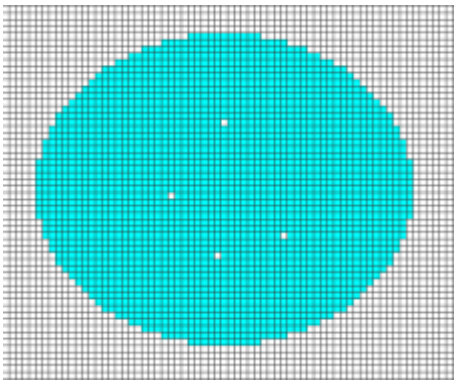
STRUČNÁ INFORMÁCIA O PROSTREDÍ MTPAINT

Program mtPaint je voľne šíriteľný, je možné ho získať z dostupných webových stránok. Inštalácia je veľmi jednoduchá. Program má dokumentáciu v anglickom a českom jazyku. Umožňuje jednoduchú prácu s pixelami obrázkov, vrstvenie obrázkov a tiež umožňuje pracovať s animáciami. Program je zaujímavý tým, že umožňuje meniť farby jednotlivých pixelov.

Jedna z možností na vykreslenie obrázku je nasledujúca:

1) Spustiť mtPaint, vyberieme v ponuke Súbor vykresliť Nový obrázok. Nastavíme veľkosť obrázka napríklad 640 x 400 a indexovanú paletu.
2) V ponuke Zobraziť vyberieme Configurate grid a nastavíme hodnoty na 10, 2, 2
3) V ponuke Efekty vyberieme Inverovať, aby sme dostali biele pozadie.
4) Pomocou ikonky Vybrať označíme vybranú oblasť v obrázku (je to obdĺžnik alebo štvorec.
5) Vybrať farbu na palete.
6) Vybrať vyplnenú elipsu, kliknúť na ňu. Vynikne obrázok s vyplnenou elipsou vybranej farby.
7) Pomocou ponuky Súbor, Uložiť ako vybrať adresár na uloženie obrázku, vybrať typ ukladaného súboru (napríklad .png), napísať meno súboru a súbor uložiť. (Vyskúšať viaceré typy ukladaných súborov a porovnať ich veľkosti.)

V mtPaint vyrobíme obrázok podobného tvaru a farby, ako je uvedené nižšie:



Obrázok 3. VYTVORENÝ OBRÁZOK PRE PRÁCU S PIXELAMI

```
Pracujeme s názvom obrázku 'gulka4.png'
```

V uvedenom obrázku máme tyrkysový objekt so 4 dierami. Farba pozadia je biela. Obrázok bude spracovávaný v programovacom jazyku Python.

POTREBNÉ INFORMÁCIE O PROGRAME V JAZYKU PYTHON NA MODIFIKÁCIU PROSTREDIA

Existuje skupina príkazov (funkcií), ktoré nevypisujú to textového ale do grafického okna. Takéto okno sa ale nevytvorí samo, musíme zadať špeciálne príkazy:

`import tkinter` **...** oznamuje, že budeme pracovať s modulom tkinter, ktorý obsahuje grafické príkazy

`canvas = tkinter.Canvas()`... vytvorí grafickú plochu a uloží jej referenciu do premennej canvas

`canvas.pack()` **...** zabezpečí zobrazenie nového okna aj s novovytvorenou grafickou plochou

Viac informácií je možné nájsť: Tkinter 8.5 reference: a GUI for Python (http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html)

Nasledujúce príkazy zobrazia obrázok v grafickej ploche canvas a na obrázok sa budeme odvolávať menom img. Nastavíme jeho šírku a výšku.

```
Sirka=640
Vyska=400
img = PhotoImage(width=Sirka, height=Vyska, file='Objekt1.png')
canvas.create_image((Sirka/2, Vyska/2), image=img)
```

SÚRADNICOVÁ SÚSTAVA

Je oproti súradnicovej sústave, ktorú poznáme z matematiky, trochu pozmenená:

- **x**-ová os prechádza po hornej hrane plátna grafickej plochy zľava doprava
- **y**-ová os prechádza po ľavej hrane plátna zhora nadol
- počiatok (0, 0) je v ľavom hornom rohu plátna
- môžeme používať aj záporné súradnice, vtedy označujeme bod, ktorý je mimo grafickú plochu

Plátno sme vytvorili pomocou tkinter.Canvas(), rozmery obrázka sú nastavené na **640x400** pixelov, preto pravý dolný roh má súradnice (639, 399). Súradnice potrebujeme pri modifikácii niektorých pixelov.

Samotná funkcia `canvas.create_image()` kreslí obrázok prečítaný zo súboru .png alebo .gif; má tri parametre: prvé dva sú súradnice stredu vykresľovaného obrázku a ďalší doplnkový parameter určuje obrázkový objekt.

Príkaz má tvar: `canvas.create_image(x, y, image=premenna)`

Pre nás dôležité príkazy sú:
`canvas.update()` - zobrazí nové zmeny v grafickej ploche
`canvas.after()` - pozdrží beh programu o zadaný počet milisekúnd

Pri kreslení v tkinter zadávame farby buď menami alebo zakódujeme ich RGB-trojicu do šestnástkovej sústavy ako 3 dvojciferné čísla (spolu 6 cifier). Pred takéto šestnástkové číslo musíme ešte na začiatok pridať znak '#', aby to tkinter odlíšil od mena farby. Napr.

- pre žltú ('yellow'): keďže platí red=255, green=255, blue=0, šestnástkovo je to trojica ff, ff, 00, a ako farba je to '#ffff00'
- pre tmavozelenú ('darkgreen'): red=0, green=100, blue=0, šestnástkovo je to trojica 00, 64, 00, a farba='"#006400'"

`funkcia canvas.after()`

- funkcia má jeden číselný parameter: počet tisícin sekundy, na ktorý sa výpočet na tomto mieste pozdrží, napr. `canvas.after(500)` pozdrží výpočet o 0,5 sekundy

GENEROVANIE LOŽISKA

Predpokladáme, že objekt v obrázku je kompaktný. V rámci objektu budeme vytvárať ložiská, čo sú malé kompaktné objekty, v tvare krížika.

Je dobré nakresliť si tieto ložiská na štvorčekový papier a jednotlivé pixely vyjadriť pomocou súradníc. V pracovnom liste je úloha o naprogramovaní krížikového ložiska.

Cieľom je vytvoriť program, ktorý vykreslí obrázok objektu s ložiskami, v nakreslenom prípade krížikovými ložiskami..

Obrázok 4. OBJEKT S LOŽISKAMI

Nakoniec modifikovaný obrázok je potrebné uložiť v súbore a môžeme ho znovu prezerať v prostredí mtPaint.

3.2 Priebeh výučby

Zámer: Oboznámiť žiakov s prácou v prostredí mtPaint a pripraviť ich na používanie grafického prostredia tkinter v Pythone:

- oboznámiť žiakov s prostredím mtPaint, ktoré je vhodné na prácu pixelami
- oboznámiť žiakov s možnosťami uloženia grafických súborov vytvorených v prostredí mtPaint;
- vytvorenie obrázku s objektom, ktorý bude v ďalšom upravovaný;

Po vysvetlení, žiaci budú pracovať s pracovným listom 3. Učiteľ v prípade potreby pripomenie niektoré príkazy jazyka Python. Pomocou pracovného listu žiak modifikuje program v jazyku Python **`P1_Pixel3.py`**, v ktorom budú generované infikované ložiská. Tento program žiak

dostane spolu s pracovným listom. Pomôckou pre učiteľa je priložený úplný program **`P1_Pixel3_ucitel.py`**.

Úloha s bádateľským prístupom:

Začnite kreslením jednoduchých obrázkov v mtPainte a iniciáciou diskusie:

OTÁZKA DO DISKUSIE

Kreslenia rôznych kombinovaných geometrických obrázkov, prázdnych, vyplnených. Vykreslenie krížika, pri ktorom je možné použiť viacero prístupov.

Nechajte žiakom možnosť vyjadriť svoje postupy a názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

VÝSKUMNÁ OTÁZKA 1

Ako by sme vedeli vyjadriť súradnice pixelov krížika, keď poznáme súradnice stredového pixelu?

SKÚMANIE A REALIZÁCIA (CCA 15 MIN.):

Nech žiaci opíšu vlastnými slovami, ako by postupovali pri riešení vyššie uvedenej úlohy. Otázkami podporujte diskusiu.

OTÁZKY DO DISKUSIE

Keby sme mali obrázok v takom digitálnom tvare, že by sme presne vedeli identifikovať riadok a stĺpec pixelu, bolo by to jednoduchšie?

ÚLOHA 1 – RIEŠTE!

Vykreslený obrázok v prostredí mtPaint uložiť pod navrhnutým menom. Budeme s ním pracovať v programe v Pythone.
K úlohe je pripravený pracovný list 3, bod 1.

Poznámka:

V tejto časti môžu mať niektorí žiaci problém s prácou v programe mtPaint. Je trochu sofistikovanejší než Paint.

VYSVETLENIE (CCA 10 MIN.):

Zámer: Vykresľovanie obrázkov s náhodne rozmiestnenými ložiskami.

Obrázok je vlastne uložený v dvojrozmernej matici, v ktorej každé políčko predstavuje pixel. Zásahy do obrázku budeme robiť cez túto dvojrozmernú maticu. Toto vysvetlenie má ozrejmiť žiakom, ako náhodne vygenerovať stred ložiska.

CVIČENIE – DISKUTUJTE!

Ako by sme vedeli pridať pixely k objektu tak, aby objekt zostal súvislý.

REALIZÁCIA (CCA 20 MIN.):

ÚLOHA 2 – RIEŠTE!

- Vytvoriť funkciu pre nájdenie pixelu určitej farby.
- Vytvoriť funkciu, ktorá zmení farbu pixelu danej farby na inú.
- Zaviesť nejaké podmienky pre doplnenie pixelu k objektu v obrázku. Vytvoriť funkciu, ktorá prefarbí niekoľko náhodne vygenerovaných pixelov tyrkysovej farby na čiernu.

Úloha je súčasťou pracovného listu 3.

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

SKÚMANIE A REALIZÁCIA (CCA 15 MIN.):

Zámer: Vytváranie zložitejších obrázkov v zmysle generovania viacerých ložísk.

ÚLOHA 3 – RIEŠTE!

- Doplniť program o funkciu, ktorá bude vytvárať ložisko „krížik“.
- Náhodne generovať nejaký počet ložísk.

Úloha je súčasťou pracovného listu 3.

HODNOTENIE (CCA 10 MIN.):

Zámer: Zistiť, či žiaci poznajú postup pri vytváraní ložiska, resp. viacerých ložísk.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

3.3 Pracovný list 3

Modifikácia rovinného obrázku, generovanie ložísk

Pracovný list obsahuje úlohy, ktoré je potrebné urobiť písomne, ale tiež naprogramovať v programovacom jazyku Python. Obrázok, s ktorým budeme pracovať, vytvoríme v programe mtPaint. Preto najprv spustíme program mtPaint.

Po vytvorení obrázka je potrebné otvoriť prostredie, v ktorom budete pracovať s programom P1_Pixel3.py. Program (súbor) dostanete spolu s pracovným listom. Program obsahuje niektoré naprogramované časti. Podľa pracovného listu je potrebné doplniť ďalšie príkazy tak, aby ste dostali funkčný program, ktorý generuje ložiská.

K práci s programom potrebujete aj súbor `gulka4.png`

1.	• Spustite mtPaint, vyberte v ponuke `Súbor, Nový obrázok`. Nastavte veľkosť obrázka napríklad 640 x 400 a indexovanú paletu. • V ponuke `Zobraziť` vyberte `Configurate grid` a nastavte hodnoty na 10, 2, 2 • V ponuke `Efekty` vyberte `Inverovať`, aby ste dostali biele pozadie. • Pomocou ikonky `Vybrať` označte vybranú oblasť v obrázku (je to obdĺžnik alebo štvorec. • Vyberte farbu na palete, odporúčame tyrkysovú (`cyan`). • Vyberte vyplnenú elipsu, kliknúť na ňu. Vznikne obrázok s vyplnenou elipsou vybranej farby. • Pomocou ponuky `Súbor, Uložiť ako` vyberte adresár vášho projektu na uloženie obrázku, vyberte typ ukladaného súboru (napríklad .png), napíšte meno súboru `gulka.png` a súbor uložte. (Vyskúšajte viaceré typy ukladaných súborov a porovnajte ich veľkosti, výsledky zapíšte.)

2.	Do vyplnenej elipsy vložte aspoň 4 malé štvorčeky bielej farby a obrázok uložte ako `gulka4.png`
	Odhadnite šírku a výšku elipsy v pixeloch. Budete to potrebovať pri zúžení prehľadávanej plochy. Odhady vložte vo svojom programe `P1_Pixel3.py` do premenných
	`SirkaEl=...`
	`VyskaEl=...`

3.	V programe je pripravená kresliaca plocha s určenou šírkou, výškou a farbou pozadia. Pretože korytnačka je pomalá, použijete na prácu s grafikou v Pythone knižnicu `tkinter`
	
	`from tkinter import *`
	`Sirka = 640`

```
Vyska = 400
window=Tk()
canvas=Canvas(window,width=Sirka,height=Vyska,bg="#ffffff")
canvas.pack()
```

Aby obrázok hneď nezmizol, na konci programu je vložený príkaz

```
mainloop()
```

4.

Načítanie a vykreslenie obrázku v strede kresliacej plochy je programe pripravené. Na obrázok sa budete odvolávať menom `img`

```
img=PhotoImage(width=Sirka, height=Vyska, file='gulka4.png')
canvas.create_image((Sirka/2, Vyska/2), image=img)
```

5.

Vypíšte v konzole informácie o niektorých pixeloch obrázka, napríklad v 240. riadku pixely od 301. po 340. stĺpec. Analyzujte, čo tieto informácie predstavujú.

6.

V programe je uvedená funkcia na výpočet hodnoty farby v šestnástkovej číselnej sústave z troch čísel v desiatkovej číselnej sústave, ktoré budú vstupnými parametrami (podobná funkcia bola používaná aj v predchádzajúcej téme)

```
def rgb_farba(r, g, b):
    return "#{:02x}{:02x}{:02x}".format(r, g, b)
```

Vytvorenú funkciu overiť na číslach 0, 255, 255 a aj iných.
Napíšte aspoň tri trojice čísel v desiatkovej číselnej sústave, pre ktoré získate v konzole výsledok. Výsledok zapíšte.

7.

V programe je uvedená funkcia, ktorá nájde v grafickej ploche pixel danej farby (vstupný parameter) a vráti jeho súradnice. V prípade, že pixel danej farby na ploche neexistuje, vráti hodnotu -1 v oboch súradniciach.

```
def najdiPixel(farba):
    xm=-1
    ym=-1
    farbaH=rgb_farba(farba[0],farba[1],farba[2])
    for y in range(Vyska):
        for x in range(Sirka):
            f_xy = img.get(x, y)
            f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
            if farbaH== f_xyH:
                xm=x
                ym=y
    return xm, ym
```

Overte túto funkciu pre nájdenie pixelu tyrkysovej a bielej farby. Zapíšte výsledok.

	Čo viete povedať o súradniciach daného pixelu, ak existuje? Ako by sa dala táto funkcia modifikovať?

8.	V programe je uvedená funkcia, ktorá zmení farbu pixelu farby `farba1` na `farba2`. Obe farby sú v šestnástkovej sústave. Ak pixel nemá farbu `farba1` zostane nezmenený.
	
	`def zmenPixel(ix, iy, farba1, farba2):`
	`  fa1 = img.get(ix, iy)`
	`  fa1H=rgb_farba(fa1[0],fa1[1],fa1[2])`
	`  if farba1==fa1H:`
	`  img.put(farba2,(ix,iy))`
	
	Overte túto funkciu napríklad zmenou tyrkysovej farby na čiernu.

9.	V programe je vygenerovaný náhodný počet n pixelov (50<n<200) v obrázku, ktoré ak mali tyrkysovú farbu, tak budú prefarbené čiernou. K náhodnému výberu je potrebné importovať knižnicu `random`. Sú pripravené farby a náhodný počet čiernych bodov n.
	
	`from random import *`
	`farbaCyH=rgb_farba(0,255,255)`
	`farba2= rgb_farba(0, 0, 0)`
	`n=random.randint(50, 200)`

10.	Doplňte uvedený cyklus na prefarbenie náhodne vygenerovaných tyrkysových bodov na čierne
	
	`for pocet in range(0,n):`
	`  x=random.randint((Sirka-SirkaEl)/2,(Sirka+SirkaEl)/2)`
	`  y=random.randint((Vyska-VyskaEl)/2,(Vyska+VyskaEl)/2)`
	`  .........`

11.	Prekreslite starý obrázok novým modifikovaným obrázkom v grafickom okne po 1000 milisekundách
	`canvas.update()`
	`canvas.after(1000)`

12.	V programe je funkcia `lozisko_krizik`, ktorá vygeneruje ložisko pixelov v tvare krížika rovnakej farby `farba2` na náhodnom mieste. Ložisko vznikne len v na náhodnej pozícii pixelu, ktorý má farbu `farba` danú ako parameter funkcie. Oba vstupné parametre sú farby dané ako čísla v šestnástkovej sústave. V uvedenej funkcii doplňte potrebné súradnice aj v programe (záleží na tom, ako si definujete ložisko, napríklad ako krížik, guličku, a pod).

	``
	def lozisko_krizik(farba,farba2):
	    x=random.randint((Sirka-SirkaEl)/2,(Sirka+SirkaEl)/2)
	    y=random.randint((Vyska-VyskaEl)/2,(Vyska+VyskaEl)/2)
	    f_xy=img.get(x,y)
	    f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
	    if f_xyH==farba:
	        img.put(farba2,(x,y))
	        img.put(farba2,(...,...))
	        img.put(farba2,(...,...))
	        img.put(farba2,(...,...))
	        img.put(farba2,(...,...))
	``
	
	Experimentovať tu možno s veľkosťou ložiska pridaním ďalších prefarbených pixelov. Funkciu odskúšajte na viacerých tvaroch ložísk.

13	Modifikovaný obrázok uložte pomocou príkazu `img.write('gulkaMod.png', format='png')`
14	Obrázok otvorte v programe mtPaint, kde je možné sledovať, čo sa jednotlivými pixelmi stalo. V obrázku pri zväčšení mierky je možné sledovať pixely so zmenenou farbou.
	Záverečné hodnotenie a sebahodnotenie
01	Napíšte funkciu, ktorá bude vytvárať ložisko tvaru tyčinky v riadku na náhodnom mieste. Veľkosť tyčinky je parametrom funkcie.
02	Napíšte cyklus, ktorý vytvorí 10 ložísk tvaru tyčinky v riadku. Vstupnými parametrami sú súradnice niektorého pixelu tyčinkového ložiska a jeho veľkosť.
03	Viem naprogramovať ložisko, ktoré má tyčinkový tvar. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem
04	Viem vytvoriť funkciu, ktorá zistí, koľko pixelov zaberá tyčinkové ložisko, ak poznám pozíciu niektorého jeho pixelu. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem

4 PRÁCA S OBRÁZKAMI, PODMIENENÉ MODIFIKÁCIE OBRÁZKU

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Modifikácia rovinného obrázku na základe stanovených podmienok.	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	***Žiakom rozvíjané spôsobilosti***
Získať zručnosti pri modifikácii rastrových obrázkov pomocou určených podmienok. Prispieť k vedomostiam v príprave na maturitu.	Programovať v jazyku Python, Zdokonaľovanie práce s obrázkami
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca so súbormi, základné príkazy jazyka Python.	
Riešený didaktický problém	
Prepojenie doposiaľ získaných vedomostí a zručnosti pri práci s pixelmi.	
Dominantné vyučovacie metódy a formy	***Príprava učiteľa a pomôcky***
Bádateľská metóda	Pomôcky: ■ pripravený program v jazyku Python – Projekt P1, program P1_Pixel4_ucitel.py ■ Pracovný list 4 a súbor obsahujúci kostru programu pre žiakov P1_Pixel4.py
Diagnostika splnenia vzdelávacích cieľov	
• Žiacka diskusia • Grafický výstup modifikovaných obrázkov	

4.1 Texty k výučbe

Ďalšie informácie o možnostiach vykresľovania objektov v jazyku Python je možné nájsť: „Tkinter 8.5 reference: a GUI for Python“ (http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html)

Všetky grafické príkazy pracujú s grafickou plochou, ku ktorej máme prístup prostredníctvom premennej `canvas`. Tieto príkazy sú v skutočnosti funkciami, ktoré budeme volať s nejakými parametrami. Všeobecný tvar väčšiny grafických príkazov je:

```
canvas.create_utvar(x,y,x,y,...,param=hodnota, param=hodnota, ...)
```

kde

- create_utvar je meno funkcie na vytvorenie grafického objektu, napr. create_line, create_rectangle, create_oval, ...
- x, y, x, y,...je postupnosť dvojíc súradníc bodov v grafickej ploche (rôzne príkazy majú rôzny počet bodov v rovine)
- param=hodnota je dvojica: meno doplnkového parametra (napr. fill, width, ...) a jej hodnota (napr. 'red', 5, 'Arial 20', ...)

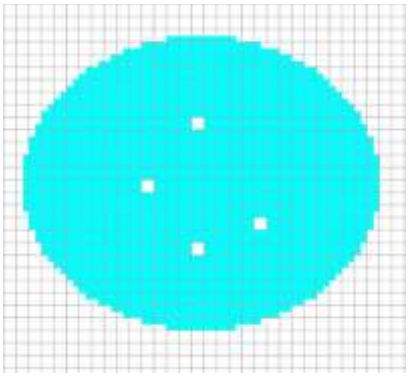
Pre všetky grafické príkazy platí, že keď pre doplnkové parametre nejaké neuvedieme hodnoty, tak tieto majú nastavené svoje inicializované (default) hodnoty (často takýmito náhradnými default hodnotami sú napr. fill='black' alebo width=1).

`canvas.create_oval()`

Parametre príkazu `canvas.create_oval()` vychádzajú z kreslenia obdĺžnika. Pre tkinter sú to presne rovnaké parametre, len elipsa ich trochu inak interpretuje: z dvoch protiľahlých vrcholov nenakreslí obdĺžnik ale vpísanú elipsu, t. j. elipsu, ktorá leží vo vnútri "mysleného" obdĺžnika a dotýka sa jeho strán. Parametrami príkazu sú súradnice dvoch protiľahlých vrcholov "mysleného" opísaného obdĺžnika v tvare: * `canvas.create_oval(x1,y1,x2,y2)` * strany tohto mysleného obdĺžnika sú rovnobežné so súradnicovými osami.

Naša práca aj naďalej bude zameraná na prácu s obrázkom, ktorý načítame do programu v Pythone.

V uvedenom obrázku 5 je farba pozadia biela, objekt je tyrkysový a má v sebe štyri diery.



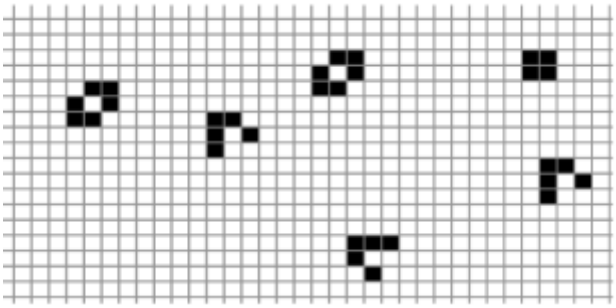
Obrázok 5. ZÁKLADNÝ OBRÁZOK PRE PRÁCU S PIXELAMI.

S objektom sme pracovali v programe P1_Pixel3.py. Do objektu sme vkladali ložiská, ktoré mali krížikový tvar čiernej farby.

MOTIVÁCIE PRE ŠTUDOVANÚ PROBLEMATIKU

Medzi najobľúbenejšie hry sa radia tie, ktoré nejakým spôsobom modelujú reálne situácie v prírode a spoločenstve: vojenské konflikty, špekulácie na burze, rôzne typy pretekov, znečisťovanie prírody atď. Najväčším úspechom v posledných rokoch sa môže pochváliť hra LIFE (život), ktorej autorom je algebraik JOHN H. CONWAY (Hvorecký, 1975).

Hra LIFE imituje život spoločenstva mikroorganizmov v prostredí, kde zdanlivo nič nebráni ich úspešnému vývinu. Spoločenstvo sa pohybuje po rozsiahlej pláni, na ktorej všetko (okrem nich samotných) tvorí ich potravu. Živočíchy sú „trojpohlavné", lebo práve traja môžu zrodiť potomka. Umierajú tak ako ostatné tvory buď na samotu, alebo premnožením. „Životné prostredie" spoločenstva tvorí nekonečné štvorčekované pole. V ňom vytvoríme počiatočné spoločenstvo tým, že niektoré štvorčeky označíme krížikmi predstavujúcimi mikroorganizmy. Každý mikroorganizmus prežije celý svoj život v tom istom štvorčeku a dĺžka jeho života závisí od toho, koľko má susedov. Ľubovoľný organizmus môže mať v krajnom prípade až osem susedov: štyroch po stranách a štyroch cez uhlopriečky.



Obrázok 6. PRÍKLAD OBJEKTOV V HRE LIFE

Hra prebieha v etapách - generačných obdobiach. Do nasledujúcej generácie prežijú z existujúcich mikroorganizmov iba tie, ktoré majú dvoch alebo troch susedov. Ak žiaden alebo iba jeden susedný štvorček je obsadený iným jedincom, organizmus zomiera na samotu, ak má štyroch alebo viac susedov, hynie nedostatkom potravy. Smrťou organizmu sa jeho políčko vyprázdni. V nasledujúcej generácii sa však môžu objaviť nové mikroorganizmy i na miestach doposiaľ prázdnych — na poliach, v susedstve ktorých žijú presné tri mikroorganizmy. Simulácie hry je možné nájsť na https://cs.wikipedia.org/wiki/Hra_%C5%BEivota

1. **Biológia, medicína –** chceme modelovať rast útvarov, kde sú pridávané rastové bunky, malé objekty.
2. **Pixel Art** je oblasť, ktorá podporuje tvorbu hier, vlastných postavičiek apod. Viac nájdete na https://v-play.net/game-resources/make-pixel-art-online

PRAVIDLÁ NA MODIFIKÁCIU OBJEKTU, KTORÁ JE ZAMERANÁ NA VZNIK A ÚPRAVY LOŽÍSK

Predpokladáme, že ložisko v objekte na obrázku je kompaktné.

