

Cvičenie 2

Inštrukcie:

- Vypracujte všetky úlohy. Na cvičení sa pokúste vypracovať čo najviac úloh a úlohy, ktoré nestihnete na cvičení, vypracujte doma.
- **POZOR!** Všímajte si, či má funkcia, ktorú máte definovať, niečo vracať alebo či má niečo vypisovať! V prípade, že má funkcia niečo vrátiť, musí v nej byť použitý príkaz **return**.
- **V prípade, že sa na niektorej úlohe zaseknete, neviete, čo máte robiť, alebo máte otázky na úlohy z predošlých týždňov, pýtajte sa cvičiacich!**

Časť 1: funkcie

Úloha č. 1

Definujte funkciu *tretia_mocnina(a)*, ktorá má 1 vstupný parameter, číslo *a*. Funkcia pre vstupný parameter *a* **vypíše na obrazovku** tretiu mocninu parametra *a*.

Ukážky vstupov a výstupov:

Volanie funkcie so vstupným argumentom hodnoty 1, t.j. volanie *tretia_mocnina(1)*, **vypíše** hodnotu 1, pretože $1*1*1 = 1$.

Volanie funkcie so vstupným argumentom hodnoty 2, t.j. volanie *tretia_mocnina(2)*, **vypíše** hodnotu 8, pretože $2*2*2 = 8$.

Volanie funkcie so vstupným argumentom hodnoty 3, t.j. volanie *tretia_mocnina(3)*, **vypíše** hodnotu 27, pretože $3*3*3 = 27$.

Riešenie:

```
def tretia_mocnina(a):  
    print(a*a*a)
```

Úloha č. 2

Definujte funkciu *tretia_mocnina(a)*, ktorá má 1 vstupný parameter, číslo *a*. Funkcia pre vstupný parameter *a* **vráti** tretiu mocninu parametra *a*.

Ukážky vstupov a výstupov:

Volanie funkcie so vstupným argumentom hodnoty 1, t.j. volanie *tretia_mocnina(1)*, **vráti** hodnotu 1, pretože $1*1*1 = 1$.

Volanie funkcie so vstupným argumentom hodnoty 2, t.j. volanie *tretia_mocnina(2)*, **vráti** hodnotu 8, pretože $2*2*2 = 8$.

Volanie funkcie so vstupným argumentom hodnoty 3, t.j. volanie *tretia_mocnina(3)*, **vráti** hodnotu 27, pretože $3*3*3 = 27$.

Riešenie:

```
def tretia_mocnina(a):  
    return(a*a*a)
```

Komentár:

POZORNE si všimnite rozdiel medzi úlohou č. 1 a úlohou č. 2!!! Pozorne si všimnite, aký je rozdiel medzi tým, že funkcia **vypíše** a že funkcia **vráti** hodnotu!

Dobre si pozrite nasledovné 2 ukážky volaní funkcií z prvej a druhej úlohy a aké hodnoty obe funkcie vrátili! Ak nerozumiete, **pýtajte sa!**

Takto to funguje pre riešenie z úlohy č. 1:

```
cvicenie.py - Z:/Documents/prednaska/cvicenie.py (3.12.6)
File Edit Format Run Options Window Help
def tretia_mocnina_ulohal(a):
    print(a*a*a)
navratova_hodnota = tretia_mocnina_ulohal(2)
print("Funkcia vratila hodnotu: ",navratova_hodnota)
```

Tento print() vypíše na obrazovku tretiu mocninu parametra

Keďže funkcia tretia_mocnina_ulohal NEMÁ return príkaz so žiadnou hodnotou, implicitne funkcia vráti hodnotu None.

8 Funkcia vratila hodnotu: None

Teda tento výpis "8" pochádza z volania funkcie "tretia_mocnina_ulohal"

Preto výpis návratovej hodnoty uloženej v premennej "navratova_hodnota" vypísal na obrazovku hodnotu None

A takto to funguje pre riešenie z úlohy č. 2:

```
cvicenie.py - Z:/Documents/prednaska/cvicenie.py (3.12.6)
File Edit Format Run Options Window Help
def tretia_mocnina_uloha2(a):
    return(a*a*a)
navratova_hodnota = tretia_mocnina_uloha2(2)
print("Funkcia vratila hodnotu: ",navratova_hodnota)
```

Táto funkcia nikde neobsahuje žiaden print(), preto táto funkcia NIČ na obrazovku nevypíše!

