LISTAT

Ohjelmoinnin peruskurssi yläkouluun

TURUN YLIOPISTO
Oppimisanalytiikan keskus

TIEDON SÄILÖMISESTÄ

- Yksittäisten merkkijonojen ja lukujen sijaan ohjelmoinnissa tarvitaan usein keino tallentaa ja käsitellä **suurempia tietomääriä**.
- Tarvitaan siis suurempia (ja monimutkaisempia) **tietorakenteita**.

LISTA

- Suurempien tietomäärien tallentamista varten useimmissa ohjelmointikielissä on jonkinlainen **listarakenne**.
- Lista on tietorakenne, johon voidaan säilöä **useita alkioita**, jotka on listassa jossakin **järjestyksessä**.
- Järjestys voi olla mikä tahansa järjestys, ja listan alkiot voidaan myös järjestellä uudestaan.

LISTAN OMINAISUUKSIA

- Lista on **mutatoituva** eli alkioita voidaan **lisätä**, **poistaa** tai **muuttaa**.
- Tämä erottaa listan esimerkiksi merkkijonosta, jonka merkkejä ei voi luomisen jälkeen enää lisätä, poistaa tai muuttaa (vaikka sen perusteella voidaan luoda uusia merkkijonoja).

LISTAN ALKIOT

- Listan alkiot voivat olla **mitä tahansa tyyppiä**: lukuja, merkkijonoja tai totuusarvoja.
- Alkioita asetettaessa voidaan käyttää myös **lausekkeita**. Lausekkeen laskettu arvo tallennetaan listaan.
- Listan alkiot voivat olla myös toisia listoja, jolloin saadaan moniulotteisia tietorakenteita.

LISTAN ALUSTAMINEN

- Pythonissa lista alustetaan **hakasulkeilla** eli [ ]-operaattorilla.
- Lista voidaan alustaa käyttäen alkuarvoja:

```
<tunniste> = [<alkiot pilkuin eroteltuna>]
```

- Tai lista voidaan alustaa tyhjä lista:

```
<tunniste> = []
```

ESIMERKKI

```
lukuja = [1, 9, 3, 0]
nimet = ["Essi", "Erno", "Eero"]
pisteet = []
vokaalit = ["a", "e", "i", "o", "u", "y", "ä", "ö"]
```

LISTAN INDEKSOINTI

- Listan alkiot on **indeksoitu** samalla tavalla kuin merkkijonojen merkit.
- **Indeksointi alkaa nollasta.**

```
                0       1       2
nimet = ["Essi", "Erno", "Eero"]
```

YHDEN ALKION JA ALILISTAN POIMIMINEN

- **Yksittäisiä listan alkioita** tai **alilistoja** voidaan poimia [ ]-operaattorilla.
- Mekaniikka ja syntaksi on sama kuin merkkien ja alijonojen poimimisessa merkkijonoista.

```
<lista>[<indeksi>]

<lista>[<alkuindeksi> : <loppuindeksi>]
```

ESIMERKKI

```
lista = [14, 12, 18, 20, 15]

alkio1 = lista[0] # 14

alkio3 = lista[2] # 18

alilista = lista[0:2] # [14, 12]

alilista2 = lista[2:3] # [18]

alilista3 = lista[3:] # [20, 15]
```

ALKIOIDEN MUUTTAMINEN

- Toisin kuin merkkijonoissa, listan alkioita voi myös muuttaa.

  Pythonissa tähän käytettään **hakasulkeita ja asetusoperaattoria** (=).

```
<lista>[<indeksi>] = <lauseke>
```

ESIMERKKI

```
lukuja = [1, 9, 3, 0]

lukuja[0] = 5 # [5, 9, 3, 0]

lukuja[2] = 6 # [5, 9, 6, 0]

lukuja[3] = 8 # [5, 9, 6, 8]


nimet = ["Essi", "Erno", "Eero"]

nimet[1] = "Erkki" # nimet = ?
```

LISTAN PITUUS

- **Listan pituuden** eli **alkioiden lukumäärän** saa selville listametodilla `len()`. Syntaksi on jälleen sama kuin merkkijonojen kanssa.

```
len(<lista>)
```

ESIMERKKI

```
lukuja = [1, 9, 3, 0]
lukumaara = len(lukuja) # 4

nimet = ["Essi", "Erno", "Eero"]
print(len(nimet))
```

ALKIOIDEN LISÄÄMINEN LISTAAN

- Listaan voidaan **lisätä uusia alkioita** append( )-metodilla. Alkio lisätään **listan loppuun**.
- Lisääminen siis kasvattaa listan pituutta yhdellä.

```
<lista>.append(<lauseke>)
```

ESIMERKKI

```
luvut = [2, 5, 1]
print(len(luvut)) # tulostaa 3
luvut.append(8) # [2, 5, 1, 8]
print(len(luvut)) # tulostaa 4
nimet = ["Heidi"]
nimet.append("Petra") # ["Heidi", "Petra"]
nimet.append("Mikko") # ["Heidi", "Petra", "Mikko"]
```

ALKIOIDEN POISTAMINEN LISTASTA

- Listasta voidaan **poistaa alkioita** kahdella metodilla.
- Metodi `pop()` poistaa alkioita niiden **indeksin perusteella**.
- Metodi `remove()` **alkioiden arvon perusteella**.
- Alkioiden poistaminen siis lyhentää listan pituutta yhdellä.

