Cvičenie 1

Sekcia 1: Premenné, výrazy a výstupy

Cieľom tejto sekcie je vyskúšať si prácu s premennými, výrazmi a funkciou print() v jazyku Python. Vašou úlohou bude naprogramovať jednoduché programy, ktoré vykonávajú základné aritmetické operácie s dátami uloženými v premenných.

Úloha č. 1

Zadanie:

Napíšte program (teda skript/zdrojový kód) v jazyku Python, ktorý realizuje nasledovnú činnosť. Program obsahuje premennú *strana_a*. Do tejto premennej sa priamo v zdrojovom kóde priradí nejaká hodnota predstavujúca veľkosť strany štvorca. Program najprv vypíše obvod a potom obsah daného štvorca.

Príklady vstupov / výstupov programu:

Ak sa do premennej *strana_a* priradí hodnota 5, potom obvod takého štvorca je 20 a obsah takého štvorca je 25. Program by teda pre situáciu, že v premennej *a* je uložená hodnota 5, mal vypísať na obrazovku čísla 20 a 25.

Riešenie:

```
strana_a = 5
obvod = 4*strana_a
obsah = strana_a**2
print(obvod)
print(obsah)
```

Riešenie s vloženými komentármi do zdrojového kódu:

```
'''
Program na vypocet obvodu a obsahu stvorca so zadanou dlzkou strany.
'''
strana_a = 5              #strana stvorca
obvod = 4*strana_a        #obvod stvorca
obsah = strana_a**2       #obsah stvorca
print(obvod)
print(obsah)
```

Úloha č. 2

Zadanie:

Napíšte program (teda skript/zdrojový kód) v jazyku Python, ktorý realizuje nasledovnú činnosť. Program obsahuje premenné *strana_a* a *strana_b*. Do týchto premenných sa priamo v zdrojovom kóde priradia nejaké číselné hodnoty predstavujúce veľkosti hrán obdĺžnika. Program najprv vypíše obvod a potom obsah daného obdĺžnika.

Príklady vstupov / výstupov programu:

Ak sa do premenných *strana_a* a *strana_b* priradia hodnoty *strana_a* = 3, *strana_b* = 8, potom obvod takého obdĺžnika je 22 a obsah takého obdĺžnika je 24. Program teda vypíše hodnoty 22 a 24 na obrazovku.

Úloha č. 3

Zadanie:

Napíšte program (teda skript/zdrojový kód) v jazyku Python, ktorý realizuje nasledovnú činnosť. Program obsahuje premennú *sekundy*. Do tejto premennej sa priamo v zdrojovom kóde priradí číselná hodnota predstavujúca ubehnutý čas v sekundách. Program vypíše, koľko hodín / minút / sekúnd predstavuje hodnota v premennej *sekundy.*

Príklady vstupov / výstupov programu:

1) Ak je v premennej *sekundy* hodnota 5043, program vypíše:

1
24
3

pretože 5043 sekúnd predstavuje 1 hodinu, 24 minút a 3 sekundy.

2) Ak je v premennej *sekundy* hodnota 3012, program vypíše:

0
50
12

pretože 3012 sekúnd predstavuje 0 hodín, 50 minút a 12 sekúnd.

3) Ak je v premennej *sekundy* hodnota 10000, program vypíše:

2
46
40

pretože 10000 sekúnd predstavuje 2 hodiny, 46 minút a 40 sekúnd.

Pomôcka:

Zvyšok po delení sa v Pythone dá zistiť pomocou operátora %, t.j. napr. výraz
110 % 4
má hodnotu 2, pretože 110 po delení 4 má zvyšok 2.

Dolná celá časť po delení (tzv. kvocient) sa v Pythone dá zistiť pomocou operátora // t.j. výraz
110 // 4
má hodnotu 27, pretože 110 po delení 4 dáva dolnú celú časť 27.

Úloha č. 4

Cyklista si sleduje prejdenú vzdialenosť v kilometroch a čas (v minútach a sekundách), za ktorý túto vzdialenosť prešiel. Naprogramujte program, ktorý na základe premenných *kilometre*, *minuty*, *sekundy* vypočíta cyklistovu priemernú rýchlosť v km/h (kilometre za hodinu) a vypíše ju na obrazovku.

