Domača vaja 1D

Še nekaj nalog, ki smo si jih tudi izposodili iz predmeta Programiranje 1 z visokošolskega študija na FRI (pri prof. dr. J. Demšarju).

Ne pozabite naloge preveriti s testnimi programi, ki jih dobite tukaj.

Kvadratna enačba (`kvadratna_enacba.py`)

Izračunaj vse realne rešitve kvadratne enačbe `ax^2 + bx + c = 0`, podane z argumenti a, b in c. Vaš program naj se obnaša kot je prikazano spodaj.

```
Vpiši a: 1
Vpiši b: 2
Vpiši c: 1
Enačba ima eno realno rešitev: -1.0

Vpiši a: 1
Vpiši b: 2
Vpiši c: 0
Enačba ima dve realni rešitvi: 0.0 in -2.0

Vpiši a: 1
Vpiši b: 2
Vpiši c: 2
Enačba nima realnih rešitev.
```

Koren števila `n` izračunate tako, da na vrhu programa dodate vrstico `import math`, nato pa pokličete funkcijo `math.sqrt(n)`.

```
import math

a = int(input('Vpiši a: '))
b = int(input('Vpiši b: '))
c = int(input('Vpiši c: '))

d = b**2 - 4 * a * c
if d < 0:
    print('Enačba nima realnih rešitev.')
elif d == 0:
    x = -b / (2 * a)
    print('Enačba ima eno realno rešitev:', x)
else:
    x1 = (-b + math.sqrt(d)) / (2 * a)
    x2 = (-b - math.sqrt(d)) / (2 * a)
    print('Enačba ima dve realni rešitvi:', x1, 'in', x2)
```

Poštevanka števila 7 (`postevanka_stevila_7.py`)

Poštevanko števila 7 se igra tako, da igralci, ki sedijo v krogu, kvadratu ali kakem drugem primernem (po potrebi nepravilnem, a po možnosti konveksnem) poligonu po vrsti govorijo števila od ena do neskončno, pri čemer morajo namesto vseh števil, ki so deljiva s 7 ali pa vsebujejo števko 7, reči BUM. Igralec, ki se zmoti, izpade in štetje se začne od začetka. Igra torej teče tako:

```
1 2 3 4 5 6 BUM 8 9 10 11 12 13 BUM 15 16 BUM 18 19 20 BUM 22 23 24 25 26
BUM BUM 29 ...
```

Napiši program, ki izpiše tole zaporedje do vključno z 100.

Klasična rešitev. Pogoj `i % 10 == 7` poskrbi za števila 7, 17, ..., 97; pogoj `i // 10 == 7` pa za 70, 71, ..., 79.

```
for i in range(1, 101):
    if i % 7 == 0 or i % 10 == 7 or i // 10 == 7:
        print('BUM', end=' ')
    else:
        print(i, end=' ')
```

Krajša rešitev:

```
for i in range(1, 101):
    if i % 7 == 0 or '7' in str(i):
        print('BUM', end=' ')
    else:
        print(i, end=' ')
```

Vsote (`vsote.py`)

Napiši program, ki izračuna vsoto prvih n števil. Če uporabnik vpiše, recimo, 7, morda program izpisati 28, saj je 1+2+3+4+5+6+7 enako 28.

Testi za to nalogo so razdeljeni na lažje in težje primere. Za potrebe tega predmeta je dovolj, da vaš program prestane lažje testne primere.

```
Vpiši število: 1
1

Vpiši število: 7
28
```

Lažjo nalogo rešimo z zanko

```
n = int(input('Vpiši število: '))

vsota = 0
for i in range(1, n + 1):
    vsota += i
print(vsota)
```

težjo pa tako, da malo razmislimo. Vzemimo vsoto prvih desetih števil, torej 1 + 2 + ... + 9 + 10. Če seštejemo prvo in zadnje število dobimo 11, če seštejemo drugo in predzadnje število dobimo spet 11, itd. Ker je takih parov 5, lahko vsoto prvih desetih števil izračunamo kot 5 * 11.

```
n = int(input('Vpiši število: '))
print((n + 1) * n // 2)
```

Kvadrati (`kvadrati.py`)

Napiši program, ki ugotovi, ali je dano število kvadrat. Tako je, na primer, 25 kvadrat, saj je enako 5 * 2. 27 pa ni kvadrat.

