

Programmeerimine

7. loeng

Täna loengus

- Listi mõiste
- Järjendid ja massiivid
- Listid, ennikud ja sõned
- Massiivide töötlemine
- Jadaiterator ja määratud tsüklid
- Eeldefineeritud operatsioonid järjenditel

Listi mõiste Pythonis

- Lihtmuutuja
- List kui andmekogum
- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja
- List kui andmekogum
- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
```

```
>>> x = 5
```

```
>>> x
```

```
5
```

- List kui andmekogum
- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
```

```
>>> x = 5
```

```
>>> x
```

```
5
```

- List kui andmekogum

- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
```

```
>>> x = 5
```

```
>>> x
```

```
5
```

- List kui andmekogum

```
>>> joogid = ['tee', 'kohv', 'mahl']
```

```
>>> temperatuurid = [80, 60, 20]
```

```
>>> koos = ['tee', 80, 'kohv', 60, 'mahl', 20]
```

- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
```

```
>>> x = 5
```

```
>>> x
```

```
5
```

- List kui andmekogum

```
>>> joogid = ['tee', 'kohv', 'mahl']
```

```
>>> temperatuurid = [80, 60, 20]
```

```
>>> koos = ['tee', 80, 'kohv', 60, 'mahl', 20]
```

- Tühi list

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
```

```
>>> x = 5
```

```
>>> x
```

```
5
```

- List kui andmekogum

```
>>> joogid = ['tee', 'kohv', 'mahl']
```

```
>>> temperatuurid = [80, 60, 20]
```

```
>>> koos = ['tee', 80, 'kohv', 60, 'mahl', 20]
```

- Tühi list

```
>>> tühi = []
```

Listi mõiste Pythonis

- Lihtmuutuja

```
>>> x = 3
>>> x = 5
>>> x
5
```

- List kui andmekogum

```
>>> joogid = ['tee', 'kohv', 'mahl']
>>> temperatuurid = [80, 60, 20]
>>> koos = ['tee', 80, 'kohv', 60, 'mahl', 20]
```

- Tühi list

```
>>> tühi = []
>>> joogid
['tee', 'kohv', 'mahl']
>>> temperatuurid
[80, 60, 20]
>>> tühi
[]
```

Listi elemendid

- Indeksid

- Elementide arv

Listi elemendid

- Indeksid

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

80	60	20
0	1	2

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

80	60	20
0	1	2

```
>>> temperatuurid[0]
```

80

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

80	60	20
0	1	2

```
>>> temperatuurid[0]
```

```
80
```

```
>>> temperatuurid[2]
```

```
20
```

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

80	60	20
0	1	2

```
>>> temperatuurid[0]
```

```
80
```

```
>>> temperatuurid[2]
```

```
20
```

- Elementide arv

Listi elemendid

- Indeksid

```
>>> temperatuurid = [80, 60, 20]
```

80	60	20
0	1	2

```
>>> temperatuurid[0]
```

```
80
```

```
>>> temperatuurid[2]
```

```
20
```

- Elementide arv

```
>>> len(temperatuurid)
```

```
3
```

Listi elemendid

- Negatiivne indeks

Listi elemendid

- Negatiivne indeks

Listi elemendid

- Negatiivne indeks

80	60	20
-3	-2	-1

Listi elemendid

- Negatiivne indeks

80	60	20
-3	-2	-1

```
>>> temperatuurid[-1]  
20
```

Listi elemendid

- Negatiivne indeks

80	60	20
-3	-2	-1

```
>>> temperatuurid[-1]  
20
```

```
>>> temperatuurid[-3]  
80
```

Listi elemendid

- Negatiivne indeks

80	60	20
-3	-2	-1