Príklady vstupov / výstupov programu:

Napríklad ak *kilometre = 8.5*, *minuty = 25* a *sekundy = 30*, potom priemerná rýchlosť cyklistu bola 20.0 km/h.
Ak *kilometre = 9.7*, *minuty = 29* a *sekundy = 55*, potom priemerná rýchlosť cyklistu bola približne 19.454 km/h.

Sekcia č. 2: Korytnačia grafika

V tejto sekcii si precvičíte prácu s korytnačou grafikou v prostredí Python, ale najmä základný koncept programovania – tvorbu zložitejších programov pomocou jednoduchých základných stavebných blokov – inštrukcií/príkazov.

Úlohy v tejto časti majú za cieľ, aby ste si premysleli v každej úlohe, ako sa daný obrazec dá vykresliť pomocou základných príkazov ako „kresli rovnú čiaru“ alebo „otoč sa o XY stupňov vľavo/vpravo“.

Korytnačia grafika je **modul** s názvom ***turtle*** v jazyku Python, ktorý umožňuje jednoduché kreslenie grafických útvarov pomocou rovných čiar. Základom je kurzor v tvare trojuholníka („korytnačka“), ktorý sa posúva po obrazovke a za sebou kreslí rovnú čiaru.

Ak chceme v jazyku Python používať nejaký už vytvorený modul, musíme ho najprv vložiť (importovať) do programu. V prípade modulu ***turtle*** to urobíme príkazom:

import turtle

Kresliaci kurzor modulu *turtle* má v každom momente nejaký smer, v ktorom sa pohybuje. Pri každom pohybe vpred/vzad kurzor **kreslí za sebou čiaru.** Pomocou takéhoto kreslenia budeme kresliť rôzne geometrické útvary.

Základné príkazy:

turtle.forward(n)	posun vpred o *n* pixelov
turtle.backward(n)	posun vzad o *n* pixelov
turtle.left(n)	otočenie vľavo o *n* stupňov
turtle.right(n)	otočenie vpravo o *n* stupňov

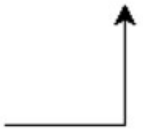
Dôležité upozornenie!!! Ak používate nejaký modul vo svojom programe, NESMIETE pomenovať svoj zdrojový kód (súbor .py) ROVNAKÝM menom, ako používaný modul! Napríklad v prípade modulu *turtle* NESMIETE pomenovať svoj zdrojový kód ako „turtle.py“, pretože program nebude fungovať!

Úloha č. 1

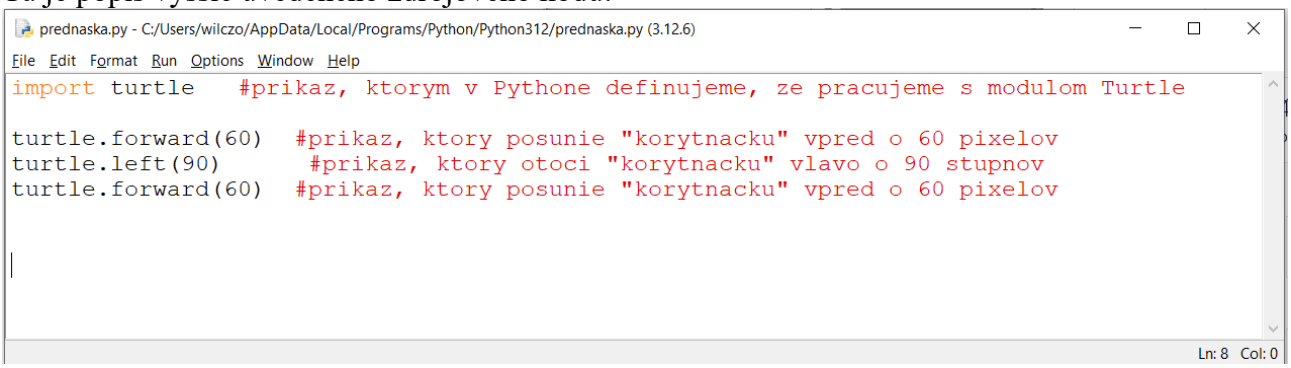
Pozrite si nasledovný zdrojový kód a spustite si ho v prostredí IDLE:

```
import turtle

turtle.forward(60)
turtle.left(90)
turtle.forward(60)
```