Bude postupne narastať, čo sme urobili v predchádzajúcom programe P1_Pixel3.py. Kritéria na rast závisia od prostredia, v ktorom sa ložisko nachádza. Návrh jednoduchých kritérií je nasledujúci:

- Ložisko narastie o pixel v pozícii (x, y), ak počet obsadených pixelov (ide o pixely objektu, nie o pixely pozadia) v jeho okolí je viac ako dva a menej ako šesť (predpokladáme 8 pixelov v najbližšom okolí).

Kritéria na zánik pixelu:

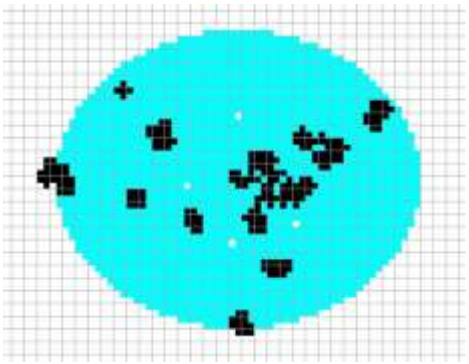
- V objekte vznikne diera v mieste pixelu v pozícii (x, y), ak v jeho okolí je aspoň šesť pixelov obsadených objektom.

Uvedené kritéria je potrebné naprogramovať. Učiteľ môže so žiakmi kritériá meniť a zasahovať aj do programu. Môže meniť počty obsadených a voľných pixelov. Pixely, ktoré nie sú obsadené (nepatria do objektu), sú voľné.

Prostredie spolu so skúmaným objektom je uložené na vykresľovanej ploche. Rast objektu je možné urobiť vyfarbovaním susediacich pixelov k objektu farbou objektu.

Nakoniec obrázok uložíme v súbore a môžeme ho znovu prezerať v prostredí mPaint (mtPaint).

Cieľový obrázok s ložiskami rôzne modifikovanými pomocou rastu a zániku má podobný tvar:



Obrázok 7. OBJEKT S RASTÚCIMI NÁDORMI

NOSNÉ FUNKCIE V PROGRAME

Funkcia `pocetSusedov`

Funkcia určí počet susedov farby farba0 pre daný pixel v pozícii rx, sy. Maximálny počet susedov je 8.
Metóda: V cykloch je skúmaných 9 pixelov, ako je to uvedené v tabuľke

Pixel (rx-1, sy-1)	Pixel (rx-1, sy)	Pixel (rx-1, sy+1)
Pixel (rx, sy-1)	Pixel (rx, sy)	Pixel (rx, sy+1)
Pixel (rx+1, sy-1)	Pixel (rx+1, sy)	Pixel (rx+1, sy+1)

Ak vytvorené súradnice padnú do kresliacej plochy, tak sa overí farba a vylúči stredový pixel.

```
def pocetSusedov(rx, sy, farba0):
    sObs=0
    for r in rx - 1, rx, rx + 1:
       for s in sy - 1, sy, sy + 1:
           if 0 <= r < Sirka and 0 <= s < Vyska:
               aa = img.get(r, s)
               aah = rgb_farba(aa[0], aa[1], aa[2])
               if aah==farba0 and (r != rx or s != sy):
                   sObs += 1
   return sObs
```

Funkcia `rastLoziska`

Rast a modifikácia ložiska so stredovým bodom (xL, yL) pomocou zmeny farby pixelov. Farby sú zadávané hexadecimálne.
Metóda: V oblasti znázornenej na obrázku

			Pixel (xL, yL)			

sa náhodne generujú súradnice potenciálneho kandidáta na pridanie do objektu. Zistí sa jeho farba a počet susedov a v prípade vhodného počtu susedov pixel sa pridáva do objektu, teda objekt porastie o tento pixel. Tento proces sa opakuje 100 krát (možno experimentovať). Deje v prírode sú náhodné, preto sme tu zaradili náhodné generovanie pozícií pixelu, ktorý môže prispieť k rastu.

```
def rastLoziska(xL, yL, farbaL, farbaN):
    for i in range(0,99):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        farba1=img.get(rr, ss)
        farba1h=rgb_farba(farba1[0],farba1[1],farba1[2])
        pocObsSus=pocetSusedov(rr, ss, farbaN)
        ### pocet nelozisk. susedov farbyN
        if pocObsSus>2 and pocObsSus<6:
            zmenPixel(rr,ss,farba1h,farbaL)
```

Funkcia `pZanik`

Funkcia zafarbi farbou pozadia náhodný pixel ložiska so stredom (xL, yL), čo znamená jeho zánik. Vstupné parametre sú pozícia stredu ložiska a dve farby v šestnástkovej sústave.

Metóda: Oblasť pre náhodné generovanie zanikajúceho pixelu je rovnaká, ako bola vo funkcii `rastLoziska`. V tomto prípade zistíme, či počet susedov je väčší ako 5 a ak áno, pixel je zafarbený farbou pozadia. Pokus o zánik sa opakuje 10 krát.

```
def pZanik(xL, yL, farbaL, farba):
    for ii in range(10):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        aap = img.get(rr, ss)
        aapH=rgb_farba(aap[0], aap[1], aap[2])
        if (farbaL==aapH):
            susedia=pocetSusedov(rr, ss, aapH)
            if susedia>5:
                zmenPixel(rr, ss, farbaL, farba)
```

Iné tvary ložísk

Je dobré nakresliť si tieto ložiská na štvorčekový papier a jednotlivé pixely vyjadriť pomocou súradníc. V programe P1_Pixel3_ziaci.py bolo naprogramované krížikové ložisko. Zaujímavá je práca aj s inými tvarmi ložísk, napríklad krúžok, štvorec, väčší krížik.

Obrázok 8. ZÁKLADNÉ TVARY LOŽÍSK

Úpravy objektu budeme robiť pomocou zmeny farieb. Ak pixel farby objektu zafarbíme farbou pozadia na obrázku, bude to vyzerať, že tam nič nie je. Rast ložiska (pridanie nejakých pixelov) je možné urobiť vyfarbovaním susediacich pixelov k ložisku farbou ložiska.

4.2 Priebeh výučby

Učiteľ motivuje žiakov ukážkami z hry Conwayov LIFE alebo ukážkami z Pixel Art, (Hvorecký, 1975), (CoBaLa, 2018). Potom im vysvetlí, že budú programovať niečo podobné ale zjednodušené. Učiteľ usúdi, do akej miery je potrebné vysvetliť tvorbu troch nosných funkcií v pracovnom liste 4.

VYSVETLENIE (CCA 20 MIN.):

Zámer:

- vytvorenie obrázku s objektom, v ktorom budú v ďalšom upravované ložiska;
- modifikácia obrázku pomocou rastu objektu (pridávania pixelov) a zániku niektorých pixelov v ložiskách.

-

Úloha s bádateľským prístupom:

Začnite kreslením jednoduchých obrázkov, v ktorých sú dané súradnice jedného pixelu (x, y) a iniciáciou diskusie o tom, ako by sme mohli vyjadriť súradnice ďalších pixelov patriacich do obrázku:

CVIČENIE – DISKUTUJTE!

Kreslenie rôznych kombinovaných geometrických obrázkov, prázdnych, vyplnených. Vykreslenie jedného konkrétne zadaného obrázku, pri ktorom je možné použiť viacero prístupov. Napríklad vykreslenie obdĺžnika. Ako vyjadríme súradnice všetkých jeho pixelov, keď poznáme súradnice jediného pixelu (x, y)? Čo je potrebné ešte vedieť?

Nechajte žiakom možnosť vyjadriť svoje postupy a názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

CVIČENIE – OTÁZKY!

Ako vyjadríme súradnice pixelov v obdĺžniku, keď poznáme súradnice jeho pixelu v ľavom hornom rohu (x,y)?

SKÚMANIE A REALIZÁCIA (CCA 15 MIN.):

Zámer: Určenie počtu susedov daného pixelu určenej farby.

CVIČENIE – OTÁZKY!

Ako by sme mohli určiť počet susedov určitej farby pre daný pixel?

Nech žiaci opíšu vlastnými slovami, ako by postupovali pri riešení v diskusii uvedenej úlohy. Otázkami podporujte diskusiu.

ÚLOHA 1 – RIEŠTE!

Vytvorte funkciu, ktorá v objekte na obrázku určí počet susedov daného pixelu určenej farby.

Úloha je zaradená v pracovnom liste 4. Spolu s pracovným listom dostanú žiaci program `P1_Pixel4.py`, ktorý budú modifikovať. Pre učiteľa je pripravený úplný program `P1_Pixel4_ucitel.py`

VYSVETLENIE (CCA 10 MIN.):

Keďže obrázok je vlastne uložený v dvojrozmernej matici, v ktorej každé políčko predstavuje pixel. Zásahy do obrázku budeme robiť cez túto dvojrozmernú maticu.

CVIČENIE – OTÁZKY!

Ako by sme vedeli pridať pixely k obrázku tak, aby zostal súvislý a mal farbu niektorého susedného pixelu?

SKÚMANIE A REALIZÁCIA (CCA 35 MIN.):

Zámer: Vytváranie a modifikácia zložitejších obrázkov pomocou úpravy farieb pixelov.

ÚLOHA 2 – RIEŠTE!

- Doplniť do programu funkcie, ktoré budú modifikovať objekt v obrázku na základe stanovených podmienok.
- Prvá funkcia pre pridanie pixelu k objektu, ak vytváraný pixel má kontakt s objektom aj s voľným priestorom, t. j. je na hranici.
- Druhá funkcia je pre zánik pixelu v objekte, t. j. prefarbenie farbou pozadia objektu.

Úloha je súčasťou pracovného listu 4.

Poznámka: Tieto úlohy vedú k biologicky motivovaným úlohám o raste nádorov.

HODNOTENIE (CCA 10 MIN.):

Zámer: Zistiť, či žiaci ovládajú prácu s pixelmi.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

Poznámka: Ako by sa dali vytvárať pohyblivé obrázky, animácie?

4.3 Pracovný list 4

Modifikácia rovinného obrázku na základe stanovených podmienok

Pracovný list obsahuje úlohy, ktoré je potrebné urobiť písomne, ale tiež naprogramovať v programovacom jazyku Python. Obrázok, s ktorým budeme pracovať, bol vytvorený v programe Paint/mtPaint a Paint/mtPaint nám poskytne možnosť lepšieho náhľadu na pixely obrázku.

Je potrebné pripraviť prostredie, v ktorom budete pracovať a otvoriť program - súbor `P1_Pixel4.py`. Pracovný list obsahuje základný postup pri modifikácii obrázka s možnosťou zasahovať farby pixelov a tiež obsahuje motiváciu pre skúmanie ďalších možností.
K práci budete potrebovať aj súbor **`gulka4.png`**

1.

V programe je pripravená kresliaca plocha s určenou šírkou, výškou a farbou pozadia. Pretože korytnačka je pomalá, použijem na prácu s grafikou v Pythone knižnicu `tkinter`

```
from tkinter import *

Sirka = 640
Vyska = 400
window=Tk()
canvas=Canvas(window,width=Sirka,height=Vyska,bg="#ffffff")
canvas.pack()
```

Aby obrázok hneď nezmizol, na konci programu je vložený príkaz `mainloop()`
Budeme pracovať s náhodnými číslami, preto do programu je vložený príkaz `import random`

2.

Načítajte a vykreslite obrázok v strede kresliacej plochy, na obrázok sa budete odvolávať menom `img`. Obrázok predstavuje teleso, ktoré budete modifikovať. Použite vytvorený obrázok **`gulka4.png`**

```
img=PhotoImage(width=Sirka, height=Vyska, file='gulka4.png')
canvas.create_image((Sirka/2, Vyska/2), image=img)
```

3.

V programe sú uvedené nasledujúce tri funkcie z programu P1_Pixel3.py, s ktorými ste už pracovali
`rgb_farba(r, g, b), lozisko_krizik(farba,farba2), zmenPixel(ix, iy, farba1, farba2)`

Napíšte, čo uvedené tri funkcie robia:

a) `rgb_farba(r, g, b)...`

b) `lozisko_krizik(farba,farba2)...`

c) zmenPixel(ix, iy, farba1, farba2)...

4.

Doplňte príkazy (miesta bodiek) vo funkcii `pocetSusedov`, ktorá určí počet susedov danej farby `farba0` pre daný pixel v pozícii `rx, sy`. Maximálny počet susedov je 8. Urobte to aj v programe.

```
def pocetSusedov(rx, sy, farba0):
    sObs=0
    for r in rx - 1, rx, rx + 1:
        for s in sy - 1, sy, sy + 1:
            if 0 <= r < Sirka and 0 <= s < Vyska:
                aa = img.get(r, s)
                aah = rgb_farba(aa[0], ......, aa[2])
                if aah==farba0 and (r != rx or s != sy):
                    sObs = ..........
    return sObs
```

Funkciu overte (zavolajte ju) na niektorých aspoň 4 náhodných príkladoch. Náhodne vygenerované pozície pixelov a ich farbu spolu s počtom susedov tej istej farby zapíšte do pracovného listu.

5.

V programe sa nachádza funkcia `rastLoziska`, ktorá aktivuje ložisko podľa nasledujúcich podmienok (`farbaL` je farba pixelu v ložisku, `farbaN` je farba rôzna od farby ložiska) :

a) Funkcia má na vstupe stred ložiska `xL, yL` a jeho aktívna oblasť je štvorcová oblasť pixelov s rohovými bodmi (x-3, y-3), (x-3, y+3), (x+3, y-3) a (x+3, y+3).
b) V tejto oblasti sa vygeneruje náhodný pixel zistí sa jeho farba (`farba1`).
c) Ak pixel susedí s 3, 4 alebo 5 pixelmi ložiska, stáva sa súčasťou ložiska (buď zmení farbu alebo ak patril k ložisku, farba mu zostáva).
d) Ak pixel susedí aspoň s aspoň šiestimi pixelmi ložiska, nemení svoju farbu.
e) Ak pixel má najviac jedného suseda, nemení svoju farbu.
f) Náhodné generovanie pixelov je potrebné opakovať aspoň 50 krát (vo funkcii je to tak nastavené).

```
def rastLoziska(xL, yL, farbaL, farbaN):
    for i in range(0,49):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        farba1=img.get(rr, ss)
```

```
            farba1h=rgb_farba(farba1[0],farba1[1],farba1[2])
            pocObsSus=pocetSusedov(rr, ss, farbaL)
            ### pocet susedov pixelu, ktore maju farbu farbaL
            if pocObsSus>2 and pocObsSus<6:
                zmenPixel(rr,ss,farba1h,farbaL)
```

Podmienka `pocObsSus>2 and pocObsSus<6` by sa dala zmeniť. Zapíšte zmenu v pracovnom liste aj programe tak, aby počet susedov ovplyvňujúcich zmenu farby bol presne 4 .

6.

Funkciu `rastLoziska` overte pri generovaní viacerých ložísk, napríklad 20. Záznam modifikácie do obrázku je možné urobiť pomocou príkazov (treba obrázok znovu nakresliť)

```
canvas.update()
canvas.after(1000)
```

Výsledný obrázok uložiť pomocou príkazu

```
img.write('nejakeMeno.png', format='png')
```

Tento obrázok si pozrite v programe Paint/mtPaint pri zväčšení, kde sú viditeľné pixely ložísk.

Popísať tvary vygenerovaných upravených ložísk. Vznikli rovnaké? Prečo vznikajú takéto ložiská?

7.

Ak by sme uvažovali o žijúcich pixeloch, ktoré pre život potrebujú živiny, záležalo by na tom, koľko ži-vín majú k dispozícii, resp. či pixely nie sú veľmi nahustené. Ak v okolí nejakého nášho pixelu ložiska žije 6, 7, alebo 8 iných pixelov, pravdepodobne sa nášmu pixelu nedostane dosť živín, a preto odumrie.

V programe je funkcia `pZanik`, ktorá sleduje odumieranie pixelov na náhodných miestach ložiska. Vstupné parametre sú pozícia stredu ložiska a dve farby sú v šestnás-tkovej sústave. `FarbaL` je farba ložiska, ktorá bude zmenená na farbu `farba`.

```
def pZanik(xL, yL, farbaL, farba):
    for ii in range(10):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        aap = img.get(rr, ss)
        aapH=rgb_farba(aap[0], aap[1], aap[2])
        if (farbaL==aapH):
            susedia=pocetSusedov(rr, ss, aapH)
            if susedia>5:
                zmenPixel(rr, ss, farbaL, farba)
```

	Funkciu overte zavolaním a experimentujte s náhodným počtom odumierajúcich pixelov. Experimentovanie popíšte.
8.	Navrhnite nejaký iný postup pri vzniku a zániku pixelov. Postup popíšte.

	Jedno možné overenie je (avšak treba premyslieť nastavenie farieb). Ložisko sa vytvorí, rastie, ale niektoré pixely v ňom zaniknú.
	
	`for PocLozisk in range(20):`
	`  xx, yy =lozisko_krizik(farba1,farba3)`
	`  ### farba3 je farba loziska`
	`  rastLoziska(xx, yy, farba3, farba1)`
	`  pZanik(xx, yy, farba3, farba2)`
	`  canvas.update()`
	`  canvas.after(1000)`

	Záverečné hodnotenie a sebahodnotenie
01	Napíšte, aká je podstata vzniku a zániku pixelov v nejakej zobrazenej štruktúre na obrázku.

02	Napíšte, čo robí nasledujúca funkcia:
	`def fFunkcia(farba):`
	`  pocet=0;`
	`  for x in range(300):`
	`  for y in range(200):`
	`  aa=img.get(x,y)`
	`  aah=rgb_farba(aa[0],aa[1],aa[2])`
	`  if aah==farba:`
	`  pocet=pocet+1`
	`  print(aah)`
	`  return pocet`

03	Viem naprogramovať rast každého ložiská, ktoré sa spomínajú v pracovnom liste. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem
04	Viem naprogramovať náhodný odumierajúci pixel. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu.
	Viem Potrebujem pomoc Neviem

5 MODELOVANIE, ZOBRAZOVANIE NAMERANÝCH HODNÔT

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Zobrazovanie nameraných hodnôt	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	***Žiakom rozvíjané spôsobilosti***
Zobrazenie vzťahov medzi atribútmi pomocou jednoduchých zobrazovacích funkcií.	Hľadanie vzťahov medzi atribútmi popisujúcimi javy v prírode, resp. v jednoduchých prípadových štúdiách
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca súbormi, príprava jednoduchého programu.	
Riešený didaktický problém	
Hľadanie závislosti medzi veličinami.	
Dominantné vyučovacie metódy a formy	***Príprava učiteľa a pomôcky***
Bádateľská metóda uplatnená pri hľadaní vzťahov a pri upravovaní programu	Pomôcky: ■ Pripravený program v jazyku Python – P2_Model1_ucitel.py ■ Pripravený pracovný list M1 a súbor obsahujúci kostru programu pre žiakov P2_Model1.py
Diagnostika splnenia vzdelávacích cieľov	
■ Žiacka diskusia ■ Grafický výstup zobrazovaných vzťahov ■ Nájdenie ďalších atribútov vhodných na zobrazenie	

5.1 Texty k výučbe

Pod **atribútmi** budeme rozumieť veličiny, pomocou ktorých je možné popísať nejaký **objekt** hmatateľný alebo nehmatateľný. Napríklad športový rekord je nehmatateľný objekt.

V tabuľke 1 máme uvedené údaje o rekordoch v skoku do diaľky sledované od roku 1901. Žijeme už niekoľko rokov v očakávaní nového rekordu, a teda by sme chceli predpovedať hodnotu rekordu napríklad v roku 2020. K tomu, aby sme túto predpoveď mohli urobiť, je potrebné sledovať, ako sa rekordy zlepšovali, aký bol ich postupný vývoj.

K sledovaniu vývoja rekordov by bolo vhodné mať rekordy vizualizované podľa rokov, to znamená vykresliť si dĺžky skokov v rokoch, kedy nastali rekordy.

Pre správne vykresľovanie a prípadné ďalšie výpočty (napríklad kedy bol najväčší prírastok k rekordu a aký veľký bol) je potrebné použiť:

- vhodné uloženie údajov a
- vhodnú metódu kreslenia.

Na uloženie údajov budeme používať pole a pri kreslení budeme používať knižnicu pyplot, ktorá obsahuje aj jednoduché príkazy pre kreslenie.

Dĺžka skoku	Skokan	Krajina	Miesto rekordu	Dátum
7,61	Peter O'Connor	Sp. kráľovstvo	Dublin	5. 8. 1901
7,69	Edwin Gourdin	USA	Cambridge	23. 7. 1923
7,76	Robert LeGendre	USA	Paríž	7. 7. 1924
7,89	DeHart Hubbard	USA	Chicago	13. 6. 1925
7,90	Edward Hamm	USA	Cambridge	7. 7. 1928
7,93	Sylvio Cator	Haiti	Paríž	9. 9. 1928
7,98	Chuhei Nambu	Japonsko	Tokio	27. 10. 1931
8,13	Jesse Owens	USA	Ann Arbor	25. 5. 1935
8,21	Ralph Boston	USA	Walnut	12. 8. 1960
8,24	Ralph Boston	USA	Modesto	27. 5. 1961
8,28	Ralph Boston	USA	Moskva	16. 7. 1961
8,31	Igor Ter-Ovanesyan	ZSSR	Jerevan	10. 6. 1962
8,31	Ralph Boston	USA	Kingston	15. 8. 1964
8.34	Ralph Boston	USA	Los Angeles	12. 9. 1964
8,35	Ralph Boston	USA	Modesto	29. 5. 1965
8,35	Igor Ter-Ovanesyan	ZSSR	Mexiko	19. 10. 1967
8.90	Bob Beamon	USA	Mexiko	18. 10. 1968
8,95	Mike Powell	USA	Tokio	30. 8. 1991

Tabuľka 2. ÚDAJE REKORDOV SKOKOV DO DIAĽKY V ODPOVEDAJÚCICH ROKOCH

PRIPOMENUTIE PRÍKAZOV PRI PRÁCI S ZOZNAMOM

S typom zoznam pracujeme úplne rovnako ako s n-ticami (tuple), ale ďalšie vlastnosti, v ktorých sa zoznamy líšia od n-tíc tomuto typu dávajú obrovskú programátorskú silu.

Čo je úplne rovnaké (resp. analogické) ako pre n-tice:

Zoznam je postupnosť hodnôt rôznych typov oddelených čiarkami, ktorá je uzavretá v [] hranatých zátvorkách. Napríklad, `Deti=['Eva',1233,'Jano',2550, 'Ema']` alebo `Roky=[1901, 1923, 1924, 1925, 1928, 1928, 1931, 1935, 1960, 1961, 1961, 1962]`

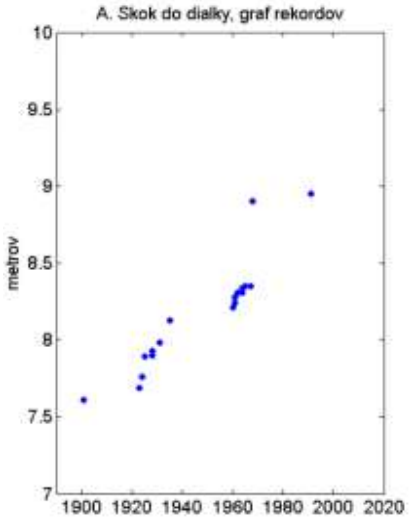
- prvkami zoznamov môžu byt' ľubovoľné typy: aj n-tice, znakové reťazce ale aj iné zoznamy
- na rozdiel od n-tíc hranaté zátvorky písať musíme (Python si veľakrát okrúhle zátvorky pre n-tice domyslí)
- pre zoznamy fungujú operácie + zreťazenia, `in` zisťovanie príslušnosti prvku
- zoznamy môžeme prechádzať `for`–cyklom (je to tiež iterovateľný typ)
- štandardné funkcie: `len()` pre počet prvkov, `sum()` pre súčet prvkov (funguje len pre číselné zoznamy), `min()` a `max()` pre najmenší a najväčší prvok (len pre polia s porovnateľnými prvkami)
- k prvkom pristupujeme pomocou indexovania, napríklad `Skoky[i+1], Skoky[i]`

- porovnávanie prvkov v zozname pomocou relácií ==, !=, <, <=, >, >= funguje na rovnakom princípe ako pre n-tice, t. j. lexikografické porovnávanie Všetky tieto vlastnosti, ak použijeme v programoch, zabezpečia, že zatiaľ sú hodnoty typu zoznam “nemenené”, t. j. všetky premenné sa správajú podobne ako n-tice.
- Pridávanie prvkov na koniec zoznamu pomocou príkazu meno_zoznamu.append. Napríklad `deltaSk.append(delta)`

DÔLEŽITÉ PRÍKAZY PRE KRESLENIE

Na obrázku 1 máme bodovo vykreslené hodnoty rekordných skokov v rokoch, kedy sa stali (x-ová os roky, y-ová os hodnoty rekordov v metroch). Našim cieľom je naučiť sa vykresliť podobné obrázky pre rôzne namerané alebo inak získané údaje.

K tomu, aby sme znázornili namerané údaje, je potrebné viac vedieť o možnostiach ich vykresľovania. Budeme postupovať tak, aby sme postupným použitím príkazov dospeli k vykresleniu obrázku.



Obrázok 9. VYKRESLENÉ DĹŽKY SKOKOV v ODPOVEDAJÚCICH ROKOCH

Budeme používať knižnicu pylab (inštalujeme ju bežným spôsobom, pylab je skratka z Python Laboratory), z ktorej vyberáme nasledujúce príkazy potrebné na vykreslenie nameraných dát:

- `figure(figsize=(12,6));` # pripraví obrázok na kreslenie viacerých menších obrázkov, (šírka, výška)
- `subplot(1,2,1);` # 1 riadok, 2 sĺpce, príprava na kreslenie prvého menšieho obrázka
- `plot(x,y,'r');` # kreslenie prvého menšieho obrázka, x, y sú súradnice bodov, r je farba
- `xlabel('Roky');` # popiska osi x
- `ylabel('Skoky');` # popiska osi y
- `title('Rekord');` # titulok prvého menšieho obrázka

- `subplot(1,2,2);` # 1 riadok, 2 sĺpce, príprava na kreslenie druhého menšieho obrázka
- `plot(x,y2,'g');` # kreslenie druhého menšieho obrázka, x, y2 sú súradnice bodov, g je farba
- `xlabel('Čas');` # popiska osi x
- `title('Doplnky');` # titulok druhého menšieho obrázka
- `suptitle('Skúmanie',fontsize=14);` # Celkový nadpis obrázka
- `plot(x,y,color='r',linestyle='dashed',linewidth=3.0);` # v plot je možné nastaviť štýl a hrúbku čiary
- `savefig('filename.png',dpi=100,bbox_inches='tight');` #uloženie obrázka

Pri zápise informácií do tretieho panelu je možné použiť príkaz

- `text,` v ktorom uvedieme počiatočnú pozíciu textu, text a prípadne premennú ktorej obsah chceme vypísať. Je potrebné previesť prevod obsahu premennej na reťazec. Napríklad

```
text(1,18,['Počet rekordných rokov: ',str(pocetRokov)])
```

- v prípade reálnej premennej je dobre použiť vhodný formát vypisovaného čísla, hlavne určiť počet desatinných miest.

