Järjendid I

Programmeerimine
8. tahvlipraktikum

Tänases tahvlipraktikumis

- Järjendid Pythonis
- Ülesandeid järjendite ühekordse läbivaatusega

Slaidide koostamisel on kasutatud
H. Nestra materjale 2010

Järjendid

Järjend on elementide lõplik jada:

$$\langle x_0, x_1, \ldots, x_{n-1} \rangle$$

kus ***n*** tähistab jada pikkust.

Massiiv (*array*) - järjend, milles elemendid on **sama tüüpi**; elementidele juurdepääs toimub indeksite abil.

Järjendid Pythonis

1. **List** (*list*)
   - Muudetav *(mutable)* kindlas järjestuses elementide jada.
   - Elemendid võivad olla eri tüüpi, sh ka järjendid

2. **Ennik** (*tuple*)
   - Mittemuudetav (*immutable*) kindlas järjestuses elementide jada.
   - Elemendid võivad olla eri tüüpi, sh ka järjendid

3. **Sõne** (*string*)
   - Mittemuudetav
   - Kontseptuaalselt sarnane ennikule
   - Märgid 8-bitised, Unicode korral 2-baidised märgid.

Kõik kolm järjenditüüpi kasutavad palju sarnast süntaksi

Oluline erinevus:

- Ennikud (*tuples*) ja sõned (*strings*) on **mittemuudetavad .**
- Listid on **muudetavad** (dünaamilise pikkusega)**.**

Vaatame operatsioone, mida saab rakendada kõikidele järjenditüüpidele.

Järjendid Pythonis I

- Listid defineeritakse kasutades kandilisi sulge ja komasid.

```
>>> li = ["abc", 3, 4.24, 21]
```

- Ennikud defineeritakse kasutades ümarsulge ja komasid.

```
>>> en = (23, "abc", 4.56, [2,3], "dem")
```

- Sõned defineeritakse kasutades ülakomasid või jutumärke (", ' või """).

```
>>> s = "Hello, World!"
>>> st = 'Hello, World!'
>>> st = """See on mitmerealine sõne, mis kasutab
kolmekordseid jutumärke."""
```

Järjendid Pythonis II

- Elementide arv:

```
>>> t = (1, 2, 4, 6)
>>> len(t)
4
```

- Juurdepääs listi, enniku või sõne üksikule elemendile toimub indeksi abil kasutades kandilisi sulge.
- Indekseerimine algab nullist

```
>>> en = (23, 'abc', 4.56, (2,3), 'def')
>>> en[1]     # Enniku teine element
 'abc'

>>> li = ['abc', 34, 4.34, 23]
>>> li[1]      # Listi teine element.
 34

>>> st = 'Hello World'
>>> st[1]   # Sõne teine märk.
 'e'
```

Positiivsed ja negatiivsed indeksid

```
>>> en = (23, 'abc', 4.56, (2,3), 'def')
```

Positiivne indeks: loendame vasakult, alustades nulliga.

```
>>> en[1]
'abc'
```

Negatiivne indeks: loendame paremalt, alustades -1.

```
>>> en[-3]
4.56
```

Viilutamine (*slicing*) I

Viilutamise tulemuseks on alamhulk

```
>>> en = (23, 'abc', 4.56, (2,3), 'def')

>>> en[1:4] # elemendid, mille indeksid on poollõigul [1, 4)
('abc', 4.56, (2,3))
```

Võib kasutada ka negatiivseid indekseid:

```
>>> en[1:-1]
('abc', 4.56, (2,3)
```

Viilutamine (*slicing*) II

```
>>> en = (23, 'abc', 4.56, (2,3), 'def')
```

Esimese indeksi ärajätmine - alustatakse algusest (vasakult)

```
>>> en[:2]     #indeksid poollõigul [0, 2)
(23, 'abc')
```

Teise indeksi ärajätmine - alustatakse esimese indeksiga elemendist kuni lõpuni

```
>>> en[2:]    #elemendid alates indeksist 2
(4.56, (2,3), 'def')
```

```
>>> en[:]  # kõik elemendid
(23, 'abc', 4.56, (2, 3), 'def')
```

Operaator *in*

- Tagastab tõeväärtuse, kas antud element on järjendis:

```
>>> t = [1, 2, 4, 5]
>>> 3 in t
False
>>> 4 in t
True
>>> 4 not in t
False
```

