Programko.sk

Roblox a Lua 1

Inštalácia Pythonu

Stránka: https://www.python.org/downloads/release/python-382/
1. Prescrollujeme na úplny spodok stránky a vyberieme váš operačný systém (pre windows: „Windows x86-64 executable installer“, pre mac: „macOS 64-bit installer“)

Files

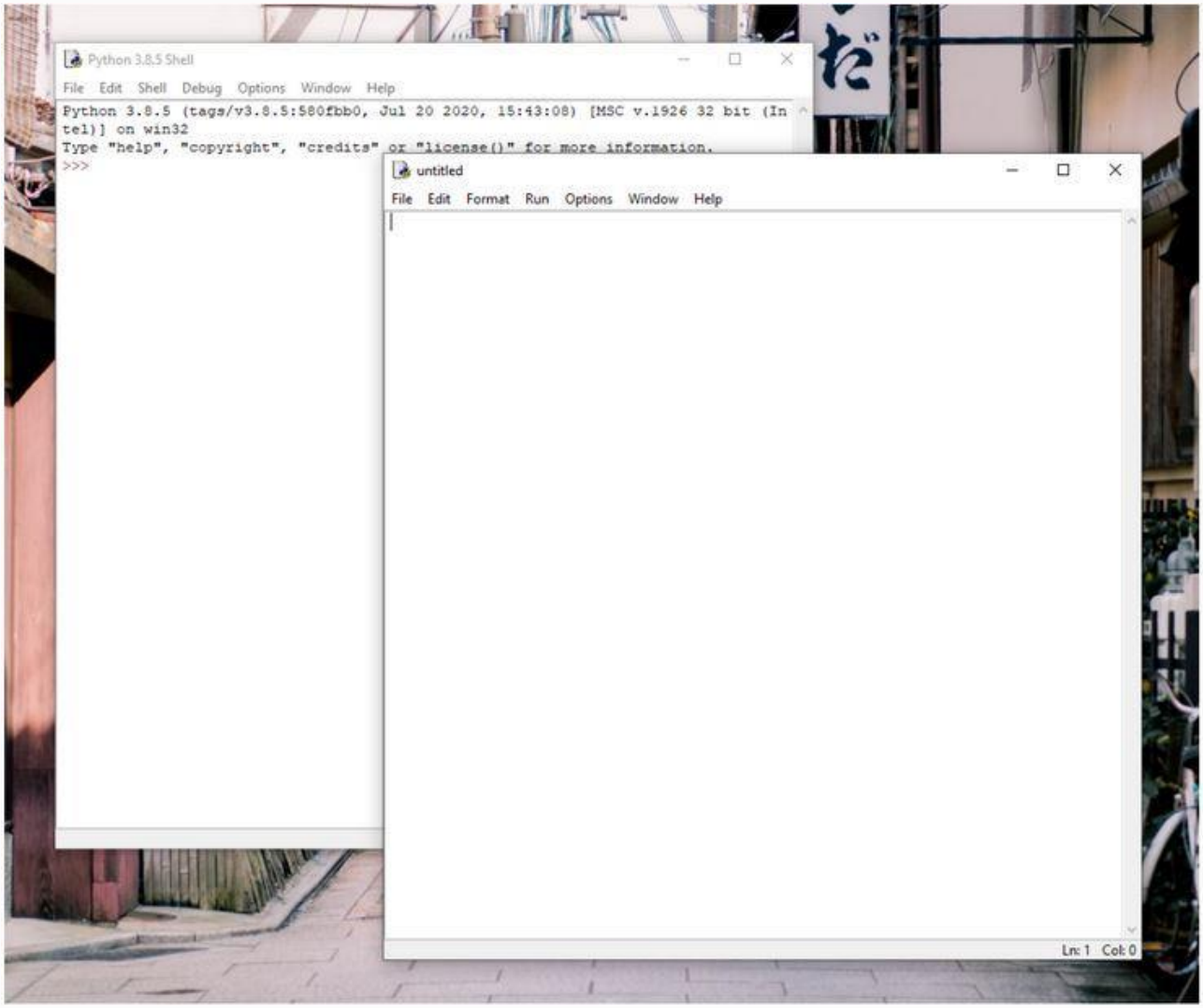
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		f9f3768f757e34b342dbc06b41cbc844	24007411	SIG
XZ compressed source tarball	Source release		e9d6ebc92183a177b8e8a58cad5b8d67	17869888	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	f12203128b5c639dc08e5a43a2812cc7	30023420	SIG
Windows help file	Windows		7506675dcbb9a1569b54e600ae66c9fb	8507261	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	1a98565285491c0ea65450e78afe6f8d	8017771	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	b5df1cbb2bc152cd70c3da9151cb510b	27586384	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	2586cdad1a363d1a8abb5fc102b2d418	1363760	SIG
Windows x86 embeddable zip file	Windows		1b1f0f0c5ee8601f160cfad5b560e3a7	7147713	SIG
Windows x86 executable installer	Windows		6f0ba59c7dbeba7bb0ee21682fe39748	26481424	SIG
Windows x86 web-based installer	Windows		04d97979534f4bd33752c183fc4ce680	1325416	SIG

