KRESLÍME SO VZORMI (PEČIATKAMI) – VLASTNÉ FUNKCIE

<table>
<tr><th>Tematický celok / Téma</th><th>ISCED / Odporúčaný ročník</th></tr>
<tr><td>Algoritmické riešenie problémov:
analýza problému,
jazyk na zápis riešenia,
pomocou postupnosti príkazov,
interpretácia zápisu riešenia,
hľadanie a opravovanie chýb.</td><td>ISCED 3 / 2. ročník</td></tr>
<tr><th colspan="2">Požiadavky na vstupné vedomosti a zručnosti</th></tr>
<tr><td colspan="2">vytvoriť a spustiť program (tvorený sekvenciou príkazov korytnačej grafiky) a uložiť ho do súboru s príponou py,
vysvetliť význam základných príkazov korytnačej grafiky (napr. <code>forward()</code>, <code>left()</code>, <code>penup()</code>, <code>dot()</code>, <code>pensize()</code>, <code>pencolor()</code>, <code>bgcolor()</code>) a aplikovať ich pri tvorbe programov vykresľujúcich obrázky pozostávajúce z úsečiek a kruhov rôznych veľkostí a farieb.</td></tr>
<tr><th colspan="2">Ciele</th></tr>
<tr><th>Žiakom osvojované vedomosti a zručnosti</th><th>Žiakom rozvíjané spôsobilosti</th></tr>
<tr><td>Analýza problému:
identifikovať vstupné informácie zo zadania úlohy,
popisovať očakávané výstupy, výsledky, akcie,
formulovať a neformálne (prirodzeným jazykom) vyjadriť ideu riešenia,
uvažovať o vlastnostiach vykonávateľa (napr. korytnačka, grafické pero, robot).
Jazyk na zápis riešenia:
používať jazyk na zápis algoritmického riešenia problému,
rozpoznávať a odstraňovať chyby v zápise.
Pomocou postupnosti príkazov:
riešiť problém skladaním príkazov do postupnosti,
aplikovať pravidlá, konštrukcie jazyka pre zostavenie postupnosti príkazov.
Interpretácia zápisu riešenia:
doplniť, dokončiť, modifikovať rozpracované riešenie,
uvažovať o rôznych riešeniach, navrhovať vylepšenie.
Hľadanie a opravovanie chýb:
hľadať chybu vo vlastnom, nesprávne pracujúcom programe a opraviť ju.
Otvoriť program a modifikovať ho podľa zadania.
Vysvetliť rozdiel medzi definovaním vlastnej funkcie a jej volaním v programe.
Vytvoriť program využitím vlastných funkcií a základných príkazov korytnačej grafiky.</td><td>Informatické myslenie:
logika,
algoritmy,
dekompozícia,
vzory,
abstrakcia.
Bádateľské spôsobilosti:
formulovať otázku/problém,
naplánovať postup (identifikovať a definovať nezávislé a závislé premenné veličiny, vzájomný vzťah),
manipulovať s pomôckami/softvérom,
zdieľať a prezentovať výsledky pred spolužiakmi.</td></tr>
</table>

Riešený didaktický problém	
Tvorbu programov využívajúcich vlastné funkcie pokladáme za veľmi dôležitú tému výučby programovania, lebo umožňuje žiakom riešiť problémy hľadaním vzorov a dekompozícou zadaných problémov na podproblémy. Preto výučbu programovania vlastných funkcií zaraďujeme ako jednu z prvých tém po tvorbe jednoduchých sekvenčných programov. Za metodicky nesprávne pokladáme nasledovné praktiky výučby: • posunutie výučby vlastných funkcií až po výučbe cyklov, vetvenia, či zložených údajových typov, čo spôsobuje, že žiaci majú menej príležitostí precvičiť: o stratégie riešenia problémov – dekompozíciu a hľadanie vzorov, o použitie funkcie bez parametrov, • absenciu úloh nevyžadujúcich si písanie programového kódu, ktoré sú zamerané na analýzu problémov (obrázkov) a hľadanie vzorov tvoriacich podproblémy (vyskytujúcich sa v obrázkoch). Funkciu predstavíme metaforicky ako pečiatku so vzorom, pomocou ktorej odtlačíme vzor na určité miesta, čím vytvoríme obrázok.	
Dominantné vyučovacie metódy a formy	*Príprava učiteľa a pomôcky*
• Bádateľská metóda (model 5E) – štruktúrované bádanie, • individuálna a skupinová forma práce žiakov.	Softvérové vybavenie: • lokálna inštalácia jazyka Python vo verzii 3.x • lokálne inštalované vývojové prostredie JetBrains PyCharm Edu (resp. IDLE) Pomôcky: • pracovný list a pracovné súbory pre žiakov: **I_SS_03_Python_PL.pdf, I_SS_03_Python_PL_pracovne.zip** • pracovný list a súbory s riešeniami úloh pre učiteľa: **I_SS_03_Python_PL_riesenia.pdf, I_SS_03_Python_PL_riesenia.zip** • zbierka úloh a súbory s riešeniami úloh pre učiteľa: **I_SS_03_Python_Z_riesenia.pdf, I_SS_03_Python_Z_riesenia.zip** • tabuľka pre učiteľa pre zápis výsledkov žiackych riešení úloh z pracovného listu: **I_SS_03_Python_Vyhodnotenie.xlsx**
Diagnostika splnenia vzdelávacích cieľov	
Sebahodnotiaci test (uvedený na poslednej strane pracovného listu) a výsledky žiackych riešení úloh z pracovného listu.	

