

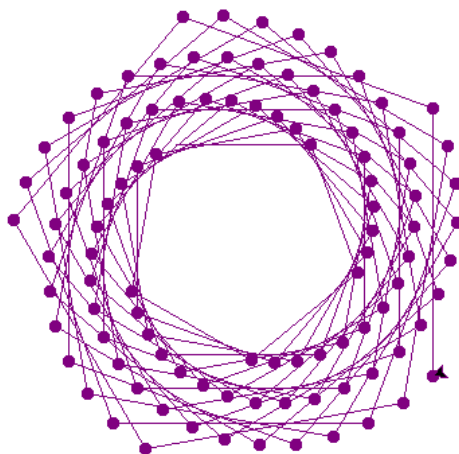
---

# PYTHON A KORYTNAČIA GRAFIKA

---

Metodický materiál pre vyučovanie základov programovania pre gymnáziá

Eva Mészárosová



Autor: Mgr. Eva Mészárosová

Názov: Python a korytnačia grafika - Metodický materiál pre vyučovanie základov programovania pre gymnáziá

Vydavateľ: Knižničné a edičné centrum FMFI UK, Bratislava, 2017

Metodický materiál bol vytvorený v rámci dizertačného výskumu s názvom *Programovanie v jazyku Python ako súčasť vyučovania informatiky* na Katedre základov a vyučovania informatiky, Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave.

Internetová adresa: [www.edi.fmph.uniba.sk/~meszarosova/Python](http://www.edi.fmph.uniba.sk/~meszarosova/Python)

© Mgr. Eva Mészárosová

**ISBN 978-80-8147-080-6**

# Obsah

Obsah .....	3
O metodike.....	4
Úvod do Pythonu a korytnačej grafiky.....	5
1. hodina: Úvod do Pythonu a IDLE a korytnačej grafiky.....	6
2. hodina: Kreslenie pomocou korytnačky .....	12
Premenné.....	15
3. hodina: Premenné.....	15
For cykly .....	22
4. hodina: For cykly .....	22
5. hodina: For cykly s funkciou range() .....	25
6. hodina: Cykly s riadiacou premennou a ďalšou premennou.....	29
7. a 8. hodina: Vnorené cykly .....	31
Funkcie .....	37
9. hodina: Funkcie .....	37
10. hodina: Cyklus vo funkcii .....	42
11 a 12. hodina: Funkcie s parametrami .....	46
Nastavovanie polohy korytnačky a náhodnosť.....	51
13. hodina: Zmena polohy korytnačky a náhodnosť .....	51
14. hodina: Zmena polohy korytnačky a náhodné čísla .....	56
Vetvenie .....	61
15. a 16. hodina: Vetvenie .....	61
Projekt.....	69
17. hodina: Zadanie projektu.....	69
Hodnotenie .....	70
Zoznam použitých príkazov.....	71
Použitá literatúra .....	73
Príloha: Pracovné listy.....	75

## O metodike

Tento materiál je určený pre učiteľa všeobecno-vzdelávacieho predmetu informatika na štvorročnom gymnáziu a je určený na vyučovanie základov programovania. Materiál je rozdelený na niekoľko tém, ktoré sa venujú úvodu do programovania v jazyku Python pomocou korytnačej grafiky.

Súčasťou metodiky je materiál pre učiteľa s textom ku každej téme, ktorý obsahuje popis vyučovacej hodiny a úlohy spolu s ich riešeniami. Ďalej sú súčasťou metodiky pracovné listy pre žiakov. Metodika obsahuje šesť tém: úvod do jazyka, premenné, for cykly, funkcie, poloha korytnačky a náhodnosť a podmienený príkaz. Témy sme následne rozdelili na jednotlivé vyučovacie hodiny.

Pre každú tému sme definovali vstupné vedomosti a zručnosti žiakov, vzdelávacie ciele jednotlivých hodín a rozvíjané kompetencie na základe výkonového štandardu podľa iŠVP (ŠPU, 2015). Ďalej nasleduje rozčlenenie témy na jednotlivé vyučovacie hodiny, návrh pedagogických postupov na hodine a popis možného priebehu vyučovacej hodiny, pričom priamo v metodike nájde učiteľ zadania úloh, ktoré žiaci nájdu vo svojich pracovných listoch. Výhodou je, že učiteľ zároveň so zadaním vidí jeho riešenie, prípadne ďalšie poznámky a odporúčania.

# Úvod do Pythonu a korytnačej grafiky

## Cieľ vyučovacích hodín

- Zoznámiť žiakov s novým programovacím jazykom Python a jeho vývojovým prostredím IDLE.
- Oboznámiť žiakov s ideou korytnačej grafiky.
- Vytvoriť prvý program, ktorý po spustení nakreslí obrázok.
- Uvažovať o vlastnostiach vykonávateľa – korytnačky.

## Nové pojmy, poznatky a zručnosti

- Importovanie modulu `turtle`, vytvorenie korytnačky, príkazy na riadenie korytnačky  
`forward()`, `back()`, `right()`, `left()`
- Príkazový a programovací režim.
- Spustenie prostredia IDLE, vytvorenie programu a jeho uloženie.
- Ukončenie práce s IDLE.

---

Téma je rozdelená na dve vyučovacie hodiny, ktoré sú zamerané na to, aby sa žiaci naučili pracovať s prostredím IDLE – príkazovým a programovacím režimom, a aby sa v ňom ľahko orientovali.

Ďalším cieľom hodiny je, aby žiaci uvažovali o vlastnostiach vykonávateľa – korytnačky. Žiaci kreslia jednoduché obrázky zamerané na automatizáciu základných príkazov na pohyb korytnačky.

Odporúčame učiteľom, aby si v rámci týchto dvoch hodín všímali, či žiaci pracujú podľa pokynov – v programovacom režime pracujú s tromi samostatnými oknami, pričom pre niektorých žiakov môže byť náročné orientovať sa v nich. Tieto hodiny sú určené na to, aby sa žiaci naučili pracovať v prostredí IDLE, aby si riešením jednoduchých úloh osvojili základné príkazy na pohyb korytnačky, aby na nasledujúcich hodinách, na ktorých sa už stretnú s novými pojmami a konštrukciami v programovaní, ovládali základné inštrukcie na pohyb korytnačky a prácu v prostredí IDLE.

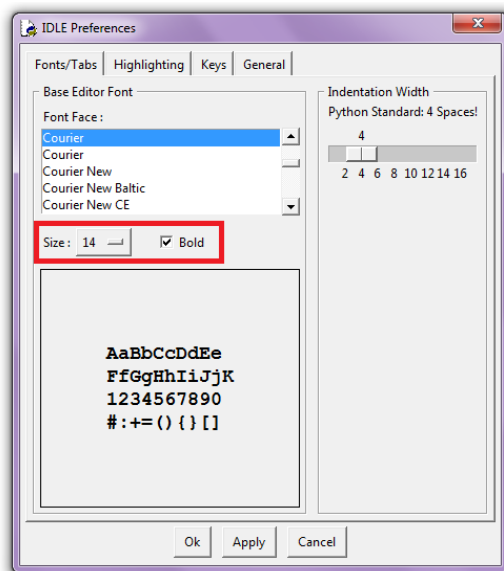
## Poznámky pre učiteľa

---

Python je voľne stiahnuteľný zo stránky [www.python.org/downloads/](http://www.python.org/downloads/) – v súčasnosti je vo verzii 3.4 alebo 3.5. V materiáloch budeme pracovať vo vývojovom prostredí IDLE, ktoré je súčasťou balíka **Python 3.5**.

Ak budete prezentovať riešenie úloh pomocou dataprojektora, je vhodné zmeniť v prostredí IDLE **veľkosť písma**, aby bolo lepšie viditeľné:

Options ► Configure ► IDLE ► Size – odporúčame veľkosť 14, Bold.



Program v Pythone musí byť pred svojím spustením uložený. Prostredie IDLE ho bežne ukladá, avšak predvolená je voľba, pri ktorej sa vždy najprv objaví okno s otázkou, či chce používateľ projekt uložiť. Takýto postup počas častého spúšťania programov zdržiava, takže je vhodné na všetkých počítačoch vypnúť okno s týmto postupom:

Options ► Configure IDLE ► General ► Autosave preferences: No Prompt

## 1. hodina: Úvod do Pythonu a IDLE a korytnačej grafiky

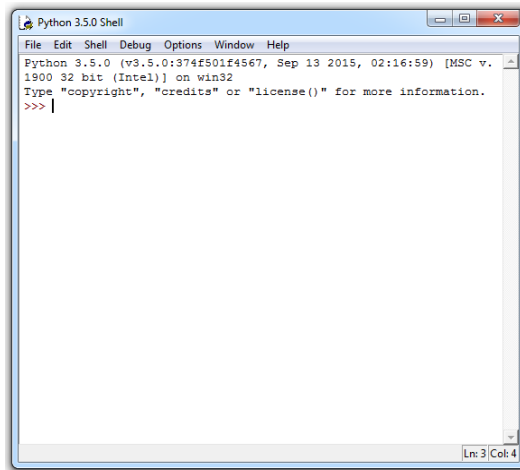
Pracovný list 1 – Úvod do Pythonu a korytnačej grafiky

### 1. Učiteľ povie žiakom niekoľko informácií o jazyku, ukáže im prostredie IDLE

- Programovací jazyk Python vytvoril Guido Van Rossum v roku 1989.
- Používajú ho aj v Google, Mozilla, Dropbox (produkt), Youtube.

Na mnohých univerzitách vyučujú úvod do programovania práve v tomto jazyku – napríklad na FMFI UK v Bratislave a Masarykovej Univerzite v Brne.

Po spustení prostredia sa objaví jedno okno, tzv. Shell, v ktorom môžeme písať priamo príkazy za znaky >>>. Takéto programovanie budeme nazývať príkazový režim.



**Príkazový režim:** Za šípkami >>> píšeme príkazy a stlačením tlačidla ENTER sa príkazy vykonajú.

## 2. Učiteľ ukáže žiakom v príkazovom režime vytvorenie korytnačky

---

```
>>> from turtle import *
```

Tento príkaz oznámi, že budeme pracovať s modulom turtle.

Okno sa ukáže až potom, ako zadáme prvý príkaz pre korytnačku.

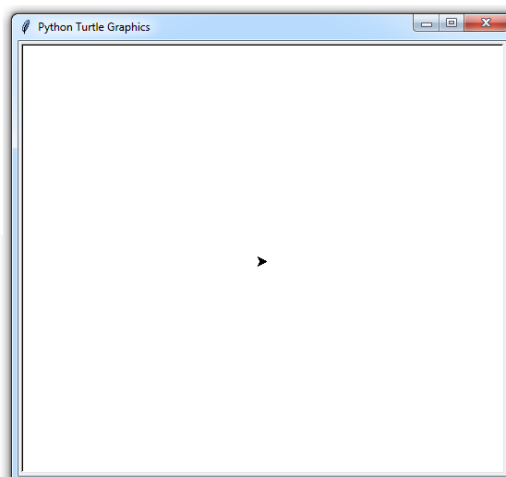
Zadajme príkaz `showturtle()`, ktorým sa korytnačka ukáže.

```
>>> showturtle()
```

Môžeme použiť aj skratku

```
>>> st()
```

Otvorí sa samostatné okno, v jeho strede sa nachádza korytnačka, ktorá je zobrazená v tvare čiernej šípky. Korytnačka je natočená k pravému okraju obrazovky, t. j. napravo.



### 3. Učiteľ ukáže žiakom príkazy na pohyb korytnačky (v príkazovom režime)

```
>>> forward(100) # skrátene fd(100)
```

Príkaz `forward(100)` posunie korytnačku dopredu o daný počet krokov. Počet krokov, ktoré má korytnačka prejsť zadávame do zátvoriek za príkazom. V predošlom príklade korytnačka prejde dopredu o 100 krokov. Každý príkaz má skratku, pre príkaz `forward(počet_krokov)` je skratka `fd(počet_krokov)`.

```
>>> back(50) # skrátene bk(50)
```

Príkaz `back(50)` posunie korytnačku vzad o 50 krokov.

```
>>> reset()
```

Príkaz `reset()` zmaže plochu a vráti korytnačku na pôvodnú pozíciu. Všimnite si, že hoci príkaz `reset()` nemá v zátvorke uvedené žiadne číslo, sú za ním prázdne zátvorky. Treba dbať na to, aby žiaci tieto zátvorky nezabudli zapísať, inak program iba vypíše správu, ale plocha sa nezmaže.

Učiteľ zadá prvú úlohu, ktorú rieši spoločne so žiakmi na tabuli. Popri tom diskutujú a ujasňujú si, ako sa korytnačka pohybuje.

1. úloha (spoločne – diskusiou)	Riešenie
Zistite, koľko krokov korytnačka prejde a akú dlhú čiaru nakreslí, ak vykoná túto postupnosť príkazov:  <pre>forward(100) forward(-90) forward(80) forward(-70) forward(60) forward(-50)</pre>	Korytnačka prejde 450 krokov a nakreslí 100 krokov dlhú čiaru.  Úlohu riešia žiaci najprv postupne bez toho, aby príkazy zadávali do príkazového režimu.  Žiaci si môžu nakresliť čiaru dlhú „100 krokov“ – rozdelenú na 10 častí a postupne vykonať jednotlivé príkazy korytnačky a tým odkrokovat úlohu.

### 4. Otočenie korytnačky

Učiteľ ukáže žiakom príkazy na otočenie korytnačky na počítači s projektorom.

```
>>> right(90) # rt(90)
```

Korytnačka sa otočí vpravo o 90 stupňov.



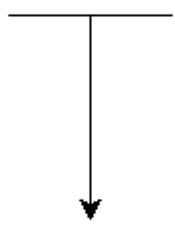
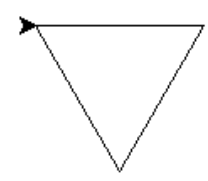
Aby žiaci videli skutočné otočenie korytnačky, medzi prvým otočením vpravo a otočením korytnačky vľavo učiteľ posunie korytnačku dopredu o niekoľko krokov, napr. fd(50).

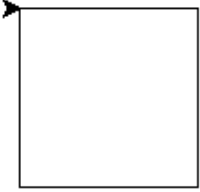
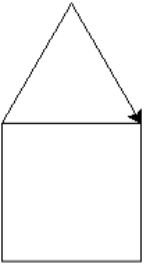
```
>>> left(90) # lt(90)
```

Korytnačka sa otočí vľavo o 90 stupňov.

## Pracovný list a riešenia úloh

Každý žiak by mal mať k dispozícii zadania na svojom počítači v PDF súbore alebo vytlačené na papieri. Žiaci riešia úlohy samostatne, pričom kreslia na papier (resp. do Skicáru), alebo zapisujú príkazy v príkazovom režime na počítači. Prvú úlohu riešia žiaci spoločne diskusiou.

2. úloha	Riešenie
<p>Zistite vzťah medzi príkazmi:</p> <pre>right(90) left(-90)</pre>	<p>right(90) otočí korytnačku do rovnakého smeru ako left(-90), t.j. ak bola otočená do smeru 35 stupňov tak po vykonaní oboch príkazov je otočená do rovnakého smeru.</p>
<p>3. úloha</p> <p>Nakreslite na papier(alebo v Skicári), čo nakreslí korytnačka, keď vykoná nasledovnú postupnosť príkazov. Svoje riešenie si overte zadáním príkazov v príkazovom režime.</p> <pre>forward(80) back(40) right(90) forward(100)</pre>	<p>Riešenie</p> <p>Žiaci by mali nakresliť na papier pohyb korytnačky, následne by si mali vyskúšať zadať tieto príkazy aj na počítači.</p> 
<p>4. úloha</p> <p>Napíšte príkazy v príkazovom režime, ktorými korytnačka nakreslí tieto obrázky</p> <p>a) rovnostranný trojuholník</p> 	<p>Riešenie</p> <pre>from turtle import * fd(100) rt(120) fd(100) rt(120) fd(100) rt(120)</pre>

<p>b) štvorec</p> 	<pre>from turtle import *  fd(100) rt(90) fd(100) rt(90) fd(100) rt(90) fd(100) rt(90)</pre>
<p>c) domček</p>  <p>Všimnite si, že domček sa skladá zo štvorca a z trojuholníka. Strany oboch útvarov majú rovnakú dĺžku, napr. 100.</p>	<pre>from turtle import *  fd(100) rt(90) fd(100) rt(90) fd(100) rt(90) fd(100) rt(30) fd(100) rt(120) fd(100)</pre>

## 5. Učiteľ ukáže programovací režim, ukladanie a spúšťanie programu

### Ukladanie programu:

Stlačením klávesovej skratky <Ctrl+S> alebo položky menu File ► Save sa uloží program. Je potrebné dávať si pozor na to, aby sme programom nedávali mená, ktoré patria do rezervovaných slov jazyka Python, ako turtle, math, random, .... Tieto môžu spôsobiť nefunkčnosť programu alebo chybu. Žiaci by sa mali naučiť používať také názvy pre svoje programy, aby aj po dlhšom čase dokázali bez otvorenia programu zistiť, čo je v programe. Ak budú riešiť úlohy z pracovných listov, môžu ich nazývať pre prvý pracovný list, napr. 01stvorec.py, 01domcek.py, 01pismL.py, atď.

### Doteraz sme pracovali v príkazovom režime:

V príkazovom režime zadávame príkazy (do riadka za znaky >>>). Každý zadaný príkaz sa vyhodnotí a korytnačka ho vykoná, resp. príkaz vyvolá inú reakciu, napr. zmazanie plochy. Po skončení vyhodnocovania riadka sa do ďalšieho riadka znovu vypíšu znaky >>>, a program očakáva zadanie ďalšieho príkazu.

## V programovacom režime:

Programovací režim znamená, že príkazy, ktoré používateľ zapíše sa nebudú vykonávať hneď, ale až po spustení celého programu, ktorý zapíšeme. Programovací režim preto potrebuje editor, v ktorom budeme zapisovať skupiny viacerých príkazov.

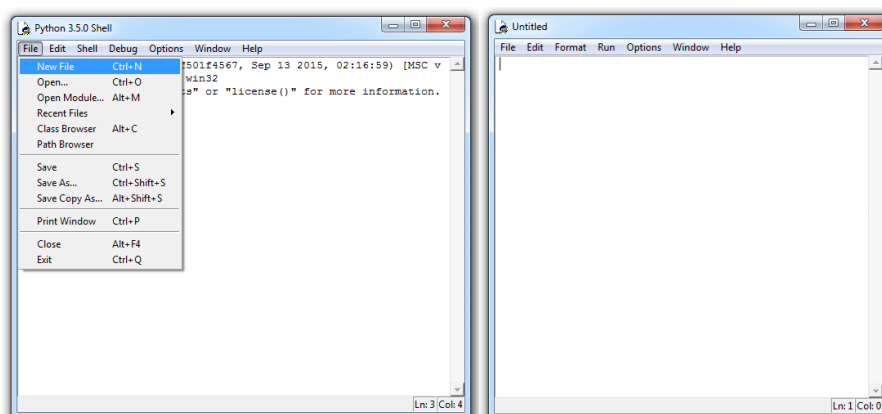
V IDLE otvoríme nové okno (položka menu File ► New Window), resp. stlačíme <Ctrl+N>. Otvorí sa nové textové okno, ale už bez znaku >>>. Do tohto okna píšeme príkazy (aj importovanie knižníc). Takto vytvorený program treba spustiť (položka menu Run, alebo stlačením klávesu <F5>). Po spustení sa v pôvodnom okne najprv celý Python reštartuje (zabudne všetko, čo sme ho doteraz naučili), následne vykoná všetky príkazy, ktoré sme zapísali do textového editora.

## Spustenie:

- stlačením tlačidla <F5>.

## Otvorenie uloženého súboru:

- uložený súbor s príponou .py môžeme otvoriť napríklad kliknutím pravého tlačidla myši na súbor a zvolením možnosti Edit with IDLE (Upraviť v IDLE)
- v príkazovom režime zvolením položky menu File ► Open.



