Järjendid (list) ehk loendid

Andmekogumitega töötamiseks pakub Python selliseid sisseehitatud tüüpe, nagu järjendid, ennikud (korteežid) ja sõnastikud.
Järjend (list) on andmetüüp, mis salvestab elementide kogumit või jada.
Järjendi loomiseks loetletakse nurksulgudes ([]) komadega eraldatuna kõik selle elemendid. Paljudes programmeerimiskeeltes on sarnane andmestruktuur, mida nimetatakse massiiviks.
Väärtusi, mis asuvad nurksulgudes ja on eraldatud komadega, nimetatakse **järjendielementideks.**

```
numbers = [1, 2, 3, 4, 5]

print(numbers)
```

```
[1, 2, 3, 4, 5]
```

Järjendi loomiseks saab kasutada ka konstruktorit list().

```
numbers1 = []

numbers2 = list()
```

Järjendi kontstruktor *list* võib loendi koostamiseks võtta teise järjendi:

In [ ]:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

numbers2 = list(numbers)
```

Sisseehitatud funktsioon list(), mis lisaks tühja järjendi loomisele suudab teatud tüüpi objekte järjenditeks teisendada.

In [ ]:

```
numbers = list(range(5))

print(numbers)
```

```
[0, 1, 2, 3, 4]
```

Selle koodi täitmisel toimub järgmine:

1. Kutsutakse funktsioon range(), kuhu edastatakse argumendina number 5;
2. See funktsioon tagastab numbrite jada 0, 1, 2, 3, 4;
3. Numbrite jada 0, 1, 2, 3, 4 edastatakse argumendina funktsioonile list();
4. Funktsioon list() tagastab loendi [0, 1, 2, 3, 4];
5. Loend [0, 1, 2, 3, 4] määratakse muutujale numbers

In [ ]:

```
even_numbers = list(range(0, 10, 2))

print(even_numbers)
```

```
[0, 2, 4, 6, 8]
```

In [ ]:

```
s = 'Python'
```

```
chars = list(s)

print(chars)
```

```
['P', 'y', 't', 'h', 'o', 'n']
```

Pythoni järjendid on sarnased teiste programmeerimiskeelte **massiividega**. Siiski on järjendite ja massiivide vahel erinevus.
Massiivi elemendid on alati sama andmetüüpi ja paiknevad arvuti mälus pideva plokina, järjendielemendid võivad aga olla mälus hajutatud ja võivad olla erinevat andmetüüpi.

Elementide läbi käimine

Elementide läbikäimiseks saab kasutada nii *for*- kui ka *while*-tsüklit.

In [ ]:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]

for item in companies:

    print(item)
```

```
Microsoft
Google
Oracle
Apple
```

In [ ]:

```
companies = ["Microsoft", "Google", "Oracle", "Apple"]

i = 0

while i < len(companies):

    print(companies[i])

    i += 1
```

```
Microsoft
Google
Oracle
Apple
```

Järjendi kuvamine

Järjendi sisu kuvamisel funktsiooni print() abil kuvatakse elemendid nurksulgudes, kusjuures kõik elemendid on eraldatud komaga. Selline kuvamine ei ole alati mugav ja eelistatav, nii et peab oskama loendi elemente muul viisil kuvata.

In [ ]:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in numbers:

    print(num, end = ' ')
```

```
0 1 2 3 4 5 6 7 8 9 10
```

Pythonis on mugav viis järjendi elementide kuvamiseks ilma for-tsüklit kasutamata.
Variant 1. Järjendi elementide kuvamine ühe tühiku kaudu:

In [ ]:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(*numbers)
```

```
0 1 2 3 4 5 6 7 8 9 10
```

Variant 2. Järjendi elementide kuvamine, igaüks eraldi real

In [ ]:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(*numbers, sep='\n')
```

```
0
1
2
3
4
5
6
7
8
9
10
```

Meetodid ja funktsioonid järjenditega töötamisel

Järjenditel on mitmesuguseid meetodeid elementide juhtimiseks. Mõned neist on:

- **append(item)**: lisab *item* elemendi järjendi lõppu
- **insert(index, item)**: lisab *item* elemendi järjendile *index* indeksi järgi
- **remove(item)**: *item* elemendi kustutamine. Eemaldatakse ainult elemendi esimene sisenemine. Kui elementi ei leita, genereeritakse erand *ValueError*.
- **clear()**: kõikide elementide kustutamine järjndist
- **index(item)**: tagastab *item* elemendi indeksi. Kui elementi ei leita, genereeritakse erand *ValueError*
- **pop([index])**: eemaldab ja tagastab elemendi indeksi järgi. Kui indeksit ei edastata, kustutab see lihtsalt viimase elemendi.
- **count(item)**: tagastab *item* elemendi esinemiste arvu järjendisse
- **sort([key])**: sorteerib elemendid. Vaikimisi sorteerib see kasvavas järjekorras. Kuid *key* parameetriga saame edastada sorteerimisfunktsiooni.
- **reverse()**: paneb kõik elemendid loetelus vastupidises järjekorras.
- **len(list)**: tagastab järjendi pikkuse.
- **sorted(list, [key])**: tagastab sorteeritud järjendi
- **min(list)**: tagastab järjendi väikseima elemendi.
- **max(list)**: tagastab järjendi suurima elemendi.

Elementide lisamine ja kustutamine

Elementide lisamiseks kasutatakse meetodeid *append()* ja *insert* ning kustutamiseks - meetodeid *remove(), pop()* ja *clear().*

In [ ]:

```
languages = ["Python", "Java", "C#"]

# lisame järjendi lõppu

languages.append("PHP")

# lisame teisele kohale

languages.insert(1, "C++")

print(languages)
```

```
['Python', 'C++', 'Java', 'C#', 'PHP']
```

In [ ]:

```
# same elemendi indeksi

i = languages.index("PHP")

# kustutame selle indeksi järgi

removed_item = languages.pop(i)


last_element = languages[-1]

# kustutame viimase elemendi

languages.remove(last_element)

print(languages)

# kustutame kõik elemendid

languages.clear()
```

```
['Python', 'C++', 'Java']
```

Elemendi olemasolu kontroll

Kui teatud elementi ei leita, genereerivad *remove* ja *index* meetodid erandi. Sellise olukorra vältimiseks saab enne operatsiooni kontrollida, kas element on olemas võtmesõna ***in*** abil:

In [ ]:

```
languages = ["Python", "Java", "C#", "PHP", "C++"]

item = "C#"  # element kustutamiseks

if item in languages:

    languages.remove(item)

 print(languages)
```

```
['Python', 'Java', 'PHP', 'C++']
```

Sortimine

Kasvavas järjekorras sortimiseks kasutatakse meetodit *sort()*:

In [ ]:

```
languages = ["Python", "Java", "C#", "PHP", "C++"]

languages.sort()

print(languages)
```

```
['C#', 'C++', 'Java', 'PHP', 'Python']
```

Kui on vaja andmeid sortida vastupidises järjekorras, saab pärast sortimist rakendada meetodit *reverse()*:

In [ ]:

```
languages = ["Python", "Java", "C#", "PHP", "C++"]

languages.sort()

languages.reverse()

print(languages)
```

```
['Python', 'PHP', 'Java', 'C++', 'C#']
```

Sortimisel võrreldakse tegelikult kahte objekti ja see, mis on "väiksem", pannakse selle ette, mis on "suurem". Mõisted "rohkem" ja "vähem" on umbkaudsed. Ja kui numbrite jaoks on see lihtne - numbrid pannakse kasvavasse järjekorda, siis sõnede ja muude objektide jaoks on olukord keerulisem. Sõne hinnatakse eelkõige esimeste sümbolite järgi. Kui esimesed sümbolid on võrdsed, hinnatakse teisi sümboleid ja nii edasi. Kusjuures numbrit loetakse "väiksemateks" kui tähestiku suurtähti ja suurtähte peetakse väiksemaks kui väiketähti.
Seega, kui järjendis kombineeritakse suur- ja väiketähtedega sõned, võib saada tulemusi, mis ei ole päris korrektsed. Selleks et muuta standardset sorteerimise käitumist, saame meetodile *sort()* parameetrina üle anda funktsiooni.

In [ ]:

```
languages = ["Python", "Java", "C#", "javascript","PHP", "C++", "pascal", "assembler"]

languages.sort()

print(languages)
```

```
['C#', 'C++', 'Java', 'PHP', 'Python', 'assembler', 'javascript', 'pascal']
```

In [ ]:

```
languages = ["Python", "Java", "C#", "javascript","PHP", "C++", "pascal", "assembler"]

languages.sort(key=str.lower)

print(languages)
```

```
['assembler', 'C#', 'C++', 'Java', 'javascript', 'pascal', 'PHP', 'Python']
```

In [ ]:

```
languages = ["Python", "Java", "C#", "javascript","PHP", "C++", "pascal", "assembler"]

sorted_languages = sorted(languages, key=str.lower) #  kõik sorteeritud elemendid paigutatakse uude järjendisse

print(sorted_languages)
```

```
['assembler', 'C#', 'C++', 'Java', 'javascript', 'pascal', 'PHP', 'Python']
```

Ülesanne 1

Kirjutage programm, mis kuvab ingliskeelse tähestiku n tähest koosneva järjendi ['a', 'b', 'c', ...] alaregistris ja teise järjendi: ['a', 'bb', 'ccc', 'dddd', 'eeeee', 'ffffff', ...]

Ülesanne 2

Koostage loend 10 juhuslikust numbrist 1 kuni 100. Kirjutage programm, mis vahetab selle loendi miinimaalse ja maksimaalse elemendid.

Ülesanne 3

Arva sõna ära
Looge sõnade järjend. Programm kuvab juhuslikult järjendis oleva sõna alljoontena, olenevalt tähtede arvust. Kasutaja sisestab tähe, kui selline täht on olemas, siis alljoone asemel kuvatakse sõna õiges kohas täht. Kui sellist tähte pole, lisatakse see täht arvamata tähtede järjendisse ja see järjend kuvatakse ka. Kui kasutaja arvas sõna ära, peab programm kuvama ka katsete arvu.