Formátovanie reťazca pomocou metódy `format()`

Ukážeme niekoľko užitočných formátovacích prvkov. Volanie má tvar:

```
'formátovací reťazec'.format(parametre)
```

• kde 'formátovací reťazec' môže obsahovať ľubovoľný text, ale pre metódu `format()` sú zaujímavé len dvojice znakov '{, }'

• parametre pri volaní metódy `format()` sú ľubovoľné hodnoty (teda môžu byť' ľubovoľných typov), týchto parametrov by malo byť' presne rovnaký počet ako dvojíc '{}' (špeciálnymi prípadmi sa tu zaoberať nebudeme)

• metóda `format()` potom dosadí hodnoty svojich parametrov za zodpovedajúce dvojice '{}'

Učiteľ môže vysvetliť tvary niektorých formátov, ich použitie bude viac skúmané v nasledujúcom probléme.

Špecifikácia formátu

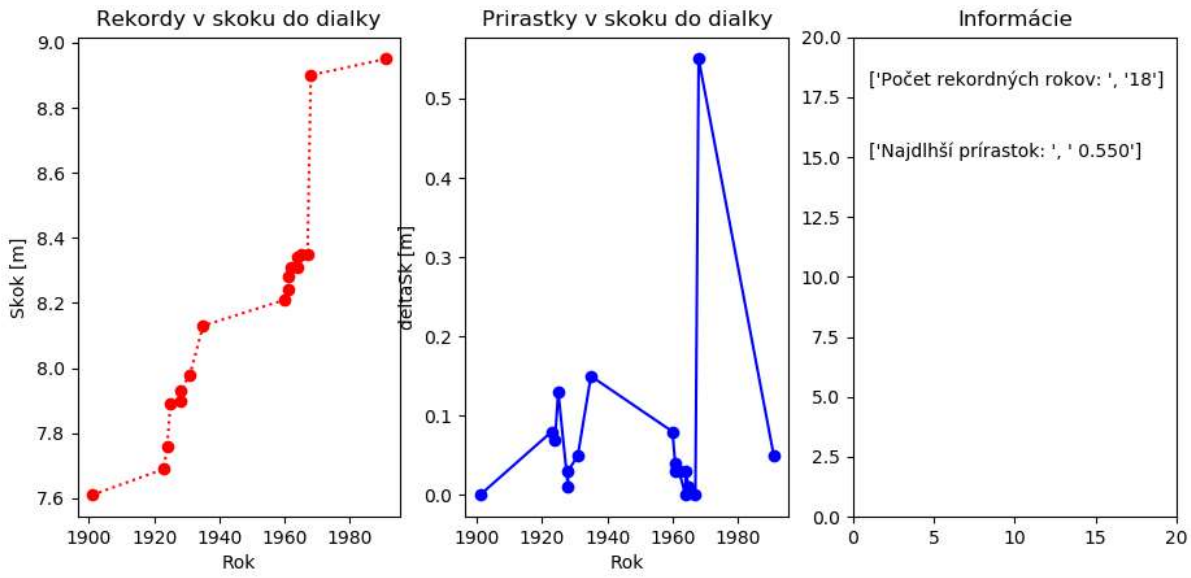
V zátvorkách '{}' sa môžu nachádzať rôzne upresnenia formátovania, napr.:

- '{:10}' - šírka výpisu 10 znakov
- '{:>7}' - šírka 7, zarovnané vpravo
- '{:<5d}' - šírka 5, zarovnané vľavo, parameter musí byť' celé číslo (bude sa vypisovať v 10-ovej sústave)
- '{:12.4f}' - šírka 12, parameter desatinné číslo vypisované na 4 desatinné miesta
- '{:06x}' - šírka 6, zľava doplnená nulami, parameter celé číslo sa vypíše v 16-ovej sústave
- '{:^20s}' - šírka 20, vycentrované, parametrom je reťazec

Najpoužívanejšie písmená pri označovaní typu parametra:

- d - celé číslo v desiatkovej sústave
- b - celé číslo v dvojkovej sústave
- x - celé číslo v šestnástkovej sústave
- s - znakový reťazec
- f - desatinné číslo (možno špecifikovať počet desatinných miest, inak default 6)
- g - desatinné číslo vo všeobecnom formáte

Pracovný list je pripravený na nakreslenie nasledujúceho obrázka 2, učiteľ sa sústredí na vysvetlenie potrebných príkazov.



Obrázok 10. V PRVOM PANELI SÚ VYKRESLENÉ REKORDY SKOKOV. V DRUHOM PANELI SÚ VYKRESLENÉ VEĽKOSTI PRÍRASTKOV K PREDCHÁDZAJÚCIM REKORDOM. DO TRETIEHO PANELU JE MOŽNÉ UMIESTNIŤ INFORMÁCIE.

5.2 Priebeh výučby (90 minút)

VYSVETLENIE (CCA 15 MIN.):

Zámer: Formou dialógu vysvetlíme možnosti zobrazovania. Je potrebné vysvetliť zobrazovanie v obrázku do panelov a tiež vysvetliť, ktoré príkazy sú dôležité. Viesť žiakov k vytvoreniu modifikovaného programu na základe pracovného listu M1. Spolu s pracovným listom žiaci dostanú pripravený program v jazyku Python. Program `P2_Model1.py` budú žiaci doplňovať podľa pokynov v pracovnom liste. Pre učiteľov je pripravený vzorový program `P2_Model1_ucitel.py`, ktorý by mal byť výsledkom práce žiakov.

ZAPOJENIE (CCA 10 MIN.):

Zámer:

- oboznámiť žiakov s problémom hľadania vzťahov medzi atribútmi
- ako pomôcku pri hľadaní vzťahov použiť vizualizáciu údajov pomocou grafiky

Začnite premietnutím tabuľky (venujte čas tomu, aby sa žiaci oboznámili s hodnotami v tabuľke). Nechajte žiakom možnosť vyjadriť svoj názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

VÝSKUMNÁ OTÁZKA 1

Rekordy nemajú pravidelný interval uskutočnenia. Ako by sme vedeli predpovedať rok ďalšieho rekordu? Vedeli by sme predpovedať dĺžku ďalšieho rekordného skoku?

Bolo by to jednoduchšie, keby rekordy boli uskutočňované pravidelne?

SKÚMANIE (CCA 10 MIN.):

Zámer: Nech žiaci opíšu slovne trend vývoja a odhadnú, kedy bude nasledujúci rekord. Body rekordov môžu byť nakreslené na tabuli alebo inak zobrazené.

OTÁZKY DO DISKUSIE

Ako by sa dal sledovať trend vývoja rekordov dĺžky skokov?

Vizualizácia dát by nám v skúmaní mohla pomôcť?

Venujte pozornosť obrázku 1 a inicializujte diskusiu:

OTÁZKA DO DISKUSIE

Obrázok predstavuje body zobrazujúce hodnoty v tabuľke. Body predstavujú diskrétne hodnoty.

Náš cieľ je vykresliť uvedený obrázok. Ako to urobiť?

SAMOSTATNÉ RIEŠENIE ÚLOHY (CCA 30 MIN.):

ÚLOHA 1 – RIEŠTE!

Vizualizácia dát pomocou grafiky.
K tomu je je pripravený pracovný list, v ktorom sa žiaci oboznámia s potrebnými príkazmi a budú experimentovať pri vytváraní jednoduchého programu.
K úlohe je pripravený pracovný list M1 a program `P2_Model1.py`, do ktorého je potrebné príkazy doplňovať.

ROZPRACOVANIE (CCA 15 MIN.):

Zámer: Uvedomiť si, že v danom probléme je možné hľadať a skúmať aj ďalšie vyhodnotenia.

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

Poznámka:

V diskusii je možné žiakov nasmerovať na výpočet prírastkov k rekordným skokom.

ÚLOHA 2 – RIEŠTE!

Naprogramujte funkciu pre výpočet prírastkov rekordných skokov.
K úlohe je pripravený pracovný list M1 a program `P2_Model1.py`

Nechajte žiakom čas na prácu na programe.

Podmienky môžu byť aj iné, podľa zručností žiakov, čo dokážu naprogramovať, a čo sami navrhnú.

ÚLOHA 3 – RIEŠTE!

Je možné sledovať aj dĺžky období, po ktorých bol nový rekord. Skúmajte, či sa tu dá sledovať nejaký trend, nejaká závislosť. Je tu možnosť znovu použiť grafické vykreslenie.

Poznámka:

Ak sú niektorí šikovní žiaci hotoví s riešením problému, je možné orientovať ich na získanie iných dát, napríklad rekordov v behoch, a pod. Údaje sa dajú nájsť na webových stránkach.

HODNOTENIE (CCA 10 MIN.):

Zámer: Žiaci majú hodnotiť, ako a čím im pomohol skúmaný problém. Hlavne, či budú vedieť samostatne postupovať pri riešení podobných úloh.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

5.3 Pracovný list M1

Modelovanie, zobrazovanie nameraných hodnôt

Pracovný list predstavuje postupné kroky k modifikácii programu na vykreslenie rekordov v skoku do diaľky. Vedie tiež k experimentovaniu s jednotlivými príkazmi v programovacom jazyku Python, preto je potrebné pracovať v prostredí Pythonu a vždy spustiť program, keď si to pracovný list vyžaduje. Niektoré odpovede a pripomienky je možné písať do pracovného listu. Spolu s pracovným listom dostanete program - súbor `P2_Model1.py`, ktorý budete v rámci svojho projektu doplňovať a modifikovať. Do tohto súboru budete písať príkazy podľa pokynov pracovného listu.

1.	K údajom `1901, 1923, 1924, 1925, 1928, 1928, 1931, 1935, 1960, 1961, 1961, 1962, 1964, 1964, 1965, 1967, 1968, 1991` je v programe vytvorený zoznam, ktorý sa nazýva Roky
2.	Pomocou príkazu `print` vypíšte prvky zoznamu v konzole. Program odskúšajte spustením.
3.	K údajom `7.61, 7.69, 7.76, 7.89, 7.90, 7.93, 7.98, 8.13, 8.21, 8.24, 8.28, 8.31, 8.31, 8.34, 8.35, 8.35, 8.90, 8.95` Je v programe vytvorený zoznam, ktorý sa nazýva Skoky
4.	Pomocou príkazu `print` vypíšte prvky poľa v konzole.
5.	Koľko prvkov je v poli Roky? Zistite to pomocou príkazu `len()` a výsledok je vložený do premennej pocetRokov Obsah premennej pocetRokov vypíšte v konzole.
6.	Ako by ste zistili, či počty prvkov v oboch poliach sú rovnaké? Vytvorte príkaz, ktorý to zistí a v konzole vypíše „rovnaký“, ak sú počty prvkov rovnaké, inak vypíše „rôzny“. Program odskúšajte spustením.

7.	Budeme pracovať s knižnicou pylab, preto do programu vložíme príkaz
	`from pylab import *`
	
	Budeme vykresľovať vzťah medzi hodnotami v poliach Roky a Skoky.
	Do programu vložíme príkaz
	`figure()`
	pomocou ktorého pripravíme obrázok na kreslenie.

8.	Obrázok môže mať viacej časti, ktoré kreslíme pomocou príkazu `subplot(počet riadkov, počet slpcov, poradové číslo časti)` V programe sa nachádzajú zobrazenia troch menších obrázkov v jednom riadku. Program spustíme a sledujeme, čo sa stalo. ‘ro:’ predstavuje štýl a farbu kreslenej čiary, r – red, : - bodkovanú čiaru, o – kreslenie guličiek v bodoch. Zmeňte farbu na zelenú v subplot(1,1,1), guličky na *, : na -.

	Experimentujte s ďalšími farbami.

9.	V prvom menšom obrázku je uvedené vysvetlenie, čo je na ktorej osi vykreslené pomocou popisiek. A tiež má nadpis.
	
	`xlabel('Rok')`
	`ylabel('Skok [m]')`
	`title('Rekordy v skoku do dialky')`

10.	Zaujímajú nás prírastky skokov, prírastok medzi (i+1) a i – tým skokom vypočítame ako rozdiel `delta=Skoky[i+1]-Skoky[i]` Vytvorte cyklus, v ktorom sa vypočítajú všetky možné prírastky do poľa deltaSk . 0-tý prírastok bude mať hodnotu 0. Doplňte cyklus do programu a výsledky poľa deltaSk vypíšte v konzole.
11.	V obrázku v druhom paneli vykreslíte prírastky rekordných skokov v jednotlivých rokoch. Do obrázka doplňte popisky a nadpis, ak chýbajú. **Pozor!** Príkaz `show()` musí byť na konci nášho programu.
12.	Pomocou príkazu `text()` je možné vkladať do obrázkov text na určenú pozíciu, podľa rozsahu súradníc oboch osí. Rozsah súradníc je možné určiť príkazom `axis([x1, x2, y1, y2])` Experimentujte v programe so zmenou súradníc v niektorom z panelov. Výsledky si zapíšte do pracovného listu.

13.	Do tretieho panelu vložte rôzne informácie. Napríklad
	
	`subplot(1,3,3)`
	`text(1,18,['Počet rekordných rokov: ',str(pocetRokov)])`
	`aa=max(deltaSk)`
	`text(1,15,['Najdlhší prírastok: ','{:6.3f}'.format(aa)])`
	
	`axis([0, 20, 0, 20])`
	`title('Informácie')`
	
	Úlohou je nájsť ďalšie vhodné informácie, ktoré by bolo vhodné vložiť do tretieho panela.

14.	`'{:6.3f}'.format(aa)` určuje výstupný formát čísla. Experimentujte so zmenami formátu. Písmeno f predstavuje, že ide o reálne číslo.
	Záverečné hodnotenie a sebahodnotenie
O1.	Napíšte príkaz/príkazy, pomocou ktorých sa určí rok najväčšieho prírastku skoku.
	Napíšte konfiguráciu obrázku s dvoma subplotmi v stĺpci

O2.	Viem, aký je rozdiel medzi príkazmi `plot` a `subplot`. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem
	Viem, ako zistím obsah prvku poľa v nejakej pozícii. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem

Pomôcka: Niektoré príkazy knižnice Pylab

- `figure(figsize=(12,6));` # pripraví obrázok na kreslenie viacerých menších obrázkov, (šírka, výška)

- `subplot(1,2,1);` # 1 riadok, 2 sĺpce, príprava na kreslenie prvého menšieho obrázka
- `plot(x,y,'r');` # kreslenie prvého menšieho obrázka, x, y sú súradnice bodov, r je farba
- `xlabel('Roky');` # popiska osi x
- `ylabel('Skoky');` # popiska osi y
- `title('Rekord');` # titulok prvého menšieho obrázka

- `subplot(1,2,2);` # 1 riadok, 2 sĺpce, príprava na kreslenie druhého menšieho obrázka
- `plot(x,y2,'g');` # kreslenie druhého menšieho obrázka, x, y2 sú súradnice bodov, g je farba
- `xlabel('Čas');` # popiska osi x
- `title('Doplnky');` # titulok druhého menšieho obrázka

- `suptitle('Skúmanie',fontsize=14);` # Celkový nadpis obrázka

- `plot(x,y,color='r',linestyle='dashed',linewidth=3.0);` # v plot je možné nastaviť štýl a hrúbku čiary

- `savefig('filename.png',dpi=100,bbox_inches='tight');` # uloženie obrázka

6 MODELOVANIE LINEÁRNYCH JAVOV

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Modelovanie športových rekordov	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	***Žiakom rozvíjané spôsobilosti***
Vyjadrenie vzťahov medzi atribútmi pomocou jednoduchých lineárnych funkcií.	Hľadanie vzťahov medzi atribútmi popisujúcimi javy v prírode, resp. v jednoduchých prípadových štúdiách
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca súbormi, poznať lineárne funkcie, vytvorenie a spustenie programov.	
Riešený didaktický problém	
Hľadanie závislosti medzi veličinami. Návrh modelu závislosti.	
Dominantné vyučovacie metódy a formy	***Príprava učiteľa a pomôcky***
Bádateľská metóda uplatnená pri hľadaní vzťahov a upravovaní programu	Pomôcky: ■ pripravený program v jazyku Python – P2_Model2_ucitel.py ■ pracovný list M2 a súbor obsahujúci kostru programu pre žiakov P2_Model2.py
Diagnostika splnenia vzdelávacích cieľov	
• Žiacka diskusia • Grafický výstup modelovaných vzťahov • Predikcia očakávaných hodnôt rekordov	

6.1 Texty k výučbe

Pod **atribútmi** budeme rozumieť veličiny, pomocou ktorých je možné popísať nejaký **objekt** hmatateľný alebo nehmatateľný. Napríklad športový rekord je nehmatateľný objekt. **Model** v našom ponímaní bude vlastne popísaný vzťah medzi vstupnými a výstupnými atribútmi nejakého objektu. Popis vzťahu sa obyčajne vyjadruje pomocou nejakej funkcie. My sa budeme zaoberať popisom vzťahov pomocou lineárnej funkcie.

V nižšie uvedenej tabuľke sú zaznamenané rekordy v skokoch do diaľky. Keďže už dlho nepadol žiadny rekord, zaujíma nás vývoj rekordov v minulosti a budeme hľadať lineárny vzťah medzi rokmi rekordov a dĺžkami skokov.

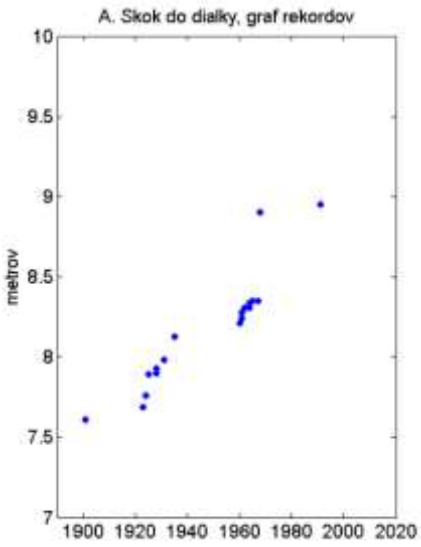
Na Obr. 11 sú vykreslené rekordy v uvedených rokoch. Je zrejmé, že rekordy rastú a rastúcim číslom roku, preto navrhneme najjednoduchší vzťah medzi nimi, a síce lineárnu závislosť dĺžky skoku od roku, t. j.

$$dlzkaSkoku = k * rok + q$$

kde k a q sú parametre vyjadrujúce daný lineárny vzťah. Ako určiť k a q?

Tabuľka 3. ÚDAJE REKORDNÝCH SKOKOV V ODPOVEDAJÚCICH ROKOCH

Dĺžka skoku	Skokan	Krajina	Miesto rekordu	Dátum
7,61	Peter O'Connor	Sp. kráľovstvo	Dublin	5. 8. 1901
7,69	Edwin Gourdin	USA	Cambridge	23. 7. 1923
7,76	Robert LeGendre	USA	Paríž	7. 7. 1924
7,89	DeHart Hubbard	USA	Chicago	13. 6. 1925
7,90	Edward Hamm	USA	Cambridge	7. 7. 1928
7,93	Sylvio Cator	Haiti	Paríž	9. 9. 1928
7,98	Chuhei Nambu	Japonsko	Tokio	27. 10. 1931
8,13	Jesse Owens	USA	Ann Arbor	25. 5. 1935
8,21	Ralph Boston	USA	Walnut	12. 8. 1960
8,24	Ralph Boston	USA	Modesto	27. 5. 1961
8,28	Ralph Boston	USA	Moskva	16. 7. 1961
8,31	Igor Ter-Ovanesyan	ZSSR	Jerevan	10. 6. 1962
8,31	Ralph Boston	USA	Kingston	15. 8. 1964
8.34	Ralph Boston	USA	Los Angeles	12. 9. 1964
8,35	Ralph Boston	USA	Modesto	29. 5. 1965
8,35	Igor Ter-Ovanesyan	ZSSR	Mexiko	19. 10. 1967
8.90	Bob Beamon	USA	Mexiko	18. 10. 1968
8,95	Mike Powell	USA	Tokio	30. 8. 1991



Obrázok 11. ZOBRAZENIE REKORDNÝCH SKOKOV V ODPOVEDAJÚCICH ROKOCH

V tabuľke máme uvedené údaje o rekordoch v skoku do diaľky sledované od roku 1901. Žijeme už niekoľko rokov v očakávaní nového rekordu, a teda by sme chceli predpovedať hodnotu rekordu napríklad v roku 2020. K tomu, aby sme túto predpoveď mohli urobiť, je potrebné sledovať, ako sa rekordy zlepšovali, aký bol ich postupný vývoj. Na obrázku 1 máme vykreslené hodnoty rekordných skokov v rokoch, kedy sa stali (x-ová os roky, y-ová os hodnoty rekordov).

Pozorovaním údajov, by sme mohli dospieť k tomu, že vývoj rekordov v rokoch je lineárny, t. j. by sme mohli nájsť priamku, ktorá by dosť dobre sledovala tento vývoj. Otázka je, čo to znamená „dosť dobre“? „Dosť dobre“ by sme mohli definovať pomocou chýb, ktoré vzniknú rozdielom rekordu a predpovede rekordu na priamke v skúmanom roku. Budeme požadovať, aby súčet štvorcov týchto chýb bol minimálny. Chybu v i-tom roku vyjadríme

Ei= dlzkaSkoku(i) - Skok(i),

kde dlzkaSkoku(i) je dĺžka skoku predpovedaná a Skok(i) je dĺžka skoku reálna v i-tom roku. Budeme teda požadovať, aby hodnota

$$E = \sum_{i=1}^{N} Ei * Ei$$

bola minimálna, N je počet rekordných rokov. Znamená to teda, že hľadáme najvhodnejšie hodnoty pre parametre k a q. K tomu potrebujeme vypočítať priemerné hodnoty roku, skoku, rok*skok a rok*rok, čo je uvedené v poslednom riadku tabuľky 4.

Tabuľka 4. V POSLEDNOM RIADKU SÚ VYPOČÍTANÉ HODNOTY PR, PS, PRS A PRR

Por. č	Rok	Skok	Rok*Skok	Rok*Rok
1	1901	7,61	14466,61	3613801
2	1923	7,69	14787,87	3697929
3	1924	7,76	14930,24	3701776
4	1925	7,89	15188,25	3705625
5	1928	7,90	15231,20	3717184
6	1928	7,93	15289,04	3717184
7	1931	7,98	15409,38	3728761
8	1935	8,13	15731,55	3744225
9	1960	8,21	16091,60	3841600
10	1961	8,24	16158,64	3845521
11	1961	8,28	16237,08	3845521
12	1962	8,31	16304,22	3849444
13	1964	8,31	16320,84	3857296
14	1964	8,34	16379,76	3857296
15	1965	8,35	16407,75	3861225
16	1967	8,35	16424,45	3869089
17	1968	8,90	17515,20	3873024
18	1991	8,95	17819,45	3964081
$(1/N)\sum_{n=1}^{N}$	1947,7	8,1739	15927,40	3793921

Matematicky vieme odvodiť nasledujúce vzťahy pre k a q. Označme

pR= priemer(Rok), pS= priemer(Skok), pRS=priemer(Rok*Skok), pRR = priemer(Rok*Rok).

Potom

$$k=[pRS– pR*pS]/[pRR - pR*pR]$$

$$q=pS – k * pR$$

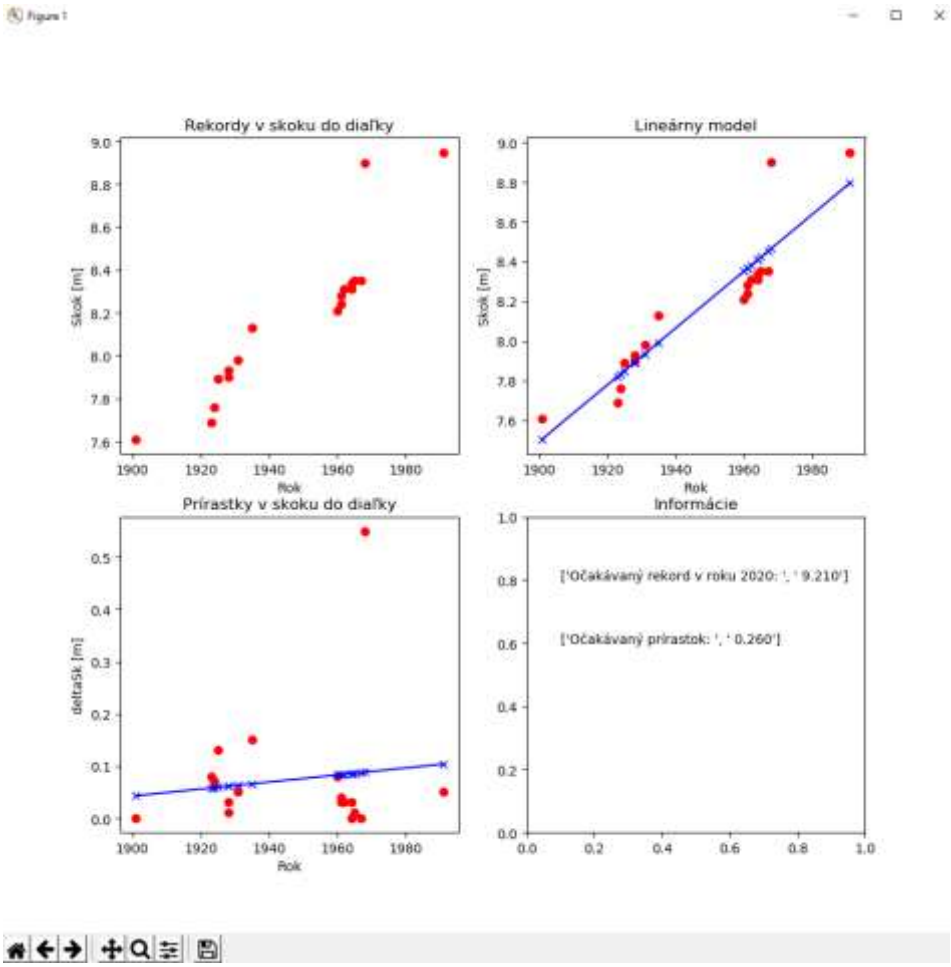
Reálny výpočet nám dáva hodnoty

$$k=0.0143,\ q=-19.7139,\ E=0.0193$$

Vykreslená priamka sa nachádza v paneli B na Obrázku 2. Táto priamka vyjadruje trend vývoja skokov a teda by sme sa mohli pýtať: Ak padne rekord v roku 2020, akú dĺžku skoku by sme mohli očakávať? Určíme to dosadením roku do rovnice priamky predstavujúcej vývoj

$$D\_2020 = 0.0143 * 2020 – 19.7139 = 9.1721 \text{ m}$$

Je potrebné poznamenať, že tieto predpovede sú orientačné a tiež je potrebné sa zamyslieť nad tým, či lineárny trend vývoja skokov je najvhodnejší.



