Ohjelmoinnin perusteet Y Python

T-106.1208

19.1.2011

Haluatko antaa palautetta luennoista?

- Ilmoittaudu mukaan lähettämällä ilmainen tekstiviesti "Vast ilmo" numeroon 16403 tai vaihtoehtoisesti täytä lomake osoitteessa http://opey.experq.com/register
- 24.1. alkaen joka luennon jälkeen ilmoittautuneiden joukosta arvotaan 80 opiskelijaa, joille lähetään yksi kysymys luennosta (esim. "Anna arvosana luennolle asteikolla 1 - 5, (5 on paras)")
- Kysymykseen voi vastata ilmaisella tekstiviestillä, joka on muotoa "Vast *num* *vapaa palaute*", esim. "Vast 3 enemmän esimerkkejä".
- Myös ne ilmoittautuneet, jotka eivät ole kysymystä luennon jälkeen saaneet, voivat lähettää vapaamuotoista palautetta ja kysymyksiä 24.1. alkaen numeroon 16403 tekstiviestillä (max 160 merkkiä), joka on muotoa "Vast *vapaa palaute*", esim "Vast enemmän esimerkkejä".
- Yhteystietoja ei luovuteta kolmansille osapuolille eikä vastauksia liitetä takaisin puhelinnumeroihin.

Arvon pyytäminen käyttäjältä

- Käyttäjän antaman arvon voi lukea `raw_input`-käskyllä.
- Käskyn sulkujen sisään kirjoitetaan käyttäjälle annettava kehote.
- Luetun arvon voi antaa arvoksi muuttujalle sijoituskäskyllä.

```
nimi = raw_input("Kerro nimesi: ")
print "Hei,", nimi
print "Tervetuloa Python-kurssille!"
```

Ajoesimerkki

```
Kerro nimesi: Minna
Hei, Minna
Tervetuloa Python-kurssille!
```

Lukuarvon lukeminen käyttäjältä

```
print "Muutan markat euroiksi."
rivi = raw_input("Anna rahasumma markkoina: ")
markat = float(rivi)
eurot = markat / 5.94573
print "Se on", eurot, "euroa."
```

Esimerkkiajo

```
Muutan markat euroiksi.
Anna rahasumma markkoina: 543.50
Se on 91.4101380318 euroa.
```

Tyypeistä

- Monissa muissa ohjelmointikielissä (esim. Java ja C) muuttujat on määriteltävä ennen käyttöä. Muuttujan määrittelyn yhteydessä kerrotaan, minkä tyyppisiä arvoja muuttujalle voi antaa.
- Python-ohjelmissa muuttujia ei määritellä, mutta arvolla on kuitenkin aina tyyppi. Tyyppi vaikuttaa esim. siihen, millaisia operaatoita arvolle voi tehdä ja miten nämä operaatiot toimivat.
- Kokonaisluvuille käytetään yleensä tyyppiä `int`.
- Desimaalilukujen esittämiseen käytetään `float`-tyyppiä. Suuria tai hyvin pieniä desimaalilukuja voi esittää eksponenttimerkinnän avulla, esim. `2.22E12` tai `1.57E-31`
- `float`-tyyppiset arvot eivät ole reaalilukuja matemaattisessa mielessä.

Tyypeistä – jatkoa

- Merkkijonojen tyyppi on `str`. Merkkijonot voidaan kirjoittaa yksin- tai kaksinkertaisten lainausmerkkien sisään, esim.
  `"Matti Virtanen"`, `'Maija Makinen'`
- Lisäksi on tyyppi `bool` totuusarvojen esittämiseen. Mahdollisia arvoja on `True` ja `False`.

Sijoituskäsky

- Muuttujalle voi antaa arvon sijoituskäskyllä. Sijoitettava arvo voi olla myös laskutoimituksen tulos:

```
leveys = 5
korkeus = 6
pinta_ala = leveys * korkeus
```

- Muuttujan vanhaa arvoa voi käyttää hyväksi uutta arvoa laskettaessa:

```
luku = 5
luku = luku + 2
korkeus = 7.0
korkeus = korkeus / 2
```

Sijoituskäsky – jatkoa

- Muuttujan vanhan arvon käyttäminen hyväksi sijoituskäskyssä on niin yleistä, että sille on lyhennysmerkintä:

  ```
  muuttuja = muuttuja + jotain
  ```

  voidaan kirjoittaa

  ```
  muuttuja += jotain
  ```

- Esimerkiksi:

  ```
  luku = 5
  luku += 1
  ```

- Vastaavasti toimivat `-=`, `*=` ja `/=`.