## Pracovný list a riešenia úloh

### 5. úloha

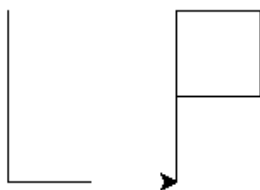
Nakreslite písmená veľkej abecedy:

L, E, H, K, M

Jednotlivé programy si ukladajte do samostatných súborov s názvom pismenoL.py, pismenoE.py, atď.

## 6. úloha

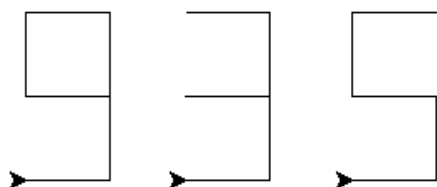
Nakreslite vaše iniciály. Iniciály sú prvé písmená vášho mena, napríklad iniciály pre Lenku Peknú:



## 7. úloha

Nakreslite digitálne číslice.

Číslicu začnite kresliť vždy v jej ľavom dolnom rohu. Po dokreslení presuňte korytnačku znovu do ľavého dolného rohu číslice. Nevadí, ak korytnačka pôjde viackrát po tej istej čiare.



## 2. hodina: Kreslenie pomocou korytnačky

Pracovný list 2 – Úvod do korytnačej grafiky s Pythonom

### 1. Zopakovanie príkazov z prvej hodiny, príkazového a programovacieho režimu

Učiteľ pracuje na počítači s projektorom, pomocou diskusie si žiaci zopakujú učivo z prvej hodiny. Učiteľ následne otvorí prostredie IDLE a v programovacom režime importuje knižnicu turtle, zopakujú si príkazy `forward()`, `back()`, `right()`, `left()`. Zopakujú si aj ukladanie nového súboru. Na tejto hodine postupne prechádzajú na skrátené názvy príkazov `fd()`, `bk()`, `rt()`, `lt()`.

### 2. Príkazy na zmenu polohy pera, farby a hrúbky

`penup()` # skrátené `pu()`

Korytnačka zdvihne pero, pri pohybe nebude kresliť.

`pendown()` # skrátené `pd()`

Korytnačka položí pero, pri pohybe bude kresliť.

`pencolor(farba)`

Nastaví korytnačke farbu pera – do zátvoriek za príkazom zapíšeme názov farby v apostrofoch napríklad: 'red', 'black', 'green', 'blue', 'brown', 'pink', 'white'. V Pythone sa môžu použiť namiesto apostrofov aj úvodzovky, napr. "blue", avšak nemožno ich miešať.

Učiteľ by sa mal rozhodnúť, či bude používať apostrofy alebo úvodzovky a programy potom písať s týmto symbolom. V našom materiáli používame apostrofy.

`pensize(hrúbka)`

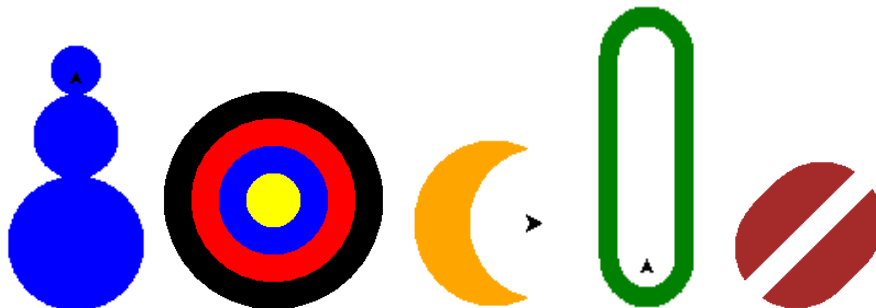
Nastaví hrúbku pera na danú veľkosť – do zátvoriek sa píše číslo, napríklad: `pensize(30)`.

## Pracovný list a riešenia úloh

Žiaci riešia úlohy z pracovného listu samostatne na počítači, pracujú v programovacom režime a riešenia ukladajú do samostatných súborov s vhodnými názvami podľa toho, čo program kreslí. Keďže ide o druhý pracovný list, navrhujeme všetky názvy programov začínať znakmi 02, napr. 02snehuliak.py, 02terc.py, atď.

### 1. úloha

Pomocou hrubých rôzne zafarbených čiar nakreslite tieto obrázky:



#### Pomôcka:

Kruh sa dá nakresliť ako veľmi hrubá veľmi krátka čiara:

```
pensize(50)
forward(0)
```

Kružnica sa dá nakresliť pomocou dvoch rôzne veľkých kruhov so spoločným stredom, pričom menší kruh má inú farbu:

```
pencolor('black')
pensize(50)
forward(0)
pencolor('white')
```

```
pensize(45)  
forward(0)
```

## 2. úloha

Neónový nápis vytvoríte nakreslením útvaru najprv hrubým perom tmavšou farbou a na následne tenším perom a svetlejšou farbou:



Tieto úlohy sú zamerané na automatizáciu základných príkazov pre korytnačku a prácu v prostredí IDLE. Žiaci by sa tiež mohli naučiť vyhľadávať opakujúce sa časti riešenia a kopírovať a vkladať príkazy v prostredí textového editora IDLE.

### Časté chyby žiakov

Pri nasadení našej metodiky do praxe sa pomerne často stávalo, že žiaci pre príkazy s parametrami pridávali medzeru medzi príkaz a ľavú zátvorku, ako napríklad: `fd (50)`. Pri takomto zápise program funguje a nevypíše chybu. Ale na ďalších hodinách sa v niektorých prípadoch stalo, že žiaci začali medzery dávať aj v názve príkazu jazyka Python ako napríklad `pen color ('blue')`, čo je už chyba, takýto príkaz v jazyku neexistuje a Python vypíše syntaktickú chybu. Preto odporúčame upozorniť žiakov na správny zápis príkazov už na prvých hodinách. Takisto sme si všimli, že žiaci často nezadávali parametre do príkazov, zabudli uviesť zátvorky za príkazmi alebo mali preklepy v názvoch príkazov. Tieto chyby si však väčšinou rýchlo opravili a bez pomoci učiteľa.

Učiteľom odporúčame všímať si, či žiaci pochopili rozdiel medzi príkazovým a programovacím režimom a či sa vedú orientovať v prostredí IDLE.

# Premenné

## Ciele vyučovacej hodiny

Žiak vie/dokáže:

- porozumieť pojmu premenná a jej využitiu v jazyku Python
- priradiť hodnotu do premennej a použiť hodnotu premennej
- identifikovať zo zadania úlohy, ktoré údaje musia byť zapamätané, resp. sa menia (a teda vyžadujú použitie premenných)
- riešiť problémy, v ktorých si treba zapamätať a neskôr použiť zapamätané hodnoty vo výrazoch
- zovšeobecniť riešenie tak, aby fungovalo nielen s konštantami.

## Nové pojmy a procesy

- premenná, meno premennej, hodnota premennej
- nastavenie hodnoty (priradenie), použitie premennej, zmena hodnoty premennej

## Odporúčanie

Pri téme premenných je dôležité krokovanie hodnoty premennej a jej znázorňovanie.

## 3. hodina: Premenné

### Pracovný list 3 - Premenné

Učiteľ pracuje na počítači s projektorom, úlohy rieši spoločne so žiakmi.

#### 1. Najprv nakreslíme štvorec

```
from turtle import *  
  
fd(100)  
rt(90)  
  
fd(100)  
rt(90)  
  
fd(100)  
rt(90)  
  
fd(100)  
rt(90)
```

Ak chceme zmeniť veľkosť strany štvorca, vidíme, že musíme dĺžku krokov v príkaze `fd(100)` opraviť štyrikrát. Dokážeme navrhnúť také riešenie, aby sme zmenili hodnotu len raz?

## 2. Použijeme premennú

---

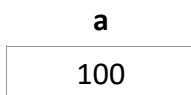
„Z hodín matematiky určite poznáte pojem premenná: je to väčšinou jednopísmenkové označenie, ktoré nahrádza nejakú číselnú hodnotu vo vzorcoch a rôznych rovniciach. Vďaka tomu môžeme niektoré vzťahy zapisovať všeobecnejšie. Podobný pojem poznajú aj programovacie jazyky, môžeme niektorú hodnotu pomenovať, a potom ďalej pracovať s týmto menom.“ (Salanci et al., 2011)

V Pythone sa pomenovanie hodnôt zapisuje:

```
>>> a = 100
```

Na pomenovanie hodnoty používame príkaz priradenia, v Pythone znak „`=`“.

Po takomto zápise si môžeme predstaviť, že v pamäti počítača sa vytvorí škatuľka (miesto) s menom `a`, v ktorej je číslo 100.



Po priradení môžeme v príkazovom režime zistiť hodnotu premennej `a`:

```
>>> a
100
```

### Použitie premennej:

Spolu so žiakmi uvažujme, ako využiť nové vedomosti o premenných a upravme program na kreslenie štvorca.

```
from turtle import *
a = 100

fd(a)
rt(90)

fd(a)
rt(90)

fd(a)
rt(90)
```



```
fd(a)
rt(90)
```

**V takto upravenom programe nám stačí zmeniť len hodnotu premennej a. Korytnačka potom nakreslí väčší alebo menší štvorec.**

Učiteľom odporúčame ozrejmiť **význam premenných** v programovacom jazyku:

Premenná je vyhradené miesto v pamäti počítača. Do premennej uložíme nejakú hodnotu, napríklad veľkosť strany štvorca. Toto číslo si premenná zapamätá. Vždy, keď budeme veľkosť strany potrebovať, pozrieme sa do premennej a zapamätané číslo prečítame z pamäte.

### 3. Druhá premenná

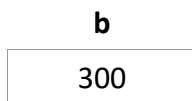
---

Učiteľ nabáda žiakov, aby mu pomohli vytvoriť premennú b s hodnotou 300

```
>>> b = 300
```

„Ako zistíme, akú hodnotu má premenná b?“

```
>>> b
300
```

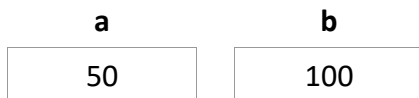


### 4. Nasleduje príklad s dvomi premennými

---

Predtým sme pracovali so štvorcom, teraz nakreslime obdĺžnik. Na veľkosti jeho strán použijeme premenné a, b:

Učiteľ nakreslí na tabuľu škatule:



Príkaz `a = 50` nastaví premennú a, príkaz `b = 100` nastaví premennú b:

Teraz vykreslime obdĺžnik pomocou premenných (v programovacom režime).

Vytvoríme súbor v IDLE:

```
from turtle import *
a = 50
b = 100
```

```
fd(a)
rt(90)
```

```
fd(b)
rt(90)
```

```
fd(a)
rt(90)
```

```
fd(b)
rt(90)
```

Učiteľ niekoľkokrát vyskúša zmenu hodnoty premenných a následné vykreslenie obdĺžnika:

Hodnota premennej a sa zmení na 120, hodnota premennej b sa zmení na 40:

a	b
120	40

## 5. Tretia premenná ako výpočet z prvých dvoch

---

Vytvorme premenné strana1, strana2:

```
>>> strana1 = 20
>>> strana2 = 120
```

strana1	strana2
20	120

Vytvorme tretiu premennú, v ktorej bude:

```
>>> sucet = strana1 + strana2
```

strana1	strana2	sucet
20	120	140

Pomocou premenných sucet vieme vyrátať obvod obdĺžnika.

```
>>> obvod = 2 * sucet
```

## 6. Zvýšenie hodnoty premennej o 10 a vykreslenie štvorca

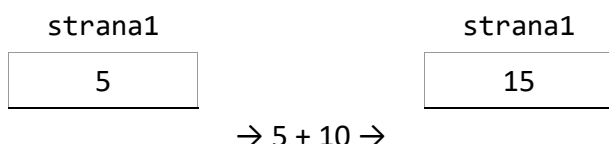
---

Máme premennú strana1, pričom chceme zvýšiť túto hodnotu o 10.

strana1
5

```
>>> strana1 = strana1 + 10
```

Počítač si zapamätá, že do premennej strana1 bude zapisovať hodnotu, prečíta hodnotu premennej strana1, zvýši túto hodnotu o 10 a následne takto získané číslo priradí premennej strana1 – zapíše do škatuľky hodnotu 30:



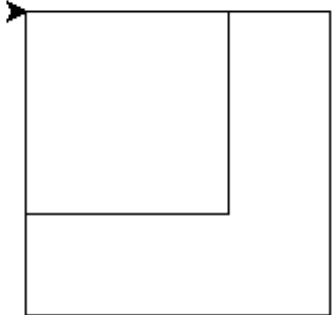
## 7. Zovšeobecnenie


Premenné v programoch majú úlohu:


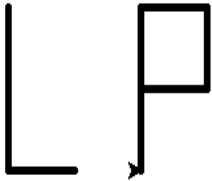
- uchovávanie údajov,
- sú nástrojom na zovšeobecnenie – umožňujú vytvárať všeobecné riešenia úloh
- pomocou nich zefektívňujeme riešenie úlohy – rovnaký výpočet realizujeme len raz a následne používame len výsledok. (Salanci et al. 2011)

### Pracovný list a riešenia úloh

Žiaci riešia na papieri alebo spoločne na tabuli – pričom znázorňujú premenné a ich hodnotu pomocou škatúľ.

1. úloha	Riešenie
<p>Nakreslite štvorec s dĺžkou strany <b>a</b> krokov, hodnotu premennej <b>a</b> nastavte na hodnotu <b>100</b>. Zvýšte hodnotu premennej <b>a</b> o <b>50</b> a opäť vykreslite štvorec s dĺžkou strany <b>a</b> krokov.</p> 	<pre>from turtle import * a = 100  fd(a) rt(90) fd(a) rt(90) fd(a) rt(90) fd(a) rt(90)  a = a + 50  fd(a) rt(90) fd(a) rt(90) fd(a) rt(90)</pre>

	<pre>fd(a) rt(90)</pre> <p>Pri tejto úlohe by si mali žiaci všimnúť, že príkazy na kreslenie štvorca môžu aj skopírovať (sú rovnaké), jediná zmena je zvýšená hodnota premennej.</p>												
2. úloha	Riešenie												
<p>Aké číslo bude v premenných, ak vykonáme nasledujúce príkazy?</p> <pre>a = 5 b = 3 b = b + 1 c = a + b d = a - 3 e = 30 - b f = 3 * b - a</pre>	<p>Po vykonaní všetkých príkazov budú v premenných hodnoty:</p> <table border="1"> <tr><td>a</td><td>5</td></tr> <tr><td>b</td><td>4</td></tr> <tr><td>c</td><td>9</td></tr> <tr><td>d</td><td>2</td></tr> <tr><td>e</td><td>26</td></tr> <tr><td>f</td><td>7</td></tr> </table>	a	5	b	4	c	9	d	2	e	26	f	7
a	5												
b	4												
c	9												
d	2												
e	26												
f	7												
3. úloha	Riešenie												
<p>Nakreslite obrázok terča použitím premenných:</p>  <p>Zmeňte veľkosť terča.</p>	<p>Kreslenie terča – premennú využijeme na nastavenie hrúbky pera, kde postupne odčítavame z hodnoty premennej, a tým zmeňujeme priemer nakresleného kruhu.</p> <pre>from turtle import * a = 80 pencolor('black') pensize(a) fd(0)  a = a - 20 pencolor('red') pensize(a) fd(0)  a = a - 20 pencolor('blue') pensize(a) fd(0)  a = a - 20 pencolor('yellow')</pre>												

	<pre>pensize(a) fd(0)</pre>
4. úloha	Riešenie
<p>Nakreslite obrázok snehuliaka použitím premenných:</p>  <p>Prichádza jar, zmenšite snehuliaka.</p>	<p>Premennú použijeme na nastavenie hrúbky pera a takisto pri počtoch prejdenných krokov medzi kruhmi:</p> <pre>from turtle import * lt(90) pencolor('blue')  a = 150  pensize(a) fd(0) pu()  fd(a/2) a = a/2 fd(a/2)  pd() pensize(a) fd(0) pu()  fd(a/2) a = a/2 fd(a/2)  pd() pensize(a) fd(0) pu()</pre>
5. úloha	Riešenie
<p>Nakreslite vaše iniciály, tak aby ste pomocou premennej vedeli meniť veľkosť písmen.</p> 	<p>Úloha č. 5 z 2. hodiny – žiaci môžu použiť riešenie z predchádzajúcej hodiny, ktorú upravia.</p>

# For cykly

## Cieľ vyučovacích hodín

Žiaci sa zoznámia s cyklami for a funkciou range(). Žiaci dokážu:

- používať príkaz cyklus for
- rozpoznávať opakujúce sa vzory, zovšeobecňovať a zapisovať riešenie pomocou cyklu
- rozpoznávať, aká časť algoritmu sa má vykonať pred, počas aj po skončení cyklu
- riešiť úlohy, v ktorých treba výsledok získať akumulovaním čiastkových výsledkov v rámci cyklu.

## Nové pojmy a poznatky

- opakovanie, konštrukcia for \_\_in\_\_:
- počet opakovaní, hranice cyklu, riadiaca premenná,
- telo cyklu, zložené telo cyklu.

---

Tému for cykly sme rozdelili do piatich vyučovacích hodín, pričom sme postupovali od for cyklov, cez for cykly s funkciou range(), for cykly s riadiacou premennou a ďalšou premennou v tele cyklu, až po vnorené cykly. Využili sme didaktický postup z učebnice Algoritmy s Logom (Varga et al., 1999). Naším cieľom je, aby žiaci dokázali rozoznať opakujúce sa vzory a dokázali zovšeobecňovať. Ako aj v téme premenné, aj tu je dôležité trasovanie programu, pomocou ktorého môžu žiaci hľadať a nachádzať určité vzťahy, môže byť tiež pomôckou pri zovšeobecnení. V jazyku Python je telo cyklu určené odsadením bloku príkazov, preto je dôležité dôkladne vysvetliť žiakom význam odsadenia a dbať na podrobné znázornenie odsadenia príkazov aj vo vnorených cykloch.

## 4. hodina: For cykly

### Pracovný list 4 – For cykly

Učiteľ píše príkazy na tabuľu, resp. ich píše **v programovacom režime** do IDLE Pythonu, spúšťa ich a premieta na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

#### 1. Kreslenie štvorca

Nakreslime korytnačkou štvorec s dĺžkou strany 100 krokov:

```
from turtle import *  
  
fd(100)  
rt(90)
```

```
fd(100)
rt(90)
```

```
fd(100)
rt(90)
```

```
fd(100)
rt(90)
```

Všimnime si opakujúce sa príkazy. Aby sme nemuseli viackrát za sebou písať rovnaké skupiny príkazov, môžeme využiť konštrukciu cyklu. Štvorec teda môžeme nakresliť jednoduchšie for cyklom:

## 2. Zavedenie konštrukcie cyklu

---

```
from turtle import *

for i in 1,2,3,4:
    forward(100)
    right(90)
```

Pre každé číslo z čísiel 1, 2, 3, 4 sa vykonajú príkazy `forward(100)` a `right(90)`.

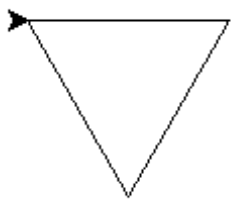
**For cyklus** funguje nasledovne:

Do premennej `i` sa postupne priradí hodnota z hodnôt, ktoré sú uvedené za `in`, a pre každú hodnotu, ktorú premenná `i` takto nadobudne sa vykonajú príkazy, ktoré sú v tele cyklu, t. j. tie príkazy, ktoré sú odsadené. V našom príklade korytnačka štyrikrát nakreslí čiaru a otočí sa doprava, pričom hodnota premennej `i` nemá na pohyb korytnačky žiadny vplyv. Dvojbodka ":" na konci riadka s príkazom `for` je povinný znak.

**Telo cyklu** tvoria príkazy, ktoré sa majú opakovať. Definujú sa odsadením príslušných riadkov, pričom odsadenie je povinné a odsadzujeme vždy o 4 medzery (alebo stlačením klávesu Tab). Telo cyklu nesmie byť prázdne, musí obsahovať aspoň jeden príkaz. (Blaho, 2016).