Obrázok 12. CIEĽOVÝ OBRÁZOK PRE PRACOVNÝ LIST 6.

Treba si tiež uvedomiť, že vo vzťahu pre výpočet k sa využíva viacero priemerných hodnôt, preto budeme vytvárať funkciu na výpočet priemeru postupnosti hodnôt. Cieľom vykreslenia je Obrázok 12. V poslednom paneli je uvedená informácia o očakávanom rekord v roku 2020.

6.2 Priebeh výučby

VYSVETLENIE (CCA 20 MIN.):

Zámer: Učiteľ vysvetlí trend vývoja rekordu skokov podľa textu. V tabuľke zobrazených hodnôt je potrebné vypočítať priemery stĺpcových hodnôt, ktoré sú potrebné pri výpočte parametrov priamky.

Učiteľ vysvetlí, ako by vyzerala priamka, keby sme ju vytvorili len z prvého a posledného rekordu. Môže túto priamku matematicky odvodiť aj nakresliť a nechá žiakov uvažovať, či neexistuje lepšia priamka, a tiež ako by sme mohli ohodnotiť, ktorá priamka je lepšia. Učiteľ zdôrazní, že pri hľadaní lepšej priamky je nutné brať do úvahy všetky rekordy, preto pri hľadaní k a q v priamke budú vystupovať priemery hodnôt a ich kombinácie.

Tiež je potrebné diskutovať o tom, že lineárny trend by v budúcnosti znamenal nekonečné zväčšovanie dĺžky skoku.

ZAPOJENIE (CCA 10 MIN.):

Zámer:

- Po oboznámení sa s problémom žiaci môžu hľadať ďalšie vzťahy medzi atribútmi, napríklad ako rástli príspevky k dĺžkam
- na základe známych vzťahov predpovedať výsledok v niektorých budúcich rokoch

CVIČENIE – DISKUTUJTE!

Obrázok predstavuje body zobrazujúce hodnoty v tabuľke. Body predstavujú diskrétne hodnoty. Medzi rekordmi sú rôzne počty rokov. Aký by bol trend, keby sme brali do úvahy len niektoré dva rekordy?

Nechajte žiakom možnosť vyjadriť svoj názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

CVIČENIE – OTÁZKY!

Ako by sme vedeli určiť trend vývoja rekordných skokov? Rekordy nemajú pravidelný interval uskutočnenia.

Bolo by to jednoduchšie, keby sa rekordy uskutočňovali pravidelne?

SKÚMANIE (CCA 20 MIN.):

Zámer: Nech žiaci opíšu slovne trend vývoja a odhadnú, kedy bude nasledujúci rekord.

CVIČENIE – DISKUTUJTE!

Ako by sa dal sledovať trend vývoja rekordov dĺžky skokov? V diskusii je možné žiakov nasmerovať na lineárny vývoj rekordov dĺžky skokov do diaľky?

ÚLOHA 1 – RIEŠTE!

K úlohe je pripravený pracovný list M2 a program `P2_Model2.py`, do ktorého budú žiaci príkazy doplňovať podľa pracovného listu. Vzorové riešenie je uvedené v programe `P2_Model2_ucitel.py`

- Overte funkciu pre výpočet priemernej hodnoty nejakej postupnosti hodnôt, ktorá je uvedená v programe `P2_Model2.py`
- Overte funkciu `mean` pre výpočet priemernej hodnoty nejakej postupnosti hodnôt, ktorá je uvedená v matematickej knižnici `numpy`.
- Overte funkciu na výpočet dvoch parametrov pre lineárny priebeh vývoja rekordov dĺžky skokov, ktorá je uvedená v programe `P2_Model2.py`

Nechajte žiakom čas na prácu s programom.

VYSVETLENIE (CCA 10 MIN.):

Zámer: Formou dialógu vysvetlíme možnosti zobrazovania pomocou príkazov knižnice Pylab, ktorá má veľmi dobre použitie v Pythone. Cieľom je ukázať žiakom možnosti lineárneho modelovania.

ROZPRACOVANIE (CCA 25 MIN.):

Zámer: Uvedomiť si, že v danom probléme je možné hľadať a skúmať aj ďalšie vyhodnotenia.

ÚLOHA 2 – RIEŠTE!

Analyzujte postupnosť zlepšenia rekordov, to znamená prírastky k dĺžke skokov. Je možné uvažovať o tom, že aj prírastky predstavujú nejaký lineárny trend? Rastúci alebo kleasjúci?

V pripravenom obrázku je potrebné vykresliť tento trend do tretieho panelu.

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

ÚLOHA 3 – RIEŠTE!

Je možné sledovať aj dĺžky období, po ktorých bol nový rekord. Skúmajte, či sa tu dá sledovať nejaký trend, nejaká závislosť. Je tu možnosť použiť grafické vykreslenie a tak diskutovať.

Poznámka: Ak sú niektorí šikovní žiaci hotoví s riešením problému, je možné orientovať ich na získanie iných dát, napríklad rekordov v behoch, a pod.

HODNOTENIE (CCA 10 MIN.):

Zámer: Žiaci majú hodnotiť, ako a čím im pomohol skúmaný problém. Hlavne, či budú vedieť samostatne postupovať pri riešení podobných úloh.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

6.3 Pracovný list M2

Lineárne modelovanie javov

Pracovný list predstavuje postupné kroky k modifikácii programu `P2_Model2.py` na vykreslenie rekordov v skoku do diaľky. Vedie tiež k experimentovaniu s jednotlivými príkazmi v programovacom jazyku Python, preto je potrebné pracovať v prostredí Pythonu a vždy spustiť program, keď si to pracovný list vyžaduje. Niektoré odpovede a pripomienky je možné písať do pracovného listu. Do programu je potrebné písať príkazy podľa pokynov pracovného listu. Výsledkom práce je program, ktorý je potrebné uložiť do určeného adresára a doplnený pracovný list.

1.	K údajom `1901, 1923, 1924, 1925, 1928, 1928, 1931, 1935, 1960, 1961, 1961, 1962, 1964, 1964, 1965, 1967, 1968, 1991` vytvorte zoznam, ktorý sa bude volať Roky
2.	Pomocou príkazu `print` vypíšte prvky zoznamu v konzole. Program odskúšajte spustením.
3.	K údajom `7.61, 7.69, 7.76, 7.89, 7.90, 7.93, 7.98, 8.13, 8.21, 8.24, 8.28, 8.31, 8.31, 8.34, 8.35, 8.35, 8.90, 8.95` vytvorte zoznam, ktorý sa bude volať Skoky

4.	V programe je uvedená funkcia `priemer`, ktorá bude počítať a dávať na výstupe priemer prvkov postupnosti (zoznamu) `post` vstupných hodnôt (funkcia musí byť volaná na postupnosti čísel, zoznam čísel). Jej tvar je
	`Def priemer(post):`
	`  n=len(post)`
	`  s=sum(post)`
	`  return s/n`
	Odskúšajte túto funkciu na hodnotách zoznamov Skoky a Roky
	Zapíšte výsledky:

5.	V programe je importovaná knižnica `numpy` . V matematickej knižnici `numpy` existuje funkcia `mean`, ktorá počíta priemer pre zadaný vstupný parameter. Funkciu zavoláme s parametrami Skoky a Roky. Zapíšte výsledky: Výsledky porovnajte a napíšte komentár.

6.	V programe je uvedená funkcia `priamka` so vstupnými parametrami pre počet prvkov v zoznamoch (postupnostiach), pre ktoré hľadáme priamku, a dve postupnosti `fRoky`, `fSkoky`. Výstupom funkcie sú parametre k a q.
	Jej tvar je
	`def priamka(n,fRoky,fSkoky):`
	`# Vypocet priemerov`

```
    fRoky_mean=priemer(fRoky)
    fRoky_mean_2=fRoky_mean*fRoky_mean
    fSkoky_mean=priemer(fSkoky)
    print("x=",fRoky_mean," y=",fSkoky_mean)
    xxpom=[]
    xypom=[]
    for i in range(0,n):
        xxpom.append(fRoky[i]*fRoky[i])
        xypom.append(fRoky[i]*fSkoky[i])

    xx_mean=priemer(xxpom)
    xy_mean=priemer(xypom)

#    Kontrolny vypis hodnot
#    print("xx=",xx_mean," xy=",xy_mean)
#   Vypocet k a q
    k=(xy_mean-fRoky_mean*fSkoky_mean)/(xx_mean-
fRoky_mean_2)
    q=fSkoky_mean- k*fRoky_mean
    return k, q
```

Vo funkcii je použitá funkcia `mean` z matematickej knižnice pre výpočet priemeru.
Funkciu odskúšajte na zoznamoch Roky a Skoky.
Zapíšte získané výsledky k a q:

Je možné použiť aj vytvorenú funkciu `priemer` na experimentovanie. Vo funkcii priamka nahraďte funkciu `mean` funkciou `priemer` a upravenú funkciu odskúšajte na zoznamoch Roky a Skoky.
Zapíšte získané výsledky k a q a komentár:

7. Budete pracovať s knižnicou pylab, preto do programu je vložený príkaz

```
from pylab import *
```

Budete vykresľovať vzťah medzi hodnotami v poliach Roky a Skoky.
Do programu je vložený príkaz

```
figure()
```

pomocou ktorého je pripravený obrázok na kreslenie.

8. Obrázok bude mať štyri panely, ktoré kreslíme pomocou príkazu

```
subplot(počet riadkov, počet stĺpcov, poradové č. časti)
```

Do programu vložíme nasledujúce 3 príkazy, ktoré nám vykreslia prvú časť

```
subplot(2,2,1)
plot(Roky, Skoky, 'ro:')
```

```
xlabel('Rok')
ylabel('Skok [m]')
title('Rekordy v skoku do dialky')
```

Príkaz `show()` umiestnime až po vykreslení celého obrázka.

9. V druhom paneli je vykreslené všetko, čo bolo vykreslené v prvej časti a je doplnené vykreslenie trendovej priamky.
Je to možné urobiť doplnením príkazov, kde vypočítate parametre priamky a údaje na nej v daných rokoch.

```
[w0,w1]=priamka(pocetRokov,Roky,Skoky)

d_Rec_yy=[]
for i in range(0,pocetRokov):
    d_Rec_yy.append(w0*Roky[i]+w1)

plot(Roky, d_Rec_yy, 'bx-')
```

10. Do obrázka doplňte tretí panel, v ktorom vykreslíte prírastky rekordných skokov v jednotlivých rokoch. Aj pre prírastky určte trendovú priamku a vykreslíte ju.

Dôležitá časť je nasledujúca, kde sú vypočítané prírastky, parametre priamky a hodnoty na tejto priamke

```
deltaSk=[0]
for i in range(0,pocetRokov-1):
    deltaSk.append(Skoky[i+1]-Skoky[i])
```

Výpočet parametrov trendovej priamky prírastkov a jej vykreslenie (podobne ako v paneli 2)

Do obrázka vložte popisky a nadpis.
Pozor!
Príkaz `show()` musí byť na konci nášho programu.

11. Do obrázku vložíme štvrtý panel s informáciami o očakávanom rekorde v roku 2020.

```
subplot(2,2,4)

[w0,w1]=priamka(pocetRokov,Roky,Skoky)
ocakRek=w0*2020+w1;
text(0.1,0.8,['Očakávaný rekord v roku
2020:','{:6.3f}'.format(ocakRek)]);
```

	`axis([0, 1, 0, 1])`
	`title('Informácie'))`
	
	Úlohou je nájsť ďalšie vhodné informácie, ktoré by bolo vhodné vložiť do štvrtého panela.

	Záverečné hodnotenie a sebahodnotenie
O1.	Napíšte postup pri výpočte očakávaného rekordu (pri lineárnom trende) v skoku do výšky v roku 2030.
	Napíšte postup, pomocou ktorého sa určí o koľko viac by mal rekordman skočiť v skoku do výšky v roku 2030 oproti rekordu v roku 1991, ak uvažujeme lineárny trend pri rekordoch v skoku.
O2.	Viem, čo to znamená lineárny trend vývoja veličiny. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem
	Viem, ako je možné určiť chybu lineárneho trendu oproti skutočnosti. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem

7 MODELOVANIE NELINEÁRNYCH JAVOV

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika/ Modelovanie priestorového počúvania	ISCED3/3.-4. ročník GY Voliteľný predmet **Informatika v prírodných vedách** 2 vyučovacie hodiny
Ciele	
Žiakom osvojované vedomosti a zručnosti	***Žiakom rozvíjané spôsobilosti***
Vyjadrenie vzťahov medzi atribútmi pomocou polynomiálnych funkcií.	Hľadanie vzťahov medzi atribútmi popisujúcimi javy v prírode, resp. v jednoduchých prípadových štúdiách
Požiadavky na vstupné vedomosti a zručnosti	
Základná práca s počítačom, práca súbormi, poznať lineárne funkcie, vytvorenie a spustenie programov.	
Riešený didaktický problém	
Hľadanie závislosti medzi veličinami. Návrh modelu závislosti.	
Dominantné vyučovacie metódy a formy	***Príprava učiteľa a pomôcky***
Bádateľská metóda uplatnená pri hľadaní vzťahov a pri upravovaní programu	Pomôcky: 1) Pripravený program v jazyku Python – P2_Model3_ucitel.py 2) Pracovný list M3 a súbor obsahujúci kostru programu pre žiakov P2_Model3.py
Diagnostika splnenia vzdelávacích cieľov	
■ Žiacka diskusia ■ Grafický výstup modelovaných vzťahov ■ Predikcia očakávaných hodnôt	

7.1 Texty k výučbe

Pod **atribútmi** budeme rozumieť veličiny, pomocou ktorých je možné opísať nejaký **objekt** hmatateľný alebo nehmatateľný. Napríklad športový rekord je nehmatateľný objekt a môže byť opísaný atribútmi rok, dosiahnutý výkon a pod. Medzi atribútmi existujú vzťahy, ktoré sú skúmané. Jeden atribút (výstupný) môže byť závislý od iných atribútov (vstupných). **Model** v našom ponímaní bude vlastne vyjadrený vzťahom medzi vstupnými a výstupnými atribútmi nejakého objektu. Popis vzťahu sa obyčajne vyjadruje pomocou nejakej funkcie. My sa budeme zaoberať popisom vzťahov pomocou polynomiálnej funkcie, túto funkciu budeme tiež nazývať trendová krivka.

ÚLOHA O LOKALIZÁCII ZVUKOV PRI RUŠENÍ RUŠIČKAMI

Každodenne sa ocitáme v situáciách, v ktorých sme vystavení počúvaniu viacerých zvukov prichádzajúcich z rôznych smerov. Či už je to na rušnej ulici, alebo v miestnosti plnej rozprávajúcich sa ľudí počas spoločenskej akcie. Pomocou nášho priestorového sluchu dokážeme odhadnúť pozície

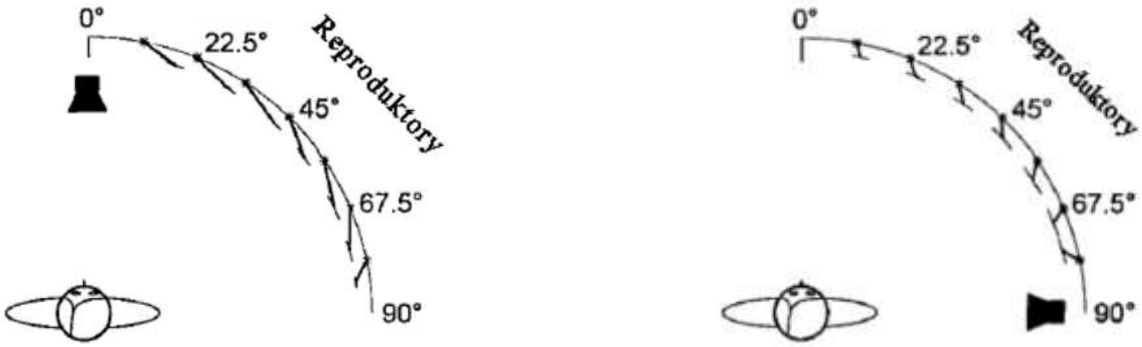
jednotlivých zdrojov zvukov (lokalizovať ich), čo nám pomáha ich lepšie vnímať. Našu schopnosť lokalizovať zvuk (ako aj sluchové vnímanie všeobecne) skúma psychoakustika.

Okrem toho mnohí ľudia, ktorí majú problém so sluchom a používajú načúvacie prístroje, vedia o tom, že dobre nastavenie parametrov prístroja je tá najdôležitejšia vec pre to, aby dobre počuli. A pre každého je potrebné iné nastavenie. Preto je dôležité poznať tiež, ako reagujeme na rôzne rušivé zvuky.

Viac informácii o takých experimentoch je možné nájsť v literatúre, napríklad (Kopčo, 2007).

Obrázok 13. USPORIADANIE REPRODUKTOROV V MIESTNOSTI, SUBJEKT UKAZUJE ŠPECIÁLNOU TYČOU, ODKIAĽ ZVUK PRICHÁDZA.

V experimente, ktorým sa budeme zaoberať, bol skúmaný vplyv rušivého zvuku – rušičky na lokalizáciu zvukov (aby sme ich odlíšili od zvukov rušičky, budeme ich nazývať cieľové zvuky).



Obrázok 14. USPORIADANIE REPRODUKTOROV (HVIEZDIČKY) A VYZNAČENIE POZÍCIE PREDNEJ (V POZÍCII 0°) A BOČNEJ RUŠIČKY (V POZÍCII 90°).

Cieľové zvuky aj rušička boli realizované ako krátke šumy z rovnomerne usporiadaných reproduktorov vo štvrťkruhu, ako je to znázornené hviezdičkami na obrázku č. 14.

Jeden pokus pozostával z prezentácie zvuku z rušičky a následne cieľového zvuku z náhodne vybraného reproduktora. Úlohou skúmaného človeka (subjektu) bolo ukázať, odkiaľ počuje cieľový zvuk a táto poloha bola zaznamenaná. Experiment pozostával z 2688 pokusov, medzi ktorými bola jedna šestina pokusov, keď pred cieľovým zvukom nezaznel zvuk z rušičky (referenčné pokusy). Porovnaním výsledkov z týchto pokusov bez rušičky s pokusmi s rušičkou vieme vyhodnotiť vplyv rušičky na lokalizáciu zvuku. V experimente bol manipulovaný aj čas medzi prezentáciou zvuku z rušičky a cieľového zvuku, v tejto úlohe sa zameriame len na merania, kde bol tento čas nastavený na 50 ms. Vybrali sme 4 subjekty, pre ktoré sme štatisticky vyhodnotili vplyv rušičky na ich odpovede pre každý z reproduktorov. Výsledky sú uvedené v tabuľke 1. Tieto údaje budeme používať v našej ďalšej práci, keď budeme modelovať chyby, ktorých sa subjekty dopúšťajú.

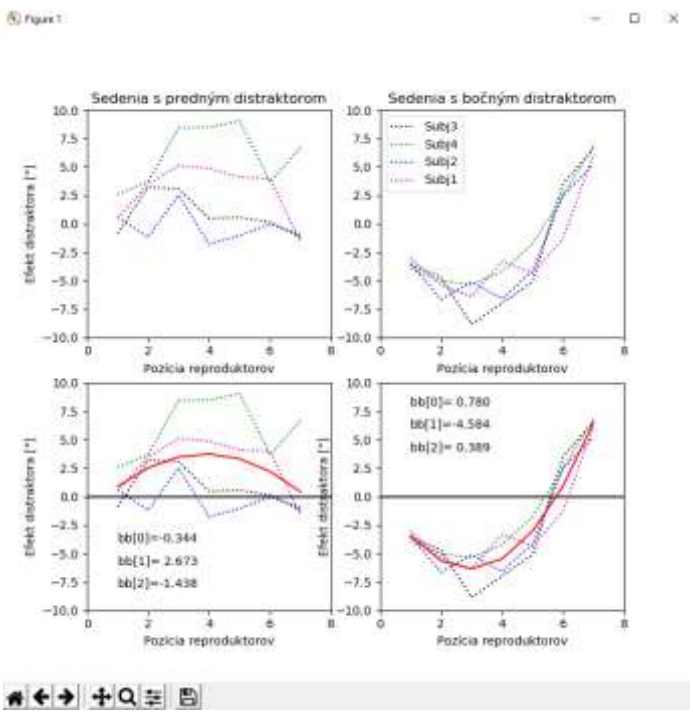
Tabuľka 5. Vplyv rušičky (rozdiel v odpovediach pri pokusoch s rušičkou a bez rušičky) u 4 subjektov. Hodnoty sú uvedené v stupňoch a kladné hodnoty znamenajú posun od rušičky doprava, záporné doľava.

	Reproduktory->	1	2	3	4	5	6	7
Predná rušička, PD	1. subjekt	-0.8992	2.5879	0.5891	0.4665	1.1900	4.3428	-5.5413
	2. subjekt	3.2087	3.6879	-1.2149	3.4914	12.3029	2.3595	1.5465
	3. subjekt	3.0810	8.4354	2.5316	5.1140	7.89520	7.0958	3.3680
	4. subjekt	0.4616	8.4873	-1.7762	4.8644	0.1417	10.0437	2.8163
Bočná rušička, BD	1. subjekt	-3.5939	-3.9389	-3.4682	-3.0279	-13.0278	-11.7808	-3.8368
	2. subjekt	-4.7074	-5.0305	-6.6589	-5.4059	-15.3588	-11.5746	-2.0842
	3. subjekt	-8.8796	-5.3403	-5.1207	-6.4005	-7.63960	-10.947	-2.3333
	4. subjekt	-6.9755	-4.2425	-6.5891	-3.3021	-3.1818	-9.0543	-1.0861

Potrebné informácie k programovaniu

Budeme používať knižnicu `numpy`, ktorá je základnou knižnicou pre vedecké výpočty. Táto knižnica pracuje s viacrozmernými poľami. Numpy pole je mriežka hodnôt, všetky hodnoty sú rovnakého typu a ich indexy sú nezáporné a celočíselné. Tvar poľa určuje n-tica celých čísel. Každý jej prvok určuje veľkosť poľa v danej dimenzii. Budeme používať nasledujúce funkcie z tejto knižnice

- `numpy.polyfit(x, y, deg)` – nájde polynóm stupňa deg, ktorý aproximuje údaje dané v poliach x, y (minimalizuje chybu pomocou minimalizácie súčtu najmenších kvadratických chýb); vypočítaný polynóm je `p(x)=p[0]*x**deg+p[1]*x**(deg-1)+ ... + p[deg];`
- `numpy.polyval(p, x)` – vypočíta hodnoty polynómu daného koeficientami v poli p pre hodnoty dané v poli x;
- `mean(x, axis=0)` – vypočíta priemer prvkov dvojrozmerného poľa x cez riadky; ak `axis=1`, počíta priemer cez stĺpce.



Obrázok 15. MODELOVANIE TRENDOVÝCH KRIVIEK. PLNÉ ČERVENÉ ČIARY ZNÁZORŇUJÚ KVADRATICKÉ TRENDOVÉ KRIVKY. BODKOVANÉ ČIARY SPÁJAJÚ ÚDAJE JEDOTLIVÝCH SUBJEKTOV. Predný distraktor predstavuje prednú rušičku, bočný predstavuje bočnú rušičku

Úlohy, ktoré tu chceme sledovať:

1) Graficky znázorniť vplyv rušičky ako funkciu pozície reproduktora (t. j. pozícia reproduktora bude na osi x, hodnota chyby na osi y), pre každý subjekt (separátne čiary v tom istom grafe), v jednom paneli pre predná rušička, v druhom paneli pre bočná rušička.
2) Graficky znázorniť priebeh týchto údajov pre priemerný subjekt.
3) Navrhnúť polynomiálne modely pre sledovanie vplyvu rušičky pre obe rušičky a modely vykresliť spolu s dátami (z bodu 2).
4) **Cieľový obrázok je č. 3**

7.2 Priebeh výučby (90 minút)

VYSVETLENIE (CCA 20 MIN.):

Zámer: Učiteľ vysvetlí priebeh experimentu a dôležitosť pozorovania odpovedí pri lokalizácii zvukov. Žiaci dostanú pracovný list M3 spolu s pripraveným programom `P2_Model3.py`, do ktoré doplňujú príkazy podľa pracovného listu. Pre učiteľa je pripravený vzorový doplnený program `P2_Model3_ucitel.py`

ZAPOJENIE (CCA 10 MIN.):

Zámer:

- Po oboznámení sa s problémom žiaci môžu hľadať ďalšie vzťahy medzi atribútmi,
- na základe známych vzťahov predpovedať výsledok v niektorých budúcich rokoch

CVIČENIE – DISKUTUJTE!

Je možné so žiakmi diskutovať o to, či poznajú ľudí, ktorí používajú načúvacie prístroje? Či sú títo ľudia s prístrojmi spokojní, atď.

Nechajte žiakom možnosť vyjadriť svoj názor na uvedený problém a diskusiu smerujte k výskumnej otázke.

CVIČENIE – OTÁZKY!