Ln: 4 Col: 41

Funkcia vratila hodnotu: 8

Vidíme, že JEDINÝ výpis na obrazovke je výpis návratovej hodnoty funkcie, teda hodnoty uloženej v premennej "navratova_hodnota", ktorá sa tam uloží POTOM, ako sa vykoná funkcia "tretia_mocnina_uloha2" a vypočítanú hodnotu vráti ako svoju návratovú hodnotu, ktorá sa následne uloží do premennej "navratova_hodnota"

Úloha č. 3:

Definujte funkciu *priemer_troch(a,b,c)*, ktorá má 3 vstupné parametre: čísla *a*, *b*, *c*. Funkcia **vypíše** aritmetický priemer týchto 3 parametrov, teda hodnotu $(a+b+c)/3$.

Vstupy a výstupy:

Volanie funkcie *priemer_troch(1,2,3)* **vypíše** hodnotu 2.0, pretože $(1+2+3)/3 = 2.0$.

Volanie funkcie *priemer_troch(1,2,4.5)* **vypíše** hodnotu 2.5, pretože $(1+2+4.5)/3 = 2.5$.

Úloha č. 4:

Upravte funkciu *priemer_troch(a,b,c)* z predošlej úlohy tak, aby **vrátila** aritmetický priemer týchto 3 parametrov, teda hodnotu $(a+b+c)/3$.

Vstupy a výstupy:

Volanie funkcie *priemer_troch(1,2,3)* **vráti** hodnotu 2.0, pretože $(1+2+3)/3 = 2.0$.

Volanie funkcie *priemer_troch(1,2,4.5)* **vráti** hodnotu 2.5, pretože $(1+2+4.5)/3 = 2.5$.

Časť 2: funkcie a for cykly

V úlohách číslo 1 – 4 sa budeme odvolávať na nasledovnú funkciu *vypis(n)*

```
1 def vypis (n) :  
2     for i in range (n) :  
3         print (i)
```

Funkcia *vypis(n)* vypíše na obrazovku čísla od 0 po $n-1$, kde n je jej vstupný parameter.

Úloha č. 1

Upravte funkciu *vypis(n)* na obrázku tak, aby **vypísala** na obrazovku čísla od 1 po n . Túto úpravu urobte tak, že zmeníte **len riadok číslo 3**, t.j. upravíte príkaz *print(i)* vhodným spôsobom!

Vstupy a výstupy:

Volanie *vypis(10)* vypíše na obrazovku čísla 1 2 3 4 5 6 7 8 9 10, každé na nový riadok.

Úloha č. 2

Upravte funkciu *vypis(n)* na obrázku tak, aby **vypísala na obrazovku** prvých n čísiel väčších ako 100. Túto úpravu urobte tak, že zmeníte **len riadok číslo 3**, t.j. upravíte príkaz *print(i)* vhodným spôsobom!

Vstupy a výstupy:

Volanie *vypis(3)* vypíše na obrazovku čísla 101 102 103, každé na nový riadok.

Úloha č. 3

Upravte funkciu *vypis(n)* na obrázku tak, aby **vypísala na obrazovku** prvých n párnych čísiel, začínajúc dvojkou. Túto úpravu urobte tak, že zmeníte **len riadok číslo 3**, t.j. upravíte príkaz *print(i)* vhodným spôsobom!

Vstupy a výstupy:

Volanie *vypis(1)* vypíše na obrazovku číslo 2.

Volanie *vypis(5)* vypíše na obrazovku čísla 2 4 6 8 10, každé na nový riadok.

Volanie *vypis(10)* vypíše na obrazovku čísla 2 4 6 8 10 12 14 16 18 20, každé na nový riadok.

Úloha č. 4

Upravte funkciu *vypis(n)* na obrázku tak, aby **vypísala na obrazovku** klesajúcu postupnosť čísiel od n po 1. Túto úpravu urobte tak, že zmeníte **len riadok číslo 3**, t.j. upravíte príkaz *print(i)* vhodným spôsobom!

Vstupy a výstupy:

Volanie *vypis(5)* vypíše na obrazovku čísla 5 4 3 2 1, každé na nový riadok.