### Pracovný list a riešenia úloh

2. úloha (spoločne – diskusiou)	Riešenie
Nakreslite príkazom <code>for</code> rovnostranný trojuholník s dĺžkou strany 100 krokov.	Mali by prísť na to, že pri kreslení trojuholníka sa trikrát opakujú príkazy <code>forward()</code> a <code>right()/left()</code> - otočenie o uhol 120.



```
from turtle import *  
  
for i in 1,2,3:  
    fd(100)  
    rt(120)
```

### 3. úloha (žiaci samostatne)

### Riešenie

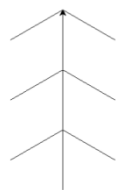
Nakreslite pravidelný 5-, 6-, 7-uholník.

```
from turtle import *  
  
for i in 1,2,3,4,5:  
    fd(100)  
    right(120)  
  
for i in 1,2,3,4,5,6:  
    fd(100)  
    rt(60)  
  
for i in 1,2,3,4,5,6,7:  
    fd(100)  
    rt(360/7)
```

Uhol otáčania nemusíme zadať číslom, stačí ak napíšeme (360/7).

### 4. úloha

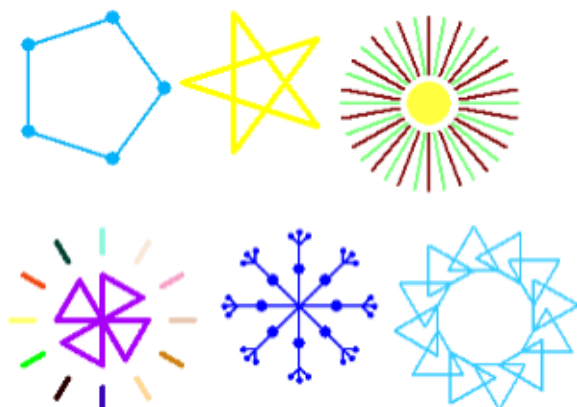
S využitím príkazu for nakreslite obrázky:





## Bonusové úlohy

S využitím príkazu for nakreslite obrázky:



(Bezáková et al, 2011)

Úlohy 1.-2. sú určené na zbieranie skúseností, po ktorom učiteľ zhrnie nové učivo a nasledujú úlohy určené na tréning poznatku.

**Zovšeobecnenie:** V programe, v ktorom sa niekoľkokrát opakuje určitá postupnosť príkazov používame konštrukciu cyklu. Príkazy, ktoré sa opakujú píšeme v tele cyklu.

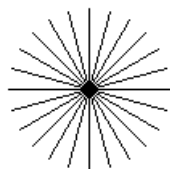
## 5. hodina: For cykly s funkciou range()

### Pracovný list 5 – For cykly s range()

Učiteľ ukazuje riešenie príkladu na počítači s projektorom, žiaci pracujú súčasne s ním na svojich počítačoch.

#### 1. Kreslenie slnka

Nakreslime slnko s 24 lúčmi:



Ako kreslíme lúče slnka?

Spoločne diskusiou: kreslíme ich tak, že korytnačku posunieme dopredu o istý počet krokov-napr. 20, potom sa korytnačka vráti na svoje pôvodné miesto – vzad o 20 krokov, a nakoniec sa otočí o  $360/24$  stupňov, lebo kreslíme 24 lúčov. Celkovo 24-krát zopakuje tieto dva príkazy.

Učiteľ zapíše:

```
from turtle import *  
  
for i in  
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24:  
    forward(50)  
    back(50)  
    right(360 / 24)
```

## 2. Zavedenie range() v konštrukcii cyklu

---

Na tomto príklade vidíme, že vypisovanie čísel je zdĺhavé. Odteraz na to môžeme použiť funkciu

```
range()
```

pomocou ktorej vieme vytvoriť postupnosť čísel, pričom hranice zadávame do zátvoriek, napríklad:

```
range(25)
```

vytvorí postupnosť čísel od 0 po 24.

Všimnime si, v `range(25)` sa začína vytvárať postupnosť čísel od čísla 0 a posledné číslo vytvorenej postupnosti je o 1 menšie, ako zadané číslo v `range(25)`.

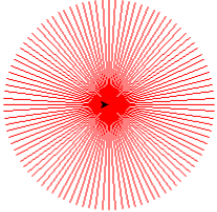



**Odporúčanie:** Učiteľ by mal dbať na to, aby žiaci pochopili hornú hranicu `range(A)`.

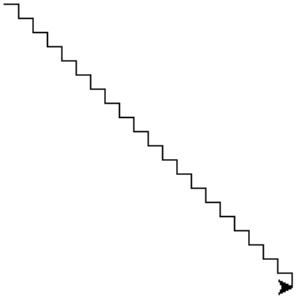
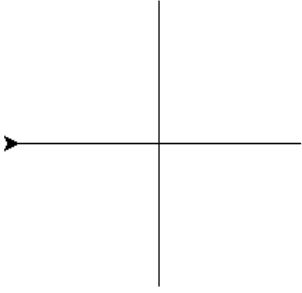
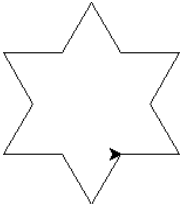
So zavedením funkcie `range(A)` sa žiaci stretnú aj s pomenovaním funkcia, ktorú zatiaľ nepoznajú, môže im preto učiteľ doplniť, že o funkciách sa viac dozvedia neskôr.

Riešenie príkladu slnka s lúčmi je teda:

```
from turtle import *  
  
for i in range(24):  
    forward(50)  
    back(50)  
    right(360 / 24)
```

## Pracovný list a riešenia úloh

2. úloha	Riešenie
<p>Nakreslite slnko s 200 lúčmi.</p> 	<pre>from turtle import * pencolor('red') for i in range(200):     fd(100)     bk(100)     rt(360/200)</pre>
3. úloha	Riešenie
<p>Nakreslite použitím range() a for nasledujúce obrázky:</p> 	<pre>from turtle import * pencolor('green') for i in range(10):     fd(20)     rt(90)     fd(40)     lt(90)     fd(20)     lt(90)     fd(40)     rt(90)</pre>
	<pre>from turtle import * pensize(5) pencolor('black') for i in range(25):     pd()     fd(0)     pu()     fd(10)</pre>
	<pre>from turtle import * pensize(5) pencolor('blue') for i in range(10):     fd(15)     rt(45)     fd(15)     lt(90)     fd(15)     rt(45)</pre>

4. úloha	Riešenie
<p>Bez toho, aby ste nasledujúce príkazy programovali, zistite, čo nakreslia. Potom príkazy napíšte a overte si svoje riešenia:</p> <p>a)</p> <pre>from turtle import *  for i in range(20):     fd(10)     rt(90)     fd(10)     lt(90)</pre> <p>b)</p> <pre>from turtle import *  for i in range(4):     fd(100)     rt(90)     fd(100)     rt(180)</pre>	<p>a)</p>  <p>b)</p> 
5. úloha	Riešenie
<p>Nakreslite 6-cípu hviezdu:</p> 	<pre>from turtle import *  for i in range(6):     fd(50)     lt(120)     fd(50)     rt(60)</pre>

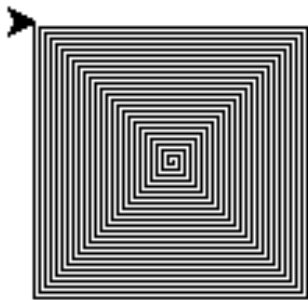
## 6. hodina: Cykly s riadiacou premennou a ďalšou premennou

### Pracovný list 6 – For cykly s premennou

Učiteľ zadá príklad a ukazuje riešenie na počítači s projektorom, žiaci pracujú súčasne s ním na svojich počítačoch.

#### 1. Učiteľ zadá žiakom úlohu na nakreslenie špirály.

---



**Diskusiou: Žiaci riešia spolu s učiteľom, buď pri tabuli alebo na projektore.**

Keďže zatiaľ žiaci poznajú premenné a for cykly, mali by vedieť túto úlohu vyriešiť pomocou zvyšovania hodnoty premennej:

```
from turtle import *  
  
a = 1  
  
for i in range(100):  
    fd(a)  
    rt(90)  
    a = a + 1
```

Učiteľ pritom môže pomôcť nakreslením špirály na tabuľu so zväčšujúcimi číslami pri stranách.

Po vyriešení príkladu pomocou premennej sa učiteľ spýta žiakov: „Ako by sme mali upraviť tento program tak, aby sme nemuseli použiť premennú a?“

#### 2. Zavedenie pojmu riadiaca premenná.

---

Učiteľ prezradí žiakom, že v zápise cyklu sme použili premennú *i*. Túto premennú nazývame riadiacou premennou cyklu a môžeme ju využiť aj pri kreslení, napríklad:

```

from turtle import *


for i in range(100):
    fd(i)
    rt(90)

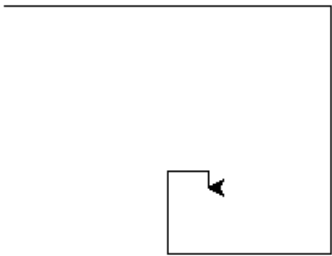

```

pre každé číslo  $i$  z čísiel 0, 1, 2 ..., 99 vykonaj príkazy: `fd(i)` a `rt(90)`.

Odporúčame tento príklad odkrokovat' pri tabuli so znázornením hodnoty riadiacej premennej v jednotlivých krokoch cyklu.

## Pracovný list a riešenia úloh

2. úloha	
<p>Postupne upravte riešenie na kreslenie špirály z úvodného príkladu tak, že zmeníte veľkosť krokov v príkaze <code>fd(i)</code> na:</p> <ul style="list-style-type: none"> <li>• <code>i * 2</code></li> <li>• <code>i ** 2</code> (takýto zápis v Pythone znamená <math>i^2</math>)</li> <li>• <code>i * 3</code></li> </ul>	
3. úloha	
<p>Postupne upravte riešenie na kreslenie špirály z úvodného príkladu tak, že zmeníte uhly otáčania na: 120, 80, 70, 60, 50.</p>	
4. úloha	Riešenie
<p>Nakreslite nasledujúce obrázky použitím premenných:</p>  <ul style="list-style-type: none"> <li>• meňte len hrúbku pera, prvý bod má veľkosť 10, každý nasledujúci je o 1 hrubší</li> <li>• korytnačka sa posúva medzi bodkami vždy o rovnaký počet krokov (25).</li> </ul>	<pre> from turtle import *  pencolor('purple')  a=10 for i in range(10):     fd(25)     pensize(i + 10)     fd(0)     pensize(1) </pre>
5. úloha	Riešenie

<p>Nakreslite na papier(alebo v Skicári), čo nakreslí korytnačka, keď vykoná nasledovnú postupnosť príkazov:</p> <pre>from turtle import *  for cislo in 200,150,100,50,25,10:     fd(cislo)     rt(90)</pre>	
<p>6. úloha</p>	<p>Riešenie</p>
<p>Nakreslite obrázok slnka, kde do premenných ukladajte dĺžku lúča a veľkosť kruhu – tela slnka. Vyskúšajte zmeniť hodnotu lúča a tela a tak nakreslite rôzne obrázky slnka.</p> 	<pre>from turtle import *  luc = 50 telo = 35 pencolor('orange') pensize(5)  for i in range(9):     fd(luc)     bk(luc)     rt(360/9) pensize(telo) pencolor('yellow') fd(0)</pre>

## 7. a 8. hodina: Vnorené cykly

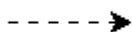
### Pracovný list 7 a 8 – Vnorené cykly

Vnorené cykly patria k náročnejším algoritmom, preto sme sem zaradili viac úloh na precvičenie. Odporúčame venovať sa tejto téme na dvoch po sebe nasledujúcich vyučovacích hodinách, pričom na druhej žiaci pracujú samostatne.

Učiteľ rieši úvodný príklad na počítači s projektorom a žiaci si všetko skúšajú spolu s ním sami na svojich počítačoch.

#### 1. Prerušovaná čiara

Nakreslime prerušovanú čiaru:



```

from turtle import *

for i in range(6):
    fd(5)
    pu()
    fd(5)
    pd()

```

Skúsme nakresliť s takouto prerušovanou čiarou štvorec:



```

from turtle import *

for i in range(6):
    fd(5)
    pu()
    fd(5)
    pd()

rt(90)

for i in range(6):
    fd(5)
    pu()
    fd(5)
    pd()

rt(90)

```

... a to isté ešte dvakrát. V tomto príklade sa opakujú príkazy for a right().

## 2. Zavedenie konštrukcie vnorených cyklov

Štvorec s prerušovanými čiarami môžeme nakresliť aj takto:

```

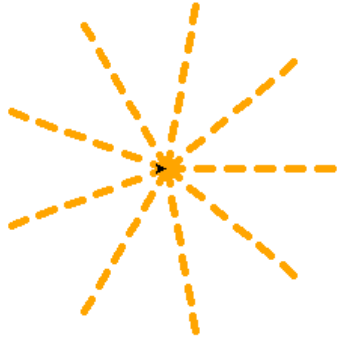
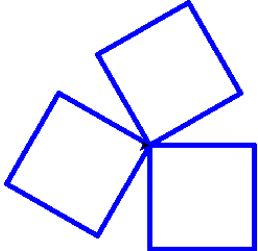
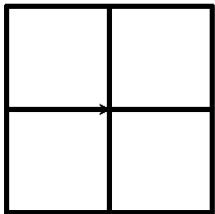
from turtle import *

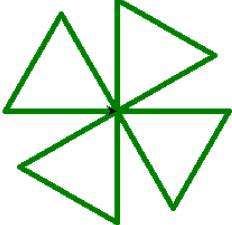
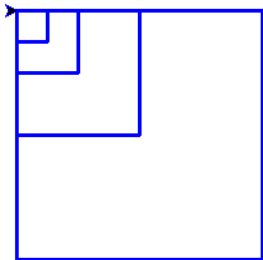
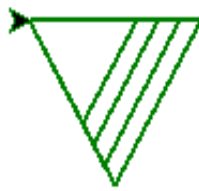
for i in range(4):
    for j in range(6):
        fd(5)
        pu()
        fd(5)
        pd()
    rt(90)

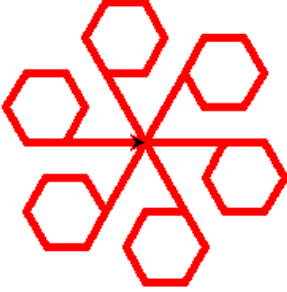
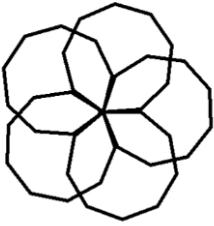
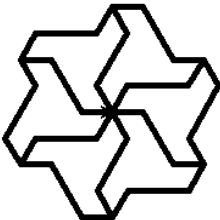
```



## Pracovný list a riešenia úloh

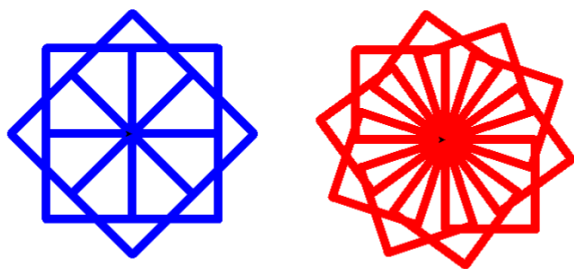
<p>3. úloha</p> <p>Nakreslite nasledujúci obrázok použitím vnorených cyklov:</p> 	<p>Riešenie</p> <pre>from turtle import *  pensize(5) pencolor('orange') for i in range(9):     for j in range(6):         forward(10)         penup()         forward(10)         pendown()     pu()     back(120)     right(360/9)     pd()</pre>
<p>4. úloha</p> <p>Nakreslite nasledujúci obrázok použitím vnorených cyklov. Korytnačka kreslí štvorce, pričom sa otáča okolo svojej osi.</p> 	<p>Riešenie:</p> <pre>from turtle import *  pensize(5) pencolor('blue')  for i in range(3):     for j in range(4):         fd(100)         rt(90)     rt(360/3)</pre>
	<pre>from turtle import *  pensize(5) pencolor('black')  for i in range(4):     for j in range(4):         fd(100)         rt(90)     rt(90)</pre>

<p>5. úloha</p> <p>Nakreslite nasledujúci obrázok použitím vnorených cyklov:</p>  <p>Korytnačka kreslí trojuholníky, pričom sa otáča okolo svojej osi.</p>	<p>Riešenie</p> <pre> from turtle import *  pensize(5) pencolor('green')  for i in range(4):     for j in range(3):         fd(100)         rt(120)     rt(90) </pre>
<p>6. úloha</p> <p>Nakreslite obrázok, v ktorom sú 4 štvorce, a platí, že strana prvého štvorca má veľkosť 25. Každý ďalší štvorec má 2-krát väčšiu stranu ako predchádzajúci štvorec.</p> 	<p>Riešenie</p> <pre> from turtle import *  pensize(3) pencolor('blue')  a = 25 for i in range(4):     for j in range(4):         fd(a)         rt(90)     a = a * 2 </pre>
<p>7. úloha</p> <p>Nakreslite obrázok, v ktorom sú 4 trojuholníky, a platí, že strana prvého trojuholníka má veľkosť 50 a každý ďalší trojuholník má o 10 krokov väčšiu stranu ako predchádzajúci trojuholník.</p> 	<p>Riešenie</p> <pre> from turtle import *  pensize(2) pencolor('green')  a = 50 for i in range(4):     for j in range(3):         fd(a)         rt(120)     a = a + 10 </pre>

8. úloha	Riešenie:
<p>Nakreslite nasledujúci obrázok použitím vnorených cyklov:</p> 	<pre> from turtle import *  pensize(5) pencolor('red')  for i in range(6):     fd(50)     for j in range(6):         fd(25)         rt(360 / 6)     bk(50)     rt(360/6) </pre>
9. úloha	Riešenie
<p>Nakreslite nasledujúci obrázok použitím vnorených cyklov:</p>  <p>Na obrázku sú 9-uholníky, pričom sa korytnačka otáčala o istý uhol okolo svojej osi.</p>	<pre> from turtle import *  pensize(3)  for i in range(5):     for j in range(9):         fd(100)         rt(360/9)     rt(360/5) </pre>
10. úloha	Riešenie:
<p>Nakreslite nasledujúci obrázok použitím vnorených cyklov:</p> 	<pre> from turtle import *  for i in range(6):     for j in range(3):         fd(25)         rt(60)         fd(50)         lt(60)         fd(25)         rt(360/3)     rt(360/6) </pre>

### 11. úloha

Nakreslite nasledujúci obrázok použitím vnorených cyklov:



Obrázky sa skladajú zo štvorcov, pričom sa korytnačka otáčala o istý uhol okolo svojej osi.

Riešenie pre modrý obrázok:

```
from turtle import *  
  
pensize(5)  
pencolor('blue')  
  
for i in range(8):  
    for j in range(4):  
        fd(100)  
        rt(90)  
    rt(360/8)
```

Riešenie pre červený obrázok:

```
from turtle import *  
  
pensize(5)  
pencolor('red')  
  
for i in range(10):  
    for j in range(4):  
        fd(100)  
        rt(90)  
    rt(36)
```

### 10. úloha

Nakreslite nasledujúci obrázok, na ktorom je 10 trojuholníkov, použitím vnorených cyklov:



Riešenie

```
from turtle import *  
  
pensize(5)  
pencolor('green')  
  
for i in range(10):  
    for j in range(3):  
        fd(100)  
        rt(120)  
    rt(36)
```

Úlohy od 1.-5. sú určené na zbieranie skúseností, po ktorých učiteľ zhrnie nové učivo a nasledujú úlohy určené na tréningovanie poznatku.

# Funkcie

## Cieľ vyučovacích hodín

Žiak vie/dokáže:

- definovať a volať funkcie
- používať funkcie s jednoduchými parametrami

## Nové pojmy a poznatky

- definícia funkcie, volanie funkcie
- meno funkcie, parametre, telo funkcie
- ako funguje odovzdávanie parametrov, použitie parametrov funkcie

## Odporúčanie

Je veľmi dôležité, aby učiteľ podrobne vysvetlil a názorne trasoval volanie funkcie s parametrami s konkrétnymi hodnotami. Popri tom odporúčame kresliť krabíčky a znázorňovať prácu s parametrami a vyhodnotenie výrazov.