Keby nebol šum, určite by počúvanie bolo menej problémové. Ako vypýva šum z rušičiek na naše počutie?

SKÚMANIE (CCA 20 MIN.):

Zámer: Nech žiaci opíšu slovne trend vývoja podľa znázornených vzťahov.

ÚLOHA 1 – RIEŠTE!

Vykreslíte efekt rušičky pre každý subjekt inou farbou. V jednom paneli pre dáta s prednou rušičkou PD a v druhom paneli pre dáta s bočnou rušičkou BD.

Úloha je súčasťou pracovného listu M3.

Nechajte žiakom čas na prácu na programe.

VYSVETLENIE (CCA 10 MIN.):

Zámer: Formou dialógu pripomenieme možnosti zobrazovania pomocou príkazov Pylabu. Cieľom je polynomiálne zobrazenie.

ROZPRACOVANIE (CCA 20 MIN.):

Zámer: Uvedomiť si, že v danom probléme je možné hľadať a skúmať aj ďalšie vyhodnotenia.

ÚLOHA 2 – RIEŠTE!

1) Do programu je potrebné doplniť výpočet priemerného subjektu.
2) Pre priemerný subjekt vypočítať kvadratickú trendovú krivku, ktorú je potrebné spolu s pôvodnými dátami zobraziť v obrázku v treťom paneli pre dáta PD (predná rušička) a vo štvrtom paneli pre dáta BD (bočná rušička).

Kým žiaci pracujú na doplnení programu, priebežne kontrolujte, ako sa im darí riešenie úlohy.

ÚLOHA 3 – RIEŠTE!

Pre dáta z bočnej rušičky experimentujte s kubickou trendovou funkciou.

Poznámka: Ak sú niektorí šikovní žiaci hotoví s riešením problému, je možné orientovať ich na získanie iných informácií, napríklad modelovať údaje každého subjektu (ako vyzerá polynomiálna krivka jedného subjektu), a pod.

HODNOTENIE (CCA 10 MIN.):

Zámer: Žiaci majú hodnotiť, ako a čím im pomohol skúmaný problém. Hlavne, či budú vedieť samostatne postupovať pri riešení podobných úloh.

OTÁZKY NA HODNOTENIE A SAMOHODNOTENIE

1) Prvý typ otázok je na overenie, či niektoré jednoduché úlohy vedia riešiť.
2) Druhý typ otázok sa týka sebahodnotenia. Tu si žiak uvedomí, aké pojmy a učivo by mal ovládať. Odpovede žiakov je potrebné brať s rezervou (a niekedy aj overiť, aká je je skutočnosť).

7.3 Pracovný list M3

Nelineárne modelovanie

Pracovný list predstavuje postupné kroky k modifikácii programu na vykreslenie nelineárneho modelu pre sledovanie chýb, ktorých sa dopúšťajú ľudia pri určovaní pozície prichádzajúceho zvuku. Vedie tiež k experimentovaniu s jednotlivými príkazmi v programovacom jazyku Python, preto je potrebné pracovať v prostredí Pythonu a vždy spustiť program, keď si to pracovný list vyžaduje. Niektoré odpovede a pripomienky je možné písať do pracovného listu. Spolu s pracovným listom dostanete súbor `P2_Model3.py`. Do tohto súboru budete písať príkazy podľa pokynov pracovného listu.

1.	Z údajov, ktoré predstavujú efekty rušičky (distraktora) v akustickom experimente, sú vytvorené v programe dva zoznamy zoznamov, PD pre odpovede pri prednej rušičke, BD pre odpovede pri bočnej rušičke. Každý jednoduchý zoznam predstavuje odpovede jedného človeka v experimente. Poradové číslo odpovede v zozname je poradové číslo reproduktora v experimente. Hodnota vyjadruje chybu, ktorej sa človek dopustil vzhľadom na pozíciu reproduktora (vyjadrená v stupňoch).
	
	`PD=[[-0.8992, 3.2087, 3.0810, 0.4616, 0.5579, 0.1426, -1.1806],`
	`[ 2.5879, 3.6879, 8.4354, 8.4873, 9.0598, 3.6582, 6.7262],`
	`[ 0.5891,-1.2149, 2.5316,-1.7762,-1.0412,-0.0320, -0.9601],`
	`[ 0.4665, 3.4914, 5.1140, 4.8644, 4.1082, 3.9664, -1.7208]]`
	
	`BD=[[ -3.5939,-4.7074,-8.8796,-6.9755,-5.1312, 3.5165, 6.6183],`
	`[ -3.9389,-5.0305,-5.3403,-4.2425,-1.7692, 2.5176, 6.8339],`
	`[ -3.4682,-6.6589,-5.1207,-6.5891,-4.1486, 2.4976, 5.2183],`
	`[ -3.0279,-5.4059,-6.4005,-3.3021,-4.3535,-1.2749, 6.1751]]`

2.	Budete pracovať s knižnicou pylab, preto do programu je vložený príkaz
	`from pylab import *`
	
	Do programu je vložený príkaz
	`figure()`
	pomocou ktorého pripravíte obrázok na kreslenie. Obrázok bude mať štyri panely.
	Nezabudnite `show()` na konci.

3.	V prvom paneli obrázka sú vykreslené priebehy týchto dát (PD) pre každého človeka inou farbou. Príkazy sú uvedené v programe. Doplňte do druhého panelu vykreslenie údajov BD, podobne ako to je v prvom paneli. Napíšte, aké zmeny bolo treba urobiť:

3.	Pozrite údaje každého subjektu a napíšte, aké trendy vývoja sa dajú sledovať u jednotlivých subjektov v prvom a druhom paneli.

4.	Na pripojenie matematickej knižnice je použitý príkaz
	`import numpy as np`
	
	Budete hľadať trend v tvare kvadratickej funkcie a použijete funkciu
	`np.polyfit(x, b, 2),`
	ktorej parametre sú údaje v smere x-ovej osi, údaje v smere y-ovej osi a 2 znamená kvadratickú funkciu. Ak chceme použiť kubickú funkciu, 2 zmeníme na 3.
	
	Odskúšajte funkciu na údajoch PD pre prvý subjekt.
	Výsledok zapíšte:

5.	Vypočítajte priemerný subjekt a pre ten budete hľadať trend. Návrh na výpočet priemerného subjektu pre údaje PD použitím funkcií z matematickej knižnice
	
	`y = np.array(PD)`
	`b = np.mean(y, axis=0)`
	Vypíšte tieto údaje v konzole a sledujte, či vyjadrujú priemer. Napíšte komentár.

6.	Pripravíme údaje na x-ovej osi podobne
	`xxr = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0]`
	`xxb=np.array(xxr)`
	
	Zavoláme funkciu na pripravených dátach
	`bb = np.polyfit(xxb, b, 2)`
	
	Vypíšte hodnoty vypočítané v bb v konzole a napíšte, čo tieto hodnoty predstavujú, keď majú reprezentovať kvadratickú funkciu.

7.	Výpočet hodnôt trendovej funkcie pre jednotlivé reproduktory `yp = np.polyval(bb, xx)` V treťom paneli sú vykreslené pôvodne údaje od všetkých ľudí ako v prvom paneli a tiež trendová funkcia. V paneli sú uvedené hodnoty trendovej funkcie. Napíšte svoj komentár k trendovej funkcii, hlavne ako vystihuje údaje, na ktorých bola vytvorená.
8.	Vo štvrtom paneli vykreslite všetko ako v treťom paneli, ale pre údaje BD. Napíšte svoj komentár k trendovej funkcii, hlavne ako vystihuje údaje, na ktorých bola vytvorená.
9.	K údajom BD by bolo možné hľadať ďalšiu trendovú funkciu, napríklad polynóm tretieho stupňa. Dá sa použiť tá istá funkcia, ale zmeníme 2 na 3. Experimentujte s touto zmenou a svoje pozorovanie napíšte.
	Záverečné hodnotenie a sebahodnotenie
O1.	Napíšte príkazy, pomocou ktorých by ste chceli použiť trendovú funkciu tretieho stupňa pre údaje BP.

	Napíšte, čo vykonajú nasledujúce dva príkazy
	`bb = np.polyfit(xxb, b, 2)`
	`yp = polyval(bb, xx)`

O2.	Viem, čo to znamená kvadratický trend vývoja veličiny . Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem

	Viem, ako je možné určiť chybu kvadratického trendu oproti skutočnosti. Podčiarknite možnosť, ktorá najlepšie vystihne vašu situáciu. Viem Potrebujem pomoc Neviem

Pomôcka:

- `numpy.polyfit(x, y, deg)` – nájde polynóm stupňa deg, ktorý aproximuje údaje dané v poliach x, y (minimalizuje chybu pomocou minimalizácie súčtu najmenších kvadratických chýb); vypočítaný polynóm je `p(x)=p[0]*x**deg+p[1]*x**(deg-1)+ ... + p[deg];`
- `numpy.polyval(p, x)` – vypočíta hodnoty polynómu daného koeficientami v poli p pre hodnoty dané v poli x;
- `mean(x, axis=0)` – vypočíta priemer prvkov dvojrozmerného poľa x cez riadky; ak `axis=1`, počíta priemer cez stĺpce.

8 PRÁCA S DATABÁZAMI 1

Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika Práca s databázami	ISCED 3 / 3.ročník (1 dvojhodinovka/90 minút)
Ciele	
Žiakom osvojované vedomosti a zručnosti	*Žiakom rozvíjané spôsobilosti*
Vyhľadať biologické, chemické a geografické informácie v dostupných online databázach.	Formulovať otázku/problém. Experimentovať s pomôckami/s podporou softvéru. Zaznamenávať výsledky.
Požiadavky na vstupné vedomosti a zručnosti	
Práca s internetom a webovým prehliadačom, vyhľadávanie na webe, práca s tabuľkovým kalkulátorom.	
Riešený didaktický problém	
Žiaci často vyhľadávajú informácie na internete z rôzne dôveryhodných zdrojov. Pri vedeckej práci je však potrebné vedieť vyhľadávať aj v odborných online databázach, systematicky selektovať a triediť dáta a následne ich vhodným spôsobom vizualizovať.	
Dominantné vyučovacie metódy a formy	*Príprava učiteľa a pomôcky*
■ Interaktívna ukážka, riadené, nasmerované bádanie, metóda E-U-R, diskusia ■ Individuálna a skupinová práca, práca vo dvojiciach	■ PC s pripojením na internet, dataprojektor ■ pracovný list ■ pracovné súbory ***dna_sekvencie.txt*** a ***referencna_DNA.txt***
Diagnostika splnenia vzdelávacích cieľov	
■ Vypracovaný pracovný list.	

8.1 Texty k výučbe

Metodika je súčasťou série informatických metodík zameraných na prácu s databázami a predstavuje úvod do danej problematiky. Je rozdelená na tri časti v zmysle modelu EUR:

- **EVOKÁCIA** – formou práce v skupinách s pracovným listom si žiaci urobia prehľad svojich doterajších vedomostí, príp. vlastných skúseností z problematiky využívania rôznych typov databáz
- **UVEDOMENIE SI VÝZNAMU** – v rámci tejto časti učiteľ facilituje žiacke bádanie pri ich práci s odbornými online databázami podľa postupu v pracovnom liste
- **REFLEXIA** – žiaci by mali byť schopní na základe predošlých aktivít porovnať možnosti rôznych druhov databáz a posúdiť ich využitie pri vedeckej práci.

Databáza (Šaling, 1977) je komplex súborov dát, ktoré sú navzájom v určitom logickom vzťahu sú prístupné pomocou systému riadenia bázy dát. V užšom zmysle je možné si predstaviť databázu

ako tabuľku, príp. zoznam, ktorý obsahuje údaje uložené v stĺpcoch (tzv. **polia databázy**) a v riadkoch (tzv. **záznamy**).

Údaje môžu byť usporiadané podľa zvoleného kritéria. Pri širšom chápaní môže databáza zahŕňať aj iné objekty, nastavenia, zostavy a výbery. Údaje v databáze môžu byť rôzneho typu – text, číslo, mena, automatické číslo, dátum, logický typ a pod. Medzi základné operácie s databázou patria:

- vkladanie a úpravy údajov,
- prezeranie databázy a tvorba výstupov,
- usporiadanie údajov,
- vyhľadávanie a filtrovanie údajov,
- matematické a štatistické operácie.

V praxi sa často môžeme stretnúť s databázami prispôsobenými pre konkrétne oblasti použitia, čoho príkladom je aj databáza **National Center for Biotechnology Information (NCBI)**, dostupná online na adrese https://www.ncbi.nlm.nih.gov/, ktorá nám v tejto metodike bude slúžiť na ukážku širokého spektra rôznych zdrojov údajov a špecifických nástrojov pre medicinu, biológiu a príbuzné oblasti. Podobné možnosti ponúkajú však aj ďalšie databázy s chemickými alebo geografickými informáciami. S praktickými aplikáciami týchto databáz sa žiaci neskôr stretnú v predmetových metodikách (napr. pre geografiu).

8.2 Priebeh výučby (90 minút)

EVOKÁCIA

Nakoľko téma databáz vo všeobecno-vzdelávacom predmete Informatika je len okrajovou témou, začneme hodinu evokáciou príkladov z bežného života, kde sa žiaci už s nejakou podobou databáz stretli – môžu to byť informačné systémy v škole, evidencia zákazníkov v obchodoch, či online databáza videí na YouTube. Žiakov rozdelíme do skupín, v ktorých pracujú bez počítača s pripravenými úlohami z pracovných listov (úloha 1 a 2).

Databáza predstavuje organizovanú množinu dát. Zamyslite sa a doplňte do tabuľky po piatich príkladoch pre rôzne dáta, ktoré môžu obsahovať databázy v týchto oblastiach:

ÚLOHA 1 – RIEŠTE!

Databáza predstavuje organizovanú množinu dát. Zamyslite sa a doplňte do tabuľky po piatich príkladoch pre rôzne dáta, ktoré môžu obsahovať databázy v týchto oblastiach:

POLÍCIA	*ŠKOLA*	*OBCHOD*	*KNIŽNICA*

ÚLOHA 2 – RIEŠTE!

Pozrite sa na YOUTUBE ako na databázu a pokúste sa navrhnúť jej štruktúru. Vyhľadajte video k svojej obľúbenej piesni – ktoré informácie sú o ňom (okrem názvu) ešte dostupné

Názov videa						

Spoločne so žiakmi zhrnieme výsledky riešených úloh, pričom zdôrazníme, že vďaka digitálnym technológiám je možné v súčasnosti spracovať a uchovať obrovské množstvo rozmanitých dát aj v prírodných vedách, čo dynamicky ovplyvňuje výskum a napredovanie v tejto oblasti. Prostredníctvom internetu sme získali jednoduchý prístup k najnovším biologickým a chemickým dátam, ktoré sa stali verejne dostupnými – nie je preto problém zistiť potrebné informácie o génoch a genetických poruchách, či preskúmať chemickú štruktúru zlúčenín, ich využitie v praxi, či prípadnú toxicitu a bezpečnostné riziká. Geografické dáta ako satelitné snímky, či výsledky meteorologických meraní a pozorovaní ponúkajú v spojení s vhodnými vizualizačnými nástrojmi cenné informácie pre skúmanie Zeme. **Databázy** umožňujú vedcom ukladať, organizovať, spracúvať a získavať potrebné informácie. Preto naším cieľom bude oboznámiť sa s databázami ako nástrojmi pre zbieranie a usporiadanie informácií – preskúmame rôzne podoby dát a existujúcich dátových súborov, prakticky vyskúšame základné možnosti vyhľadávania dát v online databázach, ako aj ich analýzu a vizualizáciu.

UVEDOMENIE SI VÝZNAMU

V tejto časti metodiky budú žiaci postupne samostatne riešiť úlohy z pracovného, pričom učiteľ facilituje žiacke skúmanie. Zameriame sa na databázu National Center for Biotechnology Information (NCBI), v súčasnosti jednu z najrozsiahlejších bioinformatických databáz dostupnú online. Objekt, o ktorom sa pokúsime vyhľadať v tejto databáze rôzne informácie, je gén BRCA1, ktorý kóduje proteín, ktorý sa podieľa na oprave DNA, preto je známy aj ako tzv. nádorový supresor („potláčač“), keďže pomáha predchádzať formovaniu nádorov prsníka. Práve mutácie na géne BRCA1 môžu viesť k narušeniu normálnej funkcie proteínu kódovanému BRCA1 a teda k potenciálnemu vzniku rakoviny. Žiakom túto informáciu na začiatku neposkytneme, necháme ich 2-3 minúty skúmať na internete, čo sa im o BRCA1 podarí nájsť, pričom nie je dôležité, aby žiaci zistili detailné informácie o géne BRCA1 – skôr ich usmerníme, aby sa pokúsili aspoň približne identifikovať, čo sa pod týmto označením skrýva (t.j. že je to gén) a aby si všimli, aké rôzne druhy informácií (texty, obrázky, prehľady vedeckých článkov a časopisov a pod.) našli. Internet poskytuje naozaj veľké množstvo informácií, no vyhľadávať týmto spôsobom by bolo pri práci vedca pomerne neefektívne a komplikované.

ÚLOHA 3 – RIEŠTE!

Pomocou internetu zistite, čo je to BRCA1. Aké rôzne druhy informácií o BRCA1 sa Vám podarilo nájsť? Z akých rôznych webových stránok ste informácie čerpali?

Spoločne so žiakmi navštívime stránku **National Center for Biotechnology Information (NCBI)**, kde pomocou úloh 4 (v pracovnom liste sú rozdelené) vo dvojiciach preskúmajú rôzne druhy databáz, ktoré na tejto stránke sú k dispozícii. Opäť upriamime ich aktivitu na uvedomenie si množstva rôznych informácií, s ktorými je možné pracovať na jednom mieste.

ÚLOHA 4 – RIEŠTE!

Jednu z najrozsiahlejších bioinformatických databáz na internete poskytuje **National Center for Biotechnology Information (NCBI)** a je dostupná na adrese https://www.ncbi.nlm.nih.gov/

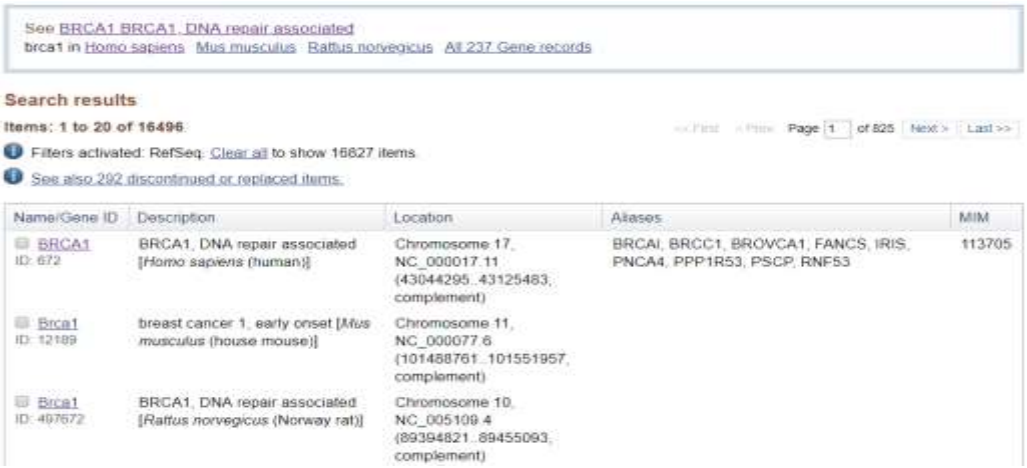
Pomocou vyhľadávacieho stroja **Entrez** vyhľadajte všetky dostupné záznamy o BRCA1. V koľkých databázach sa hľadali o ňom informácie?

V databáze **Nucleotide** sú uvedené DNA sekvencie nahrané do NCBI databázy. Koľkokrát do nej bola nahraná DNA sekvencia BRCA1?

V databáze **PubMed** sú evidované články o príslušnom géne alebo chorobe. Koľkokrát sa BRCA1 vyskytlo.

Pre štúdium génov, ako napr. BRCA1, potrebujú vedci a lekári poznať tzv. **referenčnú sekvenciu** pre príslušný gén. Vráťte sa teda na úvodnú stránku NCBI a vo vyhľadávacom stroji **Entrez** nastavte vyhľadávanie génov (v políčku „All databases“ zmeňte voľbu na „Gene“) a vyhľadajte BRCA1. Koľko rôznych záznamov pre BRCA1 bolo nájdených?

V ďalšej časti sa pokúsime vyhľadať konkrétne informácie o géne BRCA1, ktoré neskôr využijeme pri praktickej diagnostickej úlohe o mutácii sledovaného génu, na základe čoho budeme môcť priamo online otestovať génové sekvencie niekoľkých ľudí a vyhľadať tých, u ktorých je vyššie riziko vzniku rakoviny (úloha v pracovnom liste).

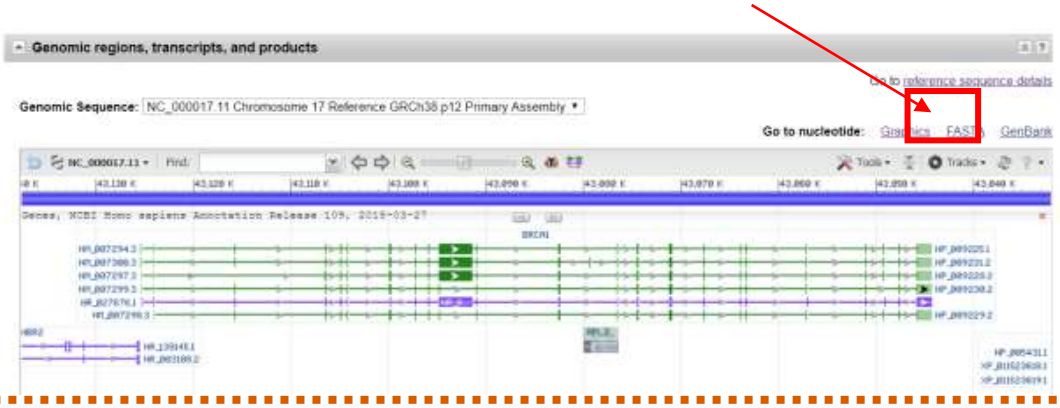


ÚLOHA 5 – RIEŠTE!

Všimnite si, že vyhľadávanie prebehlo v záznamoch rôznych organizmov. Aké zvieratá sú schované pod názvami **mus musculus**, **rattus norvegicus** alebo **canis lupus familiaris?**

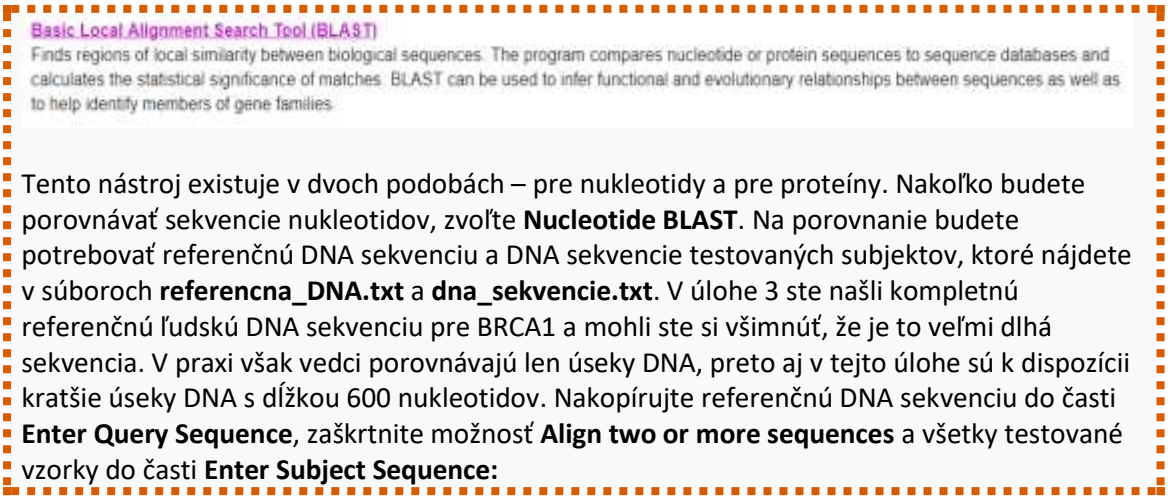
Aký je číselný identifikátor génu (t.j. „GeneID“) pre BRCA1 u človeka? Na ktorom chromozóme sa tento gén nachádza?

Kliknite na príslušný záznam, aby ste získali bližšie informácie o príslušnom géne BRCA1 u človeka a prezrite si, aké rôzne informácie ste získali. V časti „Genomic regions, transcriptions, and products“ nájdite referenčnú DNA sekvenciu vo formáte FASTA.

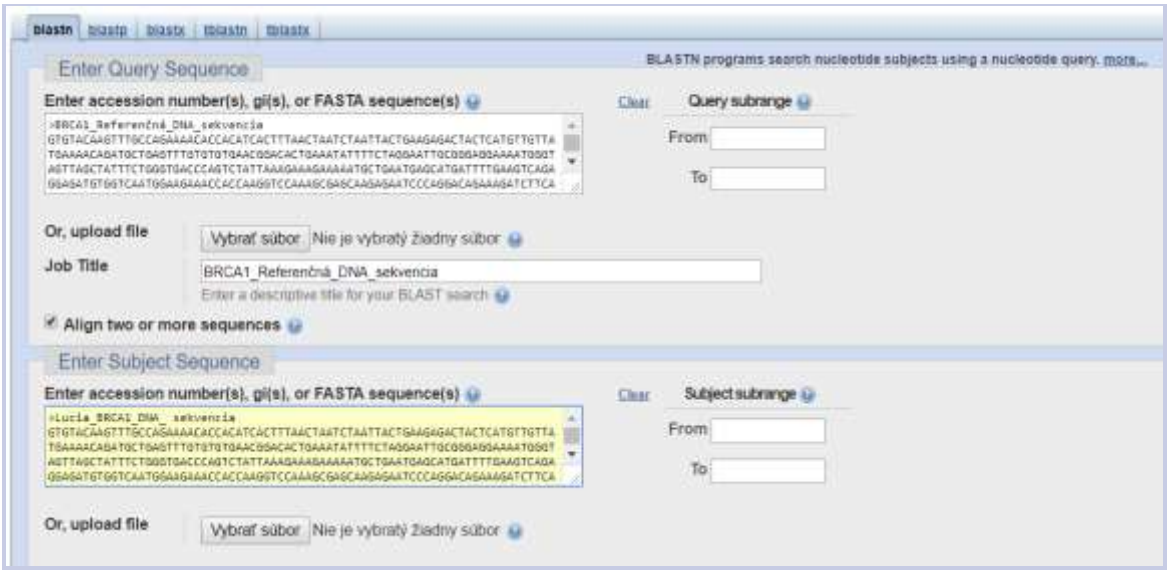


ÚLOHA 6 – RIEŠTE!