Volanie *vypis(10)* vypíše na obrazovku čísla 10 9 8 7 6 5 4 3 2 1, každé na nový riadok.

Úlohy č. 1-4 RELOADED

Vypracujte znovu úlohy č. 1-4 ale tentokrát ponechajte riadok č. 3 ako `print(i)` a vyriešte úlohy vhodnou úpravou riadku č. 2, t.j. vhodnou zmenou argumentov vo funkcii `range()`.

Na prednáške sme si hovorili, že `range(n)` vráti postupnosť čísiel $0, 1, \dots, n-1$. Avšak `range()` má aj viacero iných použití:

1. ak do funkcie `range()` vložíme 2 argumenty: `range(start, stop)`, tak `range(start, stop)` vráti postupnosť čísiel **od start po stop-1**. Teda napríklad `range(2,6)` vráti postupnosť čísiel 2,3,4,5.
2. ak do funkcie `range()` vložíme 3 argumenty: `range(start, stop, step)`, tak `range(start, stop, step)` vráti postupnosť čísiel **od start po stop-1 s krokom step**. Teda napríklad `range(3,11,2)` vráti postupnosť čísiel 3, 5, 7, 9.

Úloha č. 5

Definujte funkciu `sucet_stvorcov(n)` so vstupným parametrom n , ktorá **vráti súčet** $1^2 + 2^2 + 3^2 + \dots + n^2$ pričom tento súčet vypočítate pomocou **for-cyklu!**

Vstupy a výstupy:

Volanie `sucet_stvorcov(3)` **vráti** hodnotu 14, pretože $14 = 1^2 + 2^2 + 3^2$

Volanie `sucet_stvorcov(5)` **vráti** hodnotu 55, pretože $55 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$

Úloha č. 6

Definujte funkciu `aritmeticka_postupnost(a0,d,N)`, ktorá má 3 vstupné parametre a_0 , d a N , v tomto poradí. Funkcia **vypíše** prvých N členov aritmetickej postupnosti, ktorej prvý člen má hodnotu a_0 a diferenciu postupnosti je daná parametrom d . Pri riešení úlohy použite for-cykklus.

Hint: Ak nevíete, čo je to aritmetická postupnosť, môžete sa to dočítať napr. tu: https://sk.wikipedia.org/wiki/Aritmetick%C3%A1_postupnos%C5%A5

V skratke, aritmetická postupnosť je postupnosť čísiel, v ktorej je člen postupnosti rovný súčtu predošlého člena postupnosti a diferencie d . Ak by a_i a a_{i+1} boli 2 po sebe idúce členy postupnosti, potom $a_{i+1} = a_i + d$

Vstupy a výstupy:

Volanie `aritmeticka_postupnost(1,2,5)` **vypíše čísla** 1 3 5 7 9 (každé na nový riadok).

Zdôvodnenie: Argumenty 1,2,5 predstavujú hodnoty parametrov $a_0 = 1$, $d = 2$, $N = 5$. Teda prvý člen postupnosti je $a_0 = 1$, členy postupnosti sa od seba líšia o diferenciu $d = 2$ a chceme vypísať $N = 5$ členov postupnosti. Teda sa vypíšu čísla:

1 (lebo prvý člen postupnosti je $a_0 = 1$)

3 (lebo druhý člen postupnosti je prvý člen + diferenciu = $1 + 2 = 3$)

5 (lebo tretí člen postupnosti je druhý člen + diferenciu = $3 + 2 = 5$)

7 (lebo štvrtý člen postupnosti je tretí člen + diferenciu = $5 + 2 = 7$)

9 (lebo piaty člen postupnosti je štvrtý člen + diferenciu = $7 + 2 = 9$)

Keďže $N = 5$, vypíšeme len 5 členov postupnosti.

Volanie `aritmeticka_postupnost(5,-3,4)` **vypíše čísla** 5 2 -1 -4 (každé na nový riadok).