2. Následne sa vám stiahne inštalačný súbor, ktorý spustíme

3. (Windows inštalátor) vyberieme „Install Now“
4. Keď sa dokončí inštalácia, klikneme „Close“
5. Teraz by ste mali mať na počítači nainštalovaný python spolu s jednoduchým IDE (Integrated development environment – prostredie kde sa píše kód programovacieho jazyka), nayvaným IDLE
6. Nájdite program IDLE, buď cez štart alebo na pracovnej ploche, mal by mať meno podobné „IDLE (Python 3.8)“

Vytvorenie nového programu

1. Keď máme otvorené IDLE, klikneme na File – New File
2. Malo by sa vám otvoriť nové, prázdne okno s názvom untitled



3. Sem začneme písať náš kód. Pred jeho spustením však musíme súbor uložiť (ctrl + s a vyberieme miesto kam ho uložíme), následne program spustíme tlačítkami Run – Run Module, alebo stlačením klávesy F5.

Začíname s korytnačkou

Nasledujúci program nám otvorí okno s korytnačkou. Prvý príkaz nám umožní používať príkazy z knižnice turtle, napr. príkaz showturtle(), ktorý zobrazí korytnačku.

```
from turtle import *
showturtle()
```

Ak nepoužívame IDLE, ale napr. Visual studio Code, treba použiť aj príkaz done(), aby sa nám okno s korytnačkou hneď nezatvorilo.

```
done()
```

Príklad 1.1

Otvorte okno s korytnačkou.

Alternatívne spôsoby importovania (nepovinné)

Knižnicu na prácu s korytnačkou môžeme importnúť viacerými spôsobmi, nám bude stačiť jeden. Ale pre zaujímavosť uvádzame aj ďalšie spôsoby.

2. spôsob: Knižnicu naimportujeme po pomocou príkazu import turtle, ale museli by sme pri každom použití príkazu z knižnice turtle napísať tutle.<príkaz> tak, ako vidíte na nasledujúcom príklade:

```
import turtle
turtle.showturtle()
```

3. spôsob: Rovnako ako 2. spôsob, ale pri importnovaní knižnice si zvolíme meno, ktoré budeme používať miesto slova turtle.

```
import turtle as t
t.showturtle()
```

Pohyb korytnačky

Korytnačkou môžeme posunúť dopredu pomocou príkazu forward. Číslo v zátvorke je paramatrom funkcie forward a vyjadruje o koľko sa posúvame dopredu. Korytnačka bude za sebou zanechávať čiaru.

```
#toto je komentár - počítač bude tento riadok ignorovať
forward(100)        #dopredu o 100
forward(20)         #dopredu o 20
```

Príklad 1.2

Posuňte korytnačku dopredu. Vyskúšajte viaceré hodnoty parametra funkcie forward a porovnajte. Čo nakreslí korytnačka?

Otáčanie korytnačky

Korytnačkou môžeme otáčať pomocou príkazov left a right. Číslo v zátvorke nám vyjadruje, o koľko stupňov sa otáčame. Ak sa chceme otočiť o pravý uhol, použijeme číslo 90.

```
left(90)        #otočenie doľava o 90 stupňov
right(30)           #otočenie doprava o 30 stupňov
```

Príklad 1.3

Akým smerom sa pozerá korytnačka, ak nepoužijeme žiadne príkazy left a right?

Príklad 1.4

Použite príkaz right(90). Akým smerom sa korytnačka pozerá po použití tohto príkazu?

Príklad 1.5

Nakreslite čiaru smerom dole.
Nápoveda: Najprv korytnačku otočte, potom posuňte.

Príklad 1.6

Posuňte korytnačku, potom ju otočte a potom znovu posuňte.

Príkaz delay

Ak korytnačke zadáme viac príkazov naraz, vykoná ich pomerne rýchlo za sebou a uvidíme len výsledný obrázok. Ak chceme, môžeme korytnačku spomaliť príkazom delay.

```
delay(100)          #čím väčšie číslo, tým viac spomalíme korytnačku
```

Príklad 1.7

Dodajte do riešenia príkladu 1.6 príkaz delay. Rozhodnite sa, či chcete svoju korytnačku spomaliť a o koľko.

