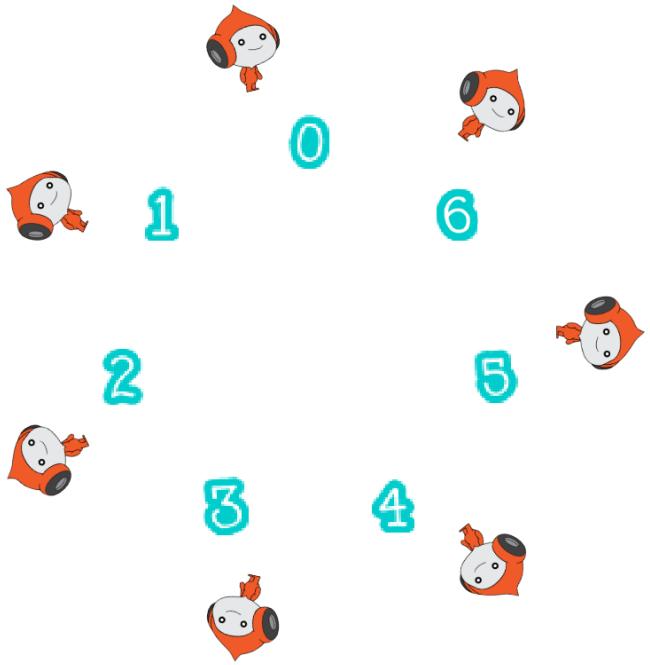
PYTHON

ZVEZA ZA TEHNIČNO KULTURO SLOVENIJE

Osnove 4

Python urejevalnik se nahaja na strani:
`https://www.w3resource.com/python-exercises/python-basic-exercises.php#EDITOR`

1. **Izštevanka** Razporedimo n otrok v krog. Zaradi enostavnosti označimo otroke s številkami od 0 do $n-1$. Vzemimo, da ima izštevanka k besed. Začnemo s štetjem pri otroku, ki smo ga označili s številko 0. Tega ne izločimo v prvem koraku, ampak šele otroka, ki je na k tem mestu. Med obema je $k-2$ mest. Če je $k < n$, potem je ta označen s številko $k-1$. Nadaljujemo z izločevanjem, dokler ne ostane en sam otrok. Program naj izračuna, s katero številko je označen otrok, pri danem n in k, ki ostane zadnji.



2. **Pitagorejske trojice** Pitagorejska trojica je trojica naravnih števil (l, m, n) za katere velja

$$l^2 + m^2 = n^2$$

.

Za dano naravno število s poiščite vse pitagorejske trojice (l, m, n) za katere velja, da je $s = l+m+n$.

Zanimale nas bodo le različne trojice, ki so urejene v naraščajočem vrstnem redu, $l < m < n$.

3. **Priponski zapis** Napišite program za evaluacijo postfiksnega (priponskega) zapisa aritmetičnega izraza.
   - Prebiramo člen za členom. Če naletimo na število, ga shranimo na sklad.
   - Če naletimo na operator, poberemo iz vrha sklada dve števili, izvršimo operacijo in rezultat vrnemo na sklad.
   - Na koncu ostane ne skladu le rezultat.

Predstavitev algoritma

```
((15/(7-(1+1)))*3)-(2+(1+1))      Izraz v infiksnem zapisu
15 7 1 1 + - / 3 * 2 1 1 + + -    Izraz v postfiksnem zapisu

15 7 1 1 + - / 3 * 2 1 1 + + -    sklad
   7 1 1 + - / 3 * 2 1 1 + + -    15
     1 1 + - / 3 * 2 1 1 + + -    15 7
       1 + - / 3 * 2 1 1 + + -    15 7 1
         + - / 3 * 2 1 1 + + -    15 7 1 1
           - / 3 * 2 1 1 + + -    15 7 2
             / 3 * 2 1 1 + + -    15 5
               3 * 2 1 1 + + -    3
                 * 2 1 1 + + -    3 3
                   2 1 1 + + -    9
                     1 1 + + -    9 2
                       1 + + -    9 2 1
                         + + -    9 2 1 1
                           + -    9 2 2
                             -    9 4
                                  5
```

