Cvicenie 2

Instrukcie:

- Vypracujte vsetky ulohy. Na cviceni sa pokuste vypracovat co najviac uloh a ulohy, ktore nestihnete na cviceni, potom vypracujte doma.
- **POZOR!** Vsimajte si, ci funkcia, ktoru mate definovat, ma nieco vracat alebo ci ma nieco vypisovat! V pripade, ze ma funkcia nieco **vratit,** musi v nej byt pouzity prikaz ***return*!**
- **V pripade, ze sa na niektorej ulohe zaseknete, pytajte sa cviciaceho alebo odborneho praktikanta.**

Cast 1: funkcie

1A.Vytvorte skript, ktory definuje funkciu *tretia_mocnina*, ktora pre vstupny argument *a* **vrati** hodnotu *a*a*a* .
Napriklad: volanie *tretia_mocnina(2)* **vrati** hodnotu 8; volanie *tretia_mocnina(3)* **vrati** hodnotu 27.

1B. Upravte skript z ulohy 1A tak, aby skript pomocou funkcie *tretia_mocnina* vypisal tretiu mocninu cisla 5.

2A.Vytvorte skript, ktory definuje funkciu *priemer_troch*, ktora pre vstupne argumenty *a,b,c* **vrati** hodnotu *(a+b+c)/3.*
Napriklad: volanie *priemer_troch(1,2,3)* **vrati** hodnotu 2.0; volanie *priemer_troch(1, 2, 4.5)* **vrati** hodnotu 2.5

2B. Upravte skript z ulohy 2A tak, aby skript pomocou funkcie *priemer_troch* vypisal priemer cisel 5,6,7.

Cast 2: funkcie a for cykly

V ulohach 1 az 4 sa budem odvolavat na nasledujuci skript:

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

Funkcia vypis(n) definovana v tomto skripte vypise cisla od 0 po n-1.

1. Vytvorte skript, ktory definuje funkciu s parametrom n, ktora **vypise** cisla od 1 po n. Vasa funkcia by mala vyzerat ako funkcia vypis(n) na obrazku vyssie a od funkcie vypis(n) by sa mala lisit iba v riadku cislo 3.
Napriklad: funkcia pre hodnotu parametra *n*=10 **vypise** cisla 1 2 3 4 5 6 7 8 9 10, kazde na novy riadok.

2. Vytvorte skript, ktory definuje funkciu s parametrom n, ktora **vypise** prvych n cisel vacsich ako 100. Vasa funkcia by mala vyzerat ako funkcia vypis(n) na obrazku vyssie a od funkcie vypis(n) by sa mala lisit iba v riadku cislo 3.
Napriklad: funkcia pre hodnotu parametra n=3 vypise cisla 101,102 a 103, kazde na novy riadok.

3. Vytvorte skript, ktory definuje funkciu s parametrom n, ktora **vypise** prvych n parnych cisel (zacinajuc dvojkou). Urobte to bez pouzitia podmienok (if-ov). Vasa funkcia by mala vyzerat ako funkcia vypis(n) na obrazku vyssie a od funkcie vypis(n) by sa mala lisit iba v riadku cislo 3.

Napriklad: funkcia pre hodnotu parametra *n*=1 **vypise** cislo 2. Funkcia pre hodnotu parametra *n=5* **vypise** cisla 2 4 6 8 10, kazde na novy riadok. Funkcia pre hodnotu parametra *n*=10 **vypise** cisla 2 4 6 8 10 12 14 16 18 20, kazde na novy riadok.

4. Vytvorte skript, ktory definuje funkciu s parametrom n, ktora **vypise** klesajucu postupnost cisel od n po 1. Vasa funkcia by mala vyzerat ako funkcia vypis(n) na obrazku vyssie a od funkcie vypis(n) by sa mala lisit iba v riadku cislo 3.
Napriklad: funkcia pre hodnotu parametra *n*=1 **vypise** cislo 1. Funkcia pre hodnotu parametra *n*=5 **vypise** cisla 5 4 3 2 1, kazde na novy riadok. Funkcia pre hodnotu parametra *n*=10 **vypise** cisla 10 9 8 7 6 5 4 3 2 1, kazde na novy riadok.

5. Vytvorte skript, ktory definuje funkciu s parametrom n, ktora **vrati** sucet $1^2+2^2+3^2+\ldots+n^2$.
(Pre tento sucet sice existuje vzorec, ale od Vas teraz chcem, aby ste to zratali pomocou for-cyklus.)
Napriklad: funkcia pre hodnotu parametra *n=1* **vrati** cislo 1, pre hodnotu parametra *n*=3 **vrati** cislo 14, pre hodnotu parametra *n=5* **vrati** cislo 55.