Príklad 1.8

Máte nejaké nápday, čo si nakresliť? Na konci každej hodiny si môžete nakresliť, čo chcete. Používať budeme len príkazy, čo sme sa naučili doteraz a po každej hodine budeme používať zložitejšie a zložitejšie príkazy. Je niečo, čo by ste si chceli nakresliť, ale zatiaľ to nevieme?

2. Hodina

Vypnutie a zapnutie pera

Ako nakresliť prerušovanú čiaru? Musíme vypnúť kreslenie a potom ho zapnúť. Použijeme nasledujúce príkazy.

```
penup()             #vypne kreslenie
pendown()           #zapne kreslenie
```

Príklad 2.1

Nakreslite prerušovanú čiaru.

Zmena farby pera

Farbu pera korytnačky môžeme zmeniť pomocou funkcie pencolor. Táto funkcia má 1 parameter - názov farby, ktorú chceme nastaviť. Názov farby je v textovom formáte, preto je v uvodzovkách

```
pencolor('blue')                #zmení farbu pera na modrú
```

Príklad 2.2

Prefarbite čiaru z minulého príkladu na modro.

Príklad 2.3

Prefarbite čiaru tak, aby každá jej časť bola inou farbou.

Zobrazenie textu

Súčasťou nášho obrázka môže byť aj text. Použijeme na to príkaz write.

```
write("nápis") #napíše text v úvodzovkách na aktuálnej pozícii korytnačky
```

Príklad 2.4

Napíšte pod obrázok "čiara".
Nápoveda: pred použitím príkazu write presuňte korytnačku pod čiaru.

Zmena hrúbky pera

Hrúbku pera môžeme nastaviť pomocou funkcie pensize. Funkcia má 1 parameter, čím väčšie číslo zadáme, tým väčšiu hrúbku pera nastavíme.

```
pensize(5)                      #nastaví hrúbku pera na 5
```

Príklad 2.5

Každej časti prerušovanej čiary nastavte inú hrúbku.

Zmena farby pozadia

Pozadie môžeme zmeniť pomocou funkcie bgcolor. Funkcia má rovnako ako pencolor 1 parameter, ktorý je v uvodzovkách.

```
bgcolor('blue')                #zmení farbu pozadia na modrú
```

Príklad 2.6

Zmeňte farbu pozadia na vašu obľúbenú farbu.

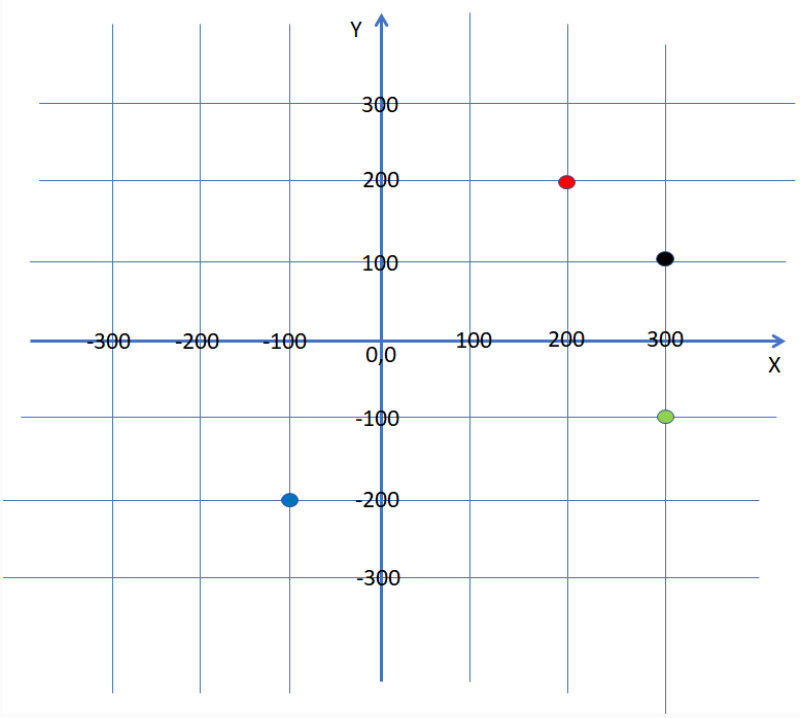
Príklad 2.7

Nakreslite si, čo chcete. Použite nové príkazy, ktoré sme sa naučili.

Súradnice

Každému bodu v okne korytnačky prislúcha dvojica čísel. Tieto čísla sa nazývajú súradnice. Každý bod má iné súradnice. Vďaka tomu môžeš povedať korytnačke: “Choď na bod so súradnicami napr. 300, 100” a ona bude vedieť, ktorý bod máš na mysli.