Malo by dôjsť k vykresleniu nasledovného obrázka:



Tu je popis vyššie uvedeného zdrojového kódu:



Mali by ste teda rozumieť, ako fungujú príkazy turtle.forward() a turtle.left(). Analogicky fungujú aj turtle.backward() a turtle.right().

Úloha č. 2

Pomocou korytnačej grafiky nakreslite **štvorec.** Veľkosť strany štvorca si môžete zvoliť ľubovoľne, napríklad 100 pixelov.

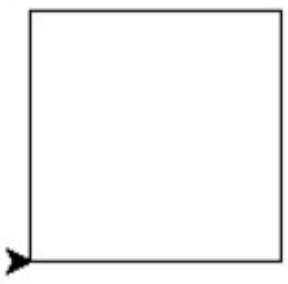
Riešenie:

Potrebujeme nasledovné príkazy:
1) import turtle – pre import modulu turtle pre prácu s korytnačou grafikou
2) turtle.forward(100) – pre kreslenie rovnej čiary o dĺžky 100 pixelov
3) turtle.left(90) – pre otočenie vľavo o 90 stupňov

Následne už len potrebujeme vymyslieť, ako pomocou takýchto jednoduchých príkazov ich vhodnou kombináciou a opakovaním nakreslíme štvorec. Keďže štvorec tvoria 4 strany, pričom zvierajú medzi sebou 90 stupňov, štvorec nakreslíme tak, že 4-krát zopakujeme kreslenie rovnej čiary a otočenie o 90 stupňov:

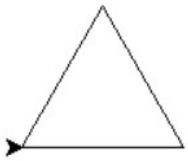
```
import turtle

turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
```

Výsledný obrázok:



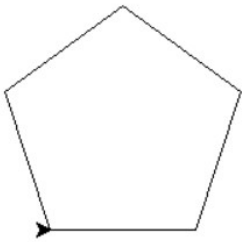
Úloha č. 3
Pomocou korytnačej grafiky nakreslite **rovnostranný trojuholník.** Veľkosť strany trojuholníka si môžete zvoliť ľubovoľne, napríklad 100 pixelov.

Požadovaný výsledok:

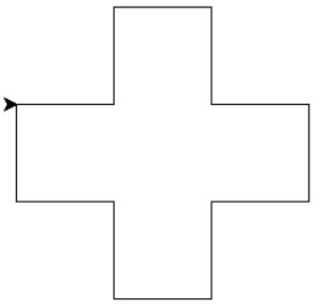


Úloha č. 4
Pomocou korytnačej grafiky nakreslite **rovnostranný päťuholník.** Veľkosť strany päťuholníka si môžete zvoliť ľubovoľne, napríklad 100 pixelov.

Požadovaný výsledok:



Úloha č. 5
Pomocou korytnačej grafiky nakreslite nasledovný obrázok**.**



Sekcia č. 3: Korytnačia grafika – pokročilé úlohy pre skúsenejších programátorov

Ak už máte s programovaním skúsenosti a ovládate koncepty ako cykly alebo funkcie, môžete skúsiť vyriešiť úlohy v tejto tretej sekcii. Ostatným z Vás odporúčame sa k týmto úlohám vrátiť neskôr, keď na prednáškach preberieme cykly a funkcie.