---

Tému funkcie sme rozdelili na štyri vyučovacie hodiny. Na prvej zavedieme definíciu funkcie. Zvolili sme kreslenie bodky ako motiváciu vytvárania podprogramov, ktoré zjednodušia riešenie zložitejších programov. Na nasledujúcej vyučovacej hodine sme zaviedli funkcie, v tele ktorých sa nachádza cyklus. Pokladáme za dôležité venovať sa tomu preto, lebo žiaci sa stretnú s dvojitém odsadením príkazov – odsadenie príkazov v tele cyklu, ktorý je zároveň vo funkcii. Následne sme sa venovali funkcii s parametrom. Je veľmi dôležité, aby učiteľ podrobne vysvetlil a názorne trasoval volanie funkcie s parametrami s konkrétnymi hodnotami. Popri tom odporúčame kresliť krabíčky a znázorňovať prácu s parametrami a vyhodnotenie výrazov.

V jazyku Pascal sme doteraz používali dva druhy podprogramov: procedúry a funkcie. V jazyku Python sa podprogramy bez návratovej hodnoty a podprogramy s návratovou hodnotou nerozlišujú, voláme ich jednotne pojmom funkcie.

## 9. hodina: Funkcie

### Pracovný list 9 – Funkcie

Učiteľ píše príkazy na tabuľu, resp. ich píše v programovacom režime do IDLE Pythonu a premieta ich na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

#### 1. Motivácia

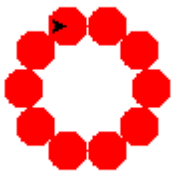
---

Učiteľ zadá nový príklad na nakreslenie obrázka:



```
from turtle import *  
  
for i in range(15):  
    pensize(20)  
    fd(0)  
    pensize(1)  
    fd(25)
```

V ďalšom príklade takisto kreslia obrázok s bodkami:



```
from turtle import *  
  
pencolor('red')  
  
for i in range(10):  
    pensize(20)  
    fd(0)  
    pensize(1)  
    fd(20)  
    rt(36)
```

## 2. Zavedenie definície funkcie

---

V týchto príkladoch vždy, keď sme chceli nakresliť bodku, museli sme napísať tieto príkazy:

```
pensize(20)  
fd(0)  
pensize(1)
```

Aby sme nemuseli znovu a znovu zapísať všetky príkazy na nakreslenie bodky, využijeme, že nám Python ponúka krajšie riešenie: vytvoríme funkciu, ktorá nám nakreslí celú bodku a v úlohách, v ktorých potrebujeme nakresliť bodku vyvoláme túto funkciu, ktorá zmení hrúbku pera a nakreslí bodku.

```

from turtle import *

def bodka():
    pensize(20)
    fd(0)
    pensize(1)

```

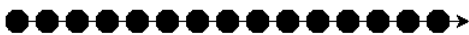

Ak program, do ktorého sme zapísali definíciu funkcie spustíme, vidíme, že korytnačka nič nekreslí. Je to preto, lebo potrebujeme korytnačke povedať, aby našu funkciu vykonala. Vyskúšajme v príkazovom režime vyvolanie funkcie:

```
>>> bodka()
```

Všimnime si, že pri definícii funkcie a aj pri volaní funkcie sú za menom funkcie zapísané prázdne zátvorky. Pripomeňme si, že aj po každom príkaze ako napríklad `pu()`, `reset()`, sme písali takéto zátvorky.

**Poznámka pre učiteľa:** Žiaci môžu zabúdať napísať tieto zátvorky vo svojich programoch. Ak ich zabudnú napísať pri definovaní funkcie, program po spustení vypíše chybu, ktorú je pomerne jednoduché identifikovať a opraviť. Ak však zabudnú napísať zátvorky pri vyvolaní funkcie, program po spustení nevypíše chybu, ale funkcia sa nevyvolá, a tak nedostanú požadovaný obrázok.

Pomocou funkcie `bodka()` vieme nakresliť predošlé obrázky:

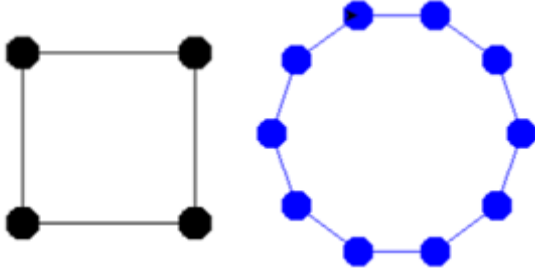
	
<pre> from turtle import *  def bodka():     pensize(20)     fd(0)     pensize(1)  for i in range(15):     bodka()     fd(25) </pre>	<pre> from turtle import *  def bodka():     pensize(20)     fd(0)     pensize(1)  pencolor('red')  for i in range(10):     bodka()     fd(20)     rt(36) </pre>

**Poznámka pre učiteľa:** Je potrebné dostatočne a aj viackrát zdôrazniť, že do funkcie patria tie príkazy, ktoré sú odsadené v bloku príkazov.

## Pracovný list a riešenia úloh

### 2. úloha

S použitím funkcie bodka() nakreslite tieto obrázky. Nemeňte funkciu bodka().

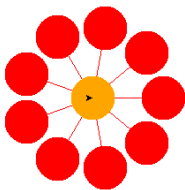


Riešenie pre 1. obrázok:

```
from turtle import *  
  
def bodka():  
    pensize(20)  
    fd(0)  
    pensize(1)  
  
for i in range(4):  
    fd(50)  
    bodka()  
    rt(90)
```

Riešenie pre 2. obrázok:

```
from turtle import *  
  
def bodka():  
    pensize(20)  
    fd(0)  
    pensize(1)  
  
pencolor('blue')  
for i in range(10):  
    fd(50)  
    bodka()  
    rt(36)
```



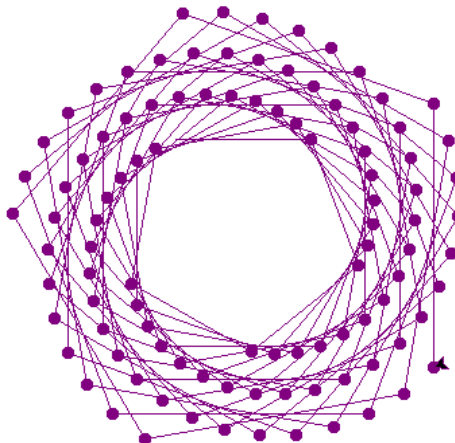
```
from turtle import *  
  
def bodka():  
    pensize(20)  
    fd(0)  
    pensize(1)  
  
pencolor('red')  
for i in range(9):  
    fd(30)  
    bodka()  
    bk(30)  
    rt(360/9)  
  
pencolor('orange')  
bodka()
```



### 3. úloha

Pamätáte sa na špirálu z minulých hodín? Do programu kreslenia špirály sme pridali funkciu `bodka()`. Pridajte volanie funkcie `bodka` do programu tak, aby sa nakreslil nasledujúci obrázok:

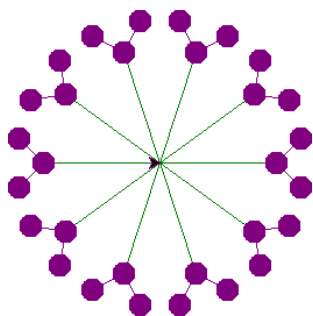
```
from turtle import *  
  
def bodka():  
    pensize(10)  
    fd(0)  
    pensize(1)  
  
pencolor('purple')  
delay(0)  
a = 100  
for i in range(100):  
    fd(a)  
    rt(70)  
    a = a + 1
```



Riešenie: volanie funkcie `bodka()` môžeme pridať do predposledného riadku kódu. Úloha je zameraná na porozumenie a opravenie už vytvoreného programu.

### 5. úloha

Nakreslite tento obrázok. S použitím funkcie `bodka()`. Zdefinujte funkciu `bodka3()`, pomocou ktorej nakreslíte trojicu fialových bodiek. Nemeňte funkciu `bodka()`.



### Riešenie

```
from turtle import *  
  
def bodka():  
    pensize(20)  
    fd(0)  
    pensize(1)  
  
def bodka3():  
    pencolor('purple')  
    bodka()  
    lt(45)  
    fd(30)  
    bodka()  
    bk(30)  
    rt(90)  
    fd(30)  
    bodka()  
    bk(30)  
    lt(45)  
  
for i in range(10):
```

```
pencolor('green')
pd()
fd(100)
bodka3()
pu()
bk(100)
rt(36)
```

## 10. hodina: Cyklus vo funkcii

### Pracovný list 10 – Cyklus vo funkcii

Učiteľ píše príkazy na tabuľu, resp. ich píše v programovacom režime do IDLE Pythonu a premieta ich na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

Na tejto hodine sa zameriame na odsadenie bloku príkazov vo funkcii. Zoznámime žiakov s funkciami, ktoré obsahujú v tele aj konštrukciu for cyklu.

#### 1. Cyklus v definícii funkcie

Učiteľ spolu so žiakmi definujú funkciu `stvorec()`.

```
from turtle import *

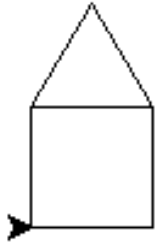
def stvorec():
    for i in range(4):
        fd(50)
        rt(90)
```

Učiteľ pritom vysvetlí žiakom význam odsadenia príkazov v tele funkcie a v tele for cyklu.

**Poznámka pre učiteľa:** Treba dostatočne zdôrazniť, že do funkcie patria tie príkazy, ktoré sú odsadené v bloku príkazov. V prípade, že použijeme v tele funkcie cyklus, telo cyklu bude znovu odsadené.

#### Pracovný list a riešenia úloh

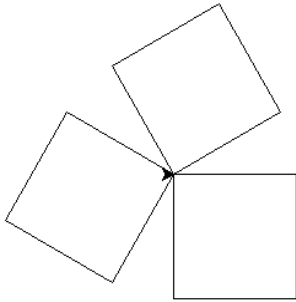
2. úloha	Riešenie
S použitím funkcie <code>stvorec()</code> nakreslite tieto obrázky. Funkciu <code>stvorec()</code> nemeňte. Korytnačka začína a končí v tom istom bode – vľavo dole.	<pre>from turtle import *  def stvorec():     for i in range(4):         fd(50)         rt(90)  stvorec() lt(60)</pre>



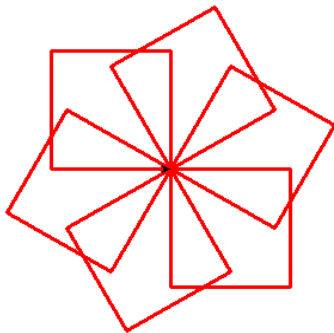
```
for i in range(3):  
    fd(50)  
    rt(120)  
lt(30)  
bk(50)  
rt(90)
```

### 3. úloha

### Riešenie

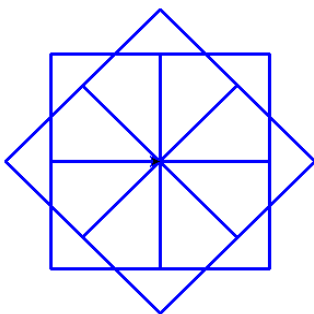


```
from turtle import *  
  
def stvorec():  
    for i in range(4):  
        fd(50)  
        rt(90)  
  
for i in range(3):  
    stvorec()  
    rt(120)
```



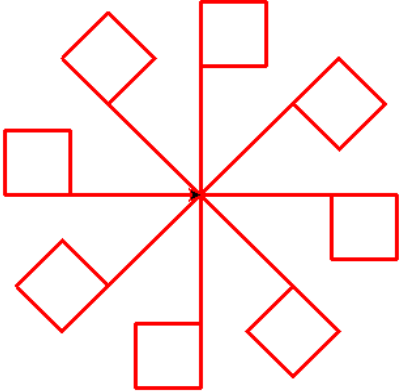
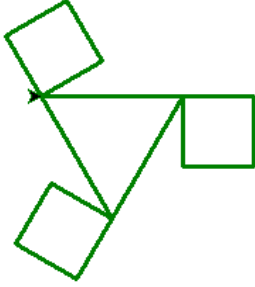
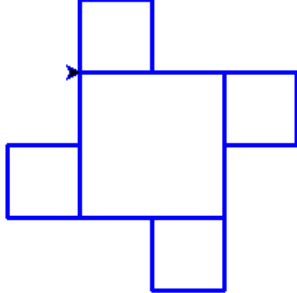
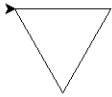
Korytnačka kreslí 6 štvorcov, pričom sa otáča okolo svojej osi

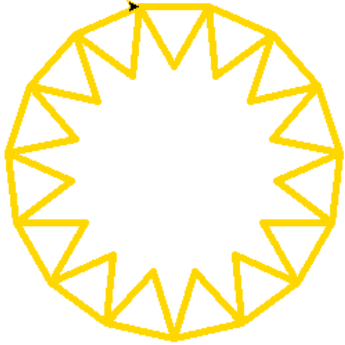
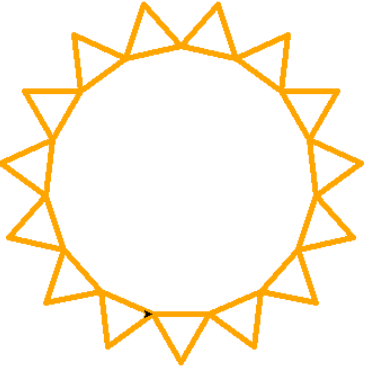
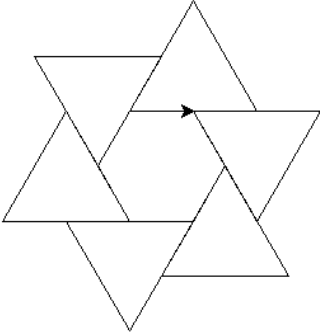
```
from turtle import *  
  
def stvorec():  
    for i in range(4):  
        fd(50)  
        rt(90)  
  
pencolor('red')  
pensize(3)  
for i in range(6):  
    stvorec()  
    rt(360/6)
```

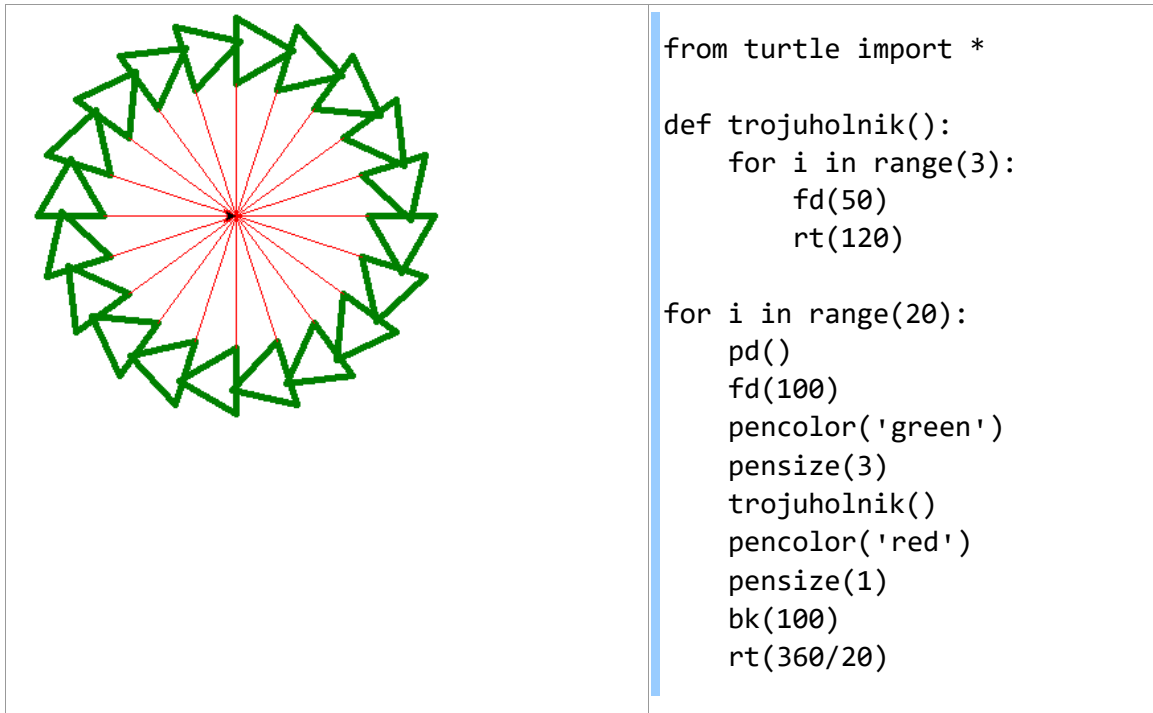


Korytnačka kreslí 8 štvorcov, pričom sa otáča okolo svojej osi.

```
from turtle import *  
  
def stvorec():  
    for i in range(4):  
        fd(50)  
        rt(90)  
  
pencolor('blue')  
pensize(3)  
for i in range(8):  
    stvorec()  
    rt(360/8)
```

4. úloha	Riešenie
<p>S použitím funkcie <code>stvorec()</code> nakreslite tieto obrázky. Funkciu <code>stvorec()</code> nemeňte.</p> 	<pre>from turtle import *  def stvorec():     for i in range(4):         fd(50)         rt(90)  pencolor('red') pensize(3) for i in range(8):     fd(100)     stvorec()     bk(100)     rt(360/8)</pre>
	
<pre>from turtle import *  def stvorec():     for i in range(4):         fd(50)         rt(90)  pencolor('green') pensize(3) for i in range(3):     fd(100)     stvorec()     rt(120)</pre>	<pre>from turtle import *  def stvorec():     for i in range(4):         fd(50)         rt(90)  pencolor('blue') pensize(3) for i in range(4):     fd(100)     stvorec()     rt(90)</pre>
5. úloha	Riešenie
<p>Definujte funkciu <code>trojuholnik()</code> s dĺžkou strany 50 krokov.</p> 	<pre>from turtle import *  def trojuholnik():     for i in range(3):         fd(50)         rt(120)</pre>

6. úloha	Riešenie
<p>S použitím funkcie trojuholnik() nakreslite tieto obrázky, funkciu trojuholnik() pritom nemeňte.</p> 	<pre> from turtle import *  def trojuholnik():     for i in range(3):         fd(50)         rt(120)  pencolor('gold') pensize(5)  for i in range(15):     trojuholnik()     fd(50)     rt(360/15) </pre>
	<pre> from turtle import *  def trojuholnik():     for i in range(3):         fd(50)         rt(120)  pencolor('orange') pensize(5)  for i in range(15):     trojuholnik()     fd(50)     lt(360/15) </pre>
	<pre> from turtle import *  def trojuholnik():     for i in range(3):         fd(50)         rt(120)  for i in range(6):     trojuholnik()     rt(60)     fd(25) </pre>



Úlohy 2. až 4. sú určené na zbieranie skúseností, po ktorom nasleduje krátke zhrnutie nového učiva a žiaci ďalej riešia úlohy na tréningovanie poznatku.

**Zovšeobecnenie:** Zložité programy, v ktorých sa viackrát opakuje istá postupnosť príkazov, môžeme zjednodušiť vytvorením funkcie, ktorá bude po zavolaní vykonávať danú postupnosť príkazov.

## 11 a 12. hodina: Funkcie s parametrami

### Pracovný list 11 a 12 – Funkcie s parametrami

Učiteľ píše príkazy na tabuľu, resp. ich píše v programovacom režime do IDLE Pythonu a premieta ich na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

Učiteľ spolu so žiakmi v rámci opakovania definuje funkciu `stvorec()`.

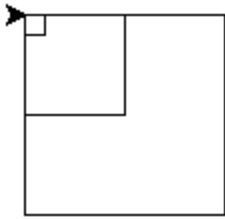
```
from turtle import *

def stvorec():
    for i in range(4):
        fd(100)
        rt(90)
```

## 1. Motivácia

---

Učiteľ zadá nový príklad na nakreslenie obrázka so štvorcami so stranami 10, 50 a 100.



Riešenie príkladu môže byť buď nakreslenie troch štvorcov bez funkcie alebo definovaním troch funkcií pre rôzne veľkosti štvorcov.