- Sõnede korral on võimalik kontrollida, kas sõne sisaldab etteantud alamsõne:

```
>>> a = 'abcde'
>>> 'c' in a
True
>>> 'cd' in a
True
>>> 'ac' in a
False
```

Ühendamine: operaator *+*

- Operaator ***+*** tekitab **uue** listi, enniku või sõne, mis saadakse operandide ühendamise teel .

```
>>> (1, 2, 3) + (4, 5, 6)
 (1, 2, 3, 4, 5, 6)

>>> [1, 2, 3] + [4, 5, 6]
 [1, 2, 3, 4, 5, 6]

>>> 'Hello' + ' ' + 'World'
 'Hello World'
```

Muudetavus: ennikud ja sõned vs. listid

Listid: muudetavad

```
>>> li = ['abc', 23, 4.34, 23]
>>> li[1] = 45
>>> li
   ['abc', 45, 4.34, 23]
```

- Listi on võimalik muuta vahetult, st muutuja ***li*** viitab endiselt samale aadressile.

Ennikud ja sõned: mittemuudetavad

```
>>> t = (23, 'abc', 4.56, (2,3), 'def')
>>> t[1] = 2

Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    t[1] = 2
TypeError: 'tuple' object does not support item assignment


>>> s = "abc"
>>> s[1] = 'b'
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    s[1] = 'b'
TypeError: 'str' object does not support item assignment
```

Listide muutmise operatsioone - I

```
>>> li = [1, 11, 3, 4, 5]

>>> li.append('a')       # elemendi lisamine lõppu
>>> li
[1, 11, 3, 4, 5, 'a']

>>> li.insert(2, 'i') # elemendi lisamine vahele
>>> li
[1, 11, 'i', 3, 4, 5, 'a']
```

Listide muutmise operatsioone - II

Mitme elemendi muutmine korraga:

```
>>> li = [1, 2, 3, 4, 5, 6]
```

Asendame elemendid indeksitega poollõigult [2, 4) nelja uuega:

```
>>> li[2:4] = [10,11, 15, 16]
>>> li
[1, 2, 10, 11, 15, 16, 5, 6]
```

Eemaldame elemendid indeksitega 2 ja 3

```
>>> li[2:4] = []
>>> li
[1, 2, 15, 16, 5, 6]
```

Listide muutmise operatsioone - III

Elemendi eemaldamine listist:

```
>>> li = [1, 2, 3, 4, 5]
>>> e = li.pop(3)
>>> e
4
>>> li
[1, 2, 3, 5]
```

Elemendi eemaldamine listi lõpust:

```
>>> li = [1, 2, 3, 4, 5]
>>> e = li.pop()
>>> e
5
>>> li
[1, 2, 3, 4]
```

Operaatorid ***extend*** ja ***+***

- ***+*** loob **uue** listi (võetakse kasutusele uus mäluosa)
- ***extend*** opereerib sama listiga.

```
>>> li = [1, 11, 3, 4, 5]
>>> li.extend([9, 8, 7])
>>> li
[1, 11, 3, 4, 5, 9, 8, 7]
```

NB!

- ***extend*** korral on argumendiks **list.**
- ***append*** korral on argumendiks **element.**

```
>>> li.append([10, 11, 12])
>>> li
[1, 11, 3, 4, 5, 9, 8, 7, [10, 11, 12]]
```

Operaator *

Paljundamine

```
>>> a = [1, 2] * 4
>>> a
[1, 2, 1, 2, 1, 2, 1, 2]
```

Operatsioone listidega - I

```
>>> li = ['a', 'b', 'c', 'b']

>>> li.index('b')      # esimese esinemise indeks
1


>>> li.count('b')      # mitmel korral esineb
2

>>> li.remove('b')     # eemaldab esimese
>>> li
 ['a', 'c', 'b']
```

Operatsioone listidega - II

```
>>> li = [5, 2, 6, 8]

>>> li.reverse()     # pöörab järjekorra vastupidiseks
>>> li
  [8, 6, 2, 5]

>>> li.sort()        # sordib listi
>>> li
  [2, 5, 6, 8]
```

Teisendused list ennikuks ja vastupidi:

```
tu = tuple(li)
li = list(tu)
```

Listi muutuja vs lihtmuutuja

Listi muutuja on viit (*reference*) objektile, mitte objekt ise.