Aritmeettisia laskutoimituksia

- Yleisimmin käytetyt aritmeettiset operaattorit kokonais- ja desimaaliluvuille ovat +, -, *, /, % ja **
- Jakolasku toimii eri tavoin kokonais- ja desimaaliluvuille. Kokonaisluvuilla jakolaskun tulos on kokonaisluku.
- Tämä voi johtaa yllättäviin tilanteisiin, esimerkiksi

```
celsius = 5 / 9 * (fahrenheit - 32)
```

- Operaattori % tarkoittaa jakojäännöstä ja ** potenssiin korotusta.

Funktiot ja pääohjelma

- Tyypillisesti ohjelman rakennetta selkiytetään jakamalla ohjelma funktioihin.
- Funktio on ohjelman osa, jolle on annettu oma nimi.
- Jos ohjelmassa *kutsutaan* funktiota, siirrytään ohjelmakoodissa sille riville, josta funktion määrittely alkaa.
- Kun funktio on suoritettu loppuun, palataan takaisin siihen kohtaan, josta funktiota kutsuttiin.
- Samaa funktiota voidaan kutsua monta kertaa ohjelman suorituksen aikana.
- Funktioiden määrittelyyn ja käyttöön tutustutaan tarkemmin myöhemmin, mutta tässä vaiheessa opetellaan määrittelemään yksi erityinen funktio, `main` eli pääohjelma.

Pääohjelman määrittely

- Pääohjelman määrittely aloitetaan kirjoittamalla

```
def main():
```

- Tämän jälkeen kirjoitetaan pääohjelmaan kuuluvat käskyt sisennettynä, esimerkiksi

```
def main():
    print "Muutan markat euroiksi."
    rivi = raw_input("Anna rahasumma markkoina: ")
    markat = float(rivi)
    eurot = markat / 5.94573
    print "Se on", eurot, "euroa."
```

- Jotta ohjelma suorittaisi pääohjelman, sitä pitää kutsua. Tämä tehdään pääohjelman määrittelyn ulkopuolella:

```
main()
```

Esimerkki: huoneen pinta-ala

- Kun käyttäjältä luetaan useampi arvo, tallennetaan kukin omaan muuttujaansa.

```
def main():
    rivi = raw_input("Anna huoneen leveys metreina: ")
    leveys = float(rivi)
    rivi = raw_input("Anna huoneen pituus metreina: ")
    pituus = float(rivi)
    pinta_ala = leveys * pituus
    print "Huoneen pinta-ala on", pinta_ala, "neliometria"

main()
```

Rivinvaihto tulosteen perään

- Goblinin tarkastusten helpottamiseksi lisätään tällä kurssilla rivinvaihto käyttäjälle annettavan kehotteen perään. Tämä voidaan tehdä merkin `\n` avulla.
- Käsky `print` lisää rivinvaihdon automaattisesti.

```
def main():
    rivi = raw_input("Anna huoneen leveys metreina.\n")
    leveys = float(rivi)
    rivi = raw_input("Anna huoneen pituus metreina.\n")
    pituus = float(rivi)
    pinta_ala = leveys * pituus
    print "Huoneen pinta-ala on", pinta_ala, "neliometria"

main()
```

Kommentit

- Kommentit ovat ohjelmaa lukevalle ihmiselle tarkoitettua selitystekstiä. Python-tulkki ohittaa ne.
- Kommentti aloitetaan #-merkillä. Kaikki sen jälkeen rivillä tuleva teksti tulkitaan kommentiksi.

```
# Ohjelma ilmoittaa sekunteina annetun ajan tunteina,
# minuutteina ja sekunteina.

def main():
    rivi = raw_input("Anna aikajakson pituus sekunteina.\n")
    pituus_sekunteina = int(rivi)
    tunnit = pituus_sekunteina / 3600
    jaannossekunnit = pituus_sekunteina % 3600
    minuutit = jaannossekunnit / 60
    sekunnit = jaannossekunnit % 60
    print "Aikajakson pituus on", tunnit, "h", minuutit, \
          "min", sekunnit, "s."
```