Zdôvodnenie: Argumenty 5,-3,4 predstavujú hodnoty parametrov $a_0 = 5$, $d = -3$, $N = 4$. Teda prvý člen postupnosti je $a_0 = 5$, členy postupnosti sa od seba líšia o diferenciu $d = -3$ a chceme vypísať $N = 4$ členov postupnosti. Teda sa vypíšu čísla:

5 (lebo prvý člen postupnosti je $a_0 = 5$)

2 (lebo druhý člen postupnosti je prvý člen + diferenciu = $5 + -3 = 2$)

-1 (lebo tretí člen postupnosti je druhý člen + diferenciu = $2 + -3 = -1$)

-4 (lebo štvrtý člen postupnosti je tretí člen + diferenciu = $-1 + -3 = -4$)

Keďže $N = 4$, vypíšeme len 4 členy postupnosti.

Úloha č. 7

Definujte funkciu *geometricka_postupnost(a0,q,N)*, ktorá má 3 vstupné parametre a_0 , q a N , v tomto poradí. Funkcia **vypíše** prvých N členov geometrickej postupnosti, ktorej prvý člen má hodnotu a_0 a kvocient postupnosti je daný parametrom q . Pri riešení úlohy použite for-cyklus.

Hint: Ak neviete, čo je to geometrická postupnosť, môžete sa to dočítať napr. tu:

https://sk.wikipedia.org/wiki/Geometrick%C3%A1_postupnos%C5%A5

V skratke, geometrická postupnosť je postupnosť čísiel, v ktorej je člen postupnosti rovný súčinu predošlého člena postupnosti a kvocientu q . Ak by a_i a a_{i+1} boli 2 po sebe idúce členy postupnosti, potom $a_{i+1} = a_i * q$

Vstupy a výstupy:

Volanie *geometricka_postupnost(1,2,5)* **vypíše čísla** 1 2 4 8 16 (každé na nový riadok).

Volanie *geometricka_postupnost(2,3,4)* **vypíše čísla** 2 6 18 54 (každé na nový riadok), pretože ak prvý člen postupnosti je 2, potom druhý člen je $2*3 = 6$, tretí člen postupnosti je $6*3 = 18$ a štvrtý člen postupnosti je $18*3 = 54$.

Úloha č. 8

Definujte funkciu *postupny_sucet_geometrickeho_radu(a0,q,N)* so vstupnými parametrami a_0 , q a N , v tomto poradí. Parametre a_0 , q , N znovu predstavujú geometrickú postupnosť, kde a_0 je jej prvý člen, q je jej kvocient a N je počet členov postupnosti, podobne ako v úlohe číslo 7. Tentokrát však naprogramujte funkciu, ktorá postupne vypíše:

- prvý člen postupnosti
- súčet prvých 2 členov postupnosti
- súčet prvých 3 členov postupnosti
- súčet prvých 4 členov postupnosti
- ...
- súčet prvých N členov postupnosti

V implementácii funkcie použite **len jeden for-cyklus!**

Vstupy a výstupy:

Volanie *postupny_sucet_geometrickeho_radu(1,2,5)* **vypíše čísla**

1 (pretože 1 je prvý člen postupnosti)

3 (pretože $3 = 1 + 2$)

7 (pretože $7 = 1 + 2 + 4$)

15 (pretože $15 = 1 + 2 + 4 + 8$)

31 (pretože $31 = 1 + 2 + 4 + 8 + 16$)

Volanie *postupny_sucet_geometrickeho_radu(1,0.5,4)* **vypíše čísla** predstavujúce postupné súčty členov postupnosti: 1, 0.5, 0.25, 0.125, teda:

1 (pretože 1 je prvý člen postupnosti)

1.5 (pretože $1.5 = 1 + 0.5$)

1.75 (pretože $1.75 = 1 + 0.5 + 0.25$)

1.875 (pretože $1.875 = 1 + 0.5 + 0.25 + 0.125$)

Úloha č. 9

Definujte funkciu $grid(n)$ s parametrom n , ktorá **vypíše na obrazovku** mriežku s n riadkami a n stĺpcami podľa obrázkov nižšie.

