Pythonin perusteita

Päivitetty viimeksi 27. toukokuuta 2019

Tähän ohjeeseen liittyviä tehtäviä

- Tähän ohjeeseen liittyviä tehtäviä ja lisätietoa löytyy osoitteesta https://kameli.hopto.org/aalto.

Python ohjelmointikielenä

- Ensimmäinen versio jo vuonna 1990
- Yksi maailman käytetyimmistä kielistä, GitHubin mukaan kolmanneksi suosituin
- Pythonia käytetään muun muassa web-kehitykseen, datankäsittelyyn ja koneoppimiseen, esimerkiksi suuri osa Youtubesta ja Redditistä on kirjoitettu Pythonilla

Ensimmäinen ohjelma

```
print("Hello world!")
```

Tässä yhden rivin mittaisessa ohjelmassa on kolme osaa:

1. `print` on funktio, joka tulostaa tekstiä näytölle.
2. Sulut `()`. Kun sulut kirjoitetaan funktion jälkeen, funktiota kutsutaan eli se suoritetaan.
3. `"Hello world!"` on tässä funktion argumentti eli parametri. `print`-funktio toimii niin, että se tulostaa argumenttinsa näytölle. Jos funktiolle haluaa antaa enemmän argumentteja, ne pitää erottaa pilkulla, esimerkiksi
   `print("Hello world!", "asdf", "123")`

Vertaa funktioihin matematiikassa: $f(x)$
f on funktion nimi ja x on muuttuja, joka annetaan funktiolle.

Muuttujat

Muuttujat on tapa tallentaa tietoa ohjelman käyttöön.

```
nimi = 4
```

Pythonissa on eri muuttajatyyppejä, jotka voivat tallentaa erilaista dataa. Pythonissa ei voi määrittää muuttujan tyyppiä suoraan, vaan se tapahtuu antamalla sille tietyn tyyppinen arvo.
Kun muuttujaan on tallennettu tietoa, sitä voidaan käyttää kirjoittamalla muuttujan nimi:

```
muuttuja = "Hello world!"
print(muuttuja)
```

Tulostaa tekstin Hello world!

Operaattorit ja vertailu

Operaattorit ovat merkkejä, joilla voi suorittaa operaatioita, esimerkiksi yhteen-, vähennys-, kerto- ja jakolaskun.

```
print(3 + 10)
print(100 - 1)
print(40 * 3)
print(19 \% 5)
```

Tulostaa: 13, 99, 120 ja 4. Vertailuoperaattoreilla saadaan totuusarvoja (True/False). Esimerkkejä:

```
print(3 == 3)
print(3 != 3)
print(3 > 3)
print(3 >= 3)
```

Tulostaa True, False, False ja True.

Ehtolauseet

Ehtolauseilla voi suorittaa koodia silloin, kun jokin ehto täyttyy.

```
if ikä >= 18:
    print("Täysi-ikäinen")
```

Ehtolauseita voi täydentää elif- ja else-rakenteilla, esimerkiksi:

```
if pisteet < 28:
    print("I")
elif pisteet < 40:
    print("A")
elif pisteet < 52:
    print("B")
elif pisteet < 64:
    print("C")
elif pisteet < 75:
    print("M")
elif pisteet < 90:
    print("E")
else:
    print("L")
```

Silmukat

Silmukoilla voidaan suorittaa ohjelman osia useampia kertoja. Pythonissa on kahdenlaisia silmukoita: for- ja while-silmukoita. For on hyvä silloin kun pitää käydä lista läpi tai tehdä jokin asia jokin määrä kertoja.

```
for i in range(100):
    print("for")
```

Ylläoleva koodi tulostaa tekstin for sata kertaa.

```
x = 1
while x <= 512:
    print(x)
    x = x * 2
```

While suoritetaan niin kauan kun ehto pysyy totena. Tämä esimerkki tulostaa luvut 1, 2, 4, 8, 16, 32, 64, 128, 256 ja 512, jokaisen omalle rivilleen

Lista ja sanakirja

Lista (list) ja sanakirja (dictionary) ovat tietorakenteita, joiden avulla voidaan tallentaa yhden muuttujan alle enemmän tietoa. Lista koostuu alkioista, jotka ovat listassa tietyssä järjestyksessä. Listan alkiot voivat olla mitä tahansa tyyppiä, mutta yleensä on niiden kannattaa olla samaa, eli esimerkiksi ei sekoita numeroita ja merkkijonoja.

```
lista = [10, 3, 2, 7]
lista.append(5)
print(lista)
```

Tuloste on `[10, 3, 2, 7, 5]`. Append-metodi lisää alkion listan loppuun.

```
print(lista[0])
print(lista[1])
```

Tuloste: `10` ja `3` .Yksittäisiä listan alkioita voi käsitellä normaalin muuttujan tavoin hakasulkumerkinnällä, jossa kerrotaan, kuinka monetta käytetään. Listoissa numerointi alkaa nollasta.

Lista ja sanakirja

Sanakirja on kuin lista, mutta siinä alkiot eivät ole järjestyksessä, vaan niitä voidaan käsitellä avainten avulla. Avaimet ovat yleensä kokonaislukuja tai merkkijonoja ja alkiot voivat olla mitä tahansa tyyppiä.

```
sanakirja = {"e":5, "c":3, "b":2, "g":7}
```

Sanakirjan alkioita voi lisätä ja käsitellä helposti:

```
sanakirja = {"e":5, "c":3, "b":2, "g":7}
print(sanakirja["b"])
sanakirja["a"] = 1
print(sanakirja["a"])
```

Tuloste: 2 ja 1