Úvod

Na predchádzajúcej hodine žiaci získali prvé skúsenosti s tvorbou Python programov zameraných na korytnačiu grafiku. Na tejto hodine rozšírime programovanie jednoduchých obrázkov na programovanie obrázkov pozostávajúcich zo vzorov. Najprv necháme žiakov analyzovať obrázky (resp. program) a hľadať v nich opakujúce sa **vzory** (resp. opakujúce sa skupiny príkazov), pre ktoré môžeme použiť metaforu **pečiatky**. Potom po preskúmaní hotového programu by mali žiaci začať chápať rozdiel medzi definovaním funkcie a jej volaním. V ďalšej časti hodiny analyzovaním, modifikovaním a vytváraním programov si žiaci precvičia poznatky o použití vlastných funkcií pri programovaní. Na základe vlastných programátorských skúseností a diskusii na tejto hodine

by mali žiaci uviesť a zdôvodniť výhody použitia vlastných funkcií (prehľadnosť kódu, znovupoužiteľnosť už raz napísaného programového kódu, lepšia lokalizácia chyby, deľba práce). Pri výučbe funkcii uvedieme žiakom aj význam komentárov v procese návrhu funkcie a tiež na zdokumentovanie funkcionality vytvorenej funkcie. V závere hodiny žiaci vyplnia sebahodnotiaci dotazník a sebahodnotiaci test.

Hlavnou pomôckou pre žiakov je pracovný list (**I_SS_03_Python_PL.pdf**) a archív (**I_SS_03_Python_PL_pracovne.zip**) s pracovnými súbormi (**03_04_corobim.py**, **03_06_zahada.py**).

Skúsenosti a poznatky získané z tejto hodiny žiaci využijú pri programovaní obrázkov s opakujúcim sa vzorom na pravidelných pozíciách, na čo bude zameraná nasledovná hodina.

Osnova vyučovacej hodiny (podľa modelu 5E):

- **Zapojenie** (6 minút) – analýza obrázkov a diskusia k vykresľovaniu obrázkov pozostávajúcich zo vzorov (úlohy 1 až 2 z pracovného listu)
- **Skúmanie** (6 minút) – skúmanie obrázkov/programov obsahujúcich opakujúce sa vzory/ skupiny príkazov smerujúce k potrebe zavedenia nového príkazu (funkcie) pre skupinu príkazov, ktorá je v programe uvedená viackrát (úlohy 3 až 4 z pracovného listu)
- **Vysvetlenie** (8 minút) – diskusia a zhrnutie spôsobu ako definovať a volať nový vlastný príkaz (funkciu) a tiež významu jeho použitia pri riešení problémov
- **Rozpracovanie** (14 minút) – programovanie grafických problémov s náročnejším vzorom či viacerými vzormi (úlohy 5 až 7 z pracovného listu)
- **Vyhodnotenie** (6 minút) – vyriešenie sebahodnotiaceho testu (úloha 8 z pracovného listu) s našim vyhodnotením

ZAPOJENIE (CCA 6 MIN)

Žiaci riešia úlohy 1 a 2, v ktorých majú analyzovať obrázky a rukou vykresliť (vyznačiť) vzory, z ktorých sú vytvorené uvedené obrázky.