Vaatleme lihtsat olukorda tavaliste muutujatega:

```
>>> a = 1
>>> b = a
>>> b = 3
>>> print b
3
>>> print a
1
```

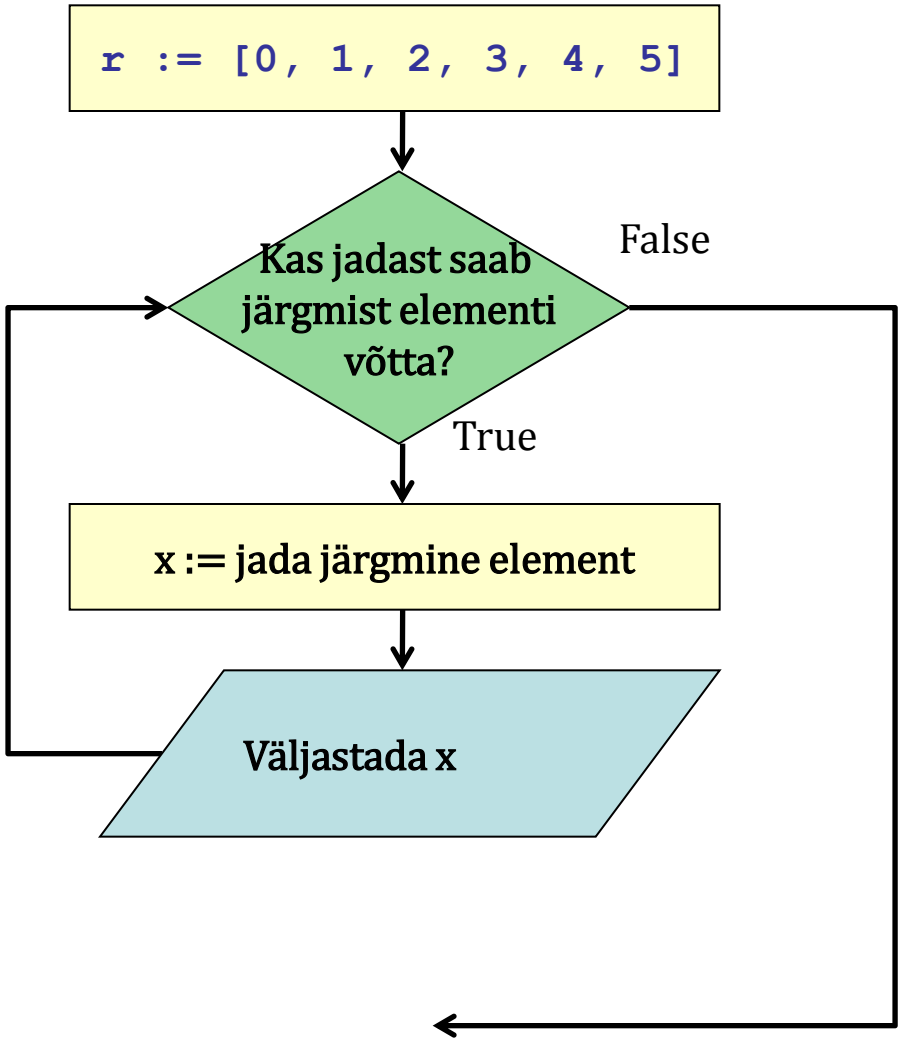
vt. 5. loeng sl 17-22

Tsüklid üle listide

For tsükkel üle listi **elementide:**

```
r = [5, 2, 3, 6.0, 1]
for x in r:
  print("x = %d" % (x))
```

```
x = 5
x = 2
x = 3
x = 6
x = 1
```