4. **Collatzevo zaporedje** Collatzevo zaporedje je zaporedje naravnih števil, ki ga tvorimo na naslednji način:
   - Izberimo si začetno naravno število $k_0 > 0$,
   - naslednika člena k_n, k_{n+1} določimo takole:

$$k_{n+1} = \begin{cases} 3k_n + 1, & \text{če je } k_n \text{ liho število,} \\ k_n/2, & \text{če je } k_n \text{ sodo število.} \end{cases}.$$

Kolikor vemo, se je do sedaj v vseh primerih zaporedje končalo s ciklom $4, 2, 1, 4, \ldots$. Collatzeva domneva je, da se to zgodi vedno, ne glede na začetno število. V matematiki je to nerešena domneva.

`https://www.youtube.com/watch?v=094y1Z2wpJg`

Prvi jo je leta 1937 postavil Lothar Collatz. Domneva je znana tudi kot domneva $3n + 1$, Ulamova domneva (po Stanislawu Marcinu Ulamu) ali sirakuški problem.

Iz standardnega vhoda preberite naravno število $n > 0$, tvorite Collatzevo zaporedje in ga končajte, ko postane izračunani člen prvič enak 1. Na standardni izhod izpišite število izračunanih členov in največjo vrednost člena v zaporedju.

5. **Benfordov zakon** Benfordov zakon, znan tudi kot zakon prve števke, trdi, v nasprotju s pričakovanjem, da na številnih seznamih številk, ki izhajajo iz resničnih podatkovnih virov, porazdelitev vodilne števke sledi specifični, neenakomerni porazdelitvi.

   `https://www.youtube.com/watch?v=XXjlR2OK1kM`

   Teoretična porazdelitev je naslednja, izražena v procentih:

   `{1: 30.1, 2: 17.6, 3: 12.4, 4; 9.6, 5; 7.9, 6: 6.6, 7: 5.7, 8: 5.1, 9: 4.5}.`

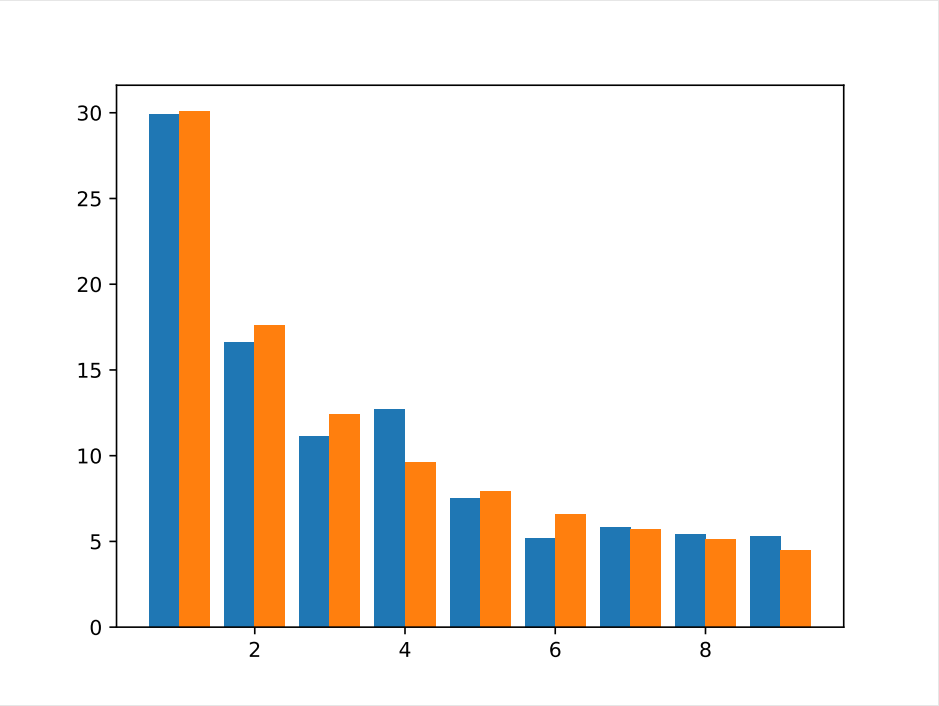
   Podatki se morajo raztezati čez več velikostnih razredov. Na primer dolžina rek na Zemlji, površina držav in podobno.