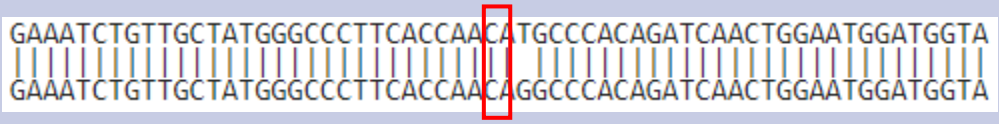
Aby bunky pracovali správne, musia byť schopné opravovať chyby na svojej DNA, ktoré vznikajú keď sa DNA kopíruje alebo keď sa DNA poškodí napr. pôsobením chemikálií alebo žiarenia. Ak poškodenia DNA nie sú opravené, môžu viesť k rakovine. Gén BRCA1 kóduje proteín, ktorý sa podieľa na oprave DNA, preto je známy aj ako tzv. nádorový supresor („potláčač“), keďže pomáha predchádzať formovaniu nádorov. Mutácie na géne BRCA1 môžu viesť k narušeniu normálnej funkcie proteínu kódovanému BRCA1 a teda k potenciál-nemu vzniku rakoviny. Na porovnávanie DNA sekvencií slúži nástroj **BLAST** (**B**asic **L**ocal **A**lignment **S**earch **T**ool), ktorý je dostupný cez úvodnú stránku NCBI – nájdete ho buď medzi obľúbenými nástrojmi (v menu vpravo) alebo cez ľavé menu v časti „All resources“ a v záložke „Tools”.



Tento nástroj existuje v dvoch podobách – pre nukleotidy a pre proteíny. Nakoľko budete porovnávať sekvencie nukleotidov, zvoľte **Nucleotide BLAST**. Na porovnanie budete potrebovať referenčnú DNA sekvenciu a DNA sekvencie testovaných subjektov, ktoré nájdete v súboroch **referencna_DNA.txt** a **dna_sekvencie.txt**. V úlohe 3 ste našli kompletnú referenčnú ľudskú DNA sekvenciu pre BRCA1 a mohli ste si všimnúť, že je to veľmi dlhá sekvencia. V praxi však vedci porovnávajú len úseky DNA, preto aj v tejto úlohe sú k dispozícii kratšie úseky DNA s dĺžkou 600 nukleotidov. Nakopírujte referenčnú DNA sekvenciu do časti **Enter Query Sequence**, zaškrtnite možnosť **Align two or more sequences** a všetky testované vzorky do časti **Enter Subject Sequence:**



*V dolnej časti formulára zaškrtnite voľbu „Show results in a new window" a kliknutím na modré tlačidlo **BLAST** spustíte testovanie. Vo výsledkoch si prezrite zarovnané a porovnané sekvencie DNA jednotlivých testovaných osôb s referenčnou vzorkou. Rovnaké nukleotidy sú spojené čiarou, ale u nerovnakých nukleotidov čiara chýba:*



U ktorých testovaných osôb neboli žiadne zmeny v DNA? Nájdite u ostatných miesta v sekvencii DNA, kde došlo k mutácii!

Na nástrojoch NCBI sme si ukázali, že verejne dostupné online biologické databázy môžu obsahovať viaceré rôzne databázy, obsahujú vyhľadávacie nástroje, ktorými môžeme získať veľké množstvo výsledkov a doplňujúcich informácií. Taktiež disponujú užitočnými nástrojmi, ktoré umožňujú vyhľadávanie aj pomocou presných DNA sekvencií, pričom tieto všetky nástroje sú dostupné zadarmo.

V úlohe 13 pracovného listu sa vrátime k myšlienke vyhľadávania informácií na internete - tentokrát o gene HFE. Nebudeme skúmať jeho funkciu, ani spôsob fungovania, ale na Wikipédii

pomocou odkazov v infoboxe o tomto géne vyhľadáme pôvodné databázy, odkiaľ boli čerpané informácie (niektoré odkazy žiakov dovedú na stránku NCBI, avšak nájdu a iné databázy ako napr. **Protein Data Bank in Europe (PDBe)**, databázu **UniProt** a iné). Podobné možnosti ponúkajú aj ďalšie databázy s chemickými alebo geografickými informáciami (ďalšie úlohy).

ÚLOHA 7 – RIEŠTE!

Vyhľadajte informácie o géne HFE na **Wikipédii** a preskúmajte stránky, z ktorých boli čerpané informácie v jeho infoboxe:

ÚLOHA 8 – RIEŠTE!

Preskúmajte informácie o nitroglyceríne (anglicky „nitroglycerine“) cez databázu **ChemSpider** dostupnú na adrese http://www.chemspider.com/ . Aké rôzne druhy informácií ste našli?

ÚLOHA 9 – RIEŠTE!

Vytvorte v tabuľkovom kalkulátore tabuľku a doplňte do nej potrebné informácie pomocou databázy **ChemSpider:**

Vzorec	*Názov (angl.)*	*3D model*	*plota topenia*	*Rozpustnosť*
$KMnO_4$				
SiO_2				
HCOOH				
$CaCO_3$				
NaOH				

ÚLOHA 10 – RIEŠTE!

Preskúmajte informácie poskytované na stránke **SHMÚ** na adrese http://www.shmu.sk/ - ich meteorologické, hydrologické, klimatologické spravodajstvo, ako aj spravodajstvo kvality ovzdušia. Podľa aktuálnych údajov o počasí vytvorte v tabuľkovom kalkulátore prehľad teplôt, rýchlosti vetra a oblačnosti pre jednotlivé slovenské mestá:

MESTO	*TEPLOTA [°C]*	*RÝCHLOSŤ VETRA [m/s]*	*OBLAČNOSŤ*

ÚLOHA 11 – RIEŠTE!

Pomocou inte Pomocou doplnku **Microsoft 3D mapy pre Excel** vytvorte v tabuľkovom kalkulátore 3D mapu aktuálnych teplôt na Slovensku – tutoriál, ako na to, nájdete na stránke https://bit.ly/2Jrvu0M

REFLEXIA

Na záver hodiny vyzveme žiakov na zhrnutie a reflexiu poznatkov, ktoré počas hodiny získali. Úlohy 17 a 18 z pracovného listu riešia samostatne a následne frontálnou metódou postupne prejdeme ich poznámky a postrehy z hodiny.

ÚLOHA 12 – RIEŠTE!

Na základe predchádzajúcich úloh doplňte:

PRÍKLADY ODBORNÝCH ONLINE DATABÁZ:	*ODBORNÉ ONLINE DATABÁZY UMOŽŇUJÚ:*	*VÝHODY ONLINE DATABÁZ:*

ÚLOHA 13 – ODPOVEDZTE!

Na základe predchádzajúcich úloh doplňte:

SOM VEDEL/A UŽ PREDTÝM:	*ČO SOM SA NAUČIL/A NOVÉ:*	*BY SOM SA EŠTE CHCEL/A NAUČIŤ:*

8.3 Pracovný list D1

Práca s databázami 1

Úloha 1. *Databáza predstavuje organizovanú množinu dát. Zamyslite sa a doplňte do tabuľky po piatich príkladoch na rôzne dáta, ktoré môžu obsahovať databázy v týchto oblastiach:*

POLÍCIA	***ŠKOLA***	***OBCHOD***	***KNIŽNICA***

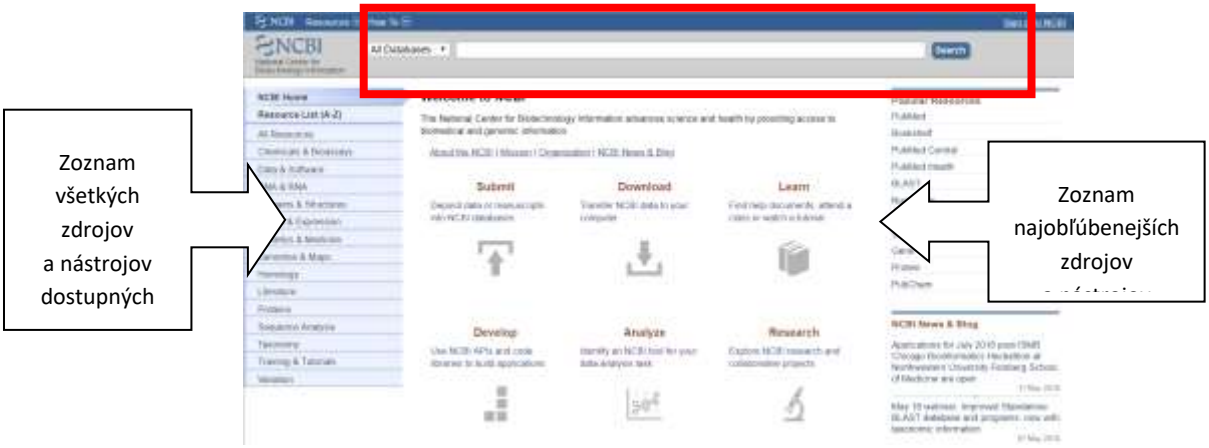
Úloha 2. *Pozrite sa na YOUTUBE ako na databázu a pokúste sa navrhnúť jej štruktúru. Vyhľadajte video k svojej obľúbenej piesni – ktoré informácie sú o ňom (okrem názvu) ešte dostupné?*

Názov videa						

Úloha 3. *Pomocou internetu zistite, čo je to BRCA1. Aké rôzne druhy informácií o BRCA1 sa Vám podarilo nájsť? Z akých rôznych webových stránok ste informácie čerpali?*

Úloha 4. *Jednu z najrozsiahlejších bioinformatických databáz na internete poskytuje* ***National Center for Biotechnology Information (NCBI)*** *a je dostupná na adrese https://www.ncbi.nlm.nih.gov/ :*

Vyhľadávací stroj
Entrez

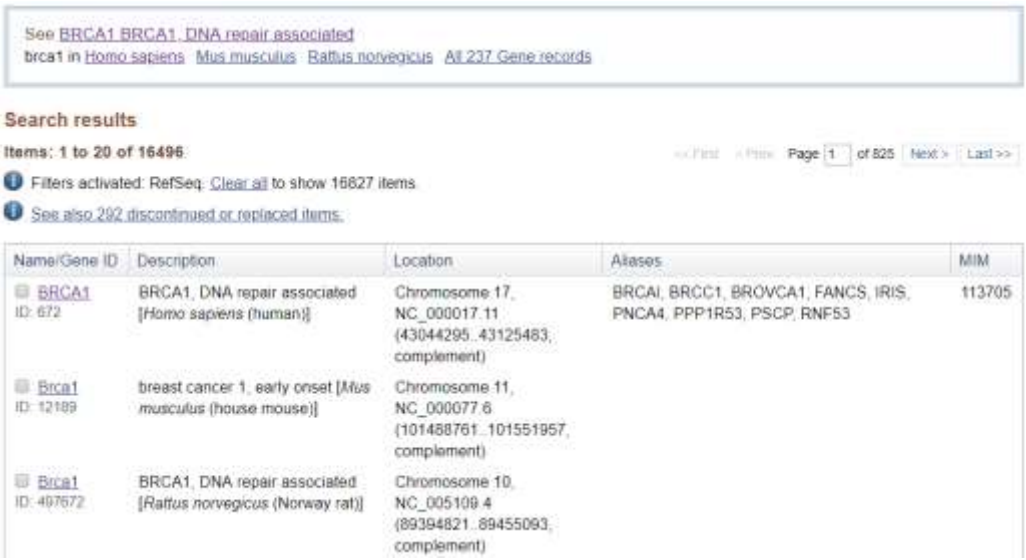


Úloha 5. *Pomocou vyhľadávacieho stroja* ***Entrez*** *vyhľadajte všetky dostupné záznamy o BRCA1. V koľkých databázach sa hľadali o ňom informácie?*

Úloha 6. *V databáze* ***Nucleotide*** *sú uvedené DNA sekvencie nahrané do NCBI databázy. Koľkokrát do nej bola nahraná DNA sekvencia BRCA1?*

Úloha 7. *V databáze* ***PubMed*** *sú evidované články o príslušnom géne alebo chorobe. Koľkokrát sa BRCA1 vyskytlo v odborných článkoch?*

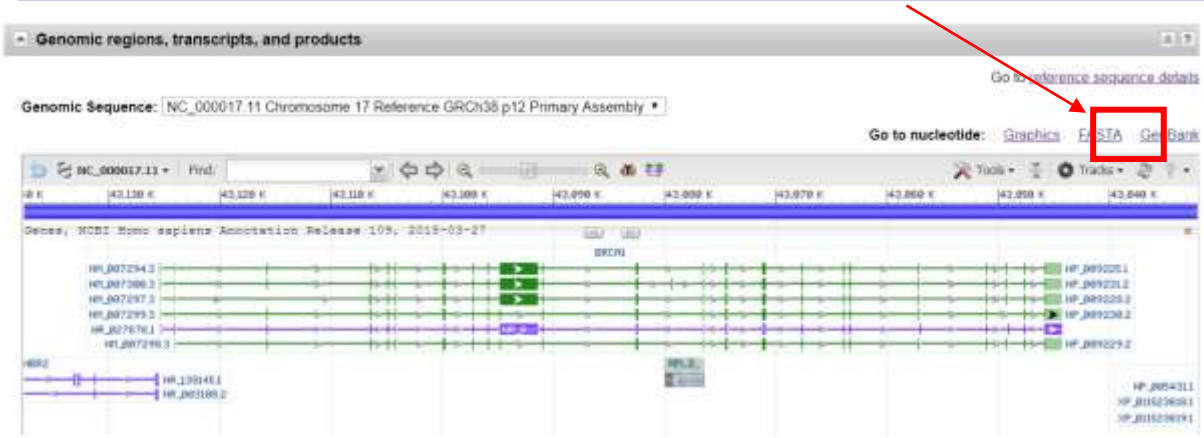
Úloha 8. *Pre štúdium génov, ako napr. BRCA1, potrebujú vedci a lekári poznať tzv.* ***referenčnú sekvenciu*** *pre príslušný gén. Vráťte sa teda na úvodnú stránku NCBI a vo vyhľadávacom stroji* ***Entrez*** *nastavte vyhľadávanie génov (v políčku „All databases“ zmeňte voľbu na „Gene“) a vyhľadajte BRCA1. Koľko rôznych záznamov pre BRCA1 bolo nájdených?*



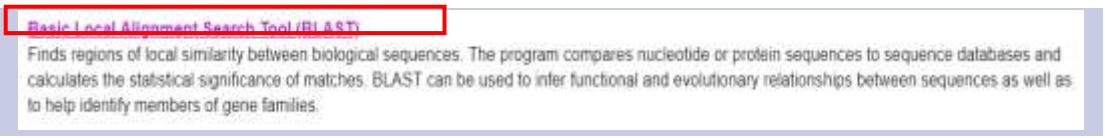
Úloha 9. *Všimnite si, že vyhľadávanie prebehlo v záznamoch rôznych organizmov. Aké zvieratá sú schované pod názvami* ***mus musculus***, ***rattus norvegicus*** *alebo* ***canis lupus familiaris****?*

Úloha 10. *Aký je číselný identifikátor génu (t.j. „GeneID“) pre BRCA1 u človeka? Na ktorom chromozóme sa tento gén nachádza?*

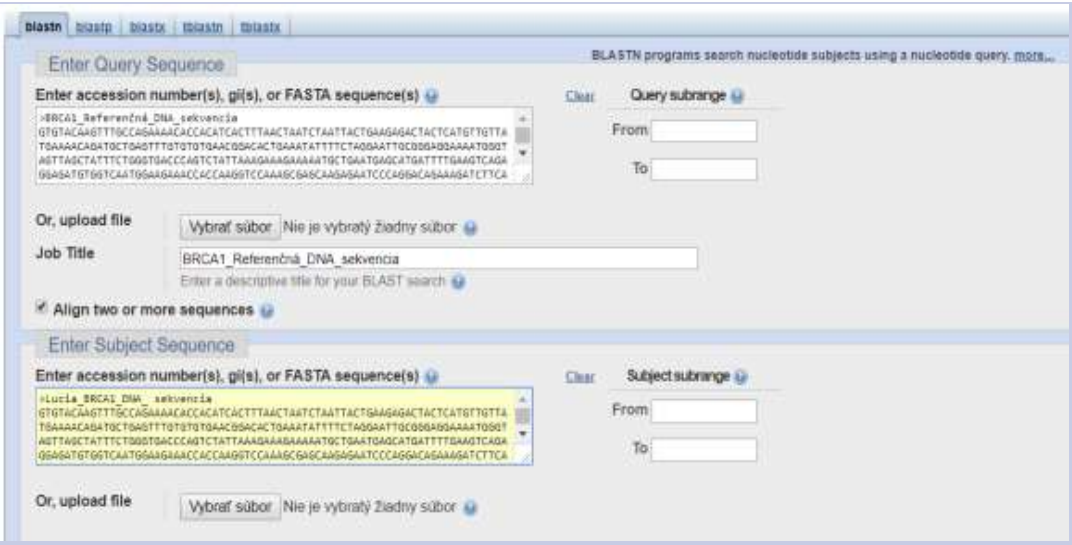
Úloha 11. *Kliknite na príslušný záznam, aby ste získali bližšie informácie o príslušnom géne BRCA1 u človeka a prezrite si, aké rôzne informácie ste získali. V časti „Genomic regions, transcriptions, and products“ nájdite referenčnú DNA sekvenciu vo formáte* ***FASTA****:*



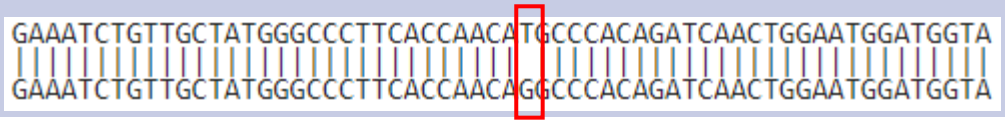
Úloha 12. *Aby bunky pracovali správne, musia byť schopné opravovať chyby na svojej DNA, ktoré vznikajú keď sa DNA kopíruje alebo keď sa DNA poškodí napr. pôsobením chemikálií alebo žiarenia. Ak poškodenia DNA nie sú opravené, môžu viesť k rakovine. Gén BRCA1 kóduje proteín, ktorý sa podieľa na oprave DNA, preto je známy aj ako tzv. nádorový supresor („potláčač"), keďže pomáha predchádzať formovaniu nádorov. Mutácie na géne BRCA1 môžu viesť k narušeniu normálnej funkcie proteínu kódovanému BRCA1 a teda k potenciálnemu vzniku rakoviny. Na porovnávanie DNA sekvencií slúži nástroj* ***BLAST*** *(**B**asic **L**ocal **A**lignment **S**earch **T**ool), ktorý je dostupný cez úvodnú stránku NCBI – nájdete ho buď medzi obľúbenými nástrojmi (v menu vpravo) alebo cez ľavé menu v časti „All resources" a v záložke „Tools":*



Tento nástroj existuje v dvoch podobách – pre nukleotidy a pre proteíny. Nakoľko budete porovnávať sekvencie nukleotidov, zvoľte ***Nucleotide BLAST****. Na porovnanie budete potrebovať referenčnú DNA sekvenciu a DNA sekvencie testovaných subjektov, ktoré nájdete v súboroch* ***referencna_DNA.txt*** *a* ***dna_sekvencie.txt****. V úlohe 3 ste našli kompletnú referenčnú ľudskú DNA sekvenciu pre BRCA1 a mohli ste si všimnúť, že je to veľmi dlhá sekvencia. V praxi však vedci porovnávajú len úseky DNA, preto aj v tejto úlohe sú k dispozícii kratšie úseky DNA s dĺžkou 600 nukleotidov. Nakopírujte referenčnú DNA sekvenciu do časti* ***Enter Query Sequence****, zaškrtnite možnosť* ***Align two or more sequences*** *a všetky testované vzorky do časti* ***Enter Subject Sequence****:*



V dolnej časti formulára zaškrtnite voľbu „Show results in a new window" a kliknutím na modré tlačidlo ***BLAST*** *spustíte testovanie. Vo výsledkoch si prezrite zarovnané a porovnané sekvencie DNA jednotlivých testovaných osôb s referenčnou vzorkou. Rovnaké nukleotidy sú spojené čiarou, ale u nerovnakých nukleotidov čiara chýba:*



U ktorých testovaných osôb neboli žiadne zmeny v DNA? Nájdite u ostatných miesta v sekvencii DNA, kde došlo k mutácii!

Úloha 13. *Vyhľadajte informácie o géne HFE na* ***Wikipédii*** *a preskúmajte stránky, z ktorých boli čerpané informácie v jeho infoboxe:*

Úloha 14. *Preskúmajte informácie o nitroglyceríne (anglicky „nitroglycerine“) cez databázu* ***ChemSpider*** *dostupnú na adrese http://www.chemspider.com/ . Aké rôzne druhy informácií ste našli?*

Úloha 15. *Vytvorte v tabuľkovom kalkulátore tabuľku a doplňte do nej potrebné informácie pomocou databázy* ***ChemSpider:***

Vzorec	***Názov (angl.)***	***3D model***	***Teplota topenia***	***Rozpustnosť***
$KMnO_4$				
SiO_2				
HCOOH				
$CaCO_3$				
NaOH				

Úloha 16. *Preskúmajte informácie poskytované na stránke* ***SHMÚ*** *na adrese http://www.shmu.sk/ - ich meteorologické, hydrologické, klimatologické spravodajstvo, ako aj spravodajstvo kvality ovzdušia. Podľa aktuálnych údajov o počasí vytvorte v tabuľkovom kalkulátore prehľad teplôt, rýchlosti vetra a oblačnosti pre jednotlivé slovenské mestá:*

MESTO	***TEPLOTA [°C]***	***RÝCHLOSŤ VETRA [m/s]***	***OBLAČNOSŤ***
...	...	...	...

Pomocou doplnku ***Microsoft 3D mapy pre Excel*** *vytvorte v tabuľkovom kalkulátore 3D mapu aktuálnych teplôt na Slovensku – tutoriál, ako na to, nájdete na stránke https://bit.ly/2Jrvu0M.*

Úloha 17. *Na základe predchádzajúcich úloh doplňte:*

PRÍKLADY ODBORNÝCH ONLINE DATABÁZ:	***ODBORNÉ ONLINE DATABÁZY UMOŽŇUJÚ:***	***VÝHODY ONLINE DATABÁZ:***

Úloha 18. *Na základe predchádzajúcich úloh doplňte:*

ČO SOM VEDEL/A UŽ PREDTÝM:	***ČO SOM SA NAUČIL/A NOVÉ:***	***ČO BY SOM SA EŠTE CHCEL/A NAUČIŤ:***

9 PRÁCA S DATABÁZAMI 2

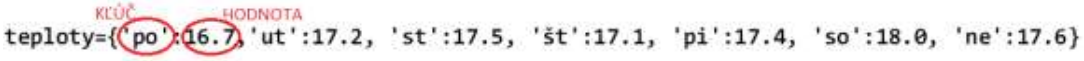
Tematický celok / Téma	*ISCED / Odporúčaný ročník*
Informatika Práca s databázami	ISCED 3 / 3.ročník (1 dvojhodinovka/90 minút)
Ciele	
Žiakom osvojované vedomosti a zručnosti	***Žiakom rozvíjané spôsobilosti***
Navrhnúť a vytvoriť vlastnú štruktúro-vanú reprezentáciu dát pomocou dátového typu slovník v programovacom jazyku Python, vkladať, modifikovať a vyhľadávať v nej informácie.	Formulovať otázku/problém. Konštruovať model a manipulovať s ním pomocou softvéru. Zisťovať hodnoty premenných. Zaznamenávať výsledky.
Požiadavky na vstupné vedomosti a zručnosti	
Základy programovania v jazyku Python (verzia 3.x).	
Riešený didaktický problém	
Pri uchovávaní dát rôzneho typu je potrebné v Pythone použiť vhodný dátový typ – okrem zoznamov sa ponúka elegantnejšia forma prostredníctvom slovníkov, ktoré využíva aj napr. BioPython. Je však potrebné postupne a systematický pristupovať k práci s týmto dátovým typom, kde budú potrebné zručnosti, ktoré už žiaci majú z práce s číslami, reťazcami a zoznamami.	
Dominantné vyučovacie metódy a formy	***Príprava učiteľa a pomôcky***
■ Interaktívna ukážka, riadené, nasmerované bádanie, metóda E-U-R, diskusia ■ Individuálna práca, práca vo dvojiciach	■ PC s pripojením na internet, dataprojektor ■ lokálna inštalácia jazyka Python (verzia 3.x) ■ pracovný list ■ pracovné súbory ***merania.py***, ***chemia.py*** a ***aminokyseliny.py*** ■ pracovné súbory (pre projekt) ***dna_sekvencie.txt*** a ***referencna_DNA.txt***
Diagnostika splnenia vzdelávacích cieľov	
■ Vypracovaný pracovný list.	