Učiteľ poukáže na rovnaké časti kódu a rozdiel pri príkaze `fd(strana)`, pričom program zjednoduší použitím funkcie `stvorec()`, ktorej zadefinuje parameter.

## 2. Odovzdávanie parametrov

---

Funkciu `stvorec()` upravíme tak, aby sme v nej použili parameter.

```
from turtle import *  
  
def stvorec(strana):  
    for i in range(4):  
        fd(strana)  
        rt(90)  
  
stvorec(10)  
stvorec(50)  
stvorec(100)
```

Čo sa deje pri volaní funkcie `stvorec(10)`?

Vznikne premenná `strana`. Parameter 10 sa priradí do premennej `strana`.

```
strana  
10
```

Vykoná sa cyklus, v ktorom sa príkaz `fd(strana)` sa vykoná s hodnotou:

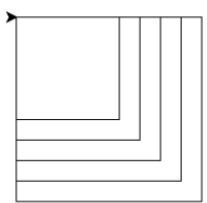
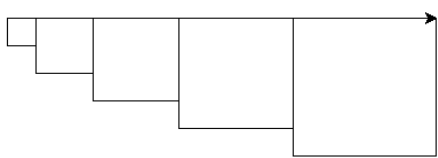
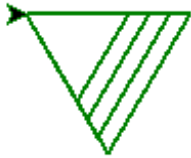

```
fd(10)
```

Nakoniec premenná `strana` zanikne.

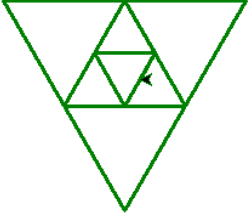
Aj pri ďalšom volaní `stvorec(50)` vznikne premenná `strana`, ale nastaví sa takto:

```
strana  
50
```

## Pracovný list a riešenia úloh

2. úloha	Riešenie
<p>S použitím funkcie <code>stvorec(strana)</code> nakreslite tieto obrázky. Funkciu <code>stvorec(strana)</code> nemeňte.</p> 	<pre>from turtle import *  def stvorec(strana):     for i in range(4):         fd(strana)         rt(90)  a = 50 for i in range(5):     stvorec(a)     a = a + 10</pre>
	<pre>from turtle import *  def stvorec(strana):     for i in range(4):         fd(strana)         rt(90)  for i in range(1,6):     stvorec(20 * i)     fd(20* i)</pre>
3. úloha	Riešenie
<p>Definujte funkciu <code>trojuholnik(s)</code>, ktorá nakreslí trojuholník so stranou veľkosti <code>s</code>. S použitím funkcie <code>trojuholnik(s)</code> nakreslite nasledujúce obrázky.</p> 	<pre>from turtle import *  def trojuholnik(s):     for i in range(3):         fd(s)         rt(120)  a = 50 for i in range(4):     trojuholnik(a)     a = a + 10</pre>
	<pre>from turtle import *  def trojuholnik(s):     for i in range(3):         fd(s)         rt(120)  for i in range(3):</pre>



	<pre>a = 50 for i in range(4):     trojuholnik(a)     a = a + 10 rt(120)</pre>
	<pre>from turtle import *  def trojuholnik(s):     for i in range(3):         fd(s)         rt(120)  strana = 200 for i in range(3):     trojuholnik(strana)     strana = strana / 2     fd(strana)     rt(60)</pre>

### 3. Funkcie s dvomi parametrami

Učiteľ zadá nový príklad.

Vytvorte funkciu, ktorá nakreslí n-uholník:

```
def Nuholnik(n, s):
    for i in range(n):
        fd(s)
        rt(360/n)
```

Učiteľ trasuje odovzdávanie parametrov na tabuli.

Čo sa deje pri volaní Nuholnik(5, 100)?

Vznikne premenná n a premenná s. Parameter 5 sa priradí do premennej n a 100 do premennej s.

n
5

s
100

Príkazy range(n) a rt(360/n) sa potom vykonajú s tými hodnotami:

```
range(5)
rt(360/5)
```

Za s sa dosadí hodnota 100. Príkaz fd(s) sa vykoná s hodnotou:

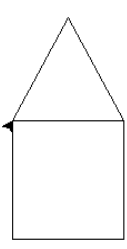
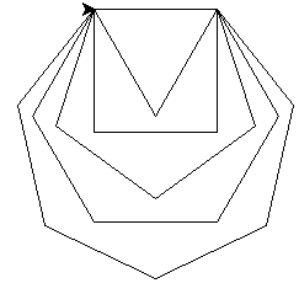
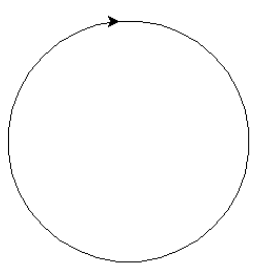
```
fd(100)
```

Nakoniec premenné n a s zaniknú.

Pri ďalšom volaní, napr. Nuholnik(7, 50) vzniknú premenné n a s, ale nastaví sa takto:

$\frac{n}{7}$	$\frac{s}{50}$
---------------	----------------

Pracovný list a riešenia úloh

4. úloha	Riešenie
<p>Definujte funkciu Nuholnik(n, s), ktorá nakreslí N-uholník so stranou veľkosti s.</p> <p>S použitím funkcie Nuholnik (n, s) nakreslite nasledujúce obrázky.</p> 	<pre>from turtle import *  def Nuholnik(n, s):     for i in range(n):         fd(s)         rt(360/n)  Nuholnik(4, 50) lt(60) Nuholnik(3, 50)</pre>
	<pre>from turtle import *  def Nuholnik(n, s):     for i in range(n):         fd(s)         rt(360/n)  for i in range(3,8):     Nuholnik(i, 50)</pre>
	<pre>from turtle import *  def Nuholnik(n, s):     for i in range(n):         fd(s)         rt(360/n)  Nuholnik(360, 1)</pre>

### Odporúčanie pre učiteľov

Odporúčame **nepoužívať** rovnaké názvy parametrov funkcií a premenných v programe. Rovnaké názvy môžu spôsobiť miskoncepce.

## Nastavovanie polohy korytnačky a náhodnosť

### Cieľ vyučovacích hodín

Žiaci sa zoznámia s generovaním náhodných čísel a príkazom na zmenu polohy korytnačky.

### Nové pojmy a poznatky

- zmena pozície korytnačky pomocou príkazu `setpos(x, y)`
- generovanie náhodných čísel, importovanie knižnice `random`

## 13. hodina: Zmena polohy korytnačky a náhodnosť

### Pracovný list 13 – Zmena polohy korytnačky a náhodnosť

Učiteľ píše príkazy na tabuľu, resp. ich píše v programovacom režime do IDLE Pythonu a premieta ich na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

#### 1. Motivácia – Hviezdna obloha

Učiteľ v spolupráci so žiakmi. Učiteľ uvedie príklad hviezd na oblohe.



Chceli by sme nakresliť podobnú hviezdnu oblohu. Ako ju môžeme kresliť?

Diskusiou sa žiaci s učiteľom dohodnú na postupe kreslenia: presunutie korytnačky na rôzne miesta na ploche a nakreslenie hviezd.

Najprv definujú funkciu, ktorá nakreslí hviezdu:

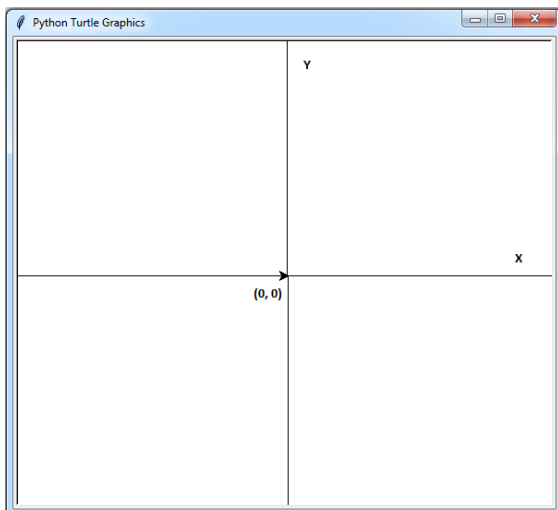
```
def hviezda():  
    for i in range(10):  
        fd(20)  
        bk(20)  
        rt(36)
```

Na to, aby sme mohli nakresliť viac hviezd (napríklad 10) na rôznych miestach, potrebujeme vedieť, ako nastaviť korytnačku na presné miesto na obrazovke. Korytnačka v Pythone pozná príkaz:

`setpos(x, y)`

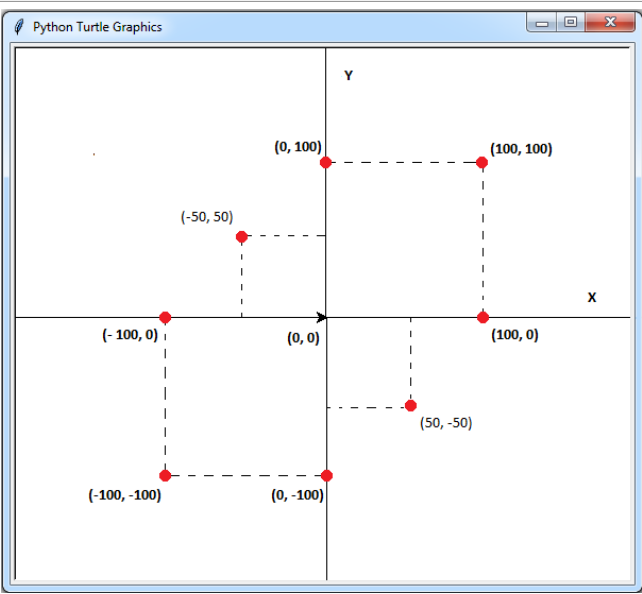
Príkaz `setpos(x, y)` presunie korytnačku na pozíciu určenú súradnicami  $x, y$ .

Učiteľ nakreslí na tabuľu plochu korytnačky a ukáže  $x$ -ové a  $y$ -ové súradnice:

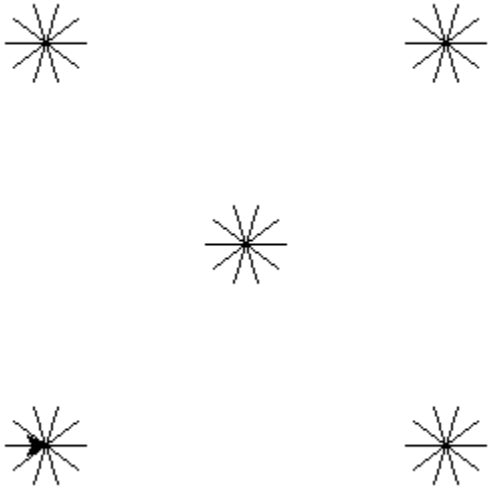


V strede plochy, kde sa korytnačka vytvorí pri spustení programu, je súradnica (0,0).

Spoločne diskusiou nakreslia na tabuľu miesta na ploche so súradnicami z 1. úlohy:

1. úloha	Riešenie
<p>Nakreslite miesta na ploche so súradnicami:</p> <ul style="list-style-type: none"><li>(0,0)</li><li>(100, 0)</li><li>(-100,0)</li><li>(0, 100)</li><li>(0, -100)</li><li>(100, 100)</li><li>(50, -50)</li><li>(-50, 50)</li><li>(-100, -100)</li></ul>	 A screenshot of the Python Turtle Graphics window showing a coordinate system with ten red dots. The dots are located at the following coordinates: (0,0), (100,0), (-100,0), (0,100), (0,-100), (100,100), (50,-50), (-50,50), (-100,-100), and (0,-100). Dashed lines connect each dot to its respective x and y coordinates on the axes.

## Pracovný list a riešenia úloh

2. úloha	Riešenie
<p>Nakreslite použitím príkazu <code>setpos()</code> a funkcie <code>hviezda()</code> nasledujúci obrázok:</p> 	<pre>from turtle import *  def hviezda():     for i in range(10):         fd(20)         bk(20)         rt(36)  hviezda() pu() setpos(100, 100) pd() hviezda() pu() setpos(100, -100) pd() hviezda() pu() setpos(-100, 100) pd() hviezda() pu() setpos(-100, -100) pd() hviezda()</pre>

## 2. Zavedenie generovania náhodných čísel

Už vieme presúvať korytnačku na ploche, ale ešte stále sa naša kresba nepodobá na hviezdnu oblohu. Na oblohe sú hviezdy na náhodných miestach. V Pythone vieme vygenerovať aj náhodné čísla.

Najprv musíme Pythonu oznámiť, že budeme používať knižnicu náhodných čísel:

```
from random import randrange as nahodne
```

Tento zápis znamená, že z knižnice `random` importujeme príkaz pre vytvorenie náhodných hodnôt `randrange` tak, aby sa funkcia, ktorá generuje náhodné čísla, volala `nahodne()`.

Pri zovaní funkcie:

```
>>> nahodne(5)
```

Python vyberie jedno náhodné číslo z čísel: 0, 1, 2, 3, 4.

Ak túto funkciu zavoláme znovu, dostaneme nové náhodne vybrané číslo.

Učiteľ nabáda žiakov, aby si v príkazovom režime vygenerovali niekoľkokrát náhodné číslo:

```
>>> nahodne(5)
4
>>> nahodne(5)
2
>>> nahodne(5)
0
```

„Všimnite si, že sa generujú čísla od 0.“

Do príkazu `nahodne()`, môžeme zadať aj dva parametre:

```
>>> nahodne(50,100)
```

V tomto prípade sa vyberie jedno náhodné číslo z čísiel od 50 do 99

### Pracovný list a riešenie úloh

Žiaci pracujú samostatne pri počítačoch.

4. úloha	Riešenie
Pomocou príkazu <code>nahodne()</code> a <code>hviezda()</code> nakreslite hviezdu náhodnej veľkosti. Zmeňte definíciu funkcie <code>hviezda()</code> tak, aby sa dĺžka lúčov hviezdy zadávala ako parameter funkcie.	<pre>from turtle import * from random import randrange as nahodne  def hviezda(d):     for i in range(10):         fd(d)         bk(d)         rt(36)  luc = nahodne(10, 50) hviezda(luc)</pre>

### 3. Príklad Hviezdna obloha

Učiteľ na počítači s projektorom doplní do programu na kreslenie hviezd generovanie náhodných pozícií.

Generovanie náhodných čísiel môžeme využiť v našom rozpracovanom príklade s hviezdou oblohou.

V predchádzajúcom príklade sme pridali generovanie náhodnej dĺžky lúčov pre hviezdu. Vytvorili sme premennú, do ktorej sme uložili náhodne vygenerovanú hodnotu. Túto

premennú môžeme využiť pri zadávaní parametra dĺžky lúča pre funkciu hviezda(). Premenné použijeme aj pri generovaní náhodných pozícií pre hviezdy.

```
from turtle import *
from random import randrange as nahodne

def hviezda(d):
    for i in range(10):
        fd(d)
        bk(d)
        rt(36)

for i in range(10):
    x = nahodne(-300, 300)
    y = nahodne(-300, 300)
    pu()
    setpos(x, y)
    pd()
    luc = nahodne(10, 50)
    hviezda(luc)
```

Takto upravený program nakreslí desať hviezd náhodnej veľkosti na náhodných pozíciách. Hviezdna obloha je však tmavomodrá a hviezdy žlté, upravme teda pozadie použitím príkazu

```
bgcolor('navy')
```

Príkaz bgcolor('navy') zafarbí pozadie plochy na zadanú farbu, v našom prípade tmavomodrú.

Do nášho programu sme pridali aj príkaz

```
delay(0)
```

Tento príkaz urýchli kreslenie.

Zmeňme počet generovaných hviezd na 50 a zmenšime veľkosti hviezd na 5 až 25:

```
from turtle import *
from random import randrange as nahodne
delay(0)

def hviezda(r):
    for i in range(10):
        fd(luc)
        bk(luc)
        rt(36)

bgcolor('navy')
pencolor('yellow')
```

```

for i in range(50):
    x = nahodne(-300, 300)
    y = nahodne(-300, 300)
    pu()
    setpos(x, y)
    pd()
    luc = nahodne(5, 25)
    hviezda(luc)

```

Výsledok je hviezdna obloha.

Organizácia tejto hodiny je náročnejšia kvôli tomu, že úlohy na seba nadväzujú. Učiteľ by mal dbať na to, aby žiaci nezaostávali. Ak je skupina žiakov nekonzistentná, t. j. ich rýchlosť riešenia úloh je rôzna, odporúčame, aby sa aj úlohy pre žiakov, ktoré sú určené na samostatnú prácu vyriešili spoločne na tabuli, prípadne na počítači s projektorom.

Niektorým žiakom môžu súradnice robiť ťažkosti. Odporúčame učiteľovi, aby týchto žiakov (aj na ďalších hodinách) vyzýval k tomu, aby si opätovne kreslili obrázok súradnicovej sústavy s x-ovou a y-ovou osou na papier alebo v Skicári, a pomocou neho si vyrátali potrebné súradnice.

## 14. hodina: Zmena polohy korytnačky a náhodné čísla

### Pracovný list 14 – Zmena polohy korytnačky a náhodné čísla

Počas tejto hodiny žiaci pracujú samostatne pri počítačoch podľa pracovného listu. V týchto úlohách si precvičia všetky doteraz prebrané pojmy a konštrukcie. Keďže žiaci pracujú samostatne, učiteľ má možnosť lepšie sledovať, ako žiaci pracujú a diagnostikovať prípadné miskoncepce.

### Pracovný list a riešenia úloh

#### 1. úloha

Napíšte program, ktorý nakreslí 20 farebných bodiek na náhodných miestach a náhodnej veľkosti. Použite pritom funkciu bodka(veľkosť, farba) s parametrami veľkosť a farba.

Ak si chceme vybrať náhodnú farbu z niekoľkých farieb, môžeme ich vybrať pomocou funkcie choice z modulu random.

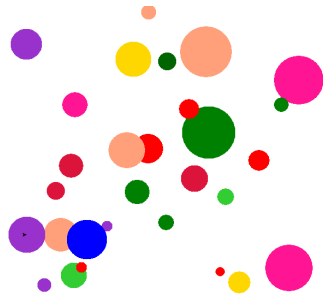
```

from random import choice as vybernahodne

farba = vybernahodne(('darkorchid', 'gold', 'red', 'purple'))

```





## Riešenie

```

from turtle import *
from random import randrange as nahodne
from random import choice as vybernahodne
delay(0)

def bodka(velkost):
    pensize(velkost)
    fd(0)

for i in range(30):
    x = nahodne(-280, 250)
    y = nahodne(-250, 290)
    pu()
    setpos(x, y)
    pd()

    farba =
vybernahodne(('limegreen', 'green', 'darkorchid', 'gold', 'red',
'purple', 'deeppink', 'lightsalmon', 'blue', 'crimson'))
    pencolor(farba)
    velkost = nahodne(15,100)
    bodka(velkost)

```

## Príklady farieb

white		forestgreen	
red		darkgreen	
green		darkcyan	
blue		midnightblue	
yellow		deepskyblue	
orange		aquamarine	
brown		cyan	
black		deeppink	
pink		chocolate	
fuschia		sandybrown	
lime		silver	
navy		lightsalmon	
purple		gold	
indigo		crimson	

## 2. úloha

Napíšte program, ktorý nakreslí lúku s farebnými kvetmi. Použite predošlý program, napíšte funkciu `kvet(velkost)` s parametrom veľkosť kvetu, ktorý určí veľkosť lupeňov kvetu, pričom bodky kreslite pomocou funkcie `bodka(velkost)` – veľkosť nakresleného bodu.