Pre $n = 2$ (mriežka s 2 „riadkami“ a 2 „stĺpcami“) by mriežka vyzerala nasledovne:

```
+ - - - + - - - +
|       |       |
+ - - - + - - - +
|       |       |
+ - - - + - - - +
```

Pre $n = 3$ (mriežka s 3 „riadkami“ a 3 „stĺpcami“) by mriežka vyzerala nasledovne:

```
+ - - - + - - - + - - - +
|       |       |       |
+ - - - + - - - + - - - +
|       |       |       |
+ - - - + - - - + - - - +
|       |       |       |
+ - - - + - - - + - - - +
```

Hint: funkcia `print`, ktorú sme používali doteraz, vždy vypisuje na nový riadok, t.j. po každom volaní funkcie `print` bude nasledovný výpis na novom riadku. Toto nastavenie sa dá zmeniť pomocou parametra `end` vo volaní funkcie `print`, keďže parameter `end` dovoľuje nastaviť iný tzv. *ukončovací znak výpisu*. Napríklad v nasledovnom príklade

```
*cvicenie.py - Z:/Documents/prednaska/cvicenie.py (3.12.6)*
File Edit Format Run Options Window Help
1 print ('+', end='')
2 print ('-')
```

+ -

vidíme, že ak nastavíme ukončovací znak na „nič“, t.j. `end=""`, po vypísaní znaku `+` bude aj ďalší výpis na tom istom riadku! Alebo v príklade:

```
*cvicenie.py - Z:/Documents/prednaska/cvicenie.py (3.12.6)*
File Edit Format Run Options Window Help
1 print ('+', end=', ')
2 print ('-')
```

+, -

vidíme, že ak nastavíme ukončovací znak `end=', '`, teda na čiarku, tak po výpise `+` sa pridala na obrazovku aj čiarka.

Úloha č. 10

Definujte funkciu `vyasob(a,b)` s 2 vstupnými parametrami, a a b . Môžete predpokladať, že a aj b sú kladné celé čísla. Funkcia **vráti** súčin $a*b$. Funkciu implementujte **bez použitia operátora $*$, t.j. bez násobenia!**

Hint: Zamyslite sa, ako je možné realizovať násobenie len pomocou opakovaného pripočítavania!

Vstupy/výstupy:

Volanie `vyasob(2,5)` vráti hodnotu 10.

Úloha č. 11

Definujte funkciu `tretia_mocnina(a)` s 1 vstupným parametrom, a . Môžete predpokladať, že a je kladné celé číslo. Funkcia **vráti** tretiu mocninu čísla a . Funkciu implementujte **bez použitia operátora $*$ a bez použitia operátora $**$ a použite pri jej implementácii funkciu `vyasob(a,b)` z predošlej úlohy!**

Úloha č. 12

Definujte funkciu `umocni(n,k)` s 2 vstupnými parametrami, n a k . Môžete predpokladať, že n aj k sú kladné celé čísla. Funkcia **vráti** k -tu mocninu čísla n , t.j. n^k . Funkciu implementujte **bez použitia operátora $*$ a bez použitia operátora $**$.**

Hint: Zamyslite sa, ako je možné pri tejto úlohe vychádzať z úloh č. 10 a č. 11, resp. použiť funkcie vytvorené v úlohách 10 a 11.

Vstupy/výstupy:

Volanie `umocni(1,5)` vráti hodnotu 1.

Volanie `umocni(2,5)` vráti hodnotu 32.

Volanie `umocni(3,4)` vráti hodnotu 81.

Úloha č. 13

Definujte funkciu `mala_nasobilka()` bez vstupných parametrov, ktorá vypíše tabuľku malej násobilky, spolu s menami riadkov a stĺpcov pre čísla od 1 po 10. Očakávaný výsledok by mal vyzerať nasledovne:

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Hint:

1) spomeňte si, ako ste v úlohe č. 9 používali parameter `end` vo funkcii `print`. Pekné odsadenie textu môžete dostať pomocou ukončovacieho znaku výpisu `\t` (tabulátor – pre jeho použitie je nutné ho uviesť do úvodzoviek/apostrofov).

2) Pre vypracovanie úlohy bude vhodné použiť tzv. cyklus-v-cykle! Teda vložiť jeden `for`-cyklus do iného `for`-cyklu!

Úloha č. 14

Definujte funkciu *od_jedna_po_n(n)* so vstupným parametrom n . Môžete predpokladať, že n je kladné celé číslo. Funkcia **vypíše** postupne na samostatné riadky čísla od 1 po n , s krokom +1, teda na prvý riadok vypíše 1, na druhý riadok 1 2, na tretí riadok 1 2 3, atď.

Vstup/výstup:

Volanie *od_jedna_po_n(5)* vypíše:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```