6. Definujte funkciu, ktora **vypise** prvych N členov **aritmetickej postupnosti** a_{i+1} = a_{i} + d na základe parametrov a_{0}, d a N. Cize na zaklade parametrov a_{0}, d a N funkcia vypise postupnost cisel a_{0}, a_{1}, a_{2}, ... , a_{N-1}, v ktorej plati, ze a_{i+1} = a_{i} + d.
Napriklad: pre hodnoty parametrov a_{0}=0, d=3, N=5, funkcia **vypise** 0,3,6,9,12. Pre hodnoty parametrov a_{0}=5, d = -3, N=4, funkcia **vypise** 5, 2, -1, -4.

7.Teraz vyrieste ulohu 6 tak, ze v definicii funkcie nepouzijete operaciu * (t.j. operaciu nasobenia).

8. Pozrite si skript *meranie_casu.py*, ktory najdete na webstranke predmetu. V tomto skripte definujem funciu arit, ktora riesi ulohu 6 a vyuziva operaciu *. V skripte takisto definujem funkciu arit2, ktora tiez riesi ulohu 6, ale operaciu * uz nepouziva. V skripte potom porovnavam rychlost tychto funkcii. Spustite skript *meranie_casu.py.* Budete si moct vsimnut, ze funkcia arit2 je rychlejsia ako funkcia arit.

9. Definujte funkciu, ktora **vypise** prvych N členov **geometrickej postupnosti** a_{i+1} = a_{i} *r na základe parametrov a_{0}, r a N. Cize na zaklade parametrov a_{0}, r a N funkcia vypise postupnost cisel a_{0}, a_{1}, a_{2}, ... , a_{N-1}, v ktorej plati, ze a_{i+1} = a_{i} *r.
Pomocka: umocnovanie sa v pythone robi pomocou symbolov ** . Napriklad 2**3 je 8.
Napriklad: pre hodnoty parametrov a_{0}=1, r=2, N=5, funkcia **vypise** 1,2,4,8,16. Pre hodnoty parametrov a_{0}=2, r=3, N=4, funkcia **vypise** 2, 6, 18, 54.

10.Teraz vyrieste ulohu 9 tak, ze v definicii funkcie nepouzijete operaciu umocnenia.

11. Ak sa funcie, ktore ste vytvorili v ulohach 9 a 10, od seba lisia, porovnajte ich rychlost. Na porovnanie rychlosti funkcii sa inspirujte skriptom *meranie_casu.py*, ktory najdete na webstranke predmetu. Pri porovnavani rychlosti mozete funkcie spustit napriklad s parametrami a_{0}=1, r=2, N=50 000. Pri porovnavani rychlosti zakomentujte volania funkcie print! Ak by vam niektory vypocet trval priliz dlho, stlacte Ctrl+C . Vypocet tym prerusite.

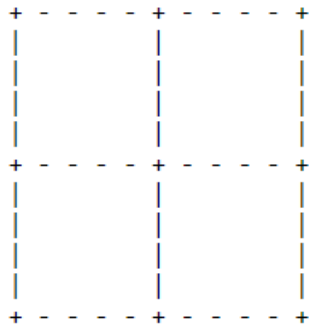
12. Definujte funkciu, ktora **vypise** prvych N členov **geometrickeho radu** na základe parametrov a_{0}, r a N. (**Geometricky rad** je definovany tak, ze i-ty člen radu predstavuje súčet prvých i členov geometrickej postupnosti.) Definujte funkciu tak, aby jej definicia obsahovala iba jeden for cyklus.
Napriklad pre hodnoty parametrov a_{0}=1, r=2, N=5, funkcia **vypise** 1, 3 (=1+2), 7 (=1+2+4), 15 (=1+2+4+8), 31 (=1+2+4+8+16).
Pre hodnoty parametrov a_{0}=2, r=3, N=4 funkcia **vypise** 2, 8, 26, 80.

13. Ak r je v absolutnej hodnote mensie ako 1, tak s rastucim N sa cleny geometrickeho radu blizia k hodnote a_{0}/(1-r). Tomuto hovorime, ze pre r v absolutnej hodnote mensej ako 1 geometricky rad konverguje k hodnote a_{0}/(1-r). Napriklad geometricky rad s parametrami a_{0}=1, r=0.5 konverguje k hodnote 2. Otestujte, ci sa cisla, ktore vypise vasa funkcia z ulohy 12 pre parametre a_{0}=1, r=0.5, N=100, blizia k hodnote 2. Ak nie, potom mate niekde chybu.
Napriklad: pre hodnoty parametrov a_{0} = 2, r = 0.5,N=100 sa hodnoty blizia k hodnote 4. Pre hodnoty parametrov a_{0} = 2, r = 0.75, N=100 sa hodnoty blizia k hodnote 8.