## Riešenie

```
from turtle import *
from random import randrange as nahodne
from random import choice as vybernahodne
delay(0)

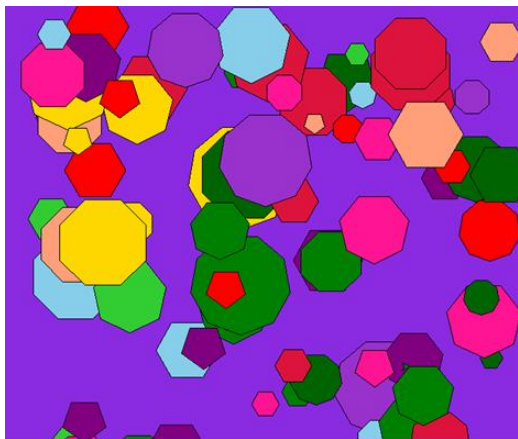
def bodka(velkost):
    pd()
    pensize(velkost)
    fd(0)
    pu()

def kvet(velkost):
    for i in range(9):
        fd(velkost-10)
        bodka(velkost)
        bk(velkost-10)
        rt(360/9)
    pencolor('yellow')
    bodka(velkost)

bgcolor('green')
for i in range(30):
    x = nahodne(-300, 300)
    y = nahodne(-300, 300)
    pu()
    setpos(x, y)
    pd()
    farba = vybernahodne(('orange', 'darkorchid', 'gold', 'red',
'purple', 'deeppink', 'lightsalmon', 'blue', 'crimson'))
    pencolor(farba)
    velkost = nahodne(15, 50)
    kvet(velkost)
```

### 3. úloha

Napište program, ktorý nakreslí na náhodných miestach 50 mnohoúhľníkov náhodnej farby a veľkosti. Využite pritom funkcie `Nuholnik(n, veľkost, farba)` s parametrami `n` – počet vrcholov, `farba` – farba výplne a `veľkost` – veľkosť strany pravidelného mnohoúhľníka.



Pri tejto úlohe je potrebné použiť nové príkazy na vyplnenie mnohoúhľníka – polygónu:

```
fillcolor(farba)
```

Príkaz `fillcolor(farba)` nastaví farbu výplne.

```
begin_fill()  
nakreslený útvar  
end_fill()
```

Príkaz `begin_fill()` zapíšeme na začiatok kreslenia útvaru, ktorý chceme vyplniť. Príkaz `end_fill()` dáme za posledný príkaz útvaru. Týmto označíme koniec jeho kreslenia a následne sa nakreslený útvar vyplní nastavenou farbou.

### Riešenie

```
from turtle import *  
from random import randrange as nahodne  
from random import choice as vybernahodne  
  
def Nuholnik(n, veľkost, farba):  
    fillcolor(farba)  
    begin_fill()  
    for i in range(n):  
        fd(veľkost)  
        rt(360/n)  
    end_fill()
```

```

bgcolor('blueviolet')
for i in range(80):
    x = nahodne(-300, 300)
    y = nahodne(-300, 300)
    pu()
    setpos(x, y)
    pd()
    n= nahodne(5,10)
    velkost = nahodne(15,50)
    farba =
vybernahodne(('limegreen', 'darkgreen', 'green', 'darkorchid', 'gold', 'red', 'purple', 'deeppink', 'lightsalmon', 'skyblue', 'crimson'))
    Nuholnik(n, velkost, farba)

```

Bonusová úloha na precvičenie pozície korytnačky

Riešenie

Nakreslite na papier (alebo v Skicári), čo nakreslí korytnačka, keď vykoná nasledovnú postupnosť príkazov:

a)

```

from turtle import *
setpos(100, 0)
setpos(-50, -50)
setpos(200, -100)

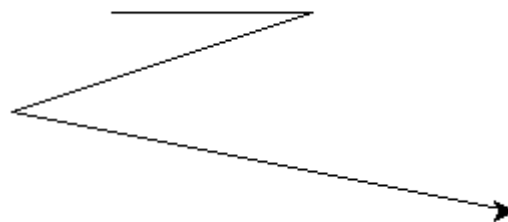
```

b)

```

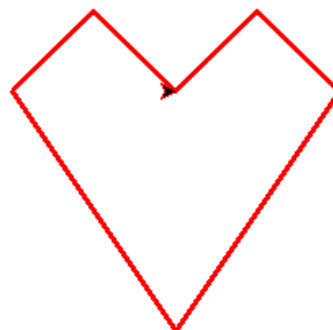
from turtle import *
pencolor('red')
pensize(3)
setpos(50, 50)
setpos(100, 0)
setpos(0, -150)
setpos(-100,0)
setpos(-50, 50)
setpos(0, 0)

```



a)

b)



# Vetvenie

## Cieľ vyučovacích hodín

Žiak vie/dokáže:

- porozumieť tomu, ako funguje vetvenie programu
- rozpoznávať situácie a podmienky, kedy treba využiť vetvenie
- rozpoznávať, aká časť algoritmu sa má vykonať pred, v rámci a po skončení vetvenia
- riešiť úlohy, v ktorých sa kombinujú cykly a vetvenia

## Nové pojmy a poznatky

- konštrukcia if ... : ... else: ...
- vetvenie, podmienka
- jednoduché podmienky v podmienenom príkaze
- zostavovanie a upravovanie vetvenia

## 15. a 16. hodina: Vetvenie

### Pracovný list 15 a 16 – Vetvenie

Učiteľ píše príkazy na tabuľu, resp. ich píše v programovacom režime do IDLE Pythonu a premieta ich na projektore. Žiaci sledujú výklad a pomáhajú mu s riešením.

### 1. Motivácia – príklad Bláznivá predpoveď počasia

Učiteľ uvedie príklad Bláznivej predpovede počasia: Nakreslíme predpoveď počasia na týždeň, v ktorej generujeme teplotu pre každý deň náhodne. Každú teplotu znázorníme stĺpcom zodpovedajúcim výške teploty.



Najprv vygenerujeme náhodnú teplotu pre jeden deň, ktorú nakreslíme tak, že korytnačku nastavíme aby smerovala hore a išla dopredu o toľko krokov, koľko bude stupňov v ten deň. Potom korytnačku vrátime na jej pôvodné miesto. Ak je teplota **vyššia ako 15 stupňov**, nakreslíme stĺpec **červenou** farbou, ak je teplota **menšia alebo rovná ako 15 stupňov**, nakreslíme ho **modrou** farbou.

**Príklad riešme postupne:** najprv nakreslime len na jeden deň:



Vyskúšajme najprv nakresliť stĺpec a stupne čiernou farbou

```
from turtle import *
from random import randrange as nahodne

lt(90)
teplota = nahodne(0, 45)

fd(teplota)
write(teplota)
bk(teplota)
```

Všimnite si nový príkaz

```
write(hodnota)
```

Po jeho zadaní korytnačka vypíše zadanú hodnotu na plochu na mieste, kde sa aktuálne nachádza, s aktuálne nastavenou farbou pera.

Žiaci doteraz nepoznali príkaz `write()`. Treba im ho vysvetliť na rôznych príkladoch a upozorniť ich, že ak chcú vypísať slová alebo znaky, musia ich písať s apostrofmi alebo v úvodzovkách, napríklad `write('Python')` vypíše slovo Python rovnako ako príkaz `write("Python")`. Zapíšeme príkazy pričom medzi nimi posunieme korytnačku, aby sme slová nepísali na seba.

```
>>> write('Python')
>>> fd(20)
>>> write("Python")
>>> fd(20)
>>> write(13)
```

## 2. Zavedenie konštrukcie podmieneného príkazu `if`

Zatiaľ máme nakreslené stĺpce pre denné teploty čiernou farbou. My ich však chceme nakresliť **červenou** alebo **modrou farbou**. Potrebujeme program, ktorý sa pre rôzne generované čísla správa dvoma rôznymi spôsobmi:

- ak sa vygeneruje číslo väčšie ako 15, farba pera sa nastaví na červenú,
- inak, pre všetky ostatné vygenerované čísla, sa farba pera zmení na modrú.



alebo

Takéto rozhodovanie zabezpečuje nová konštrukcia *if ... else ...*, ktorá zodpovedá slovenskému *ak ... inak ...* a nazýva sa **podmienený príkaz**.

```
if teplota>0:  
    pencolor('red')  
else:  
    pencolor('blue')
```

V podmienenom príkaze *if ... else ...* budeme rozlišovať dve vetvy:

1. vetvu, ktorá sa vykoná v prípade, že je *podmienka* splnená ... túto časť budeme nazývať *vetvou za if*.
2. vetvu, ktorá sa vykoná inak, teda, ak *podmienka* nie je splnená ... túto časť budeme nazývame vetvou *else*.

Odporúčame učiteľom zdôrazniť žiakom odsadenie príkazov a takisto ich upozorniť na používanie dvojbodky v konštrukcii podmienky.

Program na kreslenie bude vyzeráť takto:

```
from turtle import *  
from random import randrange as nahodne  
  
lt(90)  
teplota=nahodne(0, 45)  
  
if teplota>15:  
    pencolor('red')  
else:  
    pencolor('blue')  
fd(teplota)  
write(teplota)  
bk(teplota)
```

Teraz už máme program, ktorý vygeneruje náhodne číslo (teplotu) na jeden deň. My by sme však chceli nakresliť teploty pre celý týždeň. Potrebujeme k tomu použiť konštrukciu cyklu, pričom po nakreslení dennej teploty by sme mali posunúť korytnačku o 30 krokov doprava.



```

from turtle import *
from random import randrange as nahodne

def posun():
    pu()
    rt(90)
    fd(30)
    lt(90)
    pd()

lt(90)
for i in range(7):
    teplota=nahodne(0, 45)

    if teplota>15:
        pencolor('red')
    else:
        pencolor('blue')

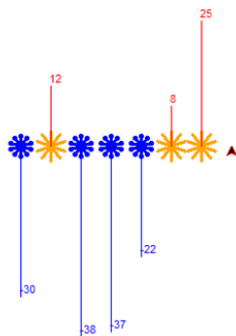
    fd(teplota)
    write(teplota)
    bk(teplota)
    posun()

```

Možné pokračovanie príkladu

**Bonusová úloha: Mimoriadne bláznivá predpoveď počasia**


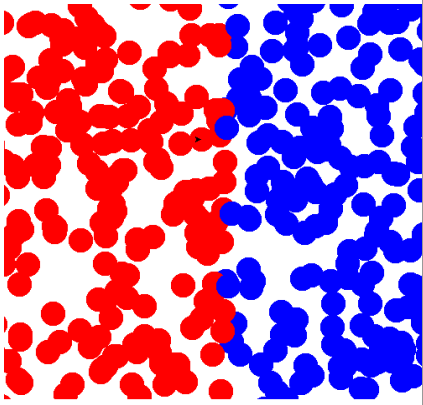
Upravte program s bláznivou predpoveďou počasia tak, aby sa nám generovali aj záporné hodnoty. Ak je teplota **vyššia ako 0 stupňov**, kreslíme stĺpec na znázornenie teploty **červenou** farbou a k stĺpci nakreslíme slnko, ak je teplota **menšia alebo rovná ako 0 stupňov**, kreslíme **modrou** a nakreslíme aj snehovú vločku.



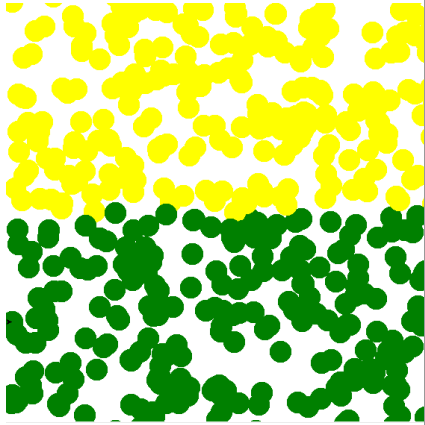


## Pracovný list a riešenia úloh

4. úloha	Riešenie
<p>Napíšte program, ktorý bude simulovať hádzanie mincou. Ak padne hlava, korytnačka vykreslí kruh. Ak padne písmo, korytnačka vykreslí písmeno A.</p> <p>Pri tejto úlohe použite príkaz <code>write('A')</code>, ktorý prikáže korytnačke, aby napísala písmeno A.</p>	<pre>from turtle import * from random import randrange as nahodne minca=nahodne(0,2) if minca==1:     pencolor('red')     pensize(15)     fd(0)     pensize(1) else:     pencolor('blue')     write('A')</pre>
5. úloha	Riešenie
<p>Upravte program tak, aby sa hodilo mincou 10-krát, do premenných hlava a pismo ukladajte počet, koľkokrát padla hlava a koľkokrát písmo. Na konci vykreslite korytnačkou stĺpce pre počty padnutí hlavy a písma, aby sme videli, čo padlo viackrát.</p> <pre> 4 6   ▲ </pre>	<pre>from turtle import * from random import randrange as nahodne  lt(90) pismo=0 hlava=0 for i in range(10):     minca=nahodne(0,2)     if minca==1:         pismo=pismo+1     else:         hlava=hlava+1  fd(pismo) write(pismo) bk(pismo)  pu() rt(90) fd(15) lt(90) pd()  fd(hlava) write(hlava) bk(hlava)</pre>

<p>6. úloha</p> <p>Upravte program pre hádzanie mincou tak, aby sa hodilo 100 krát.</p> 	<p>Riešenie</p> <p>V predchádzajúcom riešení stačí upraviť číslo 10 na 100 vo for cykle:</p> <pre>... for i in range(100): ...</pre>
<p>7. úloha</p> <p>Napíšte program, ktorý na náhodné pozície nakreslí 500 farebných bodiek, pričom tie z nich, ktoré sú v ľavej polovici plochy, budú červené a zvyšné v pravej polovici (teda else vetva) budú modré. (Aby ste nemuseli dlho čakať, môžete príkazom delay(0) zrýchliť korytnačku).</p> 	<p>Riešenie</p> <pre>from turtle import * from random import randrange as nahodne from random import choice as vybernahodne  def bodka():     pensize(30)     fd(0)  def posun(x,y):     pu()     setpos(x, y)     pd()  delay(0)  for i in range(500):     x = nahodne(-300,300)     y = nahodne(-300,300)     posun(x,y)     if x &lt; 0:         pencolor('red')     else:         pencolor('blue')     bodka()</pre>
<p>8. úloha</p> <p>Napíšte program, ktorý na náhodné pozície nakreslí 500 farebných bodiek, pričom tie z nich, ktoré sú v hornej polovici plochy, budú žlté a zvyšné v dolnej polovici (teda else vetva) budú zelené. (Aby ste nemuseli dlho čakať,</p>	<p>Riešenie</p> <pre>from turtle import * from random import randrange as nahodne from random import choice as vybernahodne  def bodka():     pensize(30)     fd(0)  def posun(x,y):</pre>

môžete príkazom `delay(0)` zrýchliť korytnačku.



```
pu()
setpos(x, y)
pd()
```

```
delay(0)

for i in range(500):
    x = nahodne(-300,300)
    y = nahodne(-300,300)
    posun(x,y)
    if y > 0:
        pencolor('yellow')
    else:
        pencolor('green')
    bodka()
```

### 9. úloha

Vytvorte program na kreslenie hviezd a kvetov náhodných veľkostí tak, aby korytnačka kreslila vo vrchnej polovici okna hviezdy a v dolnej časti kvety (hviezdy a kvety kreslite pomocou funkcií `hviezda()` a `kvet()` z minulých hodín).



### Riešenie

```
from turtle import *
from random import randrange as nahodne
from random import choice as vybernahodne
```

```
def bodka(velkost, farba):
    pensize(velkost)
    pencolor(farba)
    fd(0)
```

```
def kvetinky(velkost, farba):
    for i in range(9):
        pu()
        fd(velkost*0.75)
        pd()
        bodka(velkost, farba)
        pu()
        bk(velkost*0.75)
        pd()
        rt(360/9)
    bodka(velkost, 'yellow')
```

```
def hviezda(r):
    pencolor('yellow')
    pensize(3)
    for i in range(9):
        fd(r)
        bk(r)
        rt(360/9)
```

```
def posun(x,y):
    pu()
```

```

setpos(x, y)
pd()

delay(0)
for i in range(35):
    x= nahodne(-300,300)
    y= nahodne(-300,300)
    posun(x,y)
    v=nahodne(15,50)

f=vybernahodne(('limegreen','darkorchid',
'red', 'purple','deeppink',
'lightsalmon', 'blue', 'crimson',
'orange'))

    if y<0:
        kvetinky(v, f)
    else:
        hviezda(v-10)

```

Pri riešení úloh z cvičenia sa žiaci stretnú s podmienkou rovná sa, ktorú značíme == (dvomi symbolmi rovná sa). Učiteľ im poradí, ako zapísať podmienky:

Na číslach funguje porovnávanie pomocou znakov menší, väčší, menší alebo rovný, väčší alebo rovný, **rozdielne od matematických zápisov sú zápisy rovnosti a nerovnosti:**

podmienka	podmienka slovom	pre cislo = 50	pre cislo = 120
cislo < 90	je hodnota premennej cislo menšia ako 90	<i>True</i>	<i>False</i>
cislo <= 50	je menšie alebo rovné	<i>True</i>	<i>False</i>
cislo == 50	rovná sa	<i>True</i>	<i>False</i>
cislo != 77	nerovná sa	<i>True</i>	<i>True</i>
cislo > 100	je väčšie ako	<i>False</i>	<i>True</i>
cislo >= 90	je väčšie alebo rovné	<i>Fasle</i>	<i>True</i>
40 < cislo <= 50	je väčšie ako ... a zároveň menšie alebo rovné ...	<i>True</i>	<i>False</i>

# Projekt

## Cieľ projektu

Hlavným cieľom projektu je zafixovať si učivo. Žiak si zopakuje a precvičí si preberané učivo z predchádzajúcich vyučovacích hodín. Projekt môže slúžiť aj učiteľovi na hodnotenie výkonu žiaka.

## Použité pojmy a poznatky

- premenná a priradenie
- konštrukcia for cyklu
- funkcie s parametrami
- generovanie náhodných hodnôt
- konštrukcia if ... : ... else: ...

## 17. hodina: Zadanie projektu

### Pracovný list 17 – Zadanie projektu

Vytvorte program, ktorý vykreslí pohľadnicu alebo reklamu.

V programe musíte použiť:

- **knižnice** turtle a random
- základné príkazy na **pohyb** korytnačky
- príkazy na **zmenu polohy korytnačky**
- **zmena farby** a **veľkosti** pera korytnačky
- **premenné**
- **cykly**
- aspoň dve **funkcie**
- **príkaz vetvenia** – if
- **náhodne** generované čísla a náhodne vyberané farby

Výsledná kresba môže vyzerať napríklad nasledovne:



# Hodnotenie

Každý učiteľ má vlastný štýl hodnotenia, takisto školy majú rôzne harmonogramy a systémy hodnotení, preto sme priamo k popisu hodín nezahrnuli hodnotenie žiakov. V tejto časti uvádzame naše odporúčania k hodnoteniam.

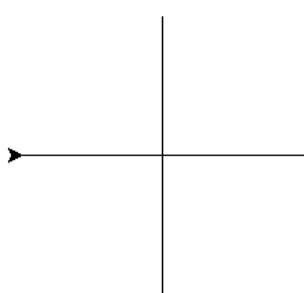
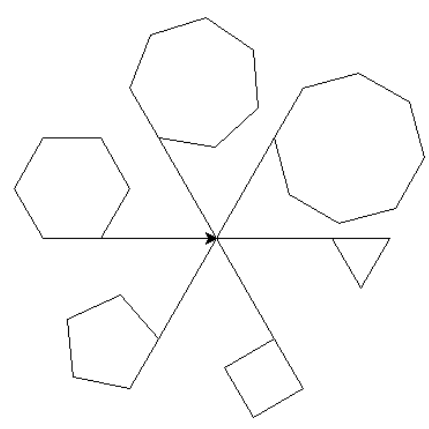
## Formatívne hodnotenie

Odporúčame žiakov priebežne slovne motivovať – pochváliť ich alebo ich opravovať, aby si nadobudli správne programátorské zvyky a aby sme udržali ich motiváciu k ďalšej práci.

## Sumatívne hodnotenie

Učiteľom odporúčame zaradiť priebežné hodnotenie po témach for cykly, funkcie a podmienený príkaz if vo forme krátkych písomiek.