```
>>> temperatuurid[-1]  
20
```

```
>>> temperatuurid[-3]  
80
```

```
>>> temperatuurid[-4]  
Traceback (most recent call last):  
  File "<pyshell#20>", line 1, in <module>  
    temperatuurid[-4]  
IndexError: list index out of range
```

List on mutableitav



$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$

List on muteritav



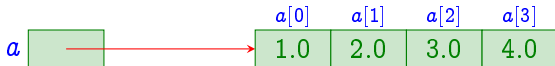
$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$



List on muteritav



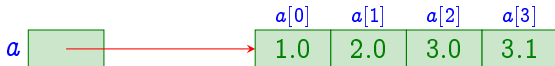
$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$



List on muteritav

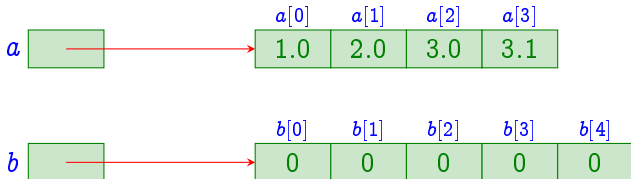
$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$



List on muteritav

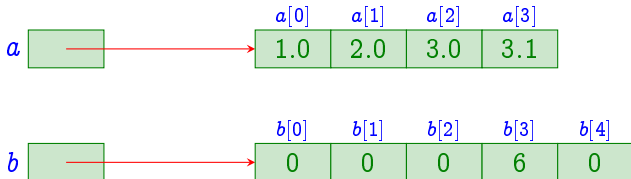
$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$



List on muteritav

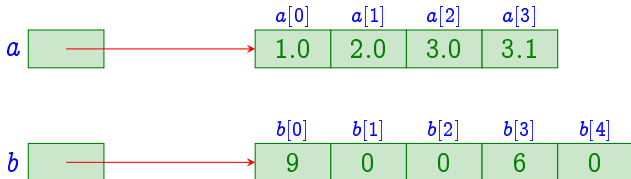
$a = [1.0, 2.0, 3.0, 4.0]$

$a[3] = 3.1$

$b = [0, 0, 0, 0, 0]$

$b[3] = 6$

$b[0] = b[3] + 3$



Listi läbivaatus

```
temperatuurid = [80, 60, 20]
i = 0
while i < len(temperatuurid):
    print(i, temperatuurid[i])
    i += 1
```

- Nullime 50st suuremad temperatuurid $i4-i[]$

Listi läbivaatus

```
temperatuurid = [80, 60, 20]
i = 0
while i < len(temperatuurid):
    print(i, temperatuurid[i])
    i += 1
```

0 80

1 60

2 20

- Nullime 50st suuremad temperatuurid $i4-i$

Listi läbivaatus

```
temperatuurid = [80, 60, 20]
i = 0
while i < len(temperatuurid):
    print(i, temperatuurid[i])
    i += 1

0 80
1 60
2 20
```

- Nullime 50st suuremad temperatuurid

Listi läbivaatus

```
temperatuurid = [80, 60, 20]
i = 0
while i < len(temperatuurid):
    print(i, temperatuurid[i])
    i += 1

0 80
1 60
2 20
```

- Nullime 50st suuremad temperatuurid

```
i = 0
while i < len(temperatuurid):
    if temperatuurid[i] > 50:
        temperatuurid[i] = 0
    i += 1
print(temperatuurid)
```

Listi läbivaatus

```
temperatuurid = [80, 60, 20]
i = 0
while i < len(temperatuurid):
    print(i, temperatuurid[i])
    i += 1
```

0 80

1 60

2 20

- Nullime 50st suuremad temperatuurid