Teda pre tých z Vás, ktorí ovládate funkcie a cykly, tu je rýchlokurz geniality, ako sa definuje funkcia v jazyku Python. Pôjde o definíciu funkcie s názvom *mocnina*, ktorá má jeden vstupný parameter, číslo *i* a ktorá vracia druhú mocninu čísla *i:*

```
def mocnina(i):
    return i*i
```

Definícia funkcie v jazyku Python má nasledovnú syntax:
1) kľúčové slovo „def“ za ktorým nasleduje identifikátor (meno) funkcie, v našom prípade „mocnina“. V zátvorkách sú potom vymenované vstupné parametre funkcie, v našom prípade má funkcia 1 vstupný parameter, premennú „*i*“.
2) Telo funkcie tvoria potom všetky príkazy, ktoré sú napísané na ďalších riadkoch a zároveň sú odsadené o nejaký počet medzier od začiatku riadku – štandardne sa používajú 4 medzery. Preto je v uvedenom príklade príkaz „return i*i“ odsadený o 4 medzery od začiatku riadku.
3) V prípade, že funkcia má vracať nejakú hodnotu, používa sa kľúčové slovo „return“ za ktorým nasleduje hodnota, ktorá sa má vrátiť. V našom prípade druhá mocnina *i*, t.j. *i*i.*

Tu je rýchlokurz geniality, ako sa používa for-cyklus v jazyku Python. Ak chcem nejaké príkazy zopakovať povedzme 4-krát, napríklad vypísať *Hello world* na obrazovku 4-krát, urobím to nasledovným spôsobom:

```
for i in range(4):
    print("Hello world")
```

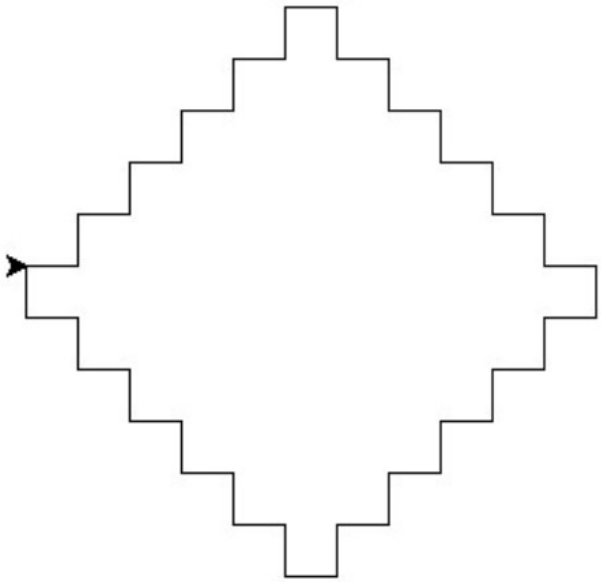
Teda použitie for cyklu v jazyku Python má nasledovnú syntax:
1) kľúčové slovo „for“ za ktorým nasleduje riadiaca premenná cyklu – v našom prípade sme použili identifikátor „i“, za ktorým nasleduje kľúčové slovo „in“, za ktorým nasleduje funkcia range(), v ktorej je ako argument požadovaný počet opakovaní, v našom prípade 4.
2) Telo cyklu tvoria príkazy, ktoré sú na ďalších riadkoch, odsadené od začiatku riadku – znovu sa štandardne používajú na odsadenie 4 medzery.

Skúste si teda sami pozrieť, čo asi urobí nasledovný kód:

```
def mocnina(i):
    return i*i

for i in range(4):
    print(mocnina(i))
```

V zvyšných úlohách v tejto tretej sekcii, s pokročilými úlohami, je teda vhodné, aby ste používali for-cykly a funkcie. Znovu opakujeme, že ak ste sa s týmito konceptami ešte nestretli, **vráťte sa k tejto sekcii v budúcnosti** a to, že v prvom týždni semestra úlohy vyriešiť neviete, nie je žiaden problém.

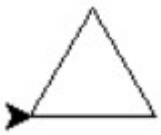
Úloha č. 1
Pomocou korytnačej grafiky a for-cyklu vykreslite nasledovný obrázok:



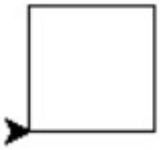
Úloha č. 2
Definujte funkciu *nuholnik(n)* so vstupným parametrom *n*, ktorá vykreslí rovnostranný *n*-uholník.