9.1 Texty k výučbe

Metodika je súčasťou série informatických metodík zameraných na prácu s databázami. Tematicky nadväzuje na základy programovania v jazyku Python, s ktorým sa žiaci mohli stretnúť v rámci hodín informatiky a ponúka nadstavbovú tému – prácu so slovníkmi ako vhodným dátovým typom na efektívne uchovávanie informácií rôzneho typu vo formáte **KĽÚČ:HODNOTA**. Je rozdelená na tri časti v zmysle modelu EUR:

1) časť **EVOKÁCIA** – formou práce v skupinách s pracovným listom si žiaci urobia prehľad svojich doterajších vedomostí, príp. vlastných skúseností z problematiky programovania v Pythone a využívania rôznych dátových typov (číslo, reťazec, zoznam)
2) časť **UVEDOMENIE SI VÝZNAMU** – v rámci tejto časti učiteľ facilituje žiacke bádanie pri ich práci s dátovým typom slovník podľa postupu v pracovnom liste
3) časť **REFLEXIA** – žiaci by mali byť schopní na základe predošlých aktivít porovnať možnosti a vlastnosti dátového typu slovník, posúdiť jeho využitie pri vedeckej práci a naprogramovať vlastný projekt využívajúci slovník

Základné dátové typy, ktoré používa jazyk Python, sú celočíselné typy (**int** a **bool**), typy s pohyblivou rádovou čiarkou (**float**, ale aj **complex** a **decimal**) a reťazce (**str**). Okrem týchto dátových typov máme možnosť zhromažďovať dátové prvky pomocou kolekcií, pre ktoré existujú v Pythone dátové typy predstavujúce postupnosti, predovšetkým zoznamy (**list**) a n-tice (**tuple**), s ktorými sa žiaci mohli stretnúť aj v predchádzajúcich metodikách venovaných modelovaniu, ďalej množinové dátové typy (**set**) a dátové typy predstavujúce mapovanie, tzv. slovníky (**dict**), ktoré nám umožňujú efektívne uchovávanie informácií rôzneho typu vo formáte **KĽÚČ:HODNOTA**,napríklad



K uloženým hodnotám sa pristupuje podobne ako pri zoznamoch, ale namiesto indexu prvku použijeme kľúč:

```
>>>teploty['po']
16.7
```

Kľúče sú odkazy na objekty , ktoré ukazujú na hashovateľné objekty, a hodnoty sú odkazy na objekty ukazujúce na objekty ľubovoľného typu. Slovníky sú **meniteľné**, teda môžeme do nich pridávať a odberať prvky, avšak sú to **neusporiadané** kolekcie, neexistuje u nich indexová pozícia a preto pri nich nemôžeme použiť rezanie (slicing), ani krokovanie. Podporujú funkciu **len()** na určenie dĺžky slovníka (t.j. počtu dvojíc kľúč:hodnota) a taktiež testovanie príslušnosti prostredníctvom operátorov **in** a **not in.**

Nový prázdny slovník **d** vytvoríme príkazom

```
>>>d={}
```

Medzi základné slovníkové metódy zaraďujeme:

Syntax	*Popis*
d.items()	Vráti všetky dvojice kľúč:hodnota zo slovníka d
d.keys()	Vráti všetky kľúče zo slovníka d
d.values()	Vráti všetky hodnoty zo slovníka d
d.clear()	Odstráni všetky prvky zo slovníka d
d.pop(k)	Vráti hodnotu spojenú s kľúčom k a odstráni prvok, ktorého kľúčom je k (príp. vyvolá výnimku KeyError, ak kľúč k nie je v slovníku d)

9.2 Priebeh výučby (90 minút)

EVOKÁCIA

Hodinu začneme aktivitami na zopakovanie práce s dátovými typmi, ktoré už žiaci z jazyka Python poznajú, pričom sa zameriame na činnosti, ktoré budú pre nich neskôr dôležité – teda výber a úprava konkrétnych dát z rôznych dátových typov a ich prípadná konverzia do iných dátových typov. Žiaci najprv pracujú samostatne bez počítača pomocou pracovných listov (úloha 1 a 2).

ÚLOHA 1 – RIEŠTE!

Pri programovaní v jazyku Python ste sa už stretli s rôznymi dátovými typmi – pozrite si príklady a určte, ktorý dátový typ predstavujú.

PRÍKLAD	*DÁTOVÝ TYP*
`2018`	
`'apríl '`	
`18.3`	
`['po', 'ut', 'st', 'št', 'pi', 'so', 'ne']`	
`'2018-04-23'`	
`['2018-04-23', '18.3°C']`	
`[['2018-04-23','18.3°C'],['2018-04-24','15.1°C'],['2018-04-25','17.5°C']]`	

ÚLOHA 2 – RIEŠTE!

Niekedy je potrebné zo zložitejšej reprezentácie dát vybrať len ich časť a upraviť ich na ďalšie spracovanie do vhodnej podoby. Akými príkazmi by ste získali nasledujúce informácie?

ČO CHCEME DOSTAŤ	*Z ČOHO TO CHCEME ZÍSKAŤ*	*AKO TO ZÍSKAME*
`18.3`	`namerana_teplota=’18.3°C’`	
`10`	`datum=‘2018-10-23’`	
`['so', 'ne']`	`dni=['po', 'ut', 'st', 'št', 'pi', 'so', 'ne']`	
`18.3`	`zaznam=['2018-04-23', '18.3°C']`	
`['18.3°C', '15.1°C', '17.5°C']`	`merania=[['2018-04-23','18.3°C'], ['2018-04-24','15.1°C'], ['2018-04-25','17.5°C']]`	
`Najnižšia teplota bola 15.1°C.` `Najvyššia teplota bola 18.3°C.`	`teploty=[18.3,15.1,17.5]`	

UVEDOMENIE SI VÝZNAMU

Po skontrolovaní výsledkov vyriešime so žiakmi formou spoločnej frontálnej diskusie úlohu 3 z pracovného listu. Túto úlohu neriešime na počítačoch, ale vyzveme žiakov, aby navrhli, príp. zapísali na tabuľu ich navrhnutú dátovú reprezentáciu dát z tabuľky.

ÚLOHA 3 – RIEŠTE!

Navrhnite v jazyku Python vhodnú dátovú reprezentáciu informácií z nasledujúcej tabuľky:

MIESTO	*DÁTUM*	*TEPLOTA (v °C)*	*RÝCHLOSŤ VETRA (v m/s)*
Košice	2018-04-20	21	4
Prešov	2018-04-20	20	6
Ružomberok	2018-04-20	18	6

Vytvoriť premennú, ktorá by uchovala všetky informácie z malej tabuľky (malej vzorky dát) nemusí byť problém. Avšak v praxi sa častejšie stretneme s pomerne veľkými súbormi dát, napr. záznamy pre každý deň v roku a každú meraciu meteostanicu. Vyhľadávať informácie z takej veľkej dátovej vzorky si vyžaduje taký dátový typ, ktorý by umožnil efektívnejší spôsob prístupu k informáciám. Poslúžiť nám na to môže dátový typ slovník, v ktorom sú dáta (prvky) uložené vo formáte dvojíc KĽÚČ:HODNOTA, napr.:

KĽÚČ HODNOTA

```
teploty={'po':16.7,'ut':17.2, 'st':17.5, 'št':17.1, 'pi':17.4, 'so':18.0, 'ne':17.6}
```

K uloženým hodnotám sa pristupuje podobne ako pri zoznamoch, ale namiesto indexu prvku použijeme kľúč:

```
>>>teploty['po']
16.7
```

Základné zručnosti pri práci so slovníkom si žiaci vyskúšajú samostatne prakticky na počítačoch pomocou úloh 4 až 7 z pracovných listov (poskytneme im pracovný súbor **merania.py**).

ÚLOHA 4 – RIEŠTE!

Otvorte súbor **merania.py** a upravte pripravený slovník **teploty** tak, aby kľúčmi boli mestá (z úlohy 3) a im prislúchajúce hodnoty boli namerané teploty.

ÚLOHA 5 – RIEŠTE!

Vytvorte v súbore **merania.py** nový slovník **rychlost_vetra** s nameranými hodnotami rýchlosti vetra v jednotlivých mestách z úlohy 3.

ÚLOHA 6 – RIEŠTE!

Máte už vytvorené dva slovníky **teploty** a **rychlost_vetra**. Odhadnite, na čo slúžia nasledujúce príkazy (zatiaľ ich neskúšajte; pokiaľ neviete, uveďte NEVIEM:

PRÍKAZ	*VÝZNAM (odhad)*
`teploty['Košice ']`	
`rychlost_vetra['Košice ']=2`	
`rychlost_vetra['Bratislava ']=4`	
`len(teploty)`	
`del teploty['Košice']`	
`'Košice' in teploty`	

ÚLOHA 7 – RIEŠTE!

Doplňte do súboru **merania.py** nasledujúci testovací kód s príkazmi z úlohy 6, spustite ho, sledujte postupne jednotlivé výpisy a zhodnoťte správnosť svojich odhadov z úlohy 6:

```
print(teploty)
print(teploty['Košice '])
input()    #stlac Enter
rychlost_vetra['Košice ']=2
print(rychlost_vetra)
input()    #stlac Enter
rychlost_vetra['Bratislava ']=4
print(rychlost_vetra)
input()    #stlac Enter
```

```
print(teploty)
print(len(teploty))
input()    #stlac Enter
del teploty['Košice ']
print(teploty)
input()    #stlac Enter
print('Košice ' in teploty)
print('Prešov ' in teploty)
```

Spoločne prekontrolujeme riešenia úloh 6 a 7, v ktorých žiaci skúmali význam príkazov pre prácu so slovníkmi. Dôležité bude ešte naučiť sa vypisovať kľúče a hodnoty zo slovníka, pri ktorých sa použijú cykly **for** a metódy **keys()** a **items**(), čo postupne vyskúšame spolu žiakmi v úlohe 8, 9 a 10. Následne žiaci môžu začať riešiť praktické aplikačné úlohy 11 a 12, pričom pracujú samostatne alebo vo dvojiciach na počítačoch s pripravenými pracovnými súbormi **chemia.py** a **aminokyseliny.py**, do ktorých dopĺňajú chýbajú zdrojový kód v programe (v programoch sú už pripravené grafické používateľské rozhrania, v ktorých môžu žiaci svoje riešenia odskúšať).

ÚLOHA 8 – RIEŠTE!

Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru **merania.py** a otestujte:

```
print(rychlost_vetra.keys())
```

ÚLOHA 9 – RIEŠTE!

Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru **merania.py** a otestujte:

```
print('Namerané teploty na Slovensku: ')
for miesto in teploty:
    print(teploty[miesto])
```

ÚLOHA 10 – RIEŠTE!

Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru **merania.py** a otestujte:

```
for key, val in rychlost_vetra.items():
    print(key, val)
```

ÚLOHA 11 – RIEŠTE!

Otvorte súbor **chemia.py**, v ktorom vytvorte 4 slovníky **H**, **He**, **Li** a **Be**, pričom každý bude obsahovať 5 podstatných informácií ku každému z uvedených prvkov (kľúče vo všetkých slovníkoch budú rovnaké, preto si zvoľte také, ktoré budú aplikovateľné na každý prvok). Spustite program a otestujte.

ÚLOHA 12 – RIEŠTE!

Otvorte súbor **aminokyseliny.py**, v ktorom vytvorte slovník **kodon** na určenie aminokyseliny na základe kodónu, teda sekvencie troch nukleotidov v molekule mRNA pomocou nasledujúcej tabuľky (postačí vytvorenie len časti slovníka pre vyznačený prvý riadok tabuľky). Spustite program a otestujte.

1st base

2nd base	U		C		A		G		3rd base
U	UUU	Phenylalanine	UCU	Serine	UAU	Tyrosine	UGU	Cysteine	U
	UUC	Phenylalanine	UCC	Serine	UAC	Tyrosine	UGC	Cysteine	C
	UUA	Leucine	UCA	Serine	UAA	Stop	UGA	Stop	A
	UUG	Leucine	UCG	Serine	UAG	Stop	UGG	Tryptophan	G
C	CUU	Leucine	CCU	Proline	CAU	Histidine	CGU	Arginine	U
	CUC	Leucine	CCC	Proline	CAC	Histidine	CGC	Arginine	C
	CUA	Leucine	CCA	Proline	CAA	Glutamine	CGA	Arginine	A
	CUG	Leucine	CCG	Proline	CAG	Glutamine	CGG	Arginine	G
A	AUU	Isoleucine	ACU	Threonine	AAU	Asparagine	AGU	Serine	U
	AUC	Isoleucine	ACC	Threonine	AAC	Asparagine	AGC	Serine	C
	AUA	Isoleucine	ACA	Threonine	AAA	Lysine	AGA	Arginine	A
	AUG	Methionine (Start)	ACG	Threonine	AAG	Lysine	AGG	Arginine	G
G	GUU	Valine	GCU	Alanine	GAU	Aspartic Acid	GGU	Glycine	U
	GUC	Valine	GCC	Alanine	GAC	Aspartic Acid	GGC	Glycine	C
	GUA	Valine	GCA	Alanine	GAA	Glutamic Acid	GGA	Glycine	A
	GUG	Valine	GCG	Alanine	GAG	Glutamic Acid	GGG	Glycine	G

Nonpolar, aliphatic Polar, uncharged Aromatic Positively charged Negatively charged

Zdroj: https://commons.wikimedia.org/wiki/File:Codontable1.PNG

REFLEXIA

Na záver hodiny vyzveme žiakov na zhrnutie a reflexiu poznatkov, ktoré počas hodiny získali. Úlohu 13 z pracovného listu riešia samostatne a následne frontálnou metódou postupne prejdeme ich poznámky a postrehy z hodiny. Posledná úloha predstavuje návrh na programátorský projekt (domácu úlohu), v ktorom žiaci prepoja poznatky z predchádzajúcej metodiky, kde sa stretli s nástrojom **BLAST** na porovnávanie DNA reťazcov, s poznatkami o práci so slovníkmi v Pythone. Učiteľ môže podľa zručností žiakov individuálne prispôsobiť náročnosť výsledného riešenia, napr. doplnením grafického používateľského rozhrania, čítaním vstupu z externého súboru alebo identifikáciou miesta mutácie.

ÚLOHA 13 – RIEŠTE!

Na základe predchádzajúcich úloh doplňte:

V ČOM SA LÍŠI SLOVNÍK OD ZOZNAMU:	***AKÉ OPERÁCIE SOM ROBIL/A SO SLOVNÍKOM:***	***KDE, RESP. NA ČO BY SA EŠTE DAL VYUŽIŤ DÁTOVÝ TYP SLOVNÍK:***

ÚLOHA 14 – RIEŠTE!

PROJEKT BLAST

Pri predošlej téme (**Práca s databázami 1**) ste sa oboznámili s nástrojom BLAST na porovnávanie sekvencií DNA. Navrhnite vlastnú jednoduchú verziu tohto nástroja, ktorá bude obsahovať referenčnú DNA (ako reťazec) a niekoľko testovaných DNA (v slovníku, kde kľúčmi budú mená testovaných osôb a hodnotami ich DNA reťazce). Program porovná DNA testovaných osôb s referenčnou DNA a pre každú osobu vypíše, či v testovanej DNA nastala alebo nenastala mutácia. Miesto mutácie nie je potrebné identifikovať

9.3 Pracovný list D2

Práca s databázami 2

Úloha 1. *Pri programovaní v jazyku Python ste sa už stretli s rôznymi dátovými typmi – pozrite si príklady a určte, ktorý dátový typ predstavujú.*

PRÍKLAD	*DÁTOVÝ TYP*
`2018`	
`'apríl '`	
`18.3`	
`['po', 'ut', 'st', 'št', 'pi', 'so', 'ne']`	
`'2018-04-23'`	
`['2018-04-23', '18.3°C']`	
`[['2018-04-23','18.3°C'],['2018-04-24','15.1°C'],['2018-04-25','17.5°C']]`	

Úloha 2. *Niekedy je potrebné zo zložitejšej reprezentácie dát vybrať len ich časť a upraviť ich na ďalšie spracovanie do vhodnej podoby. Akými príkazmi by ste získali nasledujúce informácie?*

ČO CHCEME DOSTAŤ	*Z ČOHO TO CHCEME ZÍSKAŤ*	*AKO TO ZÍSKAME*
`18.3`	`namerana_teplota='18.3°C'`	
`10`	`datum='2018-10-23'`	
`['so', 'ne']`	`dni=['po', 'ut', 'st', 'št', 'pi', 'so', 'ne']`	
`18.3`	`zaznam=['2018-04-23', '18.3°C']`	
`['18.3°C', '15.1°C', '17.5°C']`	`merania=[['2018-04-23','18.3°C'], ['2018-04-24','15.1°C'], ['2018-04-25','17.5°C']]`	
`Najnižšia teplota bola 15.1°C.` `Najvyššia teplota bola 18.3°C.`	`teploty=[18.3,15.1,17.5]`	

Úloha 3. *Navrhnite v jazyku Python vhodnú dátovú reprezentáciu informácií z nasledujúcej tabuľky:*

MIESTO	*DÁTUM*	*TEPLOTA (v °C)*	*RÝCHLOSŤ VETRA (v m/s)*
Košice	2018-04-20	21	4
Prešov	2018-04-20	20	6
Ružomberok	2018-04-20	18	6

Úloha 4. *Otvorte súbor* ***merania.py*** *a upravte pripravený slovník* ***teploty*** *tak, aby kľúčmi boli mestá (z úlohy 3) a im prislúchajúce hodnoty boli namerané teploty.*

Úloha 5. *Vytvorte v súbore* ***merania.py*** *nový slovník* ***rychlost_vetra*** *s nameranými hodnotami rýchlosti vetra v jednotlivých mestách z úlohy 3.*

Úloha 6. *Máte už vytvorené dva slovníky* ***teploty*** *a* ***rychlost_vetra****. Odhadnite, na čo slúžia nasledujúce príkazy (zatiaľ ich neskúšajte; pokiaľ neviete, uveďte NEVIEM):*

PRÍKAZ	*VÝZNAM (odhad)*
`teploty['Košice ']`	
`rychlost_vetra['Košice ']=2`	
`rychlost_vetra['Bratislava ']=4`	
`len(teploty)`	
`del teploty['Košice']`	
`'Košice' in teploty`	

Úloha 7. *Doplňte do súboru* ***merania.py*** *nasledujúci testovací kód s príkazmi z úlohy 6, spustite ho, sledujte postupne jednotlivé výpisy a zhodnoťte správnosť svojich odhadov z úlohy 6:*

```
print(teploty)
print(teploty['Košice '])
input()    #stlac Enter
rychlost_vetra['Košice ']=2
print(rychlost_vetra)
input()    #stlac Enter
rychlost_vetra['Bratislava ']=4
print(rychlost_vetra)
input()    #stlac Enter
```

```
print(teploty)
print(len(teploty))
input()    #stlac Enter
del teploty['Košice ']
print(teploty)
input()    #stlac Enter
print('Košice ' in teploty)
print('Prešov ' in teploty)
```

Úloha 8. *Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru* ***merania.py*** *a otestujte:*

```
print(rychlost_vetra.keys())
```

Úloha 9. *Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru* ***merania.py*** *a otestujte:*

```
print('Namerané teploty na Slovensku: ')
for miesto in teploty:
    print(teploty[miesto])
```

Úloha 10. *Zamyslite sa, čo bude asi robiť nasledujúci kód, doplňte ho do súboru* ***merania.py*** *a otestujte:*

```
for key, val in rychlost_vetra.items():    print(key, val)
```

Úloha 11. *Otvorte súbor* ***chemia.py****, v ktorom vytvorte 4 slovníky* ***H****,* ***He****,* ***Li*** *a* ***Be****, pričom každý bude obsahovať 5 podstatných informácií ku každému z uvedených prvkov (kľúče vo všetkých slovníkoch budú rovnaké, preto si zvoľte také, ktoré budú aplikovateľné na každý prvok). Spustite program a otestujte.*

Úloha 12. *Otvorte súbor* ***aminokyseliny.py****, v ktorom vytvorte slovník* ***kodon*** *na určenie aminokyseliny na základe kodónu, teda sekvencie troch nukleotidov v molekule mRNA pomocou nasledujúcej tabuľky (postačí vytvorenie len časti slovníka pre vyznačený prvý riadok tabuľky). Spustite program a otestujte.*

1st base

2nd base	U		C		A		G		3rd base
U	UUU	Phenylalanine	UCU	Serine	UAU	Tyrosine	UGU	Cysteine	U
	UUC	Phenylalanine	UCC	Serine	UAC	Tyrosine	UGC	Cysteine	C
	UUA	Leucine	UCA	Serine	UAA	Stop	UGA	Stop	A
	UUG	Leucine	UCG	Serine	UAG	Stop	UGG	Tryptophan	G
C	CUU	Leucine	CCU	Proline	CAU	Histidine	CGU	Arginine	U
	CUC	Leucine	CCC	Proline	CAC	Histidine	CGC	Arginine	C
	CUA	Leucine	CCA	Proline	CAA	Glutamine	CGA	Arginine	A
	CUG	Leucine	CCG	Proline	CAG	Glutamine	CGG	Arginine	G
A	AUU	Isoleucine	ACU	Threonine	AAU	Asparagine	AGU	Serine	U
	AUC	Isoleucine	ACC	Threonine	AAC	Asparagine	AGC	Serine	C
	AUA	Isoleucine	ACA	Threonine	AAA	Lysine	AGA	Arginine	A
	AUG	Methionine (Start)	ACG	Threonine	AAG	Lysine	AGG	Arginine	G
G	GUU	Valine	GCU	Alanine	GAU	Aspartic Acid	GGU	Glycine	U
	GUC	Valine	GCC	Alanine	GAC	Aspartic Acid	GGC	Glycine	C
	GUA	Valine	GCA	Alanine	GAA	Glutamic Acid	GGA	Glycine	A
	GUG	Valine	GCG	Alanine	GAG	Glutamic Acid	GGG	Glycine	G

Nonpolar, aliphatic Polar, uncharged Aromatic Positively charged Negatively charged

Zdroj: https://commons.wikimedia.org/wiki/File:Codontable1.PNG

Úloha 13. *Na základe predchádzajúcich úloh doplňte:*

V ČOM SA LÍŠI SLOVNÍK OD ZOZNAMU:	***AKÉ OPERÁCIE SOM ROBIL/A SO SLOVNÍKOM:***	***KDE, RESP. NA ČO BY SA EŠTE DAL VYUŽIŤ DÁTOVÝ TYP SLOVNÍK:***

Úloha 14. PROJEKT BLAST

*Pri predošlej téme (****Práca s databázami 1****) ste sa oboznámili s nástrojom BLAST na porovnávanie sekvencií DNA. Navrhnite vlastnú jednoduchú verziu tohto nástroja, ktorá bude obsahovať referenčnú DNA (ako reťazec) a niekoľko testovaných DNA (v slovníku, kde kľúčmi budú mená testovaných osôb a hodnotami ich DNA reťazce). Program porovná DNA testovaných osôb s referenčnou DNA a pre každú osobu vypíše, či v testovanej DNA nastala alebo nenastala mutácia. Miesto mutácie nie je potrebné identifikovať.*

BIBLIOGRAFIA

Belan, A. (2013). *PYTHON.* Bratislava: Anino Belan.

Blaho, A. (2016). *Programovanie v Pythone.* Bratislava: FMFI UK.

CoBaLa. (2018). *Pixel Art - Color By Number! Coloring Book for adults, kids and toddlers Free.* https://www.amazon.com/Pixel-Art-Number-Coloring-toddlers/dp/B07HVR65YD?ref_=fsclp_pl_dp_4.

Ferko, A. a. (1995). *Počítačová grafika a spracovanie obrazu.* Bratislava: SAPIENTIA, ISBN 80-967180-2-9.

Hvorecký, J. (1975). Conwayova hra LIFE. *Pokroky matematiky, fyziky a astronómie, Vol. 20, No. 5*, 252-257.

Šaling, S. a. (1977). *Veľký slovník cudzích slov.* Veľký Šariš: Vydavateľstvo SAMO – AAMM, ISBN 80-967524-0-5.

Švec, J. (2002). *Učebnice jazyka Python (aneb Létající cirkus).* Jan Švec.