```
i = 0
while i < len(temperatuurid):
    if temperatuurid[i] > 50:
        temperatuurid[i] = 0
    i += 1
print(temperatuurid)
```

[0, 0, 20]

Järjendid

Põhimõisted

- **Järjend** on mingite elementide lõplik jada.
- **Massiiv** (ingl. *array*) on sama tüüpi elementidega ja fikseeritud suurusega järjend, kus elementidele juurdepääs (**indekseerimine**) toimub konstantse ajaga.
- Massiivi elemendid on identifitseeritavad täisarvuliste **indeksite** kaudu.
- Klassikaliselt on massiivid **muteeritavad** (ingl. *mutable*); so. mingi elemendi muutmisel, muudetakse selle väärtus olemasolevas massiivis.
- Mõnedes keeltes võivad massiivid olla **mittemuteeritavad** (ingl. *immutable*); so. elemendi "muutmisel" luuakse uus massiiv.

Järjendid

Järjendid Pythonis

- Pythonis klassikalisi massiive keeles otseselt ei ole.
- Selle asemel on kaks üldist järjenditüüpi: **listid** ja **ennikud**.
 - Sarnaselt massiividega toimub elementide indekseerimine täisarvudega ning konstantse ajaga.
 - Mõlemal juhul põhierinevuseks, et *elemendid ei pea olema sama tüüpi*.
 - Listide korral ka see, et suurus ei ole fikseeritud (so. saab lisada uusi elemente ning olemasolevaid eemaldada).
- Lisaks on ka **sõned** Pythonis järjenditüübiks.

NB!

Edaspidi nimetame massiivideks liste, kus kõik elemendid on sama tüüpi.

Järjendid Pythonis

Listid

- **List** on dünaamilise pikkusega muteeritav järjend.
- Esitatakse elementide loendina kandiliste sulgude vahel.
 $[\textit{expr}_1, \textit{expr}_2, \dots, \textit{expr}_n]$
- Listiavaldis tekitab n -elemendilise listi, kus elementide väärtused on vastavalt avaldiste $\textit{expr}_i$ väärtused.
- Avaldised $\textit{expr}_i$ võivad olla erinevat tüüpi.

NB!

Teistes keeltes nimetatakse listideks tavaliselt dünaamilise pikkusega järjendeid, kus elementidele juurdepääs toimub järjestikku (so. lineaarse ajaga).

Järjendid Pythonis

Ennikud

- **Ennik** (ingl. *tuple*) on fikseeritud pikkusega mittemuteeritav järjend.
- Esitatakse elementide loendina (ümar-)sulgude vahel.
$$(\textit{expr}_1, \textit{expr}_2, \dots, \textit{expr}_n)$$
- Ennikuavaldis tekitab n -elemendilise korteeži, kus elementide väärtused on vastavalt avaldiste $\textit{expr}_i$ väärtused.
- Avaldised $\textit{expr}_i$ võivad olla erinevat tüüpi.

Sõned

- **Sõne** (ingl. *string*) on tähemärkidest koosnev mittemuteeritav järjend.
- Esitatakse ülakomade või jutumärkide vahel.

Järjendid Pythonis

Näiteid listidest ja ennikutest

```
a = [1, 7, 3, 5]
b = [2, 8, 0, 2]
d = ["Hello", 1, 5]      # erinevat tüüpi elementidega
e = [ ]                  # tühi list

f = (1, 'World', a)
g = (3,)                  # üheelemendiline ennik
h = ()                    # tühi ennik
```

Järjendid Pythonis

Järjendi elementidele juurdepääs

seqName [*index*]

- *seqName* on järjendimuutuja nimi (või suvaline avaldis, mis väärtustub järjendiks).
- *index* on täisarvuline avaldis, mis näitab mitmendale elemendile juurdepääsu soovime.
- Järjendite indekseerimine algab nullist.
- Listide korral võib indeksavaldist kasutada ka L-väärtusena (so. omistuslause vasakul poolel).

Näide

```
a    = [1, 7, 3, 5]
b    = a[0] + a[3]      # b = 6
a[2] = 0                # a = [1, 7, 0, 5]
```

Järjendid Pythonis

Järjendi elementidele juurdepääs

- Kasutada võib ka negatiivseid indekseid, misjuhul indekseeritakse paremalt.
 - Indeksiga -1 on parempoolseim element.
- Indeks i peab olema vahemikus $-N \leq i \leq N - 1$ (kus N on listi elementide arv), vastasel korral on täitmisaegne viga.

Näide

```
a = [1, 7, 3, 5]
d = a[-1]          # d = 5
e = a[-4]          # e = 1
```

Jadaiteraator

- Pythonis on for-tsükkel üldine jadaiteraator:

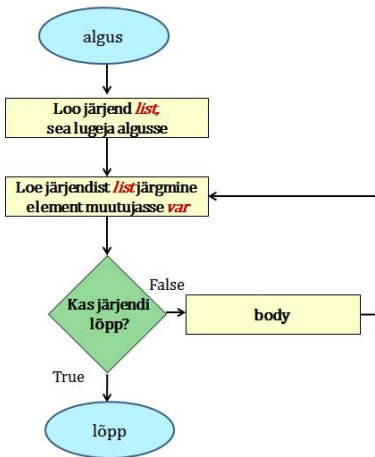
```
for var in list:  
    body
```

- *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.

Jadaiteraator

Plokkskeem

```
for var in list:  
    body
```



Jadaiteraator

Näide – jada elementide väljatrükkimine