   Benfordov zakon je uporaben pri preverjanju resničnosti podatkov. Ponarejeni podatki običajno sledijo prepričanju, da morajo biti enakomerno razporejeni.

   Zakon je poimenovan po fiziku Franku Benfordu, ki ga je oblikoval leta 1938, čeprav ga je pred tem leta 1881 omenil astronom Simon Newcomb.

   - Iz standardnega vhoda preberite naravno število n.
   - Sestavite slovar, kjer so ključi števke od 1 do 9, njihove vrednosti postavite na 0.
   - Tvorite Collatzevo zaporedje preberete prvo števko (ključ) tekočega člena in povečate vrednost ključa v začetnem seznamu.
   - Na koncu vrednosti v končnem seznamu delite z vsoto števil v seznamu in pomnožite s 100, da dobite procente.

Slika



Za vrednost $n = 9663$ ima Collatzevo zaporedje 185 členov in največja vrednost je 27114424. Prikaz porazdelitve za vrednost $n = 9663$ modro in teoretične poradeliteve oranžno.

Rešitve 4

Izštevanka

```python
#!/usr/bin/env python3

def odstevanka(n, k):
    nn = list(range(n))
    r = 0
    k = k - 1
    while len(nn) > 1:
        r = (r + k) % len(nn)
        del nn[r]
    return 'Lovi: %d' % nn[0]

if __name__ == "__main__":
    line = input('Vpiši število otrok in število besed -> ')
    n, k = [int(x) for x in line.split()]
    print(odstevanka(n,k))
```

Pitagorejske trojice

```python
#!/usr/bin/env python3

def pythagora_triples1(n):   # Direktno
    trip = set()
    for i in range(1,n):
        for j in range(i,n):
            if i**2+j**2 == (n-j-i)**2:
                trip.add((i,j,n-i-j))
    return trip
```

Priponski zapis

```
#!/usr/bin/env python3

opfunc = {'+': lambda x, y: x + y, '-': lambda x, y: y - x,
          '*': lambda x, y: x * y, '/': lambda x, y: y / x}
         # pri - in / obrnemo vrstni red operandov

def evaluate(izraz):              # evaluacija priponskega izraza
    sklad = []
    izraz = izraz.split()
    for ch in izraz:
        if ch in opfunc:
           sklad.append(opfunc[ch](sklad.pop(),sklad.pop()))
        else:
           sklad.append(float(ch))
    return sklad
```

Collazevo zaporedje

```
#!/usr/bin/env python3
# Se zapiše na datoteko collatz.py za uporabo v naslednji nalogi.
import matplotlib.pyplot as plt

def collatz(n):
    m = n % 2
    if m:
        return 3 * n + 1
    else:
        return n // 2

def list_collatz(n):
    res = [n]
    while True:
        n = collatz(n)
        res.append(n)
        if n == 1:
            break
    return res

if __name__ == '__main__':
    n = int(input('n --> '))
    x = list_collatz(n)
    print(max(x), len(x))
```

Benfordov zakon

```python
#!/usr/bin/env python3
from collatz import *
from random import randint, seed

def first_digit(n):
    assert type(n) == int, print('no go')
    return str(n)[0]

def benford_collatz(n):
    benford_dict = dict([(str(i), 0) for i in range(1,10)])
    for m in yield_collatz(n):
        benford_dict[first_digit(m)] += 1
    return benford_dict

if __name__ == '__main__':
    n = int(input('n -> '))
    bfdic = benford_collatz(n)
    s = sum(benford_dict.values())
    print([int(x/s*100) for x in benford_dict.values()])
```