### Príklady priebežných písomiek

Cykly	Riešenie
<p>Nakreslite, čo nakreslí korytnačka po spustení programu:</p> <pre>from turtle import *  for i in range(4):     fd(100)     rt(90)     fd(100)     rt(180)</pre>	
<p>Nakreslite, čo nakreslí korytnačka po spustení programu:</p> <pre>from turtle import *  def utvar(cislo):     for i in range(cislo):         fd(50)         rt(360/cislo)  for i in range(6):     fd(100)     utvar(i+3)     bk(100)     rt(360/6)</pre>	

Ako poslednú časť metodiky sme zaradili **projekt**, ktorý slúži jednak pre fixáciu poznatku, ale aj pre sumatívne hodnotenie žiakov.

## Zoznam použitých príkazov

Príkazy	Vysvetlenie
<code>from turtle import *</code>	Použi príkazy z knižnice <code>turtle</code>
<code>from random import randrange as nahodne</code>	Použi príkaz <code>randrange</code> z knižnice <code>random</code> ako <code>nahodne</code>
<code>from random import choice as vyber</code>	Použi príkaz <code>choice</code> z knižnice <code>random</code> ako <code>vyber</code>
<code>showturtle()</code> <code>st()</code>	Ukáž korytnačku
<code>hideturtle()</code> <code>ht()</code>	Skry korytnačku
<code>forward(100)</code> <code>fd(100)</code>	Dopredu o 100 bodov
<code>back(100)</code> <code>bk(100)</code>	Dozadu o 100 bodov
<code>reset()</code>	Zmaž plochu a posuň korytnačku do stredu plochy
<code>right(90)</code> <code>rt(90)</code>	Vpravo o 90 stupňov
<code>left(90)</code> <code>lt(90)</code>	Vľavo o 90 stupňov
<code>penup()</code> <code>pu()</code>	Pero hore
<code>pendown()</code> <code>pd()</code>	Pero dole
<code>pencolor(farba)</code>	Nastav farbu pera
farby napríklad: <code>'red'</code> , <code>'black'</code> , <code>'green'</code> , <code>'blue'</code> , <code>'brown'</code> , <code>'pink'</code> , <code>'white'</code>	
<code>pensize(hrúbka)</code>	Nastav hrúbku pera
<code>setpos(x,y)</code>	Nastav pero na súradnice <code>x</code> , <code>y</code>
<code>bgcolor(farba)</code>	Vyfarbí pozadie farbou zadanou v parametri
<code>write(text)</code>	Vypíše text
<code>delay(0)</code>	Zrýchli pohyb korytnačky

### Konštrukcia cyklu

```
for n in range(cislo):
    príkazy
```

### Definícia funkcie

```
def MenoFunkcie(parameter1, parameter2):
    príkazy
```

## Náhodné číslo

```
from random import randrange as nahodne

n = nahodne(5)           # do n priradí náhodné číslo od 0 do 4
x = nahodne(-300,301)    # do x priradí náhodné číslo od -300 do
300
```

## Náhodná farba

```
from random import choice as vybernahodne

farba = vybernahodne(('darkorchid','gold','red', 'purple','deeppink',
'lightsalmon', 'skyblue', 'crimson'))
```

## Vyfarbovanie ohraničenej oblasti

```
fillcolor(farba)
begin_fill()
...kreslenie ohraničenej oblasti...
end_fill()
```

## Podmienенý príkaz if – príkaz vetvenia

```
if podmienka:           # ak podmienka platí, vykonaj 1. skupinu príkazov
    prikazy
else:                   # ak podmienka neplatí, vykonaj 2. skupinu príkazov
    prikazy
```



## Použitá literatúra

**Bezáková, Daniela, Lovászová, Gabriela a Kučera, Peter. 2009.** *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika - Programovanie 1.* Bratislava : Štátny pedagogický ústav, 2009. ISBN 978-80-89225-65-1.

**Blaho, Andrej. 2012.** *Informatika pre stredné školy: Programovanie v Delphi a Lazaruse.* Bratislava : Slovenské pedagogické nakladateľstvo - Mladé letá, s.r.o., 2012. ISBN: 978-80-10-02308-0.

**Blaho, Andrej. 2016.** *Programovanie v Pythone 1 (prednášky k predmetu Programovanie (1)).* [Online] 2016. [Dátum: 30. 9. 2016.] Dostupné na: [www.input.sk](http://www.input.sk).

**Mészárosová, Eva.** *Vývoj vzdelávacieho obsahu pre vyučovanie programovania v jazyku Python.* Rigorózná práca. Bratislava: Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, 2017.

**Salanci, Ľubomír, Tomcsányiová, Monika a Blaho, Andrej. 2011a.** *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika - Didaktika programovania pre stredné školy 1.* Bratislava : Štátny pedagogický ústav, 2011. ISBN 978-80-8118-079-8.

**Salanci, Ľubomír, Tomcsányiová, Monika a Blaho, Andrej. 2011b.** *Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika - Didaktika programovania pre SŠ 2.* Bratislava : Štátny pedagogický ústav, 2011. ISBN 978-80-8118-090-3.

**Salanci, Ľubomír. 2016.** *Programovanie v C++.* Bratislava, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. [Online] [Dátum: 29. 11. 2016.] Dostupné na: <https://www.edi.fmph.uniba.sk/~salanci/C/index.html>.

**ŠPÚ - Štátny pedagogický ústav. 2015.** *Inovovaný Štátny vzdelávací program - príloha Informatika - gymnázium so štvorročným a päťročným vzdelávacím programom.* Bratislava, 2015. Dostupné na: [http://www.statpedu.sk/sites/default/files/dokumenty/inovovany-statny-vzdelavaci-program/informatika\\_g\\_4\\_5\\_r.pdf](http://www.statpedu.sk/sites/default/files/dokumenty/inovovany-statny-vzdelavaci-program/informatika_g_4_5_r.pdf).

**Varga, Mário, et al. 1999.** *Informatika pre gymnáziá: Algoritmy s Logom.* Bratislava : Media Trade, spol. s r. o., 1999. ISBN 80-08-02965-X.



## Príloha: Pracovné listy



## Pracovný list 1 – Úvod do Pythonu a korytnačej grafiky

- 1 Zistite, koľko krokov korytnačka prejde a akú dlhú čiaru nakreslí, ak vykoná túto postupnosť príkazov:

```
forward(100)
forward(-90)
forward(80)
forward(-70)
forward(60)
forward(-50)
```

- 2 Zistite vzťah medzi príkazmi:

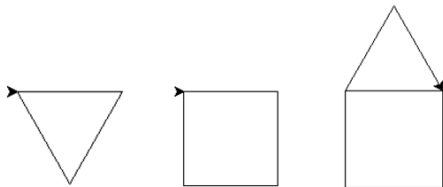
```
right(90)
left(-90)
```

- 3 Nakreslite na papier (alebo v Skicári), čo nakreslí korytnačka, keď vykoná nasledovnú postupnosť príkazov:

```
forward(80)
back(40)
right(90)
forward(100)
```

Svoje riešenie si overte zadaním príkazov v príkazovom režime.

- 4 Napíšte príkazy v príkazovom režime, ktorými korytnačka nakreslí tieto obrázky:

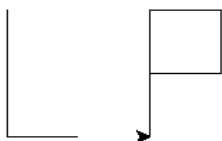


Všimnite si, že domček sa skladá zo štvorca a z trojuholníka. Strany oboch útvarov majú rovnakú dĺžku, napr. 100.

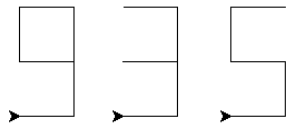
- 5 Nakreslite písmená veľkej abecedy: L, E, H, K, M

Jednotlivé programy si ukladajte do samostatných súborov s názvom `pismoL.py`, `pismoE.py`, atď.

- 6 Nakreslite vaše iniciály. Iniciály sú prvé písmená vášho mena, napríklad iniciály pre Lenku Peckú:



7 Nakreslite digitálne číslice

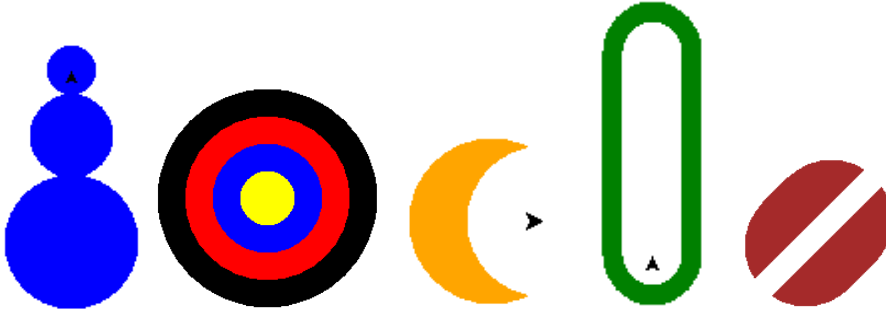


Číslicu začnite kresliť vždy v jej ľavom dolnom rohu. Po dokreslení presuňte korytnačku znovu do ľavého dolného rohu číslice. Nevadí, ak korytnačka pôjde viackrát po tej istej čiare.

---

## Pracovný list 2 - Úvod do korytnačej grafiky s Pythonom

8 Pomocou hrubých rôzne zafarbených čiar nakreslite tieto obrázky:



**Pomôcka:**

Kruh sa dá nakresliť ako veľmi hrubá veľmi krátka čiara:

```
pensize(50)
forward(0)
```

Kružnica sa dá nakresliť pomocou dvoch rôzne veľkých kruhov so spoločným stredom, pričom menší kruh má inú farbu:

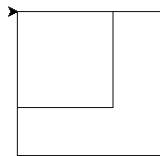
```
pencolor('black')
pensize(50)
forward(0)
pencolor('white')
pensize(45)
forward(0)
```

9 Neónový nápis vytvoríte nakreslením útvaru najprv hrubým perom tmavšou farbou a na následne tenším perom a svetlejšou farbou:



## Pracovní list 3 – Premenné

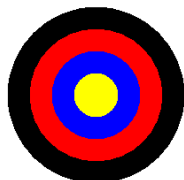
- 1 Nakreslite štvorec s dĺžkou strany **a** krokov, hodnotu premennej **a** nastavte na hodnotu **100**. Zvýšte hodnotu premennej **a** o **50** a opäť vykreslite štvorec s dĺžkou strany **a** krokov.



- 2 Aké čísla budú v premenných, ak vykonáme nasledujúce príkazy?

	a	b	c	d	e	f
<b>a = 5</b>						
<b>b = 3</b>						
<b>b = b + 1</b>						
<b>c = a + b</b>						
<b>d = a - 3</b>						
<b>e = 30 - b</b>						
<b>f = 3 * b - a</b>						

- 3 Nakreslite obrázok terča použitím premenných:



Zmeňte veľkosť terča.

- 4 Nakreslite obrázok snehuliaka použitím premenných:



Prichádza jar, zmenšite snehuliaka.

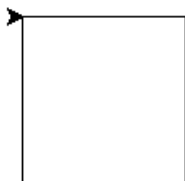
- 5 Nakreslite vaše iniciály, tak aby ste pomocou premennej vedeli meniť veľkosť písmen.



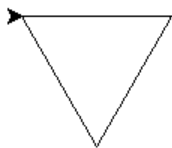


## Pracovní list 4 - For cykly

1 Nakreslite štvorec.

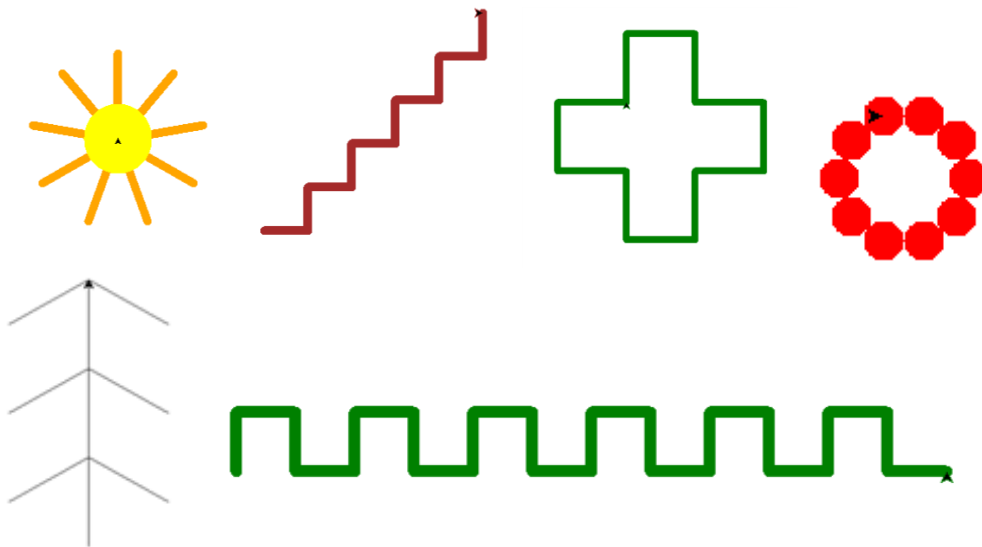


2 Nakreslite príkazom **for** rovnostranný trojuholník s dĺžkou strany 100 krokov.



3 Nakreslite pravidelný 5-, 6-, 7-uholník.

4 S využitím príkazu **for** nakreslite obrázky:

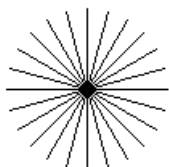


5 Bonusové úlohy - S využitím príkazu **for** nakreslite obrázky:

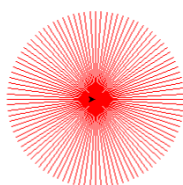


## Pracovný list 5 – For cykly s range()

- 1 Nakreslime slnko s 24 lúčmi.



- 2 Nakreslite slnko s 200 lúčmi.



- 3 Nakreslite použitím `range()` a `for` nasledujúce obrázky:



- 4 Bez toho aby ste nasledujúce príkazy programovali, pokúste sa zistiť, čo nakreslia. Potom príkazy napíšte a overte tak svoje riešenia:

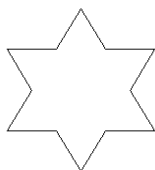
a)  
`from turtle import *`

```
for i in range(20):  
    fd(10)  
    rt(90)  
    fd(10)  
    lt(90)
```

b)  
`from turtle import *`

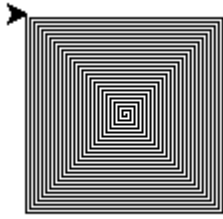
```
for i in range(4):  
    fd(100)  
    rt(90)  
    fd(100)  
    rt(180)
```

- 5 Nakreslite 6-cípu hviezdu:



## Pracovní list 6 – For cykly s premennou

1 Nakreslite špirálu:



2 Postupne upravte riešenie na kreslenie špirály tak, že zmeníte veľkosť krokov v príkaze `fd(i)` na:

- `i*2`
- `i**2` (takýto zápis v Pythone znamená  $i^2$ )
- `i*3`

3 Postupne upravte riešenie na kreslenie špirály tak, že zmeníte uhol otáčania na: 120, 80, 70, 60, 50

4 Nakreslite nasledujúce obrázky použitím premenných:

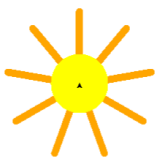


- meňte te len hrúbku pera, prvý bod má veľkosť 10, každý nasledujúci je o 1 hrubší
- korytnačka sa posúva medzi bodkami vždy o rovnaký počet krokov (25).

5 Nakreslite na papier (alebo v Skicári), čo nakreslí korytnačka, keď vykoná nasledovnú postupnosť príkazov:

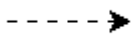
```
from turtle import *  
  
for cislo in 200,150,100,50,25,10:  
    fd(cislo)  
    rt(90)
```

6 Nakreslite obrázok slnka, kde do premenných ukladajte dĺžku lúča a veľkosť kruhu – tela slnka. Vyskúšajte zmeniť hodnotu lúča a tela a tak nakreslite rôzne obrázky slnka.



## Pracovní list 7 a 8 – Vnorené cykly

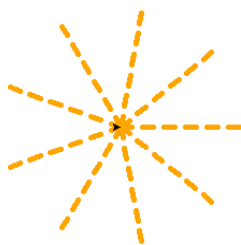
- 1 Nakreslite prerušovanú čiaru



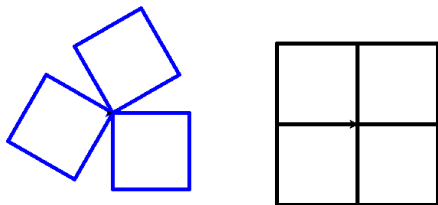
- 2 Prerušovanou čiarou nakreslite štvorec



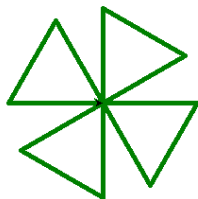
- 3 Nakreslite nasledujúce obrázky použitím vnorených cyklov:



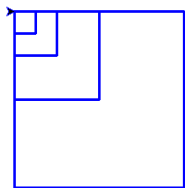
- 4 Nakreslite nasledujúce obrázky použitím vnorených cyklov. Korytnačka kreslí štvorce, pričom sa otáča okolo svojej osi.



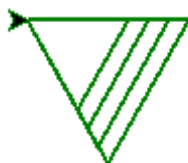
- 5 Nakreslite nasledujúce obrázky použitím vnorených cyklov. Korytnačka kreslí trojuholníky, pričom sa otáča okolo svojej osi.



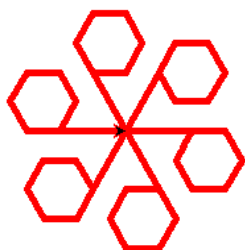
- 6 Nakreslite obrázok, v ktorom sú 4 štvorce, a platí, že strana prvého štvorca má veľkosť 25 a každý ďalší štvorec má 2-krát väčšiu stranu ako predchádzajúci štvorec.



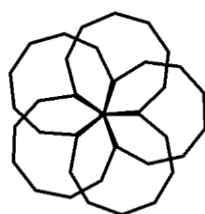
- 7 Nakreslite obrázok, v ktorom sú 4 trojuholníky, a platí, že strana prvého trojuholníka má veľkosť 50 a každý ďalší trojuholník má o 10 krokov väčšiu stranu ako predchádzajúci trojuholník.



- 8 Nakreslite nasledujúce obrázky použitím vnorených cyklov.

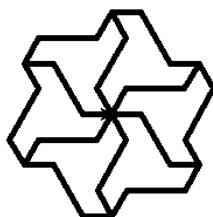


- 9 Nakreslite nasledujúci obrázok použitím vnorených cyklov:

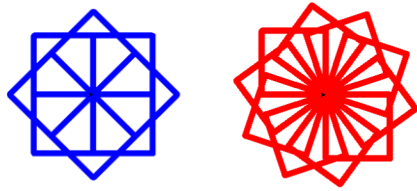


Na obrázku sú 9-uholníky, pričom sa korytnačka otáčala o istý uhol okolo svojej osi.

- 10 Nakreslite nasledujúci obrázok použitím vnorených cyklov:



11 Nakreslite nasledujúci obrázok použitím vnorených cyklov:



Obrázky sa skladajú zo štvorcov (8 a 10), pričom sa korytnačka otáčala o istý uhol okolo svojej osi.