```
for c in "Hello, World!":  
    print(c)
```

```
for i in [1,7,2,3,12]:  
    print(i)
```

```
for e in (3,8,2,15):  
    print(e)
```

Määratud tsükliid

- 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.

<i>range(5)</i>	$\Rightarrow$	<i>[0, 1, 2, 3, 4]</i>
<i>range(1, 6)</i>	$\Rightarrow$	<i>[1, 2, 3, 4, 5]</i>
<i>range(1, 6, 2)</i>	$\Rightarrow$	<i>[1, 3, 5]</i>
<i>range(6, 1, -2)</i>	$\Rightarrow$	<i>[6, 4, 2]</i>

Määratud tsükliid



```
sum = 0
```

```
for i in range(1, 5):
```

```
    sum = sum + i
```

```
...
```

Määratud tsükliid



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

sum 0

Määratud tsükliid



```
sum = 0
```

```
for i in range(1, 5):
```

```
    sum = sum + i
```

```
...
```

sum

0

i

1

Määratud tsükliid



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

<i>sum</i>	1
<i>i</i>	1

Määratud tsükliid



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

<i>sum</i>	1
<i>i</i>	1

Määratud tsükliid



```
sum = 0
```

```
for i in range(1, 5):
```

```
    sum = sum + i
```

```
...
```

sum

1

i

2

Määratud tsükliid



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

<i>sum</i>	3
<i>i</i>	2

Määratud tsükliid



sum = 0

for *i* **in** *range*(1, 5):

sum = *sum* + *i*

...

sum

3

i

2

Määratud tsükliid



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

<i>sum</i>	3
<i>i</i>	3

Määratud tsükliid



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

<i>sum</i>	6
<i>i</i>	3

Määratud tsükliid



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

<i>sum</i>	6
<i>i</i>	3

Määratud tsükliid



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

<i>sum</i>	6
<i>i</i>	4

Määratud tsükliid



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

<i>sum</i>	10
<i>i</i>	4

Määratud tsükliid



```
sum = 0
```

```
for i in range(1, 5):
```

```
    sum = sum + i
```

```
...
```

sum

10

i

4

Määratud tsükliid

sum = 0

for *i* **in** *range*(1, 5):

sum = *sum* + *i*

sum

10

i

4



...

Massiivide töötlemine

Massiivi elementide summeerimine

```
def sumListValues(a):  
    sum = 0  
    for elem in a:  
        sum += elem  
    return sum
```

Näide

```
A = [7, 2, 4]  
s = sumListValues(A)  
print(s)
```

Trükkib: 13

Massiivide töötlemine

Massiivi elementide summeerimine (ver. 2)

```
def sumListValues(a):  
    sum = 0  
    for i in range(len(a)):  
        sum += a[i]  
    return sum
```

Näide

```
A = [7, 2, 4]  
s = sumListValues(A)  
print(s)
```

Trükkib: 13

Massiivide töötlemine

Massiivi negatiivsete elementide loendamine

```
def countNegValues(a):  
    count = 0  
    for i in range(len(a)):  
        if a[i] < 0:  
            count += 1  
    return count
```

Näide

```
A = [7, -2, 4, -1]  
n = countNegValues(A)  
print(n)
```

Trükkib: 2

Massiivide töötlemine

Massiivist otsimine

```
def findListValue(x, a):  
    for i in range(len(a)):  
        if a[i] == x:  
            return True  
    return False
```

Näide

```
A = [7, 2, 4]  
if findListValue(2,A):  
    print("Arv leiti!")  
else:  
    print("Arvu ei leitud!")
```

Trükkib: "Arv leiti!"

Massiivide töötlemine

Massiivist minimaalse elemendi leidmine

```
def minListValue(a):  
    min = a[0]  
    for i in range(1,len(a)):  
        if (a[i] < min):  
            min = a[i]  
    return min
```

Näide

```
A = [7, -2, 4, -1]  
m = minListValue(A)  
print(m)                # Trükkib: -2
```

Massiivide töötlemine

Listi ümberpööramine

```
def reverseList(a):  
    n = len(a)  
    b = list(range(n))  
    for i in range(n):  
        b[i] = a[n-1-i]  
    return b
```

Näide

```
A = [7, 2, 4]  
B = reverseList(A)  
print(A[0])           # Trükib: 7  
print(B[0])           # Trükib: 4
```

Suur tänu osalemast

ja

kohtumiseni!