Prvá súradnica **X** vyjadruje, ako ďaleko sa bod nachádza od stredu obrazovky vo vodorovnom smere. Ak prvá súradnica je 300, bod sa nachádza o 300 pixelov napravo od stredu obrazovky vo vodorovnom smere (-300 by bolo 300 pixelov naľavo od stredu obrazovky).
Druhá súradnica **Y** vyjadruje, ako ďaleko sa bod nachádza od stredu obrazovky vo zvislom smere. Ak druhá súradnica je 100, bod sa nachádza o 100 pixelov hore od stredu obrazovky vo vodorovnom smere (-100 by bolo dole od stredu obrazovky).
Bod so súradnicami 300, 100 sa teda nachádza 300 pixelov doprava a 100 pixelov hore od stredu. Je vyznačený na obrázku čiernou farbou.



Príklad 3.1

Aké súradnice majú modrý, zelený a červený bod na obrázku?

Zistenie súradníc korytnačky

Ak chceme zistiť súradnice korytnačky, použijeme kombináciu príkazov write() a pos().

```
write(pos())                #vypíše súradnice korytnačky
```

Príklad 3.2

Zistite počiatočné súradnice korytnačky.

Príklad 3.3

Skúste určiť, aké súradnice bude mať korytnačka po použití príkazu forward(100). Zapíšte si váš výsledok a potom ho overte programom.

Príklad 3.4

Skúste určiť, aké súradnice bude mať korytnačka po použití príkazov forward(100), right(90), forward(250). Zapíšte si váš výsledok a potom ho overte programom.

Presun korytnačky pomocou súradníc

Ak chceme zistiť súradnice korytnačky, použijeme nasledujúce príkazy:

```
goto(300, -100)          #presunie korytnačku na bod so súradnicami 300, -100
```

Príklad 3.5

Nakreslite trojuholník. Bude sa vám kresliť lepšie pomocou príkazov forward, right, left alebo pomocou goto? Prečo?

Príklad 3.6

Nakreslite niečo vlastné použitím len príkazov goto.

Premenná

Premenná nám umožňuje uložiť nejakú hodnotu. Môžeme ju prirovnať ku krabičke v pamäti počítača, do ktorej môžeme vložiť hodnotu, ktorú chceme uložiť. Každá premenná má názov a hodnotu, táto hodnota sa môže meniť.

Názov premennej si môžeme zvoliť sami, ale musí začínať písmenom alebo podčiarkovníkom, musí obsahovať len písmená a čísla a nemôže byť kľúčovým slovom jazyka python. Napr. zvolíme názov pocetKrokov. Túto premennú vytvoríme nasledovne:

```
pocetKrokov = 50
```

Vytvorili sme premennú s názvom pocetKrokov a uložili sme do nej hodnotu 50. Ak tento program spustíme, nenakreslí sa nám nič, lebo zmena sa udeje len v pamäti počítača. Ak sa chceme presvedčiť, že naozaj sa do premennej pocetKrokov uložilo číslo 50, môžeme to spraviť napr. takto:

```
forward(pocetKrokov)
```

Korytnačka sa posunie o 100 krokov dopredu, lebo v premennej pocetKrokov je uložená hodnota 100. Keby sme chceli hodnotu premennej zmeniť na 100 urobíme to nasledovne:

```
pocetKrokov = 100
```

Príkaz vyzerá rovnako ako príkaz na vytvorenie premennej, rozdiel je v tom, že na ľavej strane použijeme názov premennej, ktorá už je vytvorená. Opäť sa môžeme presvedčiť o zmene hodnoty premennej použitím prikazu forward.

```
forward(pocetKrokov)
```

Tentokrát sa nám korytnačka posunie až o 100 krokov, lebo aktuálna hodnota premennej pocetKrokov je 100.

Príklad 4.1

O koľko krokov sa posunie korytnačka po vykonaní týchto príkazov?

```
pocet = 100
color("blue")
pocet = 20
forward(pocet)
```

Ako by sa premenná dala využiť? Predstavte si, že chceme nakresliť písmeno štvorec, ale nevieme sa rozhodnúť, ako veľký má byť. Radi by sme si vyskúšali štvorce rôznych veľkostí.

Príklad 4.2

Nakreslite štvorec (zatiaľ nepoužívame cyklus).

Príklad 4.3

Skúste tento štvorec zväčšiť. Koľko čísel musíte prepísať?

Príklad 4.4

Vytvorte premennú stranaStvorca a použite ju tak, aby nám na zväčšenie štvorca stačilo prepísať 1 číslo.

Príklad 4.5

Nakreslite obdĺžnik.

Príklad 4.6

Skúste zmeniť šírku obdĺžnika (zatiaľ nemeníme výšku). Koľko čísel musíte prepísať? Použite premennú, aby sme mohli meniť šírku obdĺžnika prepísaním jedného čísla.