Testi za to nalogo so razdeljeni na lažje in težje primere. Za potrebe tega predmeta je dovolj, da vaš program prestane lažje testne primere.

```
Vpiši število: 25
Število je kvadrat

Vpiši število: 27
Število ni kvadrat
```

Lažjo nalogo rešimo z zanko

```
n = int(input('Vpiši število: '))

je_kvadrat = False
for i in range(n + 1):
```

```
    if i**2 == n:
        je_kvadrat = True

if je_kvadrat:
    print('Število je kvadrat')
else:
    print('Število ni kvadrat')
```

Težjo pa z binarnim iskanjem

```
n = int(input('Vpiši število: '))

lo = 0
hi = n
while lo < hi:
    mid = (lo + hi) // 2
    if mid**2 < n:
        lo = mid + 1
    else:
        hi = mid

if lo**2 == n:
    print('Število je kvadrat')
else:
    print('Število ni kvadrat')
```

Kocke (`kocke.py`)

V trgovini s kockami pakirajo kocke v škatle kvadratne oblike. Imajo škatle širine 1, 2, 3, 4, 5... in tako naprej; v te škatle gre 1, 4, 9, 16, 25 ... kock. Če stranka naroči, npr. 20 kock, potrebujemo škatlo širine 5. Napiši program, ki mu uporabnik vpiše število naročenih kock, program pa izpiše potrebno velikost škatle.

Testi za to nalogo so razdeljeni na lažje in težje primere. Za potrebe tega predmeta je dovolj, da vaš program prestane lažje testne primere.

```
Vpiši število kock: 20
Potrebujemo škatlo širine 5
```

Rešitev lažje naloge

```
n = int(input('Vpiši število kock: '))

for i in range(n + 1):
    if i**2 >= n:
        print('Potrebujemo škatlo širine', i)
        break
```

Težjo lahko zopet rešimo z binarnim iskanjem

```
n = int(input('Vpiši število kock: '))

lo = 0
hi = n
while lo < hi:
    mid = (lo + hi) // 2
    if mid**2 < n:
        lo = mid + 1
    else:
        hi = mid

print('Potrebujemo škatlo širine', lo)
```

Kocke z luknjo (`kocke_z_luknjo.py`)

Napiši podoben program kot v prejšnji nalogi, ki pa naj poleg tega pove še, za koliko kock prostora je še ostalo v škatli.

Testi za to nalogo so razdeljeni na lažje in težje primere. Za potrebe tega predmeta je dovolj, da vaš program prestane lažje testne primere.

```
Vpiši število kock: 20
Potrebujemo škatlo širine 5 v kateri je prostora še za 5 kock
```

Naloga je zelo podobna prejšnji.

Lažja:

```
n = int(input('Vpiši število kock: '))

for i in range(n + 1):
    if i**2 >= n:
        print('Potrebujemo škatlo širine', i, 'v kateri je prostora še za',
i**2 - n, 'kock')
        break
```

Težja:

```
n = int(input('Vpiši število kock: '))

lo = 0
hi = n
while lo < hi:
    mid = (lo + hi) // 2
    if mid**2 < n:
        lo = mid + 1
    else:
        hi = mid

print('Potrebujemo škatlo širine', lo, 'v kateri je prostora še za', lo**2
- n, 'kock')
```

Delitelji (`delitelji.py`)

Napiši program, ki izpiše vse delitelje danega števila.

```
Vpiši število: 18
1
2
3
6
9
18
n = int(input('Vpiši število: '))

for i in range(1, n + 1):
    if n % i == 0:
        print(i)
```

Trikotnik iz zvezdic (`trikotnik_iz_zvezdic.py`)

Programerska tradicija narekuje, da je prvi program, ki ga napišemo v novem programskem jeziku, program, ki izpiše `"Hello world"`. Žal je za to že prepozno.

Nadalje se spodobi, da je eden prvih programov, ki ga napišemo z uporabo zank, program, ki nariše trikotnik iz zvezdic. Držimo se vsaj tega in napišimo program, ki vpraša uporabnika za višino trikotnika, nato pa izpiše takšen trikotnik iz zvezdic.

```
Vpiši višino: 4
*
```

```
**
***
****
```

V Pythonu si bomo olajšali delo, če vemo, da lahko niz pomnožimo s številom. Tri zvezdice dobimo tako, da niz `'*'` pomnožimo s 3, torej `'*' * 3`.

```
n = int(input('Vpiši višino: '))

for i in range(1, n + 1):
    print('*' * i)
```

Smrekica (`smrekica.py`)

Napišite program, ki izriše smrekico iz znakov `'*'`.

```
Vpiši višino: 4
   *
  ***
 *****
*******
```

```
n = int(input('Vpiši višino: '))

for i in range(1, n + 1):
    print(' ' * (n - i) + '*' * (2 * i - 1))
```