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



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

Uveďte koľkokrát sa otočil valček so vzorom, ktorý vymaľoval na stenu uvedený obrázok: .....-krát

Riešenie:

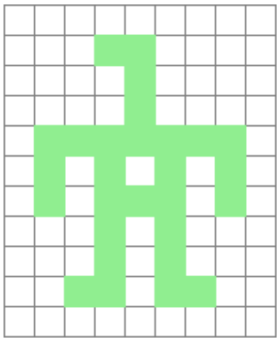
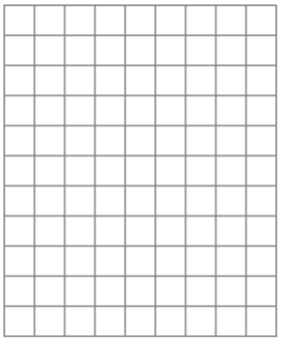
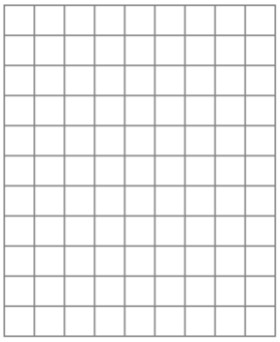


3-krát

V **úlohe 1** je obrázok s ľudovým motívom tvorený jedným vzorom, ktorého pozície sú pravidelne lineárne rozmiestnené. Táto úloha je propedeutickou pre programovanie cyklov, ale tiež

pre programovanie viacerých vnorených funkcií. Je tiež inšpiráciou pre neskoršie vykresľovanie známych ľudových či vlastných ozdobných motívov (STEAM úloha).

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

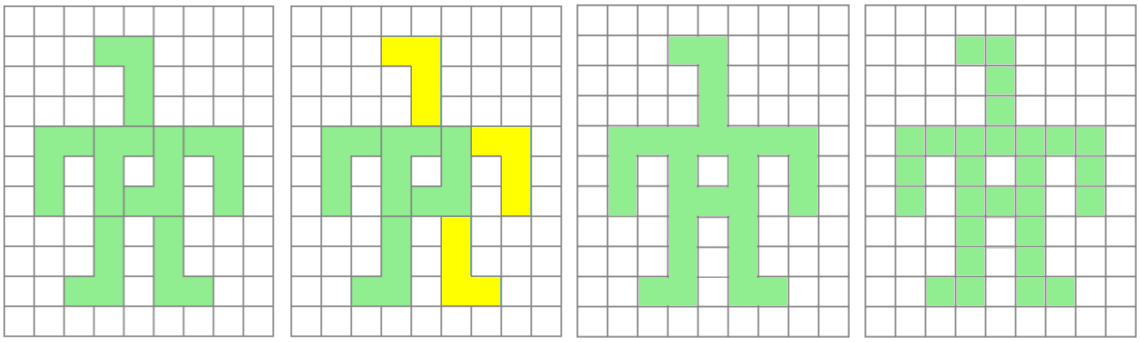
Uveďte, či stačí na opečiatkovanie celého tvaru panáka pečiatka s jedným vzorom: áno – nie

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

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

Riešenie:

Úloha môže mať viacero riešení, ktoré sú uvedené nižšie:



Na opečiatkovanie panáčika stačí pečiatka s jedným vzorom (riešenia na obrázkoch 1, 3 a 4). Ďalšie riešenie úlohy dostaneme, ak vzor v tvare písmena L (s plochou 4 základných štvorcov) rozdelíme na dva rovnaké vzory – obdĺžniky s veľkosťou 1×2 základné štvorce.

V **úlohe 2** sa od žiakov očakáva nájdenie vzorov, z ktorých pozostáva zelená postavička panáka vykreslená v štvorcovej sieti. Úloha je formulovaná ako divergentná s viacerými možnými riešeniami, pričom sú žiakom ako pomôcka k dispozícii ďalšie dve štvorcové siete. Jedným z možných riešení je dvojica vzorov – písmena L tvoreného 4 základnými štvorcami a jeho osovo symetrického obrazu. Ak pripustíme použitie obojstrannej pečiatky, možným riešením je aj vzor s písmenom L tvoreným 4 základnými štvorcami. Ďalšími riešeniami s jedným vzorom môžu byť: základný štvorec, obdĺžnik z dvoch základných štvorcov, celá postavička panáka. Túto divergentnú úlohu uzavrieme diskusiou k možným riešeniam a tiež výhodám (obmedzeniam) vykresľovania obrázkov obsahujúcich vzory.