Príklad 4.7

Vytvorte druhú premennú, aby sme mohli meniť aj výšku obdĺžnika prepísaním len jedného čísla.

Matematické operácie

V pythone môžeme používať matematické operácie ako sú +, -, * (krát) a / (deleno).
Príklady použitia:

```
write(5 + 5)
write(2 * 3)
write(10 - 6 / 2)
```

Príklad 5.1

Skúste určiť, čo spraví kód uvedený vyššie. Vaše výsledky overte programom.

Príklad 5.2

Vyskúšajte nasledujúce 2 príkazy:
write(5+5)
write("5+5")
Dostaneme rovnaký výsledok? Ak nie, prečo?

Matematické operácie a premenné

```
dlzka = 10                  #vytvorenie premennej dlzka s hodnotou 10
dlzka2 = 10 + 5             #vytvorenie premennej dlzka2 s hodnotou 15
dlzka3 = dlzka + 5          #vytvorenie premennej dlzka3 s hodnotou 15
dlzka = dlzka + 2           #zvýšenie premennej dlzka o 2
```

Príklad 5.3

Akú hodnotu bude mať premenná x po vykonaní týchto príkazov?
x = 10
x = 3 + x/2
Ako by sme to overili programom?

Príklad 5.4

Akú hodnotu bude mať premenné x, y a z po vykonaní týchto príkazov?
x = 10
y = x/2
z = x + y
Ako by sme to overili programom?

Príklad 5.5

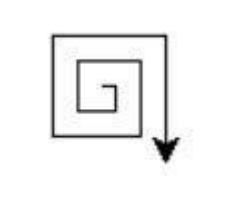
Nakreslite písmeno L (Zvislá čiara je 2x tak dlhá ako vodorovná).

Príklad 5.6

Upravte riešenie predchádzajúceho príkladu. Vytvorte premennú vodorovnaCiara a nastavte ju na 50. Vytvorte premennú zvislaCiara a nastavte jej hodnotu na polovicu hodnoty premennej vodorovnaCiara. Obe premenné použite pri kreslení L. Akú výhodu má použitie premenných?

Príklad 5.7

Nakreslite hranatú špirálu ako na obrázku. Prvá strana má dĺžku 5 a každá ďalšia je o 5 dlhšia.



Cykly

V predchádzajúcom príklade sa stále opakovali rovnaké 3 riadky kódu. Namiesto kopírovania tých istých príkazov môžeme použiť cyklus.

```
for i in range(10):
    #tieto príkazy sa vykonajú 10-krát
```

Do zátvorky za "range" dáme počet opakovaní. Toľkokrát sa vykonajú nasledujúce príkazy. Všetky príkazy, ktoré chceme opakovať odsadíme tabulátorom.

Príklad 6.1

Vylepšite kreslenie špirály z minulej hodiny použitím cyklu.

Príklad 6.2

Napíšte pod špirálu text "Špirála". Napísať text chceme len raz, preto tento príkaz nemôže byť súčasťou cyklu.

Príklad 6.3

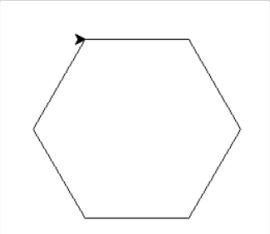
Nakreslite švorec.

Príklad 6.4

Nakreslite prerušovanú čiaru.

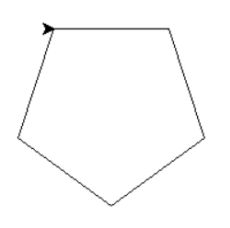
Príklad 6.5

Nakreslite rovnostranný šesťuholník.
Nápoveda: Šesťuholník má 6 uhlov, súčet všetkých uhlov v mnohouholníku je 360.
360/6 = 60



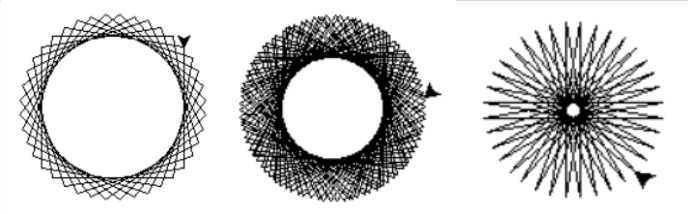
Príklad 6.6

Nakreslite rovnostranný päťuholník.
Nápoveda: 360/5 = 72



Príklad 6.7

V predchádzajúcom príklade zmeňte uhol (odporúčam hodnoty od 30 do 179) a zvýšte počet opakovaní.