Príklad vstupu/výstupu:

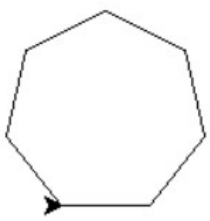
Volanie *nuholnik(3)* vykreslí rovnostranný trojuholník.



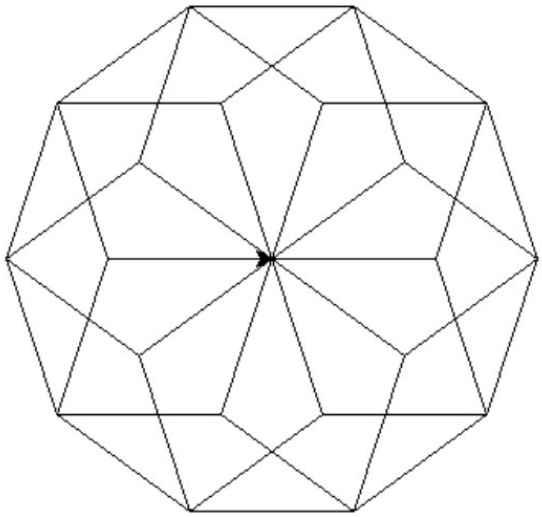
Volanie *nuholnik(4)* vykreslí štvorec.



Volanie *nuholnik(7)* vykreslí rovnostranný sedemuholník.

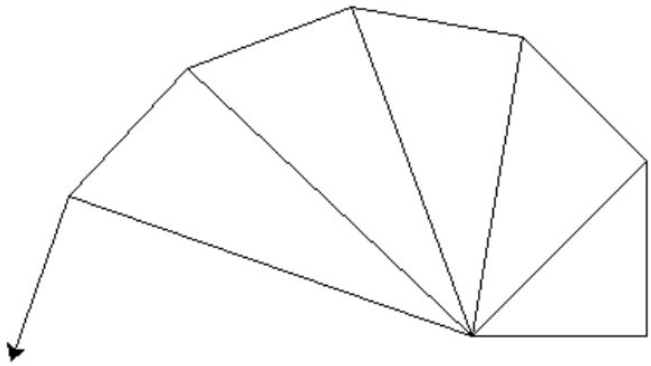


Úloha č. 3
Pomocou korytnačej grafiky a for-cyklu vykreslite nasledovný obrázok.



Hint: Všimnite si, že obrázok pozostáva z päťuholníkov, rotovaných okolo stredu obrázku!

Úloha č. 4
Pomocou korytnačej grafiky a for-cyklu vykreslite nasledovný obrázok.



Hint: Všimnite si, že špirála na obrázku pozostáva z na seba naukladaných pravouhlých trojuholníkov a že strany trojuholníkov, ktoré tvoria obvod špirály majú vždy rovnakú dĺžku.

Hint č. 2: Pre jednoduchšie kreslenie si preštudujte, aké všetky možné príkazy viete využívať v module s korytnačou grafikou, na stránke https://docs.python.org/3/library/turtle.html

Konkrétne Vás môžu zaujímať / preštudujte si nasledovné funkcie:
setheading(), position(), towards(), distance()

Úloha č. 5

Definujte funkciu *odmocnina(n)* so vstupným parametrom *n*, ktorá pomocou príkazov korytnačej grafiky a obrázka z predchádzajúcej úlohy vráti odmocninu z *n*.

Vstupy a výstupy:

Volanie funkcie *odmocnina(3)* vráti hodnotou **približne** 1.732
Volanie funkcie *odmocnina(4)* vráti hodnotou **približne** 2.0

Váš program pravdepodobne nebude počítať odmocninu úplne presne, keďže ju bude odhadovať na základe kresleného obrázka, takže si z toho nerobte ťažkú hlavu :).

Hint: Zamyslite sa, ako súvisí počítanie odmocniny s obrázkom špirály z úlohy č. 4. Napríklad ak by odvesny „prvého“ pravouhlého trojuholníka a následne všetky „vonkajšie“ hrany špirály mali dlžku 1, aké by boli hodnoty prepôn v pravouhlých trojuholníkoch?