Možné otázky v diskusii:

1. Ako by vyzeralo riešenie s čo najmenším jedným vzorom? (Možná odpoveď: vzorom je základný štvorec)
2. Ako by vyzeralo riešenie s čo najmenším počtov odtlačkov jedného vzoru? (Možná odpoveď: vzorom je celá postavička panáka)
3. Ako by vyzeralo riešenie s čo najväčším vzorom (vzormi) opakujúci sa aspoň dvakrát v obrázku? (Možná odpoveď: jedným vzorom je písmeno L zložené zo 4 základných štvorcov a druhým vzorom je nepriamo zhodný osovo symetrický obraz písmena L)
4. Aké výhody a obmedzenia má vykresľovanie obrázkov pečiatkovaním vzorov oproti vykresľovaniu bez pečiatkovania? (Možná odpoveď: výhodou je znovupoužitie pečiatky a možno aj kratší zápis riešenia, obmedzením je, že s danými pečiatkami vzorov nevieme vykresliť ľubovoľný obrázok)

Ak máme dostatok času a patrične motivovaných žiakov, môžeme zaradiť aj ďalšiu otázku:

5. Doplňte obrázok postavičky panáčika o jeden základný štvorec tak, aby bol celý obrázok osovo symetrický podľa zvislej osi. Aké budú iné možné riešenia úlohy s použitím jedného vzoru okrem riešenia okrem so vzorom jedného základného štvorca a riešenia so vzorom celej postavičky panáčika? (Možná odpoveď: vzhľadom na prvočíselný počet základných štvorcov tvoriacich postavičku panáčika, neexistuje žiadne iné riešenie s jedným vzorom)

SKÚMANIE (CCA 6 MIN)

V tejto časti hodiny necháme žiakom riešiť úlohy 3 a 4 z pracovného listu, v ktorých majú preskúmať obrázky a tiež programový kód, ktoré obsahujú opakujúce sa vzory.

Úloha 3 *Na uvedenom obrázku nájdite a vyznačte vzor, z ktorého vieme zostaviť daný obrázok.*



V programe na vykreslenie daného obrázku vyznačte časti, ktoré vykresľujú nájdený vzor.

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

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

tabula.mainloop()
```

V **úlohe 3** majú žiaci preskúmať obrázok aj odpovedajúci programový kód a v oboch nájsť opakujúci sa vzor. Prvá časť úlohy je (podobne ako úlohy 1 a 2) zameraná na nájdenie opakujúceho sa

vzoru. V druhej časti úlohy majú žiaci v programovom kóde vyznačiť časti (skupiny príkazov), ktoré vykresľujú nájdený vzor. Žiakom môžeme prípadne pripomenúť, že kruhy sa dajú vykresľovať aj so zdvihnutým perom. Ak by bolo pero dolu, tak by sa pri posunoch do nových pozícii vykresľovali farebné úsečky. Je zaujímavé pozorovať, koľkým žiakom bude prekážať, že sa 3-krát opakujú rovnaké skupiny príkazov. Možno si z predchádzajúcich kurzov programovania niektorí spomenú, že takúto rovnakú skupinu príkazov by mohli označiť ako nový príkaz a potom ho v programe uviesť na patričných miestach. Ak nie, nič sa nedeje, lebo sa k tomu vrátime v nasledujúcej časti Vysvetlenie.

Úloha 4 *Preskúmajte uvedený program* ***03_04_corobim.py****. (Poznámka: Na konci riadku 3 s novým príkazom* `vzor()` *je uvedená dvojbodka. Riadky 4 až 8 sú odsadené dohodnutým počtom medzier, napr. 4.)*