Funkcie

Funkcie vytvárame pomocou kľúčového slova “def”, potom nasleduje názov funkcie, okrúhle zátvorky a dvojbodka. Na ďalšom riadku nasledujú príkazy vnútri funkcie odsadené tabulátorom. Tieto príkazy sa nevykonajú, kým funkciu nezavoláme.

```
def stvorec():
        #príkazy funkcie stvorec
stvorec()                       #volanie funkcie
```

Názov funkcie si môžeme zvoliť sami, ale musí začínať písmenom alebo podčiarkovníkom, môže obsahovať len písmená, čísla a podčiarkovníky (nemôže obsahovať napr. medzeru).

Príklad 7.1

Vytvorte funkciu, ktorá nakreslí štvorec.

Príklad 7.2

Zavolajte funkciu z predchádzajúcej úlohy. Potom korytnačku posuňte a zavolajte túto funkciu znova.

Príklad 7.3

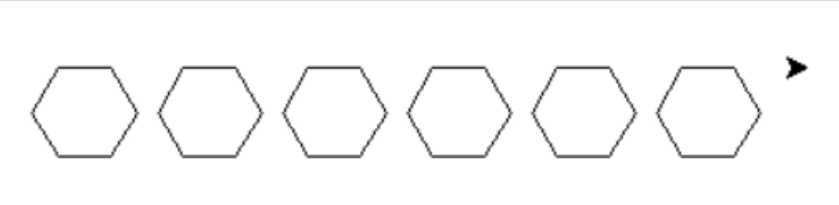
Vytvorte funkciu, ktorá nakreslí rovnostranný šesťuholník.

Príklad 7.4

Zavolajte funkciu z predchádzajúcej úlohy. Potom korytnačku posuňte a zavolajte funkciu, ktorá kreslí štvorec.

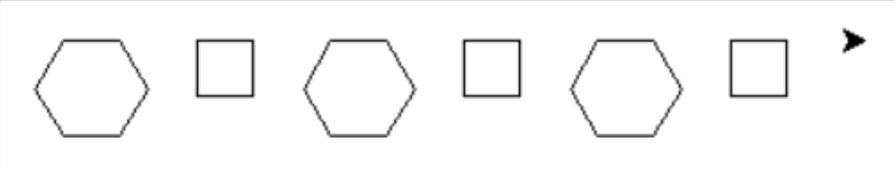
Príklad 7.5

Nakreslite viac šesťuholníkov za sebou za sebou. Skombinujte cyklus a funkciu, ktorá kreslí šesťuholník.



Príklad 7.6

Zmeňte predchádzajúci príklad tak, aby každý druhý vykreslený útvar bol štvorec.



Funkcie s parametrom

Na minulej lekcii sme sa naučili vytvárať funkcie bez parametrov, dnes sa naučíme vytvárať funkcie s parametrom. Pri volaní funkcie dáme do zátvoriek hodnotu parametra.

```
stvorec()                   #volanie funkcie bez parametra
stvorec(10)                 #volanie funkcie s parametrom 10
stvorec(20)                 #volanie funkcie s parametrom 20
```

Rozdiel bude aj pri vytváraní funkcie. Do zátvoriek dáme názov parametra. Názov parametra si môžete vymyslieť, platia rovnaké pravidlá ako pre názvy premenných.

```
def stvorec():              #vytvorenie funkcie bez parametra
def stvorec(strana):        #vytvorenie funkcie s parametrom strana
def stvorec(vzdialenost):   #vytvorenie funkcie s parametrom vzdialenost
```

Parameter strana môžeme potom využiť vnútri funkcie:

```
def stvorec(strana):        #vytvorenie funkcie s parametrom strana
    forward(strana)         #využitie parametra strana
```

Ak zavoláme túto funkciu s hodnotou parametra 10, vo funkcii sa miesto “strana” dosadí číslo 10.
Ak zavoláme túto funkciu s hodnotou parametra 20, vo funkcii sa miesto “strana” dosadí číslo 20.

Príklad 8.1

Vytvorte funkciu stvorec s parametrom strana.

```
stvorec(100)        #nakreslí štvorec so stranou 100
stvorec(200)        #nakreslí štvorec so stranou 200
```

Čím váčši číslo, tým väčší štvorec.

Príklad 8.2

Zavolajte funkciu z predchádzajúceho príkladu s hodnotou parametra 50. Potom korytnačku posuňte doprava a zavolajte tú istú funkciu s hodnotou parametra 100.

Príklad 8.3

Vytvorte funkciu stvorec s parametrom vzdialenost. V tomto prípade bude funkcie kreslit rovnako veľké štvorce, parameter bude určovať, ako ďaleko od stredu sa štvorec nakreslí. Korytnačka najprv prejde istú vzdialenosť doprava, až potom nakreslí štvorec so stranou dĺžky 100.