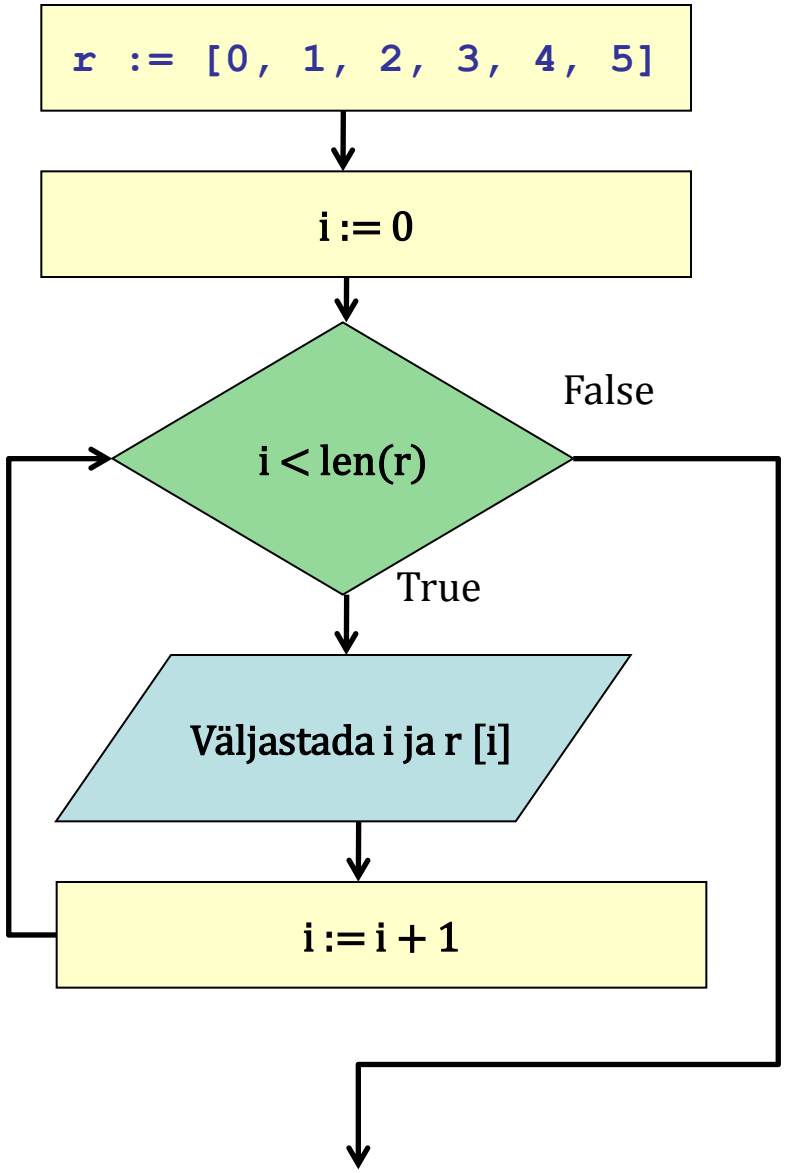


Tsüklid üle listide

For tsükkel üle listi **indeksite:**

```
r = [5, 2, 3, 6.0, 1]
for i in range(len(r)):
  print( "r[%d] = %d" % (i, r[i]))
```

```
r[0] = 5
r[1] = 2
r[2] = 3
r[3] = 6
r[4] = 1
```



Tüüpilised ülesanded

1. Jada **summa**. Leida antud jada kõigi elementide summa.
2. Jada liikmete **keskmine**. Leida jada elementide keskmine väärtus.
3. Jada liikmete **korrutis**. Leida jada kõigi elementide korrutis.
4. **Maksimum** (**miinimum**). Leida antud jada maksimaalne (minimaalne) element.
5. **Esimene** negatiivne (positiivne). Leida antud jada kõige esimene negatiivne (positiivne) element. Tagastada 0, kui negatiivset (positiivset jne) ei leidu.
6. **Viimane** positiivne (negatiivne). Leida antud jada kõige viimane positiivne (negatiivne) element. Tagastada 0, kui positiivset (negatiivset) ei leidu.
7. Nullide **arv** (negatiivsete/positiivsete) arv. Leida mitu null-elementi (positiivset / negatiivset elementi) on antud jadas.
8. Arv K. Leida, **kas antud arv K kuulub** jadasse. S.t. kas leidub i nii, et $a_i=K$.
9. **Libisev keskmine**. Leida jada libisev keskmine 3 (m) elemendi kaupa.
10. Kolm kõige suuremat (väiksemat) elementi. Leida jada kolm kõige suuremat (väiksemat) elementi.