---

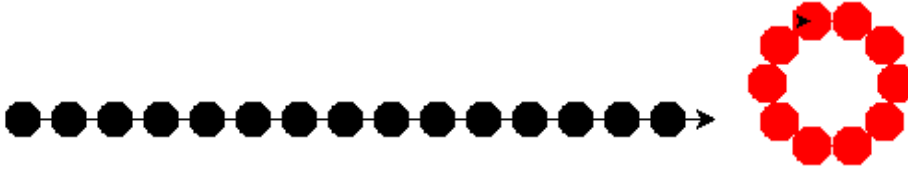
12 Nakreslite nasledujúci obrázok, na ktorom je 10 trojuholníkov, použitím vnorených cyklov:



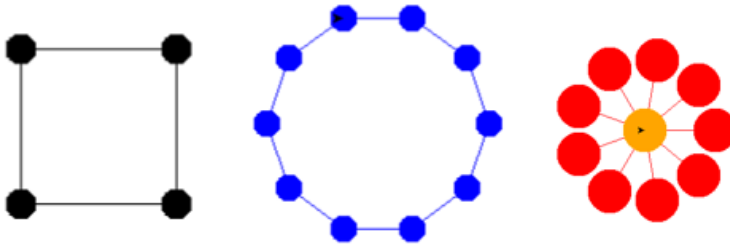
---

## Pracovní list 9 - Funkcie

1 Nakreslite obrázky:

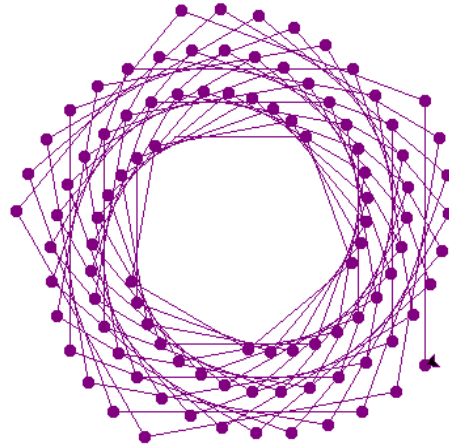


2 S použitím funkcie bodka() nakreslite tieto obrázky. Nemeňte funkciu bodka().

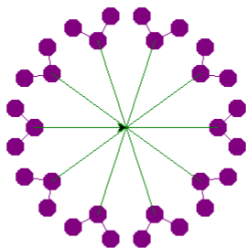


3 Pamätáte sa na špirálu z minulých hodín? Do programu kreslenia špirály sme pridali funkciu bodka(). Pridajte volanie funkcie bodka do programu tak, aby sa nakreslil nasledujúci obrázok

```
from turtle import *  
  
def bodka():  
    pensize(10)  
    fd(0)  
    pensize(1)  
  
pencolor('purple')  
delay(0)  
a = 100  
for i in range(100):  
    fd(a)  
    rt(70)  
    a = a + 1
```

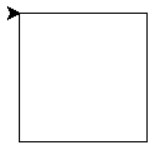


4 Nakreslite tento obrázok. S použitím funkcie bodka() zdefinujte funkciu bodka3(), pomocou ktorej nakreslíte trojicu fialových bodiek. Nemeňte funkciu bodka().

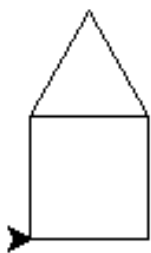


## Pracovní list 10 – Cyklus vo funkcii

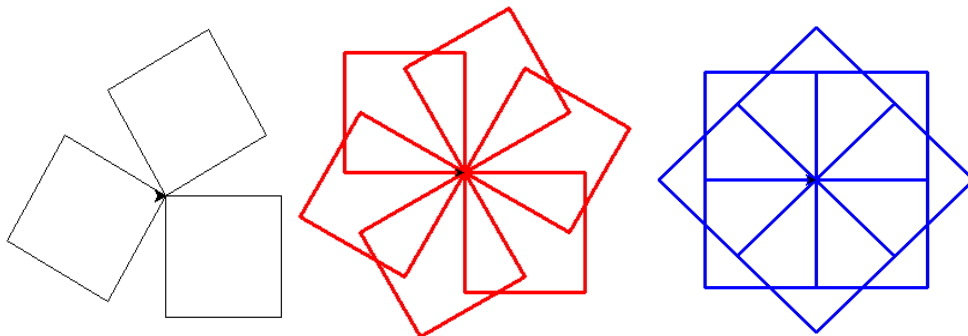
- 1 Definujte funkciu `stvorec()`.



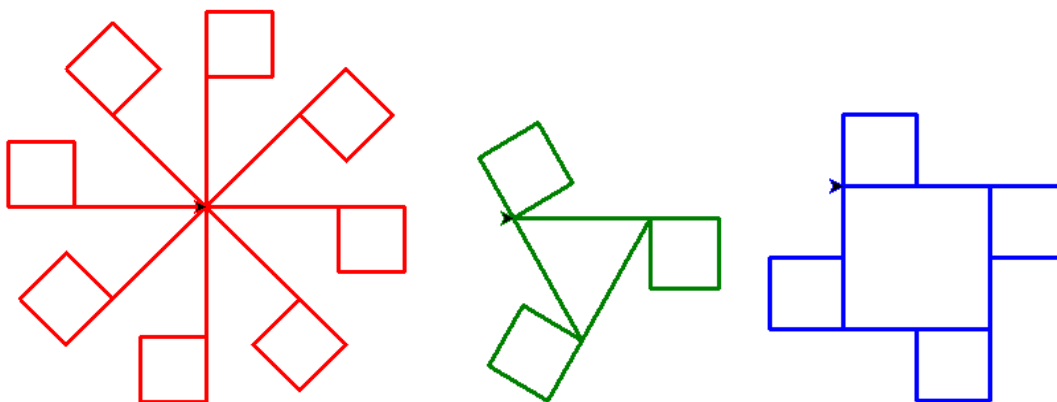
- 2 S použitím funkcie `stvorec()` nakreslite dom. Funkciu `stvorec()` nemeňte. Korytnačka začína a končí v tom istom bode – vľavo dole.



- 3 S použitím funkcie `stvorec()` nakreslite tieto obrázky. Funkciu `stvorec()` nemeňte. Korytnačka začína a končí v tom istom bode, kreslí 3, 6 a 8 štvorcov, pričom sa otáča okolo svojej osi.

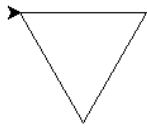


- 4 S použitím funkcie `stvorec()` nakreslite tieto obrázky. Funkciu `stvorec()` nemeňte.



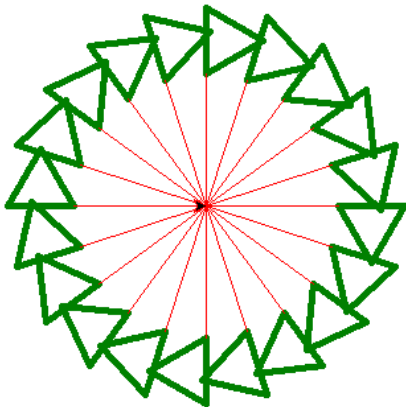
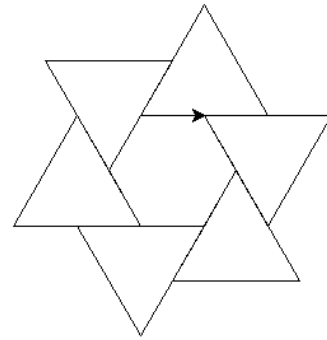
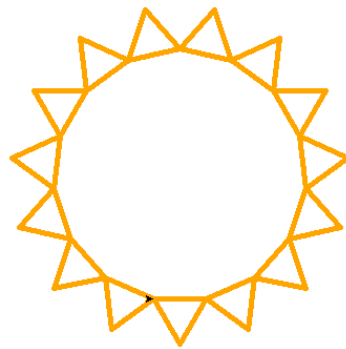
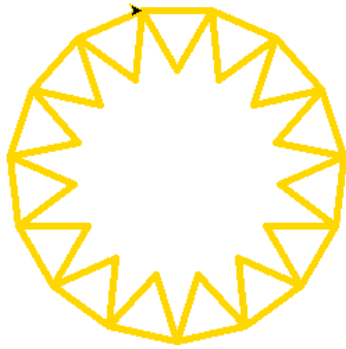


5 Definujte funkciu trojuholnik().



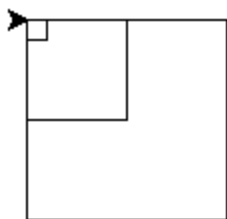
---

6 S použitím funkcie trojuholnik() nakreslite tieto obrázky, funkciu trojuholnik() pritom nemeňte.

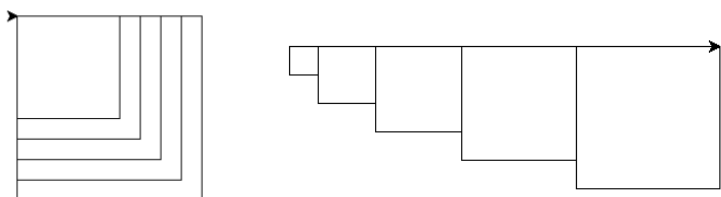


## Pracovní list 11 a 12 – Funkcie s parametrami

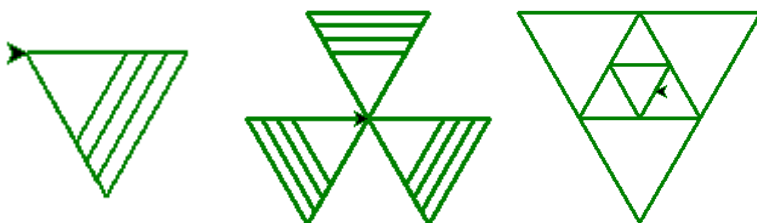
- 1 Nakreslite obrázok so štvorcom so stranami 10, 50 a 100.



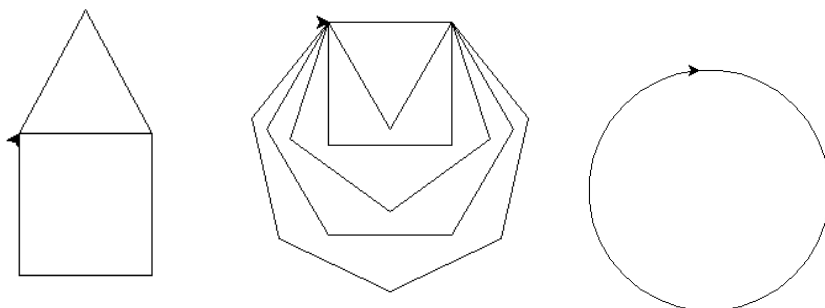
- 2 S použitím funkcie `stvorec(strana)` nakreslite tieto obrázky. Funkciu `stvorec(strana)` nemeňte.



- 3 Definujte funkciu `trojuholnik(s)`, ktorá nakreslí trojuholník so stranou veľkosti  $s$ . S použitím funkcie `trojuholnik(s)` nakreslite nasledujúce obrázky.



- 4 S použitím funkcie `Nuholnik(n, s)`, ktorá nakreslí  $N$ -uholník so stranou veľkosti  $s$ . S použitím funkcie `Nuholnik(n, s)` nakreslite nasledujúce obrázky.



## Pracovní list 13- Zmena polohy korytnačky a náhodnosť

Spoločný príklad: Hviezdna obloha



1 Nakreslite miesta na ploche so súradnicami:

(0,0)

(100, 0)

(-100,0)

(0, 100)

(0, -100)

(100, 100)

(50, -50)

(-50, 50)

(-100, -100)

---

2 Nakreslite použitím príkazu `setpos()` a funkcie `hviezda()` nasledujúci obrázok:



---

3 Pomocou príkazu `nahodne()` a `hviezda()` nakreslite hviezdu náhodnej veľkosti. Zmeňte definíciu funkcie `hviezda()` tak, aby sa dĺžka lúčov hviezdy zadávala ako parameter funkcie.

---

4 . Pomocou funkcie `setpos(x,y)` premiestnite korytnačku na **náhodnú pozíciu**, kde nakreslite hviezdu. Zmeňte farbu pozadia pomocou príkazu `bgcolor('navy')` a hviezdy kreslite žltou farbou.

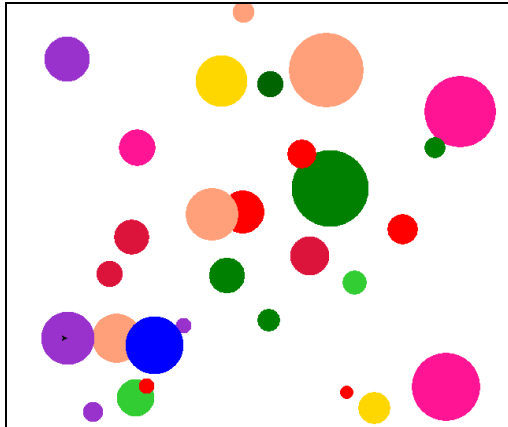
---

5 Upravte program na kreslenie hviezdnej oblohy tak, aby mali hviezdy náhodnú veľkosť a hrúbku lúčov.

---

## Pracovný list 14- Zmena polohy korytnačky a náhodné čísla

- 1 Napíšte program, ktorý nakreslí 20 farebných bodiek na náhodných miestach a náhodnej veľkosti. Použite pritom funkciu `bodka(veľkosť, farba)` s parametrami `veľkosť` a `farba`.



Ak si chceme vybrať náhodnú farbu z niekoľkých farieb, môžeme ich vybrať pomocou funkcie `choice` z modulu `random`:

```
from random import choice as vybernahodne
```

```
farba = vybernahodne(('darkorchid', 'gold', 'red',  
'purple'))
```

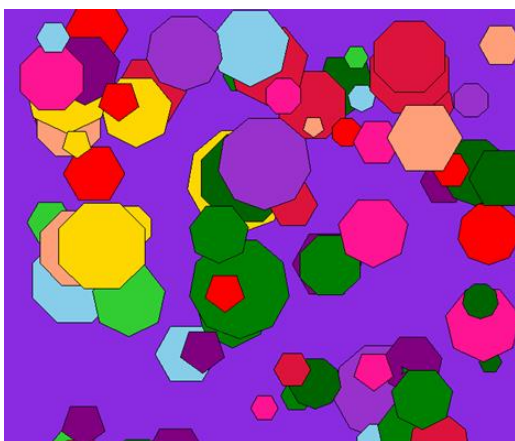
Príklady farieb:

white		forestgreen	
red		darkgreen	
green		darkcyan	
blue		midnightblue	
yellow		deepskyblue	
orange		aquamarine	
brown		cyan	
black		deeppink	
pink		chocolate	
fuschia		sandybrown	
lime		silver	
navy		lightsalmon	
purple		gold	
indigo		crimson	

- 2 Napíšte program, ktorý nakreslí lúku s farebnými kvetmi. Použite predošlý program, napíšte funkciu `kvet(velkost)` s parametrom veľkosť kvetu, ktorý určí veľkosť lupeňov kvetu, pričom bodky kreslite pomocou funkcie `bodka(velkost)` – veľkosť nakresleného bodu.



- 3 Napíšte program, ktorý nakreslí na náhodných miestach 50 mnohoúhelníkov náhodnej farby a veľkosti. Využite pritom funkcie `Nuholnik (n, veľkost, farba)` s parametrami `n` – počet vrcholov, `farba` – farba výplne a `veľkost` – veľkosť strany pravidelného mnohoúhelníka.



Pri tejto úlohe je potrebné použiť nové príkazy na vyplnenie mnohoúhelníka – polygónu:

```
fillcolor(farba)
```

Príkaz `fillcolor(farba)` nastaví farbu výplne.

```
begin_fill()  
...nakreslený útvar...  
end_fill()
```

Príkaz `begin_fill()` zapíšeme na začiatok kreslenia útvaru, ktorý chceme vyplniť. Príkaz `end_fill()` dáme za posledný príkaz útvaru. Týmto označíme koniec jeho kreslenia a následne sa nakreslený útvar vyplní nastavenou farbou.

## Pracovný list 15 a 16- Vetvenie

### Spoločný príklad

Vytvorte program *Bláznivá predpoveď počasia*. Nakreslite predpoveď počasia na týždeň, v ktorej generujete teplotu pre každý deň náhodne. Každú teplotu znázorníte stĺpcom zodpovedajúcim výške teploty.



Najprv vygenerujeme náhodnú teplotu pre daný deň, ktorú nakreslíme tak, že korytnačku nastavíme aby smerovala hore, a išla dopredu o toľko krokov, koľko bude stupňov v ten deň, potom korytnačku vrátime na jej pôvodné miesto. Ak je teplota **vyššia ako 15 stupňov**, kreslíme stĺpec **červenou** farbou, ak je teplota **menšia alebo rovná ako 15 stupňov**, kreslíme **modrou**.

### Príklad riešte postupne:

- 1 Nakreslite znázornenie predpovede počasia len na jeden deň, čiernou farbou.



- 2 Opravte program tak, aby ste kreslili stĺpce červenou alebo modrou farbou podľa výšky dennej teploty.



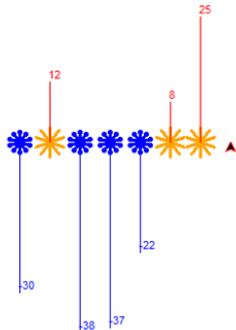
alebo

- 3 Opravte program tak, aby kreslil teploty pre celý týždeň. Po nakreslení dennej teploty posuňte korytnačku o 30 krokov doprava.



Bonusová úloha: Mimoriadne bláznivá predpoveď počasia

Opravte program o bláznivej predpovedi počasia tak, aby generoval aj záporné hodnoty. Vytvorte funkcie `slnko()` a `vlocka()`, pomocou ktorých nakreslite slnko alebo snehovú vločku k stĺpcom podľa výšky dennej teploty.



- 
- 4 Napíšte program, ktorý bude simulovať hádzanie mincou. Ak padne hlava, korytnačka vykreslí kruh. Ak padne písmo, korytnačka vykreslí písmeno A.

Pri tejto úlohe použite príkaz `writeln("A")`, ktorý prikáže korytnačke, aby napísala písmeno A.

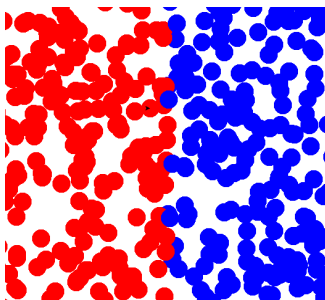
- 
- 5 Upravte program tak, aby sa hodilo mincou 10-krát, do premenných **hlava** a **pismo** ukladajte počet koľkokrát padla hlava a koľkokrát písmo. Na konci vykreslite korytnačkou stĺpce pre počty padnutí hlavy a písma, aby sme videli, čo padlo viackrát.



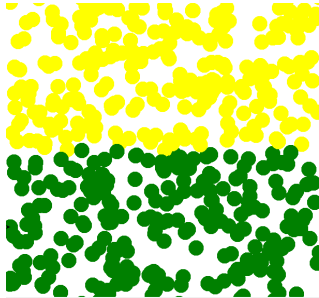
- 
- 6 Upravte program pre hádzanie mincou tak, aby sa hodilo 100 krát.



- 
- 7 Napíšte program, ktorý na náhodné pozície nakreslí 500 farebných bodiek, pričom tie z nich, ktoré sú v ľavej polovici plochy, budú červené a zvyšné v pravej polovici (teda else vetva) budú modré. (Aby ste nemuseli dlho čakať, môžete príkazom `delay(0)` zrýchliť korytnačku).



- 8 Napíšte program, ktorý na náhodné pozície nakreslí 500 farebných bodiek, pričom tie z nich, ktoré sú v hornej polovici plochy, budú žlté a zvyšné v dolnej polovici (teda else vetva) budú zelené.



- 9 Vytvorte program na kreslenie hviezd a kvetov náhodných veľkostí tak, aby korytnačka kreslila vo vrchnej polovici okna hviezdy a v dolnej časti kvety (hviezdy a kvety kreslite pomocou funkcií **hviezda()** a **kvet()** z minulých hodín).





## Pracovný list 17 – Zadanie projektu

Vytvorte program, ktorý vykreslí pohľadnicu alebo reklamu.

V programe musíte použiť:

- knižnice **turtle** a **random**,
- základné príkazy na ovládanie pohybu pera,
- príkazy na zmenu polohy, farby a veľkosti pera,
- **premenné**,
- **cykly**,
- aspoň dve **funkcie**,
- **príkaz vetvenia - if**,
- náhodne generované čísla a náhodne vyberané farby.

Výsledná kresba môže vyzeráť napríklad nasledovne:



## **Python a korytnačia grafika**

Metodický materiál pre vyučovanie základov programovania pre gymnáziá

Autor: Mgr. Eva Mészárosová

Vydavateľ: Knižničné a edičné centrum FMFI UK, Bratislava, 2017

Vydanie: prvé

Rok vydania: 2017

Náklad: 35 ks

Počet strán: 73 s., 25 s. prílohy

Internetová adresa: [www.edi.fmph.uniba.sk/~meszarosova/Python](http://www.edi.fmph.uniba.sk/~meszarosova/Python)

© Mgr. Eva Mészárosová

**ISBN 978-80-8147-080-6**