Toinen dokumentointimahdollisuus

- #-merkillä aloitettavien kommenttien lisäksi Pythonissa on myös toinen mahdollisuus kommentoida ohjelman kokonaisuuksia, dokumentointimerkkijono (documentation string, docstring).
- Se on lainausmerkkien sisään pantu kommentti, jota voidaan käyttää joko ohjelmatiedoston alussa tai heti funktion tai luokan otsikkoa seuraavalla rivillä.
- Python-tulkki pystyy käyttämään hyväksi dokumentointimerkkijonoja ja niiden avulla voidaan myös generoida automaattisesti ohjelman dokumentteja.
- Tällä kurssilla dokumentointimerkkijonoja ei käsitellä tämän enempää, mutta nykyisin käytettävä Pydev-versio lisää oletuksena dokumentointimerkkijonon (kolmen lainausmerkin sisässä) uuden ohjelmatiedoston alkuun.
- Opiskelija saa valintansa mukaan joko jättää tuotetun dokumentointimerkkijonon tiedoston alkuun ja kirjoittaa sen sisään ohjelman alkukommentit tai poistaa dokumentointimerkkijonon ja korvata sen tavallisilla kommenteilla.

Valintakäsky if

- Tähänastiset ohjelmat ovat toimineen aina samalla tavalla. Usein ohjelman pitäisi kuitenkin muuttaa toimintaansa käyttäjän syötteen mukaan.
- Esimerkki: kirjoita ohjelma, joka pyytää käyttäjältä tentin pistemäärän ja kertoo, menikö tentti läpi, kun läpipääsyraja on 50 pistettä.
- Valinta voidaan tehdä `if`-käskyn avulla. Yleinen muoto:

```
if ehto:
    kasky1
else:
    kasky2
```

Tenttiesimerkki

```
def main():
    syote = raw_input("Kerro tenttipisteesi.\n")
    pisteet = int(syote)
    if pisteet >= 50:
        print "Tentti meni lapi!"
    else:
        print "Reputit!"


main()
```

Toinen esimerkki: luvun itseisarvo

```
def main():
    print "Ohjelma laskee desimaaliluvun itseisarvon."
    rivi = raw_input("Anna luku.\n")
    luku = float(rivi)
    if luku < 0:
        itseisarvo = - luku
    else:
        itseisarvo = luku
    print "Sen itseisarvo on", itseisarvo


main()
```

Useampi suoritettava käsky if-käskyssä

- Sisennyksillä osoitetaan, mitkä käskyt kuuluvat suoritettavaan vaihtoehtoon.

```
def main():
    print "Ohjelma laskee desimaaliluvun itseisarvon."
    rivi = raw_input("Anna luku.\n")
    luku = float(rivi)
    if luku < 0:
        itseisarvo = - luku
        print "Sen itseisarvo on", itseisarvo
    else:
        print "Sen itseisarvo on", luku


main()
```

If-käsky ilman else-osaa

- Else-osa voi myös puuttua. Tällöin siirrytään suoraan ohjelmassa eteenpäin (if-käskyä seuraavaan käskyyn), jos ehto on epätosi.

```
def main():
    print "Ohjelma laskee desimaaliluvun itseisarvon."
    rivi = raw_input("Anna luku.\n")
    luku = float(rivi)
    if luku < 0:
        luku = - luku
    print "Sen itseisarvo on", luku


main()
```

Vertailuoperaattoreita

>	suurempi kuin
<	pienempi kuin
==	yhtäsuuri kuin
!=	erisuuri kuin
>=	suurempi tai yhtäsuuri kuin
<=	pienempi tai yhtäsuuri kuin

- Huomaa yhtäsuuruusoperaattorin == ja sijoitusoperaattorin = ero.
- Desimaalilukujen yhtäsuuruutta ei yleensä kannata tutkia, koska pyöristysvirheet voivat aiheuttaa yllätyksiä.