ALKIOIDEN POISTAMINEN: pop()

- Metodi pop() **poistaa siis alkion annetun indeksin kohdalta**.

```
<lista>.pop(<indeksi>)
```

ESIMERKKI

```
luvut = [50, 40, 80, 60, 40]
luvut.pop(2) # [50, 40, 60, 40]
luvut.pop(0) # [40, 60, 40]

nimet = ["Erno", "Lauri", "Milla"]
nimet.pop(1) # ["Erno", "Milla"]
```

ALKIOIDEN POISTAMINEN: `remove()`

- Metodi `remove()` sen sijaan **poistaa alkioiden perusteella**.
- Jos alkiota ei löydy listasta, tulee **virheilmoitus**.

```
<lista>.remove(<alkio>)
```

ESIMERKKI

```
luvut = [50, 40, 80, 60, 40]
luvut.remove(50) # [40, 80, 60, 40]
luvut.remove(40) # [80, 60, 40]

nimet = ["Erno", "Lauri", "Milla"]
nimet.remove("Milla") # ["Erno", "Lauri"]
```

LISTOJEN ITEROINTI

- Ohjelmoidessa on usein tarpeellista **käydä läpi listan kaikki alkiot**. Sitä kutsutaan **iteroimiseksi**.
- **Iteroimiseen käytetään silmukoita**. Sekä `while`- että `for`-silmukka toimivat.
- Iteroimisessa hyödynnetään usein listan indeksejä. Luodaan siis indeksimuuttuja, joka silmukassa käy läpi kaikki listan indeksit.

ITEROIMINEN while-SILMUKALLA

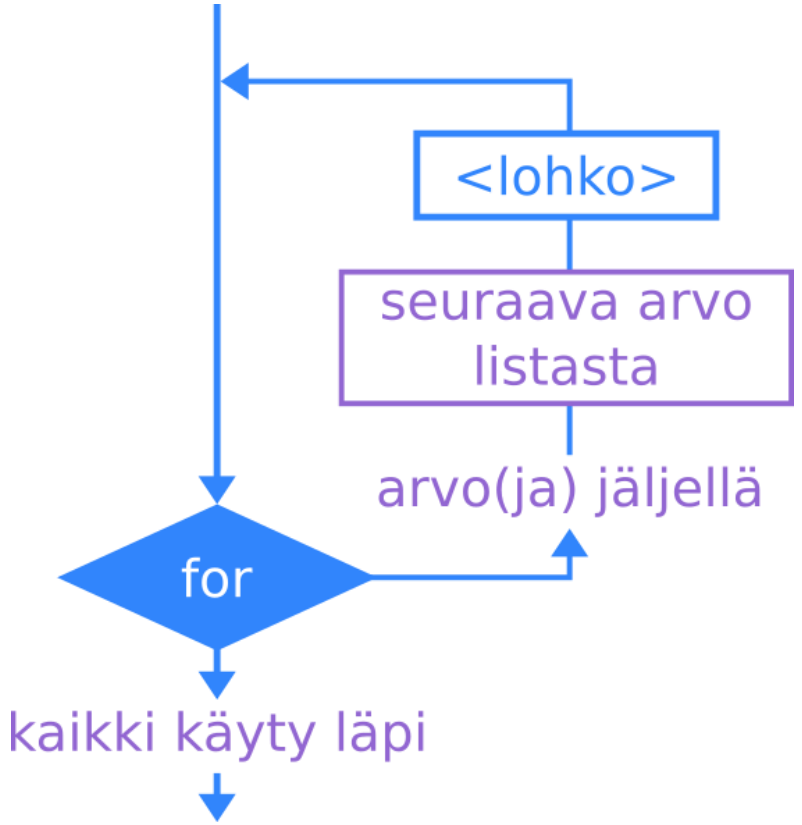
```
lista = [1, 9, 4, 6, 2]
# Pidetään kirjaa indeksistä muuttujassa.
indeksi = 0
# Käydään silmukalla läpi kaikki listan indeksit.
while indeksi < len(lista):
    print(lista[indeksi])
    # Seuraavan alkion indeksi.
    indeksi = indeksi + 1
```

ITEROIMINEN for-SILMUKALLA

```
lista = [1, 9, 4, 6, 2]

# Laitetaan rangen aloitukseksi 0 ja lopetukseksi pituus.

# Askel on 1 jotta käydään kaikki indeksit (alkiot) läpi.

for indeksi in range(0, len(lista), 1):

    print(lista[indeksi])

    # range hoitaa indeksin kasvattamisen.
```

ITEROIMINEN for-SILMUKALLA

- Lista voidaan iteroida for-silmukalla myös **asettamalla muuttujan arvoksi aina listan alkio**.
- **Käydään siis läpi listan alkiot eikä indeksit.**



ESIMERKKI

```
# Luodaan funktio, joka palauttaa listan
# viisikirjaimisista nimistä.
def viisiKirjaimiset(nimet):
    kirjaimia5 = [] # Viisikirjaimiset lisätään tähän.
    for nimi in nimet: # Käydään nimet läpi.
        if len(nimi) == 5: # Tarkistetaan pituus.
            kirjaimia5.append(nimi) # Lisätään.
    return kirjaimia5 # Palautetaan.
```

ESIMERKKI

- Samalla tavalla voidaan käydä läpi myös merkkijonojen merkit.

```
lause = "Ulkona paistaa aurinko."
for merkki in lause:
    print(merkki)
```

TURUN
YLIOPISTO
Oppimisanalytiikan keskus