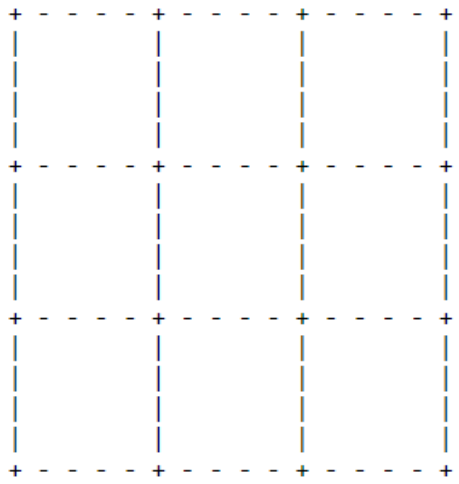
14.Teraz vyrieste ulohu 12 tak, ze v definicii funkcie nepouzijete operaciu umocnenia. Definicia by stale mala obsahovat iba jeden for cyklus.

15. Porovnajte rychlost vasich funkcii z uloh 12 a 14 a rychlost funkcie, ktoru najdete v skripte *neefektivny_skript.py* na webstranke predmetu. Funkcia v tomto skripte tiez riesi ulohu 12, ale robi to pomocou dvoch do seba vnorenych for cyklov. Na porovnanie rychlosti funkcii sa opat inspirujte skriptom *meranie_casu.py*, ktory najdete na webstranke predmetu. Ak budete porovnavat s funkciou zo skriptu *neefektivny_skript.py* , mozete napriklad pouzit parametre a_{0}=1, r=2, N=1000. Ak budete porovnavat iba funkcie z uloh 12 a 14, potom mozete napriklad pouzit parametre a_{0}=1, r=2, N=50000. Pri porovnavani rychlosti zakomentujte volania funkcie print! Ak by vam niektory vypocet trval priliz dlho, stlacte Ctrl+C . Vypocet tym prerusite.

16. Vytvorte skript, ktory pomocou for-cyklov a funkcie print definuje funkciu *grid* s parametrom *n*, ktora do konzoly vykresli mriezku s *n* riadkami a *n* stlpcami. *Vsimnite si pomocku uvedenu nizsie!* Mriezka s dvomi riadkami a dvomi stlpcami (teda n=2) by mala vyzerat nasledovne:



Pre n=3 by mriezka zase mala vyzerat nasledovne:



Pomocka: funkcia print automaticky vypisuje na novy riadok. Toto nastavenie sa ale da zmenit zmenou hodnoty parametra end. Napriklad prikazy:

print('+', end=' ')
print('-')

vypisu:
+ -

Dalsi priklad:
Prikazy:

print('+', end=',')
print('-')

vypisu:
+,-

17. Definujte funkciu *vynasob(a,b)*, ktora vrati sucin kladnych celych cisiel *a* a *b*. Funkciu implementujte bez operatora nasobenia *.
Napriklad volanie *vynasob(2,5)* vrati hodnotu 10.
Poznamka: Vo vseobecnosti nie je potrebne sa operatoru nasobenia vyhybat. V tejto ulohe sa operatoru nasobenia vyhybame iba preto, aby ste si viac potrapili hlavicky :)

18. Definujte funkciu *tretia_mocnina(a)*, ktora vrati tretiu mocninu kladneho celeho cisla *a*. Funkciu implementujte bez operatora nasobenia * alebo operatora umocnenia **.
Napriklad volanie *tretia_mocnina(3)* vrati hodnotu 27.
Poznamka: Vo vseobecnosti nie je potrebne sa operatorom nasobenia a umocnenia vyhybat. V tejto ulohe sa im vyhybame iba preto, aby ste si viac potrapili hlavicky :)

18b. V pripade, ze ste tak neurobili, vyuzite v implementacii funkcie *tretia_mocnina(a)* funkciu *vynasob(a,b)*.

19. Definujte funkciu *mocnina(n,k)*, ktora vrati *k*-tu mocninu cisla *n*, t.j. hodnotu n^k. Predpokladajte, ze *n* a *k* su kladne cele cisla. Pri implementacii funkcie nepouzite operatory nasobenia * ani umocnenia **.
Poznamka: Vo vseobecnosti nie je potrebne sa operatorom nasobenia a umocnenia vyhybat. V tejto ulohe sa im vyhybame iba preto, aby ste si viac potrapili hlavicky :)

20. Definujte funkciu *mala_nasobilka()*, ktora vypise tabulku malej nasobilky, spolu s menami riadkov a stlpcov pre cisla od 1 po 10. Ocakavany vysledok by mal vyzerat 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
```

21. Definujte funkciu *od_jedna_po_n(n)*, ktora pre kladne cele cislo *n* vypise postupne na samostatne riadky cisla od 1 po *n,* s krokom +1. Na prvom riadku bude len cislo 1, na druhom riadku cisla 1 2, na tretom riadku 1 2 3 a tak dalej, az na *n*-tom riadku cisla 1 2 3 … n. Napriklad volanie *od_jedna_po_n(5)* vypise postupne:

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