PRÍLOHA A – TEXTY PROGRAMOV

Program k pracovnému listu P1, P1_Pixel1.py

```
# K pracovnému listu 1, celý program vytvárajú žiaci sami
#
# Vykreslenie obrazku pomocou korytnacej grafiky
# Pouzitie kniznice s korytnacou grafikou
import turtle

# Priprava grafickeho okna na vykreslenie
okno = turtle.Screen()
okno.bgcolor(0.7, 0, 0.5)
okno.title("Vykreslenie obsahu 0/1 pola")

# Priprava pera a rychlosti a farby korytnacky kor
myPen = turtle.Turtle()
myPen.speed(0)
myPen.color("#000000")
#########################################################################
# Funkcia vykresli stvorcek, jeho strany a potom ho vyplni pouzitim
# funkcie fill
# Parameter fstvorcek predstavuje velkost stvorceka
#########################################################################
def stvorcek(fvelkost):
    myPen.begin_fill()
    # 0 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 90 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 180 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 270 deg.
    myPen.forward(fvelkost)
    myPen.end_fill()
    myPen.setheading(0)
#########################################################################
# Funkcia vykresli pole pixelov pixs s velkostou stvorceka fstvorcek
#########################################################################

def kresli(pixs, fvelkost):
    farba="#00FF00"
    for i in range(0, len(pixs)):
        for j in range(0, len(pixs[i])):
            if pixs[i][j] == 1:
```

```
                myPen.color(farba)
                stvorcek(fvelkost)
            myPen.penup()
            myPen.forward(fvelkost)
            myPen.pendown()
        myPen.setheading(270)
        myPen.penup()
        myPen.forward(fvelkost)
        myPen.setheading(180)
        myPen.forward(fvelkost * len(pixs[i]))
        myPen.setheading(0)
        myPen.pendown()

# Ulozenie 0/1 PixelArt (pouzitim "zoznamu v zozname")
# Tak vieme namodelovat akykolvek 0/1 zoznam v zozname
# Nastavenie pozicie pera korytnacky na poziciu, kde zacne kreslenie;
# Na zaciatku je v lavom hornom rohu
myPen.penup()
myPen.forward(-90)
myPen.setheading(90)
myPen.forward(90)
myPen.setheading(0)

velkost = 10
print("Velkost boxu pre pixel je", velkost)

pixels1 =[1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,1]

pixels =     [[0, 0, 0, 1, 1, 1, 1, 0, 0, 0]]
pixels.append([0, 0, 1, 0, 0, 0, 0, 1, 0, 0])
pixels.append([0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
pixels.append([0, 1, 0, 1, 0, 0, 1, 0, 1, 0])
pixels.append([0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
pixels.append([0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
pixels.append([0, 1, 0, 1, 1, 1, 1, 0, 1, 0])
pixels.append([0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
pixels.append([0, 0, 1, 0, 0, 0, 0, 1, 0, 0])
pixels.append([0, 0, 0, 1, 1, 1, 1, 0, 0, 0])

myPen.penup()
myPen.forward(90)
myPen.pendown()
kresli(pixels, velkost)
myPen.getscreen().update()
input("Stlacte Enter!")
```

Program k pracovnému listu P2, P1_Pixel2_ucitel.py

```
#
# Spustenim programu overite, ci je syntakticky spravny
# Vykreslenie konkretneho obrazku pomocou korytnacej grafiky
# PIL je package, ktory je prospesny pri vykreslovani

import turtle
from PIL import Image
#######################################################################
# Otvorenie obrazku, jeho nazov je img, vypis mena suboru a velkosti
#######################################################################
meno = 'obrPixely1.png'
print(meno)

img = Image.open(meno)
print("Velkost obrazka: ", img.size)
##################### Doplnte, PL5, PL6
#### V konzole vypiste velkost obrazka.
# Vypisu sa dve cisla, prve je sirka a druhe je vyska obrazka
# Obrazok je mozne ukazat v pripojenom programe pomocou nasledujuceho
# prikazu
# img.show()

pxl_mapa = img.load()
##################### Doplnte, PL8
#### Vypis ciselnych farieb pixelu pxl_mapa[5,7], podla prac. listu
print(pxl_mapa[5,7])

######################################### PL7
### Ulozenie obrazka
print('R=',pxl_mapa[5,7][0])
print('G=',pxl_mapa[5,7][1])
print('B=',pxl_mapa[5,7][2])
print('A=',pxl_mapa[5,7][3])

print('Mapa - informacia o jednom pixeli',pxl_mapa[5,7])

##################### Doplnte, PL8
#### Vypis ciselnych farieb pixelu pxl_mapa[5,7], podla prac. listu
# Vypis informacii v pixloch 3. riadku

for stlpec in range(0, img.size[0]):
    print(pxl_mapa[2,stlpec])

# Korytnacka vykresluje
okno = turtle.Screen()
okno.bgcolor("yellow")
okno.title("Práca s reálnym obrázkom")

myPen = turtle.Turtle()
```

```
myPen.speed(0)

velkost=10
print("Velkost stvorceka pre pixel je", velkost)
######################################################################
# Funkcia vykresli stvorcek, jeho strany a potom ho vyplni pouzitim
# funkcie fill
# Parameter fstvorcek predstavuje velkost stvorceka
######################################################################

def stvorcek(fvelkost):
    myPen.begin_fill()
    # 0 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 90 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 180 deg.
    myPen.forward(fvelkost)
    myPen.left(90)
    # 270 deg.
    myPen.forward(fvelkost)
    myPen.end_fill()
    myPen.setheading(0)
######################################################################
# Funkcia namiesa farbu z troch dekadickych cisel;
# vrati vysledok farba v sestnastkovej sustave
######################################################################

def namiesajFarbu(fR,fG,fB):
    farbaR=256*256*fR
    farbaG=256*fG
    farbaB=fB
    farba='#'+'{:06x}'.format(farbaR+farbaG+farbaB)
    return farba

######################################################################
# Overenie funkcie. Namiesanie farby a vykreslenie pixelu touto farbou
######################################################################

farbaPix=namiesajFarbu(100,0,255)
myPen.color(farbaPix)
stvorcek(velkost)
######################################################################
# Funkcia na vykreslenie obrazku korytnackou
# Posledne dva parametre urcuju pocty riadkov a stlpcov
######################################################################

def kresli2(pixs, fvelkost, pocRiadkov, pocStlpcov):
    for i in range(0, pocStlpcov-1):
```

```
        for j in range(0, pocRiadkov-1):
            farba1=namiesajFarbu(pixs[j,i][0],pixs[j,i][1],pixs[j, i][1])
            myPen.color(farba1)
            stvorcek(fvelkost)
            myPen.penup()
            myPen.forward(fvelkost)
            myPen.pendown()
        myPen.setheading(270)
        myPen.penup()
        myPen.forward(fvelkost)
        myPen.setheading(180)
        myPen.forward(fvelkost * (pocRiadkov-1))
        myPen.setheading(0)
        myPen.pendown()

############ Nastavenie pozicie pera korytnacky
myPen.penup()
myPen.forward(-100)
myPen.setheading(90)
myPen.forward(100)
myPen.setheading(0)

pr=img.size[0]
ps=img.size[1]
print(pr,ps)
myPen.pendown()
kresli2(pxl_mapa, velkost, pr, ps)

# Zmena farby pixelov v 3. riadku na cervenu farbu
for stlpec in range(0, img.size[0]):
    pxl_mapa[stlpec,2]=(255, 0, 0, 0)

# Nove vykreslenie obrazku
myPen.up()
myPen.setheading(-90)
myPen.forward(10)
myPen.setheading(0)
myPen.pendown()
kresli2(pxl_mapa, velkost, pr, ps)

# Zatvorenie kresliaeho okna kliknutim mysi
okno.exitonclick()
```

Program k pracovnému listu P3, P1_Pixel3.py

```
## K pracovnemu listu 3
#
# Vytvorenie lozisk

from tkinter import *
import random

Sirka = 640
Vyska = 400

window = Tk()
canvas = Canvas(window, width=Sirka, height=Vyska, bg="#ffffff")
canvas.pack()

# Nacitanie a vykreslenie obrazku
img = PhotoImage(width=Sirka, height=Vyska, file='gulka4.png')
canvas.create_image((Sirka/2, Vyska/2), image=img)

# Predpokladajme, ze sirka vykreslenej elipsy je 60 pixelov a vyska 50
pixelov
SirkaEl=200
VyskaEl=140

# Vypis obsahu pixelov v riadku 240, 301.-240. stlpec
for j in range(301, 341):
    print(img.get(j,239))

# Funkcia vytvori sestnastkovu hodnotu farby
def rgb_farba(r, g, b):
    return "#{:02x}{:02x}{:02x}".format(r, g, b)

# Overenie funkcie rgb
farbaB=rgb_farba(255, 255, 255)
print("16-tkova hodnota farby (255, 255, 255) ",farbaB)

######################################################################
# Funkcia najde v celej ploche pixel danej farby s najvacsimi indexami
# x a y
# Vstupny parameter farba je trojica dekadickych cisel v zatvorkach
# Porovnanie farieb ako hexadecimalne cisla
######################################################################

def najdiPixel(farba):
    xm=-1
    ym=-1
    farbaH=rgb_farba(farba[0],farba[1],farba[2])
    for x in range(Sirka):
        for y in range(Vyska):
            f_xy = img.get(x, y)
            f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
            if farbaH== f_xyH:
                xm=x
                ym=y
    return xm, ym

# Overenie funkcie najdiPixel
farbaCy=(0, 255, 255)
[x_sur, y_sur]=najdiPixel(farbaCy)
print("Pozicie najdeneho pixelu = ", x_sur, y_sur)

farba2= rgb_farba(255, 255, 255)
print("Farba2=",farba2)
```

```
########################################################################
#
# Funkcia zmeni farbu pixelu v pozicii ix, iy farby farba1 na definovanu
# farba2
# inak ho nakresli jeho farbou
# Farby su dane v hexadecimalnej sustave
########################################################################

def zmenPixel(ix, iy, farba1, farba2):
    fa1 = img.get(ix, iy)
    fa1H=rgb_farba(fa1[0],fa1[1],fa1[2])
    if farba1!=fa1H:
        img.put(fa1H,(ix,iy))
    else:
        img.put(farba2,(ix,iy))

# Overenie funkcie zmenPixel; Zmena farby nahodne vygenerovanych n pixelov;

farbaCyH=rgb_farba(0,255,255)
print(farbaCyH)
farba2= rgb_farba(0, 0, 0)
n=random.randint(50, 200)
print(n)
for pocet in range(0,n):
    x = random.randint((Sirka-SirkaEl)/2, (Sirka+SirkaEl)/2)
    y = random.randint((Vyska-VyskaEl)/2, (Vyska+VyskaEl)/2)
    f_xy=img.get(x,y)
    f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
    zmenPixel(x, y, farbaCyH, farba2)

########################################################################
# Vytvorenie lozisk
# Lozisko_krizik je pixel a jeho 4 susedia (dvaja v horizontalnom
# a dvaja vo vertikalnom smere) ciernej farby
# Lozisko vznikne len v na nahodnej pozicii pixelu, ktory ma farbu farba.
# Pixely loziska budu zafarbene farbou farba2
########################################################################

def lozisko_krizik(farba,farba2):
    x = random.randint((Sirka-SirkaEl)/2, (Sirka+SirkaEl)/2)
    y = random.randint((Vyska-VyskaEl)/2, (Vyska+VyskaEl)/2)
#    print('x=',x,y)
    f_xy=img.get(x,y)
    f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
    if f_xyH==farba:
        img.put(farba2,(x,y))
        img.put(farba2,(x-1,y))
        img.put(farba2,(x+1,y))
        img.put(farba2,(x,y-1))
        img.put(farba2,(x,y+1))

# Overenie funkcie lozisko_krizik na desiatich krizikoch
for ii in range(10):
    lozisko_krizik(farbaCyH,farba2)

canvas.update()
canvas.after(1000)

## Ulozenie obrazka do suboru
img.write('gulkaMod.png', format='png')

## Zvacsenie obrazku je mozne sledovat v mtPaint
mainloop()
```

Program k pracovnému listu P4, P1_Pixel4.py

```
# K pracovnemu listu 4, lozisko je krizik
#
# Rast loziska

from tkinter import *
import random

Sirka = 640
Vyska = 400

window = Tk()
canvas = Canvas(window, width=Sirka, height=Vyska, bg="#ffffff")
canvas.pack()

img = PhotoImage(width=Sirka, height=Vyska, file='gulka4.png')
canvas.create_image((Sirka/2, Vyska/2), image=img)

# Predpokladajme, ze sirka vykreslenej elipsy je 60 pixelov a vyska 50
pixelov
SirkaEl=180
VyskaEl=140
#########################################################################
####   Funkcie z programu P1_Pixel3
#########################################################################

# Funkcia vytvori sestnastkovu hodnotu farby
def rgb_farba(r, g, b):
    return "#{:02x}{:02x}{:02x}".format(r, g, b)

## Lozisko v tvare krizika
def lozisko_krizik(farba,farba2):
    x = random.randint((Sirka/2-SirkaEl), (Sirka/2+SirkaEl))
    y = random.randint((Vyska/2-VyskaEl/2), (Vyska/2+VyskaEl/2))
    f_xy=img.get(x,y)
    f_xyH=rgb_farba(f_xy[0],f_xy[1],f_xy[2])
    if f_xyH==farba:
        img.put(farba2,(x,y))
        img.put(farba2,(x-1,y))
        img.put(farba2,(x+1,y))
        img.put(farba2,(x,y-1))
        img.put(farba2,(x,y+1))
    return x, y

# Funkcia zmeni farbu pixelu v pozicii ix, iy farby farba1 na definovanu
farba2
# inak ho nakresli jeho farbou
# Farby su dane hexadecimalne
# Funkcia definovana v P1_Pixel3

def zmenPixel(ix, iy, farba1, farba2):
    fa1 = img.get(ix, iy)
    fa1H=rgb_farba(fa1[0],fa1[1],fa1[2])
```

```
    if farba1!=fa1H:
        img.put(fa1H,(ix,iy))
    else:
        img.put(farba2,(ix,iy))

#########################################################################
#      Nove funkcie
#########################################################################

# Funkcia urci pocet susedov farby farba0 pre dany pixel v pozicii rx,sy
# Maximalny pocet susedov je 8
def pocetSusedov(rx, sy, farba0):
    sObs=0
    for r in rx - 1, rx, rx + 1:
        for s in sy - 1, sy, sy + 1:
            if 0 <= r < Sirka and 0 <= s < Vyska:
                aa = img.get(r, s)
                aah = rgb_farba(aa[0], aa[1], aa[2])
                if aah==farba0 and (r != rx or s != sy):
                    sObs += 1
    return sObs
#########################################################################
# Overenie funkcie pocetSusedov
#########################################################################

farba1=(0, 255, 255)
rrx = random.randint((Sirka / 2 - SirkaEl), (Sirka / 2 + SirkaEl))
ssy = random.randint((Vyska / 2 - VyskaEl / 2), (Vyska / 2 + VyskaEl /
2))
aap = img.get(rrx, ssy)
aaph = rgb_farba(aap[0], aap[1], aap[2])
print("Pozicia a farba pixelu=", rrx, ssy, aaph)
pocSusedov=pocetSusedov(rrx, ssy, aaph)
print("Pocet susedov= ",pocSusedov)

#########################################################################
# Rast a modifikacia loziska pomocou zmeny farby pixelov
# Farby su zadavane hexadecimalne
#########################################################################

def rastLoziska(xL, yL, farbaL, farbaN):
    for i in range(0,99):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        farba1=img.get(rr, ss)
        farba1h=rgb_farba(farba1[0],farba1[1],farba1[2])
        pocObsSus=pocetSusedov(rr, ss, farbaN) ### pocet neloziskovych
susedov farbyN
        if pocObsSus>2 and pocObsSus<6:
            zmenPixel(rr,ss,farba1h,farbaL)

#########################################################################
# Overenie funkcie rastLoziska

farba1= rgb_farba(0,255,255)   ### Cyan
farba3= rgb_farba(0, 0, 0)     ### Cierna
```

```
for PocLozisk in range(20):
    xx, yy =lozisko_krizik(farba1,farba3)  ### farba3 je farba loziska
    rastLoziska(xx, yy, farba3, farba1)
    canvas.update()
    canvas.after(1000)

########################################################################
#  Funkcia, ktora nahodny pixel loziska zafarbi farbou pozadia, co
#  znamena jeho zanik
#  Vstupne parametre su: pozicia stredu loziska a farby v sestnastkovej
%  sustave
########################################################################

def pZanik(xL, yL, farbaL, farba):
    for ii in range(10):
        rr=random.randint(xL-3,xL+3)
        ss=random.randint(yL-3,yL+3)
        aap = img.get(rr, ss)
        aapH=rgb_farba(aap[0], aap[1], aap[2])
        if (farbaL==aapH):
            susedia=pocetSusedov(rr, ss, aapH)
            if susedia>5:
                zmenPixel(rr, ss, farbaL, farba)

########################################################################
# Overenie funkcie pZanik pre 20 nahodne vygenerovanych pixelov
# Farba zaniku je urcena na cervenu kvoli viditelnosti
########################################################################

farbaC=farba1
xx, yy =lozisko_krizik(farbaC,farba3)

farba1= rgb_farba(0,255,255)   ### Cyan
farba3= rgb_farba(0, 0, 0)     ### Cierna
farba2= rgb_farba(255,0,0)     ### cervena alebo zmenime na bielu

for PocLozisk in range(20):
    xx, yy =lozisko_krizik(farba1,farba3)  ### farba3 je farba loziska
    rastLoziska(xx, yy, farba3, farba1)
    pZanik(xx, yy, farba3, farba2)
    canvas.update()
    canvas.after(1000)
########################################################################
##  Bonusove funkcie pre snazivych studentov
########################################################################

# Funkcia urci pocet pixelov danej farby v celej kresliacej ploche
def pocetPixelovFarby(farba):
    pocet=0;
    for x in range(Sirka):
        for y in range(Vyska):
            aa=img.get(x,y)
            aah=rgb_farba(aa[0],aa[1],aa[2])
            if aah==farba:
                pocet=pocet+1
```

```
    print(aah)
    return pocet

# Overenie funkcie pocetPixelov hladanej farby
farbaCyh=rgb_farba(0,255,255)
pocetCy=pocetPixelovFarby(farbaCyh)
print("Pocet= ", pocetCy)


## Ulozenie obrazka do suboru
img.write('nejakeMeno.png', format='png')

mainloop()
```

Program k pracovnému listu M1, P2_Model1.py

```
### Graficke zobrazenie rekordnych skokov do dialky

from pylab import *

Roky=[1901, 1923, 1924, 1925, 1928, 1928, 1931, 1935, 1960, 1961,
1961, 1962, 1964, 1964, 1965, 1967, 1968, 1991]

Skoky=[7.61, 7.69, 7.76, 7.89, 7.90, 7.93, 7.98, 8.13,
8.21, 8.24, 8.28, 8.31, 8.31, 8.34, 8.35, 8.35, 8.90, 8.95]

###  Doplnte vypis oboch zoznamov v konzole
###                          Doplnte
print(Roky)
print(Skoky)

pocetRokov=len(Roky)
### Vypis poctu rokov v konzole
###                          Doplnte
print('Pocet rokov',pocetRokov)

### Je pocet prvkov v oboch suboroch rovnaky
if pocetRokov == len(Skoky):
    print('Rovnaky')
else:
    print('Rozny')

#### Graficke znazornenie dat a informacii o nich v jednom obrazku
figure()

subplot(1,3,1)
plot(Roky, Skoky, 'ro-')
xlabel('Rok')
ylabel('Skok [m]')
title('Rekordy v skoku do diaľky')

################  Zobrazenie prirastkov skokov
################  Doplnit, najprv vytvorit zoznam prirastkov PrirastokSk

subplot(1,3,2)
prirastokSk=[0.0]
for i in range(0,pocetRokov-1):
    delta=Skoky[i+1]-Skoky[i]
    prirastokSk.append(delta)

plot(Roky, prirastokSk, 'bo-')
xlabel('Rok')
ylabel('prírastokSk [m]')
title('Prírastky v skoku do diaľky')

#### Doplnenie roznych informacii

subplot(1,3,3)
text(1,18,['Počet rekordných rokov: ',str(pocetRokov)])
```

```
aa=max(prirastokSk)
text(1,15,['Najdlhší prírastok: ','{:6.3f}'.format(aa)])
axis([0, 20, 0, 20])
title('Informácie')
show()
```

Program k pracovnému listu M2, P2_Model2.py

```
# K pracovnemu listu M2

from pylab import *
import numpy

Roky=[1901, 1923, 1924, 1925, 1928, 1928, 1931, 1935, 1960, 1961, 1961,
1962, 1964, 1964, 1965, 1967, 1968, 1991]

Skoky=[7.61, 7.69, 7.76, 7.89, 7.90, 7.93, 7.98, 8.13, 8.21, 8.24, 8.28,
8.31, 8.31, 8.34, 8.35, 8.35, 8.90, 8.95]

pocetRokov=len(Roky)
#########################################################################
# Funkcia na vypocet priemeru
#########################################################################
def priemer(post):
    n=len(post)
    s=sum(post)
    return s/n
#########################################################################
# Oskusanie funkcie priemer
ps1=priemer(Roky)
print('Priemerny rok', str(ps1))
ps2=mean(Roky)
print('Priemerny rok', str(ps2))

#########################################################################
# Funkcia na vypocet parametrov priamky,
# pouzitie numerickej funkcie mean na vypocet priemerov
#########################################################################

def priamka(n,fRoky,fSkoky):
    fRoky_mean=priemer(fRoky)
    fRoky_mean_2=fRoky_mean*fRoky_mean
    fSkoky_mean=priemer(fSkoky)
    print("x=",fRoky_mean," y=",fSkoky_mean)
    xxpom=[]
    xypom=[]
    for i in range(0,n):
        xxpom.append(fRoky[i]*fRoky[i])
        xypom.append(fRoky[i]*fSkoky[i])

    xx_mean=priemer(xxpom)
    xy_mean=priemer(xypom)

#    Kontrolny vypis hodnot
#    print("xx=",xx_mean," xy=",xy_mean)

    k=(xy_mean-fRoky_mean*fSkoky_mean)/(xx_mean-fRoky_mean_2)
    q=fSkoky_mean- k*fRoky_mean
    return k, q
#########################################################################
# Vykreslenie trendu vyvoja rekordov skokov v obrazku
#########################################################################
```

```
figure()
subplot(2,2,1)
plot(Roky, Skoky, 'ro')
xlabel('Rok')
ylabel('Skok [m]')
title('Rekordy v skoku do diaľky')

################  Vykreslenie ziskanej priamky
subplot(2,2,2)

plot(Roky, Skoky, 'ro');

[w0,w1]=priamka(pocetRokov,Roky,Skoky)
d_Rec_yy=[]
for i in range(0,pocetRokov):
    d_Rec_yy.append(w0*Roky[i]+w1)

plot(Roky, d_Rec_yy, 'bx-')
xlabel('Rok')
ylabel('Skok [m]')
title('Lineárny model')

################### Prirastky a vykreslenie priamky prirastkov
############                                Doplnte
subplot(2,2,3)
deltaSk=[0]
for i in range(0,pocetRokov-1):
    deltaSk.append(Skoky[i+1]-Skoky[i])

[w0,w1]=priamka(pocetRokov,Roky,deltaSk)

for i in range(0,pocetRokov):
    d_Rec_yy[i]=w0*Roky[i]+w1

###############  Vypis roznych informacii
############                                Doplnte
plot(Roky, deltaSk, 'ro')
plot(Roky, d_Rec_yy, 'bx-')
xlabel('Rok')
ylabel('deltaSk [m]')
title('Prírastky v skoku do diaľky')

subplot(2,2,4)
[w0,w1]=priamka(pocetRokov,Roky,Skoky)
ocakRek=w0*2020+w1;
text(0.1,0.8,['Očakávaný        rekord        v        roku        2020:
','{:6.3f}'.format(ocakRek)]);
aa=w0*2020+w1-Skoky[pocetRokov-1];
text(0.1,0.6,['Očakávaný prírastok: ','{:6.3f}'.format(aa)])
axis([0, 1, 0, 1])
title('Informácie')
show()
```

Program k pracovnému listu M3, P2_Model3.py

```
# Efekt rusicky
from pylab import *
import numpy as np

PD=[[-0.8992,    3.2087,    3.0810,    0.4616,    0.5579,    0.1426,   -1.1806],
    [ 2.5879,    3.6879,    8.4354,    8.4873,    9.0598,    3.6582,    6.7262],
    [ 0.5891,   -1.2149,    2.5316,   -1.7762,   -1.0412,   -0.0320,   -0.9601],
    [ 0.4665,    3.4914,    5.1140,    4.8644,    4.1082,    3.9664,   -1.7208]]

BD=[[ -3.5939,   -4.7074,   -8.8796,   -6.9755,   -5.1312,    3.5165,    6.6183],
    [ -3.9389,   -5.0305,   -5.3403,   -4.2425,   -1.7692,    2.5176,    6.8339],
    [ -3.4682,   -6.6589,   -5.1207,   -6.5891,   -4.1486,    2.4976,    5.2183],
    [ -3.0279,   -5.4059,   -6.4005,   -3.3021,   -4.3535,   -1.2749,    6.1751]]

#### Priprava farieb pre subjekty
farba=['k:','g:','b:','m:']
figure()
xx = [1, 2, 3, 4, 5, 6, 7]

#### Vykreslenie udajov PD jednotlivych subjektov v prvom subplote

subplot(2, 2, 1)
for subj in range(4):
    pp=[]
    for i in range(7):
        pp.append(PD[subj][i])
    plot(xx, pp, farba[subj])

xlabel('Pozícia reproduktorov')
ylabel('Efekt rusičky [°]')
axis([0, 8, -10, 10])
title('Sedenia s prednou rušičkou')

#### Vykreslenie udajov BD jednotlivych subjektov v druhom subplote
###                                          Doplnte

subplot(2,2,2)
xx = [1, 2, 3, 4, 5, 6, 7]
for subj in range(4):
    pp=[]
    for i in range(7):
        pp.append(BD[subj][i])
    plot(xx, pp,farba[subj])

legend({'Subj1','Subj2','Subj3','Subj4'})
xlabel('Pozícia reproduktorov')
title('Sedenia s bočnou rušičkou')
axis([0, 8, -10, 10])

#### Vykreslenie udajov jednotlivych subjektov v jednom subplote
#### Vykreslenie modelovacej krivky

subplot(2, 2, 3)
xxr = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0]
```

```
y = np.array(PD)
for subj in range(4):
    plot(xx, y[subj], farba[subj])

plot([0, 8], [0, 0], 'k-')
b  = np.mean(y, axis=0)
xxb=np.array(xxr)

print(b)
print(xxb)

bb = np.polyfit(xxb, b, 2)
yp = polyval(bb, xx)
plot(xx, yp, 'r-')
text(1, -4,'bb[0]={:6.3f}'.format(bb[0]))
text(1, -6,'bb[1]={:6.3f}'.format(bb[1]))
text(1, -8,'bb[2]={:6.3f}'.format(bb[2]))
axis([0, 8, -10, 10])
xlabel('Pozícia reproduktorov')
ylabel('Efekt rušičky [°]')

#### Vykreslenie udajov jednotlivych subjektov v jednom subplote
#### Vykreslenie modelovacej krivky
###
subplot(2, 2, 4)
x = np.array(BD)
for subj in range(4):
    plot(xx, x[subj], farba[subj])
plot([0, 8], [0, 0], 'k-')

b  = np.mean(x, axis=0)
xxb=np.array(xxr)

print(b)
print(xxb)

bb = np.polyfit(xxb, b, 2)
yp = polyval(bb, xx)
plot(xx, yp, 'r-')
text(1, 8,'bb[0]={:6.3f}'.format(bb[0]))
text(1, 6,'bb[1]={:6.3f}'.format(bb[1]))
text(1, 4,'bb[2]={:6.3f}'.format(bb[2]))
xlabel('Pozícia reproduktorov')
axis([0, 8, -10, 10])

show()
```

PRÍLOHA B – ZOZNAM POMOCNÝCH SÚBOROV

1. Program P1_Pixel1.py
2. Program P1_Pixel2.py
3. Program P1_Pixel3.py
4. Program P2_Model1.py
5. Program P2_ Model2.py
6. Program P2_Model3.py
7. Program P2_Model4.py
8. Program P1_Pixel1_ucitel.py
9. Program P1_Pixel2_ucitel.py
10. Program P1_Pixel3_ucitel.py
11. Program P2_Model1_ucitel.py
12. Program P2_ Model2_ucitel.py
13. Program P2_Model3_ucitel.py
14. Program P2_Model4_ucitel.py
15. PracovnyList_P1.docx
16. PracovnyList_P2.docx
17. PracovnyList_P3.docx
18. PracovnyList_M1.docx
19. PracovnyList_M2.docx
20. PracovnyList_M3.docx
21. PracovnyList_M4.docx

Informatika v prírodných vedách a matematike, Zošit Informatika

Spracované s finančnou podporou národného projektu
http://www.itakademia.sk, IT Akadémia -- vzdelávanie pre 21. storočie
Učebné texty pre učiteľov

Autori: doc. RNDr. Gabriela Andrejková, CSc.
Ing. Zuzana Tkáčová

Vydavateľ: Centrum vedecko-technických informácií SR, Bratislava

Rok vydania: 2020
Počet strán: 127
Rozsah: 6 AH
Vydanie: prvé

ISBN 978-80-89965-69-4
EAN 9788089965694 (e-publikácia)