Lahenduste tabel

Koostame tabeli, kus iga esitatud 10 ülesandega seome järgmised
veerud:

a) Ülesande nimi.
b) Minimaalne jada pikkus, mille puhul see ülesanne on defineeritud.
c) Vastuse tähis (nt muutuja, millesse vastus salvestatakse)
d) Lahendus minimaalse jadapikkuse korral.
e) Oletame, et ülesanne on lahendatud *i-1* pikkuse jada korral.

Mis juhtub, kui lisame jada lõppu veel ühe liikme?

Lahenduste tabel

Ülesanne	Min jada pikkus	Vastuse tähis	Lahendus min. jadaga	i-1 ⇒i
1. Summa	0	S	S=0	$S = S + x_i$
2. Keskmine	1	K	$K = x_1$	$K = \frac{K*(i-1)+x_i}{i}$
3.Korrutis	0	P	P = 1	$P = P * x_1$
4.Maksimum	1	M	$M = x_0$	Kui $x_i > M$, siis $M = x_i$
5. Esimene negatiivne	0	E	E=0	Kui $x_i < 0$ ja E=0, siis $E = x_i$
6. Viimane positiivne	0	V	V=0	Kui $x_i > 0$, siis $V = x_i$
7. Nullide arv	0	N	N=0	Kui $x_i = 0$, siis N=N+1
8. Kas K on jadas?	0	α	α=väär	Kui $x_i = K$, siis α=tõene
9. Libisev keskmine	2	Y	Y= < >	$Y=Y,(x_{i-2}+x_{i-1}+x_i)/3$
10. Kolm suurimat	3	A,B,C	A=max(x_0,x_1,x_2) B=keskmine(x_0,x_1,x_2), C=min(x_0,x_1,x_2)	Kui $x_i > A$, siis C=B, B=A, $A = x_i$ Vastasel juhul Kui $x_i > B$, siis C=B, $B = x_i$ Vastasel juhul kui $x_i > C$, siis $C = x_i$

Ülesandeid

1. Koostada funktsioonid kõikidele tabelis esitatud ülesannetele.

2. Koostada funktsioon, mis leiab ja väljastab ekraanile etteantud täisarvude järjendi kõigi selliste elementide indeksite summa, mis jaguvad etteantud täisarvuga.

Näide:

Antud: täisarvujärjend `[0, 7, 4, 2, 5, 10, 7]` ja täisarv `2`.
Tulemus: `0+2+3+5=10`.

3. On antud kaks täisarvujärjendit - poiste pikkuste järjend ja tüdrukute pikkuste järjend (pikkused on antud sentimeetrites). Poisse ja tüdrukuid on võrdselt. Kumbki järjend on pikkuste järgi järjestatud. Koostada programm, mis moodustab tantsupaarid (poiss ja tüdruk) nii, et kõige pikem poiss tantsib kõige pikema tüdrukuga, pikkuselt järgmised on omavahel paaris jne. Väljastada lähteandmed (poiste ja tüdrukute pikkused) ning tantsupaarid.

4. Järjendite „paarsusühildamine“. Antud: kaks täisarvujärjendit. Tulemus: tagastatakse uus täisarvujärjend, mis on saadud nii, et esimesest etteantud järjendist on võetud paaritu indeksiga elemendid ja teisest paarisindeksiga elemendid.
Näiteks lähtejärjendite $\{0, 1, 2, 3, -1\}$ ja $\{2, 4, -3\}$ puhul sobib tulemuseks järjend $\{1, 3, 2, -3\}$.

5. Ühte järjendisse on salvestatud võidusõidul osalevate autode hetkekaugused (*km*) finišist, teise järjendisse aga nende autode kiirused (*km/h*). Tuleb kirjutada funktsioon, mis leiab, mitu autot on t tunni pärast finišisse jõudnud. (Teepikkus s, kiirus v ja aeg t on teatavasti seotud valemiga $s = vt$.)

Sobivad lisafunktsioonid ülesande lahendamiseks:

1) Antud on autode kauguste järjend, kiiruste järjend ning aeg. Muuta kauguste järjendi elementide väärtusi järgmisel viisil: iga auto kaugust vähendada antud aja jooksul läbitud teepikkuse võrra; negatiivne uus kaugus asendada nulliga.

2) Antud on järjend autode kaugustega finišist. Leida nende autode arv, mille kaugus finišist on null.