Príklad 8.4

Vytvorte funkciu pismeno s parametrom dlzka. Funkcie nakreslí písmeno L, parameter bude určovať dĺžku dlhšej strany. Kratšia strana bude mať polovičnú dĺžku.

Viac parametrov

Ak má funkcia 2 a viac parametrov, oddeľujeme ich čiarkami.

```
def obdlznik(a, b):                #vytvorenie funkcie s parametrami a, b
def stvorec(x, y, strana)          #vytvorenie funkcie s parametrami x, y, strana
```

Príklad 8.5

Vytvorte funkciu obdlznik s parametrami a, b.

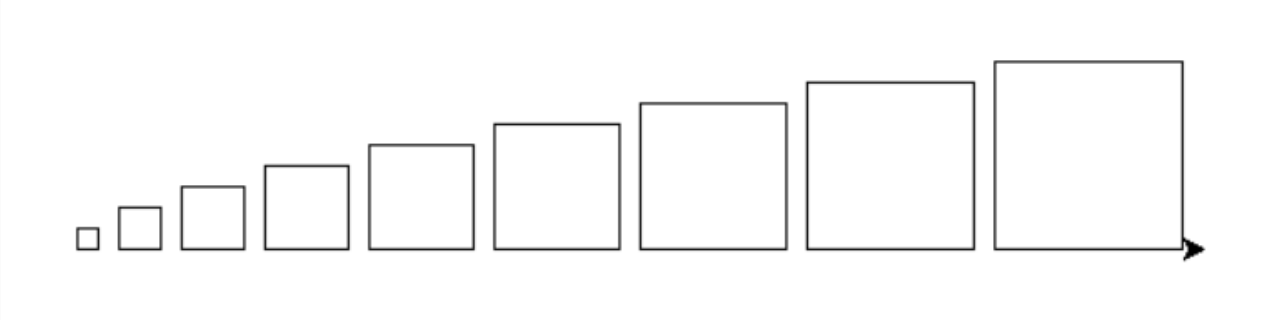
```
obdlznik(50, 100)          #nakreslí obdĺžnik s dĺžkami strán 50 a 100
obdlznik(80, 120)          #nakreslí obdĺžnik s dĺžkami strán 80 a 120
```

Príklad 8.6

Vytvorte funkciu stvorec s parametrami x, y, strana. Prvé 2 prametre x, y budú určovať súradnice ľavého dolného rohu štvorca, 3. pramater bude určovať dĺžku strany štvorca.

Príklad 8.7*

Nakreslite viacero útvarov (napr. štvorcov) za sebou, každý útvar bude väčší ako predchádzajúci.



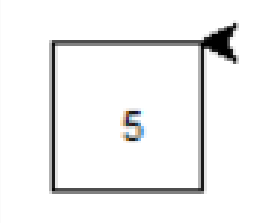
Generovanie náhodných čísel

Náhodné čísla generujeme pomocou funkcie randint. Táto funkcia sa nachádza v knižnici random, ktorú musíme najprv naimportovať (podobne ako korytnačku). Keďže z knižnice random budeme používať len funkciu randint, nebudeme importovať celú knižnicu, ale len jej funkciu randint.

```
from random import randint   #import funkcie random z knižnice randint
randint(0, 10)               #generovanie náhodného čísla od 0 do 10
write(randint(0,10))         #generovanie a vypísanie náhodného čísla
```

Príklad 9.1

Naprogramujte hádzanie kockou. Nakreslite švorec s náhodným číslom od 1 do 6 uprostred.



Príklad 9.2

Nakreslite štvorec náhodnej veľkosti.
Nápoveda:
1. Vygenerjute náhodné číslo (a uložte ho do premennej).
2. Nakreslite štvorec. Použite premennú z bodu 1.

Príklad 9.3

Nakreslite štvorec na náhodnej pozícii.
Nápoveda:
1. Vygenerjute 2 náhodné čísla (a uložte ich do premennej).
2. Presuňte korytnačku na pozíciu s vygenerovanými súradnicami.
3. Nakreslite štvorec.

