Programmeerimine

3. loeng

Täna loengus

- Loogilised avaldised
- Hargnemisdirektiivid
  - Lihtne if-lause
  - if-else-lause
- Tsüklidirektiivid
  - Eelkontrolliga tsükkel
  - Tsükli kontrolldirektiivid
  - Jadaiterator ja määratud tsüklid

Loogilised avaldised

Loogilised avaldised

- Avaldisi, mille väärtus on tõeväärtustüüpi, nimetatakse loogilisteks avaldisteks.
- Loogilised avaldised koosnevad muutujatest, tõeväärtuskonstantidest, relatsioonilistest- ja loogilistest operaatoritest.
- Tõeväärtustüüp Pythonis:

Tüübi nimi	*bool*
Tõene väärtus	`True`
Väär väärtus	`False`

Relatsioonilised operaatorid

- Relatsioonilised operaatorid võimaldavad võrrelda sama tüüpi väärtusi.

Operaator	Kirjeldus
==	võrdus
!=	mittevõrdus
<	väiksem
<=	väiksem või võrdne
>	suurem
>=	suurem või võrdne

Loogilised avaldised - näiteid

```
a >= 10

k.upper() == 'KOOL'

(a + b > 0) == True

summa != 100
```

Tingimuslaused

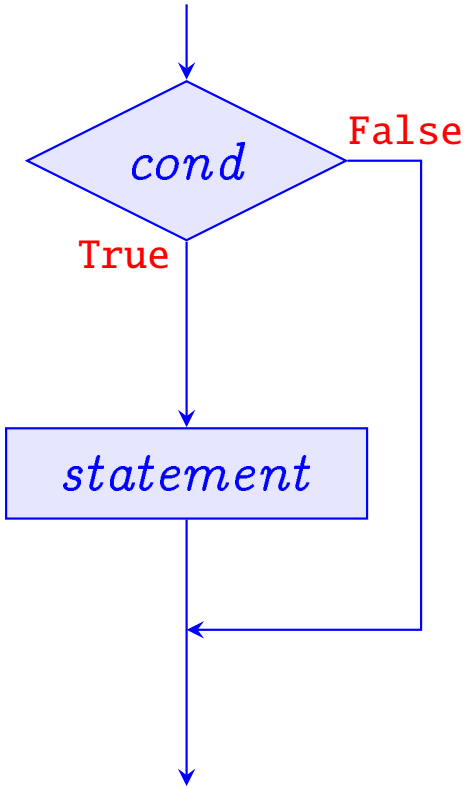
Tingimuslaused

- Lihtne if-lause:

  if *cond*:

  statement

- Tingimus *cond* on loogiline avaldis.
- Lause *statement* täidetakse ainult juhul, kui tingimus *cond* on tõene.
- *statement* võib olla kas üksik lause või lausete plokk.
- Pythonis on plokk määratud taandega.



Tingimuslaused

Näide – kahe arvu järjestamine

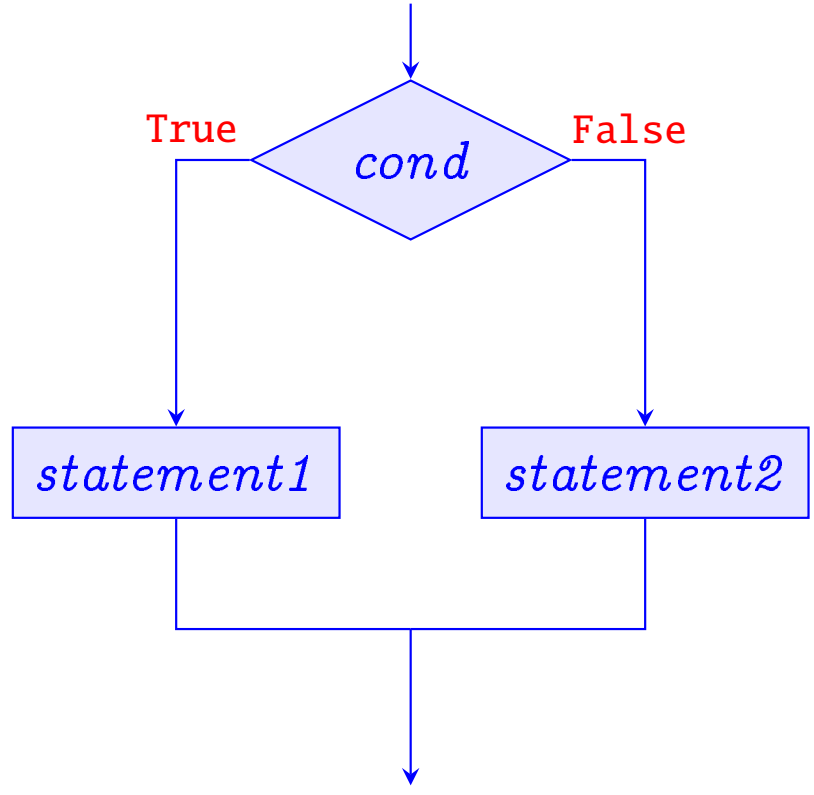
```
print("Sisestage kaks arvu:")
arv1 = int(input())
arv2 = int(input())

# Kui esimene on suurem, siis vahetada
if arv1 > arv2:
    tmp = arv1
    arv1 = arv2
    arv2 = tmp

print("Arvud kasvavalt:", arv1, "ja  siis ", arv2)
```