```
1  import turtle
2
3  def vzor():
4      # definovanie vykreslenia vzoru
5      pero.dot(50, "black")
6      pero.forward(50)
7      pero.dot(50, "lightgray")
8      pero.forward(50)
9
10 tabula = turtle.Screen()
11 pero = turtle.Turtle()
12 pero.penup()
13
14 vzor()           # vykreslenie vzoru
15 pero.right(90)   # otocenie sa o 90 stupnov vpravo
16 vzor()           # vykreslenie vzoru
17
18 tabula.mainloop()
```

a) Popíšte alebo nakreslite obrázok, ktorý vykreslí tento program: Vykreslí obrázok z úlohy 3 bez 2 kruhov

b) Vysvetlite aký je rozdiel medzi príkazom `def vzor():` *na riadkoch 3 až 8 a príkazom* `vzor()` *na riadkoch 14 a 16: Pomocou* `def vzor()` *definujeme ako sa má vykresliť vzor a pomocou novej funkcie* `vzor()` *tento vzor vykreslíme.*

c) Uveďte akú zmenu v uvedenom programe treba urobiť, ak by ste chceli vykresliť obrázok z úlohy 3: Doplniť pred riadok 14 ešte jedno volanie funkcie `vzor()`

V nadväznosti na úlohu 3, z ktorej možno vzišla potreba použitia nového príkazu (funkcie), necháme žiakom v **úlohe 4** preskúmať hotový program **03_04_corobim.py**. V ňom je uvedený kód pre definovanie a volanie nového príkazu (funkcie), s ktorým sa prvýkrát stretnú pri programovaní v jazyku Python.

VYSVETLENIE (CCA 8 MIN)

Po skúsenostiach žiakov zo skúmania riešení úloh 3 a 4 rozvinieme diskusiu, pomocou ktorej odhalíme prvotné žiacke predstavy o vytvorení a použití vlastnej funkcie a nasmerujeme žiakov k ich správnemu pochopeniu:

1. V čom je upravené riešenie úlohy 4 využívajúce vlastné funkcie lepšie či horšie ako riešenie úlohy 3 bez vlastných funkcií? (Možná odpoveď: programový kód je kratší a prehľadnejší)
2. V ktorej časti programu definujeme novú funkciu a v ktorej ju voláme? Mohli by sme najprv volať funkciu a až potom ju definovať? (Možná odpoveď: funkciu definujeme na začiatku programu a voláme ju až po jej definovaní. Poznámka: Odporúča sa definovať funkcie na začiatku programu po klauzule import)
3. Koľkokrát definujeme novú funkciu a koľkokrát ju môžeme volať? (Možná odpoveď: funkciu definujeme raz, ale volať ju môžeme koľkokrát chceme/potrebujeme)
4. Z akých častí pozostáva definovanie novej funkcie? (Možná odpoveď: z hlavičky s názvom novej funkcie a z tela odsadeného medzerami)
5. Kde končí definícia novej funkcie? (Možné odpovede: ak za príkazmi tela nasleduje prázdny riadok; ak za príkazmi tela nasledujú príkazy, ktoré už nie sú odsadené)
6. Ako sa zmení vykonávanie programu ak neuvedieme v ňom komentáre? (Možná odpoveď: nijako, komentáre nemajú vplyv na beh programu)
7. Aký význam pre programátora majú komentáre? Nie je to zbytočná strata času pri ich uvádzaní? (Možné odpovede: áno je to zbytočnosť; komentáre slúžia na zdokumentovanie časti programu, aby sme rýchlejšie (hlavne s odstupom času) pochopili čo robia jednotlivé časti programu)

Po diskusii so žiakmi zhrnieme nové učivo – uvedieme **dekompozíciu** ako jednu zo stratégii riešenia problémov, čo ilustrujeme pomocou schematických zápisov dvoch príkladov funkcií – na vykreslenie obrázku zvieraťa pozostávajúceho z hlavy, trupu a končatín; na vypísanie povinnosti v pracovnom týždni uvedením povinností v jednotlivých pracovných dňoch.

Program na vykreslenie zvieraťa	Program na výpis povinností pracovného týždňa

`def hlava():`
`    # príkazy na vykreslenie hlavy`
 	
`def trup():`
`    # príkazy na vykreslenie trupu`
 	
`def končatiny():`
`    # príkazy na vykreslenie končatín`
 	
`hlava()`
`trup()`
`končatiny()`
	`def pondelok():`
	`    # výpis zoznamu povinnosti v pondelok`
	
	`def utorok():`
	`    # výpis zoznamu povinnosti v utorok`
	
	`def streda():`
	`    # výpis zoznamu povinnosti v stredu`
	