Príklad 9.4

Pozrite si funkciu pod týmto príkladom. Čo myslíte, čo robí táto funkcia? Zavolajte ju, vyskúšajte viaceré hodnoty parametrov.

```
def hviezda(x, y, velkost):
    penup()
```

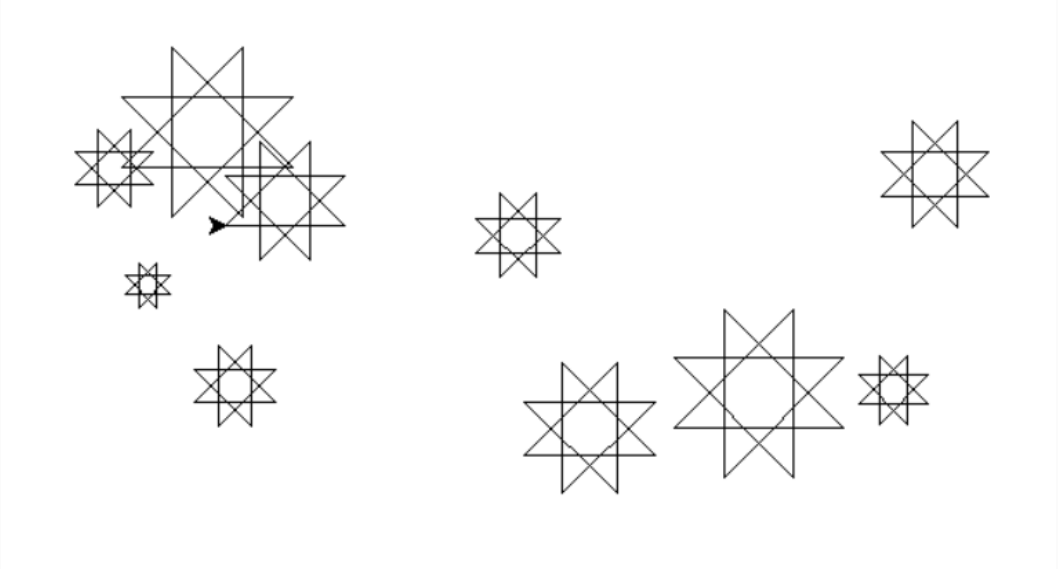
```
    goto(x,y)
    pendown()
    for i in range(8):
        forward(velkost)
        left(135)
```

Príklad 9.5

Nakreslite štvorec na náhodnej pozícii.
Nápoveda:
1. Vygenerjute 2 náhodné čísla (a uložte ich do premennej).
2. Presuňte korytnačku na pozíciu s vygenerovanými súradnicami.
3. Nakreslite štvorec.

Príklad 9.6

Použite túto funkciu na nakreslenie viacerých hviezdičiek na náhodných miestach. Veľkosť môžete zvoliť akú chcete alebo vygenerovať náhodnú.



Vetvenie

Na vetvenie programu nám slúži príkaz if. Používame ho vtedy, ak chceme, aby sa niečo stalo, ak platí určitá podmienka. Ak táto podmienka neplatí, stane sa niečo iné.
Príklad:

```
if y < 10:
        write("platí podmienka")
else:
        write("neplatí podmienka")
```

Príklad 10.1

A) Vytvorte premennú y a nastavte ju na 50. Potom použite program v ružovom nad týmto príkladom. Čo vypíše korytnačka? Prečo?
B) Zmeňte hodnotu premennej y na 6. Čo vypíše korytnačka?

Príklad 10.2

Vygenerujte náhodné číslo a uložte ho do premennej. Ak ste vygenerovali číslo 1, pohnite sa dopredu o 50. Ak ste vygenerovali číslo 2, pohnite sa dopredu o 200. Program spustite viackrát.

Príklad 10.3

Vygenerujte náhodné číslo a uložte ho do premennej. Ak ste vygenerovali číslo 1, nakreslite hviezdičku (môžete použiť funkciu z minulej lekcie). Ak ste vygenerovali číslo 2, nakreslite štvorec.

Elif

Čo ak by sme mali 3 možnosti?

```
if cislo > 10:
        write("platí 1. podmienka")
elif cislo < 10:
        write("platí 2. podmienka")
else:
        write("neplatí ani 1. ani 2. podmienka")
```

Príklad 10.4

Zmeňte predchádzajúci príklad tak, aby okrem hviezdičky a štvorca občas nakreslil iný tvar.

Nepovinná vetva else

Vetva else je nepovinná, nemusíme ju použiť. V takom prípade ak podmienka neplatí, nič sa nestane.

```
if y < 10:
        write(“platí podmienka”)
```

Na nasledujúci príklad dáme žiakom funkciu, ktorá kreslí strom:

```
def strom():
    left(90)
    forward(25)
    right(90)
    forward(25)
    for i in range(2):
        left(120)
        forward(50)
    left(120)
    forward(25)
    right(90)
    forward(25)
    left(90)
```

Príklad 10.5

Posuňte korytnačku dopredu o 50. Vygenerujte náhodné číslo od 1 do 2. Ak ste vygenerovali 1, nakreslite strom. Opakujte viackrát.

Príklad 10.6

Presuňte korytnačku na náhodné miesto. Ak y > 0, nakreslite hviezdičku. Inak nakreslite strom. Opakujte viackrát.