Tingimuslaused

- if-else-lause:

```
if cond:
    statement1
else:
    statement2
```

- Kui tingimus *cond* on tõene, siis täidetakse lause *statement1*.
- Kui tingimus *cond* on väär, siis täidetakse lause *statement2*.



Tingimuslaused

Näide – kahest arvust maksimaalse leidmine

```
print("Sisestage kaks arvu:")
arv1 = int(input())
arv2 = int(input())

if arv1 < arv2:
    max = arv2
else:          # arv1 >= arv2
    max = arv1

print("Maksimaalne on arv", max)
```

Tingimuslaused

Näide – keskmise arvutamine (v. 1)

```
arv = int(input("Sisestage arv: "))
sum = int(input("Sisestage summa: "))

if arv > 0 and sum/arv > 10:
    print("Keskmine on suurem kui 10")
else:
    print("arv <= 0 voi keskmine <= 10")
```

Tsüklidirektiivid

- Iteratsioon ehk tsükkel on kontrollstruktuur, mis võimaldab lausete korduvtäitmist.
- Sõltuvalt korduste arvu määramise viisidest eristatakse järgnevaid tsüklikonstruktsioone:
  - määratud tsükkel – korduste arv on fikseeritud enne tsüklikonstruktsiooni täitmist;
  - eelkontrolliga tsükkel – korduste arv määratakse dünaamiliselt tsüklilõpu tingimusega, kusjuures tingimust kontrollitakse enne iga korduse algust;
  - järelkontrolliga tsükkel – sarnaselt eelmisega määratakse korduste arv dünaamiliselt, kuid tingimust kontrollitakse pärast igat kordust.
- NB! Pythonis on sisseehitatud tsüklidirektiivideks eelkontrolliga tsükkel ja jadaiteraator.

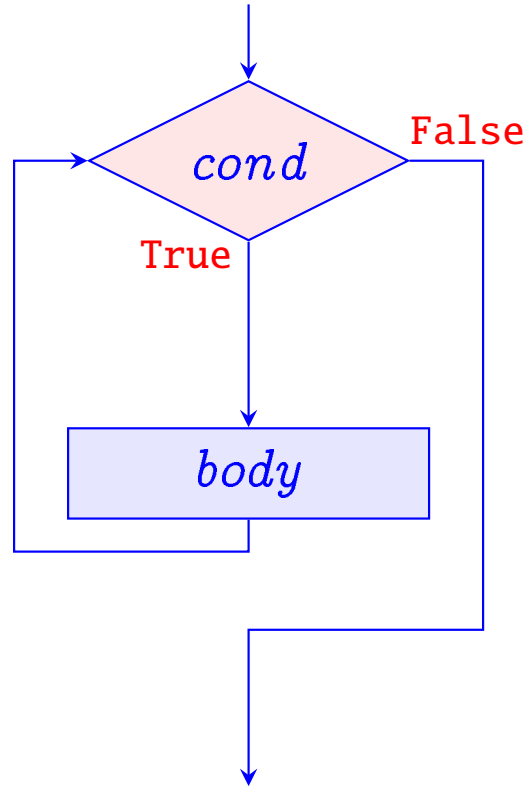
Eelkontrolliga tsükkel

- while-tsükkel:

  while *cond*:

  body

- Tingimus *cond* on loogiline avaldis.
- Tsüklikeha *body* võib olla kas üksik lause või lausete plokk.
- Kui tingimus on tõene, siis täidetakse tsüklikeha üks kord ja kontrollitakse tingimust uuesti.
- Protsessi korratakse kuni tingimus on väär, misjuhul tsüklikeha rohkem ei täideta, vaid jätkatakse tsüklile järgnevate lausete täitmisega.



Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum | 0

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	0
i	1

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	0
i	1

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	1
i	1

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	1
i	2

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	1
i	2

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	1
i	2

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	3
i	2

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	3
i	3

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	3
i	3

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	3
i	3

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	6
i	3

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	6
i	4

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	6
i	4

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	6
i	4

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	10
i	4

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	10
i	5

Eelkontrolliga tsüklid


```
sum = 0
i = 1
while i < 5:
    sum = sum + i
    i = i + 1

...
```

sum	10
i	5

Eelkontrolliga tsüklid

$sum = 0$
$i = 1$
while $i < 5$:
    $sum = sum + i$
    $i = i + 1$

...

sum	10
i	5

Eelkontrolliga tsükkel

Näide – kolmega jaguvate arvude trükkimine

```
i = 0
while (i < 10):
    if i % 3 == 0:
        print(i)
    i = i + 1
```

Eelkontrolliga tsükkel

Näide – faktoriaal

```
fact = 1
n = 5
while n > 1:
    fact *= n
    n = n - 1
```

Eelkontrolliga tsükkel

- Reeglina tuleb *tsüklimuutujat* tsüklikehas muuta selliselt, et "vahe" tsüklitingimusega väheneks.
- Vastasel korral võib tsükkel mittetermineeruda.

Eelkontrolliga tsükkel

- Reeglina tuleb *tsüklimuutujat* tsüklikehas muuta selliselt, et "vahe" tsüklitingimusega väheneks.
- Vastasel korral võib tsükkel mittetermineeruda.

Näide – lõpmatu tsükkel (1)

```
i = 0
while (i < 10):
    print(i)
```

Eelkontrolliga tsükkel

- Reeglina tuleb *tsüklimuutujat* tsüklikehas muuta selliselt, et "vahe" tsüklitingimusega väheneks.
- Vastasel korral võib tsükkel mittetermineeruda.

Näide – lõpmatu tsükkel (2)

```
i = 0
while (i < 10):
    print(i)
    i = i - 1
```

Tsükli kontrolldirektiivid

- Tsüklikonstruktsioonid lubavad kontrollida jätkutingimust enne igat tsüklikeha täitmist.
- Mõnikord on loomulikum kontrollida tingimust tsüklikeha keskel, lõpus või isegi mitmes kohas.
- Tingimusi saab kontrollida if-lausega ning tsüklikeha täitmist on võimalik katkestada kontrolldirektiividega **break** ja **continue**.
- Käsk **break** lõpetab koheselt tsükli täitmise ning programm jätkab tsüklile järgneva lause täitmisega.
- Käsk **continue** lõpetab tsüklikeha täitmise ning täitmist jätkatakse tsüklitingimuse kontrollimisega; kui see on tõene, siis jätkatakse tsükli täitmist edasi.
- NB! Üksteisesse sisestatud tsüklite korral mõjutavad käsud **break** ja **continue** ainult sisemise tsükli täitmist.

Tsükli kontrolldirektiivid

Näide – sisendi korrektsuse kontrollimine

```
while True:
    arv = int(input("Sisesta positiivne täisarv: "))
    if arv > 0:
        break
    print("Sisestatud arv ei olnud positiivne!")
```

Tsükli kontrolldirektiivid

Näide – paarisarvude trükkimine

```
x = 10
while (x>0):
    x = x - 1
    if x % 2 != 0: continue
    print(x)
```

Eelkontrolliga tsükli üldkuju

- while-tsükli üldkuju:

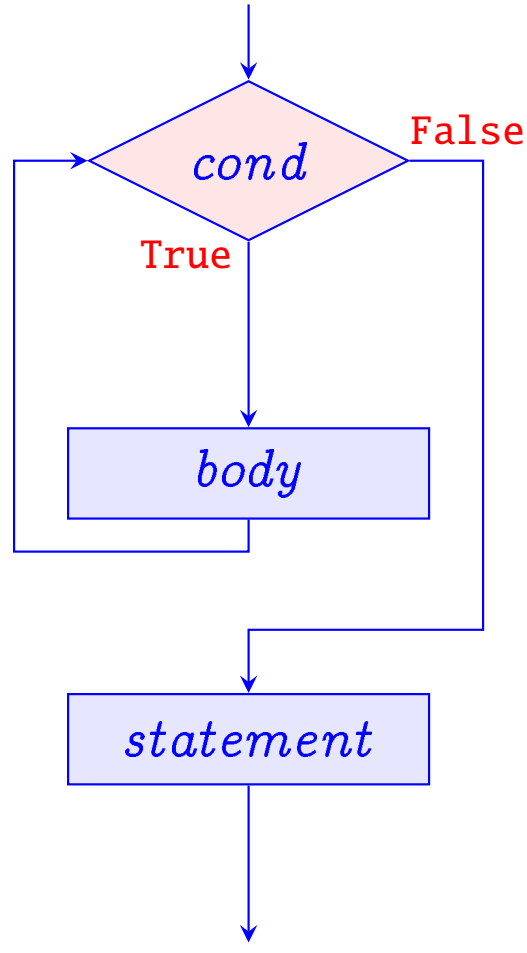
  while *cond*:

  ⟶ *body*

  else:

  ⟶ *statement*

- Kui tingimus on tõene, siis täidetakse tsüklikeha üks kord ja kontrollitakse tingimust uuesti.
- Protsessi korratakse kuni tingimus on väär, misjuhul täidetakse else-haru (üks kord) ja väljutakse tsüklist.
- Kui tsüklikehas on kontrolldirektiiv **break**, siis selle täitmisel hüpatakse tsüklikehast välja ilma else-haru täitmata.



Eelkontrolliga tsükli üldkuju

Näide – algarvulisuse kontroll

```
import math

n = int(input("Sisesta arv: "))
d = int(math.sqrt(n))

while d > 1:
    if n % d == 0:
        print("Arv", n, "jagub arvuga", d)
        break
    d = d - 1
else:
    print("Arv", n, "on algarv!")
```

Jadaiteraator

- Pythonis on for-tsükkel üldine jadaiteraator:

  for *var* **in** *list*:
      body

- *var* on tsüklimuutuja ning *list* on mingi jada.
- Tsüklimuutuja saab algul väärtuseks jada esimese elemendi väärtuse ja igal järgneval iteratsioonisammul jada järgmise elemendi väärtuse.
- Tsükkel lõpeb, kui jadas rohkem elemente pole.

Määratud tsüklid

- Määratud tsükli tarvis saab jada genereerimiseks kasutada funktsiooni *range*.
- Kõige üldisem versioon, $range(start, stop, step)$, saab kolm täisarvulist argumenti:
  - argument *start* on jada esimene element;
  - argument *stop* on jada ülemine piir (kõik jada elemendid on sellest "väiksemad");
  - argument *step* on samm, mille võrra jada järgmine element on eelmisest "suurem".
- Kaheargumendilisena, $range(start, stop)$, on *step* üks.
- Üheargumendilisena, $range(stop)$, on *start* väärtuseks null.

$$
\begin{aligned}
range(5) &\Longrightarrow [0, 1, 2, 3, 4] \\
range(1, 6) &\Longrightarrow [1, 2, 3, 4, 5] \\
range(1, 6, 2) &\Longrightarrow [1, 3, 5]
\end{aligned}
$$

Määratud tsüklid

Näiteid määratud tsüklitest

```
for i in range(6):
    print(i, end=" ")        # Trükib: 0 1 2 3 4 5

for i in range(-2):
    print(i, end=" ")        # Ei trüki midagi

for i in range(-2,2):
    print(i, end=" ")        # Trükib: -2 -1 0 1

for i in range(2,-2):
    print(i, end=" ")        # Ei trüki midagi

for i in range(2,-2,-1):
    print(i, end=" ")        # Trükib: 2 1 0 -1
```

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

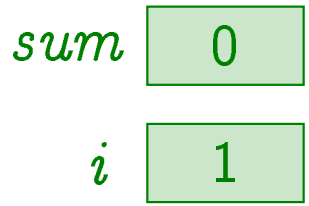
Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum 0

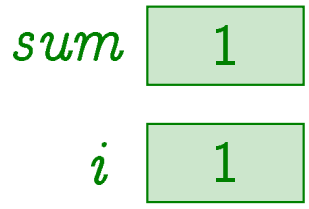
Määratud tsüklid


```
sum = 0
for i in range(1, 5):
    sum = sum + i

...
```



Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```



Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	1
i	1

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	1
i	2

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	3
i	2

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	3
i	2

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	3
i	3

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	6
i	3

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	6
i	3

Määratud tsüklid


```
sum = 0
for i in range(1, 5):
    sum = sum + i

...
```

sum	6
i	4

Määratud tsüklid

```
sum = 0
for i in range(1, 5):
    sum = sum + i

...
```


sum	10
i	4

Määratud tsüklid


```
sum = 0
for i in range(1,5):
    sum = sum + i

...
```

sum	10
i	4

Määratud tsüklid

```
sum = 0
for i in range(1, 5):
    sum = sum + i
```

sum	10
i	4

...

Järgmiseks korraks

- Lugeda läbi õpiku peatükid:
  - Ptk. 3 *"Liitlaused"*
  - (Ptk. 4 *"Algoritm ja plokkskeem"*)
- NB! Järgmisel nädalal arvutipraktikumides (24.–28. sept.) esimene arvutikontrolltöö!

Suur tänu osalemast

ja

kohtumiseni!