	`def štvrtok():`
	`    # výpis zoznamu povinnosti vo štvrtok`
	
	`def piatok():`
	`    # výpis zoznamu povinnosti v piatok`
	
	`pondelok()`
	`utorok()`
	`streda()`
	`štvrtok()`
	`piatok()`

Vysvetlíme, že pomocou kľúčového slova **def** definujeme **vlastné funkcie** (`hlava()` ... , `pondelok()`...), ktorých vykonávanie v programe vyvoláme uvedením ich názvu doplnenom o okrúhle zátvorky (podobne ako ich majú uvedené aj ostatné štandardné príkazy, napr. `forward()` či `penup()`).

Ďalej uvedieme **hľadaj vzor** ako ďalšiu stratégiu riešenia problémov, pomocou ktorej vieme rozpoznať opakujúce sa podproblémy (vzory), čo ilustrujeme pomocou schematických zápisov dvoch príkladov funkcií – na vykreslenie obrazu s jedným, resp. dvoma vzormi.

Program na vykreslenie obrazu s 1 vzorom	Program na vykreslenie obrazu s 2 vzormi

`def pečiatka():`
`    '''`
`    príkazy na vykreslenie vzoru,`
`    z ktorého pozostáva obrázok`
`    '''`
 	
`def presun1():`
`    # príkazy na 1. presun grafického pera`
 	
`def presun2():`
`    # príkazy na 2. presun grafického pera`
 	
`pečiatka()`
`presun1()`
`pečiatka()`
`presun2()`
`pečiatka()`
	`def pečiatka1():`
	`    # príkazy na vykreslenie vzoru 1`
	
	`def pečiatka2():`
	`    # príkazy na vykreslenie vzoru 2`
	
	`def presun1():`
	`    # príkazy na 1. presun grafického pera`
	
	`def presun2():`
	`    # príkazy na 2. presun grafického pera`
	
	`pečiatka1()`
	`presun1()`
	`pečiatka2()`
	`presun2()`
	`pečiatka1()`

Zdôrazníme význam použitia komentárov na sprehľadnenie zápisov jednotlivých funkcií programu, a to jednak v procese návrhu funkcie (ako poznámky, čo treba urobiť – „to do“ či na znefunkčnenie časti kódu, ale to ukážeme až vtedy, ak nastane takáto situácia) a tiež na zdokumentovanie funkcionality funkcie (ako komentáru čo robí daná funkcia). Okrem jednoriadkových komentárov ukážeme žiakom príklad viacriadkového komentára vymedzeného počiatočnou a koncovou trojicou apostrofov (úvodzoviek). Dokumentačné reťazce uvedieme až pri funkciách s parametrami.

Napokon prediskutujeme so žiakmi a následne zhrnieme výhody použitia vlastných funkcií (písanie prehľadnejšieho kódu odpovedajúcemu riešenému problému; lepšia lokalizácia a opravovanie chýb; skrátenie kódu viacnásobnou znovupoužiteľnosťou už raz napísaného programového kódu; deľba práce medzi viacerých programátorov). Pochopeniu významu použitia vlastných funkcií napomôže ak žiaci uvedú príklady objektov alebo činností zo sveta okolo nás, ktoré obsahujú jeden či viaceré opakujúce sa vzory a tiež uvedenie výhod, ktoré nám prináša rozpoznanie vzorov, z ktorých pozostávajú objekty alebo činnosti zo sveta okolo nás.

ROZPRACOVANIE (CCA 14 MIN)

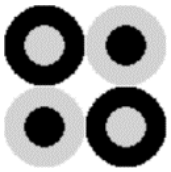
V tejto časti hodiny si žiaci analýzou, modifikáciou a tvorbou programov uvedených v úlohách 5 až 7 z pracovného listu precvičia programovanie vlastných funkcií, použitie stratégií riešenia problémov – dekompozíciu a hľadaj vzor, precvičia osvojené príkazy korytnačej grafiky. Žiakom zdôrazníme, že pri riešení grafických úloh je výbornou pomôckou náčrt obrázku s uvedenými rozmermi (v štvorcovej či inej mriežke), ktorý je veľmi nápomocný vo fáze rozboru úlohy. Pri programovaní funkcii dbáme na používanie prirodzených názvov vlastných funkcií zodpovedajúcich ich správaniu, používanie komentárov k popisu správania funkcie, písanie programov podľa štýlu PEP-8 (funkcie uvádzame spolu na začiatku programu oddelené dvomi prázdnymi riadkami).

Úloha 5 *Vytvorte programy na vykreslenie uvedených obrázkov, v ktorých použijete vlastnú funkciu pre vykreslenie vzoru.*

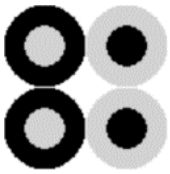
A



B



C



Riešenie:

```
import turtle

def vzor():
    # vykreslenie vzoru s dvoma dvojicami sustrednych kruhov
    pero.dot(50, "black")
    pero.dot(25, "lightgray")
    pero.forward(50)
    pero.dot(50, "lightgray")
    pero.dot(25, "black")
    pero.forward(-50)


def obrazok_A():
    # vykreslenie obrazku A
    vzor()
    pero.forward(2 * 50)
    vzor()


def obrazok_B():
    # vykreslenie obrazku B
    vzor()
    pero.forward(50)
    pero.right(90)
    pero.forward(50)
    pero.right(90)
    vzor()


def obrazok_C():
    # vykreslenie obrazku C
    vzor()
    pero.right(90)
    pero.forward(50)
    pero.right(-90)
    vzor()


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

obrazok_A()
#obrazok_B()
#obrazok_C()

tabula.mainloop()
```

Úloha 5 je analogická s úlohami 3 a 4, je zameraná na precvičenie tvorby vlastnej funkcie pri kreslení troch obrázkov, v ktorých sa opakujú rovnaké vzory. Očakávame, že v žiackom riešení súčasťou funkcie vykresľujúcej vzor bude aj posun grafického pera na nasledujúcu pozíciu v smere kreslenia vzoru. Žiakom tu môžeme ukázať aj alternatívne riešenie, v ktorom funkcia vykreslí vzor

a posunie pero na počiatočnú pozíciu kreslenia vzoru. Toto alternatívne riešenie je vhodné pre vykreslenie obrázku C a tiež pre podobné prípady, keď nie je zreteľné postupné nadväzujúce vykresľovanie vzorov za sebou v určitom smere. Príkaz `pero.hideturtle()` predstavíme žiakom až vtedy, kedy si ho sami vyžiadajú – v situácii, keď im prekáža grafické pero pri kompletnom zobrazení vykresleného obrázka.

Úloha 6 *Preskúmajte uvedený program* ***03_06_zahada.py****.*

```
1   import turtle
2
3   def obrazok():
4       pero.forward(100)
5       pero.left(120)
6       pero.forward(200)
7       pero.left(-120)
8       pero.forward(100)
9       pero.left(-120)
10      pero.forward(200)
11      pero.left(120)
12
13  tabula = turtle.Screen()
14  pero = turtle.Turtle()
15  pero.pensize(10)
16  pero.pencolor("green")
17  pero.fillcolor("yellow")
18
19  pero.begin_fill()
20  obrazok()
21  pero.end_fill()
22
23  tabula.mainloop()
```



a) Popíšte obrázok, ktorý vykreslí tento program: Vykreslí presýpacie hodiny so zelenou obrubou a žltou výplňou

b) Vysvetlite význam príkazov `fillcolor()`, `begin_fill()`, `end_fill()`*:*

`fillcolor()` *nastaví farbu výplne, kresba medzi príkazmi* `begin_fill()` *a* `end_fill()` *sa vykreslí s nastavenou farbou pera a farbou výplne*

c) Uvedený program upravte, aby zobrazil vedľa seba obrázky s opačnými farbami pera a výplne.

Do riadku 22 doplníme nasledovný kód:

```
pero.penup()
pero.forward(110)
pero.pendown()

pero.pencolor("yellow")
pero.fillcolor("green")
pero.begin_fill()
presypacky()
pero.end_fill()
```

Úloha 6 je zameraná na analýzu programového kódu obsahujúceho vlastnú funkciu uloženého v pracovnom súbore **03_06_zahada.py**, na pochopenie príkazov na vypĺňanie uzatvoreného útvaru tvoreného postupnosťou čiar (`fillcolor()`, `end_fill()`, `end_fill()`) a na modifikáciu programu na vykreslenie dvoch obrázkov vedľa seba s opačnými farbami pera a výplne.

Úloha 7 *Vytvorte program na vykreslenie uvedeného obrázku.*



