Tsükkel while

Tsüklit for kasutatakse, kui on ette teada, kui palju kordusi (iteratsioone) on vaja teha.

While'i tsükkel täitus enne mõne sündmuse toimumist ja sel juhul ei ole võimalik iteratsioonide arvu ette hinnata.

While'i tsükli struktuur Pythonis näeb välja selline:

```
    while tingimus:

        koodiplokk
```

In [ ]:

```
i = 0
while i < 5:
    print('Hello, world')
    i += 1
```

```
Hello, world
Hello, world
Hello, world
Hello, world
Hello, world
```

Sellise koodi võib asendada tsükliga for

Näide. Programm loeb numbreid ja leiab nende summa, kuni kasutaja sisestab sõna stop

In [ ]:

```
text = input()
total = 0
print('Numbrite summa ', end='')
while text != 'stop':
    num = int(text)
    print(num, end=' ')
    total = total + num
    text = input()
print(f' võrdub {total}')
```

```
Numbrite summa 1 2 3  võrdub 6
```

Tsükli õigeks toimimiseks on vajalik: **muutuja** text **muutmine tsükli while sees**.

Kui tsükli sees ei muudeta muutujat num (text = input()), siis on võimalik saada nn **lõpmatu tsükkel**, mida tehakse lõputult palju kordi. Lõpmatu tsükkel kordub, kuni programm katkeb. Lõpmatud tsüklid ilmuvad tavaliselt siis, kui programmeerija unustab tsükli sisse kirjutada tarkvarakoodi, mis muudab kontrollitava tingimuse valeks. Enamasti tuleks vältida lõpmatute tsüklite kasutamist.

Tsükkel while on väga **sarnane if tingimusoperaatoriga**. Erinevus seisneb selles, et tingimusoperaatori puhul tehakse vastavat koodiplokki ainult üks kord, samas kui while tsükliga tehakse koodiplokki mitmekordselt. Tsükkel while sai oma nime oma töö iseloomu tõttu: see täidab mingit ülesannet, kuni while (veel) tingimus on tõene.

While'i tsüklit nimetatakse **tsükliks eeltingimustega**, sest tsüklikeha täitmisele eelneb tingimuse kontrollimine (kõigepealt kontrollitakse tingimust ja seejärel täidetakse tsüklikeha).

Tsüklikeha ühekordset täitmist nimetatakse tsükli **iteratsiooniks**.

Tsükkel while ei pruugi kordagi täituda.

Ülesanne 1

Programmi sisend on paarisarvude jada. Jada lõpp on suvaline arv, mis ei jagu 2-ga. Kirjutage programm, mis kuvab kõik selles jadas sisestatud arvud, sisestatud arvude arvu ja summa.

Ülesanne 2

Programmi sisendiks on üks naturaalarv, kauba hind eurodes. Programm peab väljastama maksmiseks minimaalset võimalikku rahatähtede ja müntide arvu (2 ja 1 euro).

Näiteks, 123 – Ekraanile kuvatakse 100 20 2 1

Ülesanne 3

Arva number ära. Kirjutage programm, mis "mõtleb" juhuslikku numbrit vahemikus 1 kuni 10 ja palub kasutajal ära arvata. Prindib, kui teie arv on suurem või väiksem. Printige välja katsete arv.

Ülesanne 4

Sürakuusa hüpotees ütleb, et mis tahes naturaalarvu saab taandada üheks, tehes sellega järgmised toimingud:

a) kui arv on paaris, siis jagada see pooleks,

b) kui see on paaritu, korrutada 3-ga, lisada 1 ja tulemus jagada 2-ga.

Korrake uuesti saadud arvuga tehteid a) või b) olenevalt selle paarsusest. Varem või hiljem muutub see arv võrdseks 1-ga.

Kasutaja sisestab numbri. Printige välja, millised numbrid saadakse tehete a) ja b) sooritamisel.

Arvu numbrite töötlemine

Kasutades while´i tsüklit ja kahte operatsiooni: täisarvu jagamine // ja ühe täisarvu teisega jagamise jäägi leidmise operatsiooni %, saab töödelda suvalise numbrikohaga arvu numbreid.

Näide programmist, mis loeb naturaalarvu (positiivset täisarvu) ja töötleb selle numbreid.

In [ ]:

```
n = int(input())
print(f'Number {n}')
while n != 0:  # kuni arvus on numbrid
    last_digit = n % 10  # saada viimane number
    # viimase numbri töötlemiskood - numbrite kuvamine, numbrite summa, korrutise leidmine, suurima või väikseima numbri leidmine ...
    print(last_digit, end=" ")
    n = n // 10  # viimase numbri kustutamine arvust
```

```
Number 2789
9 8 7 2
```

Ülesanne 5

On antud naturaalarv. Kirjutage programm, mis arvutab:

- selle numbrite summa;
- numbrite arv selles;
- selle esimese ja viimase numbri summa.

Ülesanne 6

On antud naturaalarv. Kirjutage programm, mis otsustab, kas antud arv koosneb samadest numbritest.

Tsükli katkestamise operator break ja iteratsiooni katkestamise operaator continue

Mõnikord tuleb tsükli täitmist enneaegselt katkestada.

Break operaator katkestab lähima for või while'i tsükli. Tsükli katkestamise operaator break võimaldab programme kiirendada, sest üleliigseid iteratsioone ei tehta.

In [ ]:

```
# programm, mis määrab, kas kasutaja sisestatud arv sisaldab numbrit 7.

num = int(input())
number = num
flag = False
while num != 0:
    last_digit = num % 10
    if last_digit == 7:
        flag = True
        break        # katkestame tsükli töö, sest arv sisaldab numbrit 7
    num //= 10

if flag == True:
    print('Arv', number, 'sisaldab numbrit 7')
else:
    print('Arv', number, 'ei sisalda numbrit 7')
```

```
Arv 12347 sisaldab numbrit 7
```

Mõnikord on lõpmatu tsükli abil võimalik programmikood loetavamaks muuta. Lihtsam on tsükli lõpetamine, mis põhineb tsüklikehas olevatel tingimustel, mitte selle pealkirja tingimustel:

while True:

```
    ...

    if tingimus 1:  # tsükli peatamise tingimus

        break
    ...

    if tingimus 2:  # veel üks tsükli peatamise tingimus

        break
    ...

    if условие 3:  # veel üks tsükli peatamise tingimus

        break
    ...
```

Operaator `continue` võimaldab liikuda tsükli for või while järgmisele iteratsioonile, kuni kõigi tsüklikehas olevate käskude täitmiseni.

In [ ]:

```
for i in range(1, 20):
    if i in [7, 17]:
        continue  # liikume järgmisele iteratsioonile
    print(i, end=' ')
```

```
1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 18 19
```