```
Riešenie:

import turtle

def listnaty():
    # vykreslenie listnateho stromu
    pero.pendown()
    pero.pencolor("brown")
    pero.forward(100)
    pero.penup()
    pero.dot(100, "green")
    pero.forward(-100)


def ihlicnaty():
    # vykreslenie ihlicnateho stromu
    pero.pendown()
    pero.pencolor("brown")
    pero.forward(130)
    pero.pencolor("green")
    pero.fillcolor("green")
    pero.begin_fill()
    pero.left(150)
    pero.forward(80)
    pero.left(120)
    pero.forward(80)
    pero.left(120)
    pero.forward(80)
    pero.right(30)
    pero.end_fill()
    pero.penup()
    pero.forward(-130)


def posun():
    # posunutie sa na nasledovnu poziciu vpravo
    pero.right(90)
    pero.forward(100)
    pero.left(90)


tabula = turtle.Screen()
pero = turtle.Turtle()
pero.pensize(20)
pero.left(90)

ihlicnaty()
posun()
posun()
listnaty()
posun()
ihlicnaty()
posun()
listnaty()

pero.hideturtle()
tabula.mainloop()
```

V **úlohe 7** si žiaci precvičia použitie vlastnej funkcie vykresľovaním obrázku tvoreného dvoma vzormi (pre ihličnatý a listnatý strom) a príkazov na vypĺňanie uzatvoreného útvaru tvoreného postupnosťou čiar. Očakávame, že v žiackom riešení sa pri vykresľovaní vzorov grafické pero presunie na počiatok vykresľovania vzorov (do dolnej časti kmeňa ihličnatého, resp. listnatého stromu).

Týmto je zavŕšená precvičovacia časť hodiny. Pre ďalšie prípadne precvičenie tvorby a použitia vlastných funkcií či domácu úlohu poslúžia **úlohy 9** až **14** uvedené v zbierke úloh (**I_SS_03_Python_Z_riesenia.pdf**).

VYHODNOTENIE (CCA 6 MIN)

V závere hodiny necháme žiakom vyriešiť sebahodnotiaci test (úloha 8 z pracovného listu).

Úloha 8 *Zakrúžkovaním vyberte skupinu (skupiny) príkazov, pomocou ktorej (ktorých) sa vykreslí uvedený obrázok.*



Skupina príkazov A

```
def vzor():
    pero.dot(40, "black")
    pero.forward(40)
    pero.dot(40, "yellow")
    pero.forward(40)
    pero.dot(40, "red")
    pero.forward(40)
    pero.dot(40, "blue")
    pero.forward(40)
    pero.dot(40, "yellow")
    pero.forward(40)
    pero.dot(40, "red")
    pero.forward(40)


vzor()
vzor()
~~vzor()~~
```

Skupina príkazov B

```
def belgicko():
    pero.dot(40, "black")
    pero.forward(40)
    pero.dot(40, "yellow")
    pero.forward(40)
    pero.dot(40, "red")
    pero.forward(40)


def rumunsko():
    pero.dot(40, "blue")
    pero.forward(40)
    pero.dot(40, "yellow")
    pero.forward(40)
    pero.dot(40, "red")
    pero.forward(40)


belgicko()
rumunsko()
belgicko()
rumunsko()
```

Skupina príkazov C

```
def dvojgulka():
    pero.dot(40, "yellow")
    pero.forward(40)
    pero.dot(40, "red")
    pero.forward(40)


def sestgulka():
    pero.dot(40, "black")
    pero.forward(40)
    dvojgulka()
    pero.dot(40, "blue")
    pero.forward(40)
    dvojgulka()


sestgulka()
sestgulka()
```

Skupinu s chybným riešením opravte. Stručne okomentujte uvedené riešenia úlohy: Skupina príkazov A obsahuje navyše 1 volanie funkcie vzor(), ostatné skupiny príkazov B a C sú správnymi riešeniami úlohy. Skupina príkazov C využíva vnorené volanie funkcii (vo funkcii `sestgulka()` sa používa funkcia `dvojgulka()`), je na našom rozhodnutí podľa podmienok v triede, či zadáme a prípadne vyhodnotíme aj túto skupinu príkazov.

V uvedenej úlohe majú žiaci vybrať skupinu (skupiny) príkazov, ktoré vykreslia obrázok s radom farebných kruhov. Po vyriešení úlohy žiakom zdôrazníme rozdiel medzi definovaním a volaním vlastnej funkcie a poskytneme správne riešenie úlohy. Výsledky žiackych riešení úloh a sebahodnotiaceho testu z pracovného listu zapíšeme do tabuľky v súbore **I_SS_03_Python_Vyhodnotenie.xlsx**, resp. do tabuľky na cloude. Uvedené výsledky žiakov nám poslúžia pri identifikovaní problematického učiva a následnej našej intervencii v ďalšej výučbe a rovnako autorom metodiky pre vylepšenie metodiky, pracovného listu či ostatných príloh metodiky.