Ohjelmoinnin perusteet Y Python

T-106.1208

15.2.2010

Merkkijonot

- Merkkijonojen avulla ohjelmassa voi esittää tekstitietoa, esim. nimiä, osoitteita ja erilaisia tunnuksia.
- Merkkijonon tyyppi Pythonissa on `str`.
- Yksittäisiä merkkejä varten ei ole omaa tyyppiä, vaan ne ovat yhden merkin pituisia merkkijonoja.
- Merkkijono esitetään yksin- tai kaksinkertaisten lainausmerkkien avulla.

```
mjono = 'appelsiini'
mjono = "appelsiini"
```

Merkkijonojen käsittely

- Merkkijonoja voidaan käsitellä monessa tapauksessa samalla tavalla kuin listoja, esim.

```
>>> sana = "sitruuna"
>>> print sana[3]
r
```

- Olennainen ero: merkkijonon sisältöä ei voi muuttaa sen jälkeen, kun merkkijono on luotu, esim.

```
>>> sana[3] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Merkkijonon läpikäynti

- Merkkijonon merkit voi käydä läpi for-käskyn avulla samalla tavalla kuin listan alkiot:

```
>>> mjono = "matti"
>>> for merkki in mjono:
...     print merkki
...
m
a
t
t
i
```

Alimerkkijonot

- Merkkijonosta voi ottaa myös alimerkkijonoja samaan tapaan kuin listoista alilistoja:

```
>>> mjono = "appelsiini"
>>> print mjono[2:5]
pel
>>> print mjono [:4]
appe
>>> print mjono [:-1]
appelsiin
```

Merkkijonon pituus ja merkin esiintymisen tutkiminen

- Merkkijonon pituuden saa selville funktiolla `len`.

```
>>> mjono = "appelsiini"
>>> print len(mjono)
10
```

- Operaattorin `in` avulla voi tutkia, esiintyykö merkki merkkijonossa.

```
>>> print "i" in mjono
True
```

- Operaattorin `in` avulla voi myös tutkia, esiintyykö pitempi merkkijono osana toista.

```
>>> print "else" in mjono
False
>>> print "elsi" in mjono
True
```

Merkin esiintymiskohta merkkijonossa

- Metodin `index` avulla voidaan hakea parametrina annetun merkin ensimmäinen esiintymispaikka merkkijonosta:

```
>>> mjono = "appelsiini"
>>> print mjono.index("i")
6
```

- Saman metodin avulla voidaan hakea myös pitempää merkkijonoa:

```
>>> print mjono.index("lsi")
4
```

- Ohjelma kaatuu, jos haettua merkkijonoa ei löydy :

```
>>> print mjono.index("lse")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

Sijoitus merkkijonomuuttujaan

- Merkkijonon sisältöä ei voi muuttaa sen jälkeen, kun merkkijono on luotu.
- Voidaan kuitenkin tehdä uusi merkkijono ja sijoittaa se arvoksi vanhalle muuttujalle:

```
>>> mjono = "mansikka"
>>> print mjono
mansikka
>>> mjono = "mustikka"
>>> print mjono
mustikka
```

Uusi merkkijono vanhan avulla

- Uusi merkkijono voi kuitenkin riippua jotenkin vanhasta merkkijonosta. Metodi `upper` luo uuden merkkijonon, joka sisältää muuten samat merkit kuin vanha merkkijono, mutta kaikki pienet kirjaimet on muutettu isoiksi:

```
>>> mjono = "mustikka"
>>> mjono = mjono.upper()
>>> print mjono
MUSTIKKA
```

- Metodi `lower` luo uuden merkkijonon, joka sisältää muuten samat merkit kuin vanha merkkijono, mutta kaikki pienet kirjaimet on muutettu isoiksi:

```
>>> print mjono
MUSTIKKA
>>> mjono = mjono.lower()
>>> print mjono
mustikka
```

Merkkijonojen liittämiset

- Useampi merkkijono voidaan liittää peräkkäin +-operaattorilla.

```
>>> etunimi = "Matti"
>>> sukunimi = "Virtanen"
>>> kokonimi = etunimi + " " + sukunimi
>>> print kokonimi
Matti Virtanen
```

Jos merkkijonoon halutaan liittää muuntyyppisten muuttujien arvoja, pitää ensin suorittaa tyypinmuunnos `str`-operaattorilla.

```
>>> tunteja = 50
>>> tuntip = 12.5
>>> palkka = tunteja * tuntip
>>> tulosrivi = str(tunteja) + " h * " + str(tuntip) + \
... " eur / h = " + str(palkka)
>>> print tulosrivi
50 h * 12.5 eur / h = 625.0
```

Merkkijonojen monistaminen

- Operaattorin * avulla voidaan tehdä merkkijono, joka sisältää pienemmän merkkijonon monta kertaa.
- Rivinvaihtomerkin saa tarvittaessa mukaan lisäämällä merkkijonoon erikoismerkin "\n":

```
>>> merkit = "*!*"
>>> rivi = 5 * merkit
>>> print rivi
*!**!**!**!**!*
>>> rivit = 3 * (rivi + "\n")
>>> print rivit
*!**!**!**!**!*
*!**!**!**!**!*
*!**!**!**!**!*
```

Erikoismerkkejä

- Merkkijonoihin on mahdollista liittää erikoismerkkejä (engl. escape characters), jotka aiheuttavat tulostuksessa esimerkiksi rivinvaihdon tai kursorin siirron seuraavaan tabulointikohtaan.
- Tärkeimpiä erikoismerkkejä:

`\n`	rivinvaihto
`\t`	tabulaattori
`\'`	yksinkertainen lainausmerkki
`\"`	kaksinkertainen lainausmerkki
`\\`	yksi kenoviiva

Tyhjien merkkien poisto syötteen alusta ja lopusta

- Usein halutaan poistaa käyttäjän syötteestä käyttäjän mahdollisesti vahingossa alkuun tai loppuun kirjoittamat ns. tyhjät merkit (välilyönnit, tabuloinnit, rivinvaihdot).
- Tämä on helppo tehdä metodin `strip` avulla:

```
>>> syote = raw_input("Anna tekstia.\n")
Anna tekstia.
        kirjoitetaan jotain
>>> print "*%s*" % (syote)
*       kirjoitetaan jotain      *
>>> muutettu_syote = syote.strip()
>>> print "*%s*" % (muutettu_syote)
*kirjoitetaan jotain*
```

- Jos tyhjät merkit halutaan poistaan vain merkkijonon alusta tai lopusta, voidaan käyttää metodeita `lstrip` tai `rstrip`.

Merkkijonon jakaminen

- Monesti on tarve jakaa merkkijono osiin jonkun merkin (esimerkiksi välilyönnin kohdata).
- Esimerkki: halutaan erottaa samalla rivillä annettu etunimi ja sukunimi tai sana ja sen käännös toiselle kielelle toisistaan.
- Merkkijono voidaan jakaa metodilla `split`. Se palauttaa listan, joka sisältää jaetun merkkijonon eri osat.
- Oletusarvoisesti `split`-metodi jakaa merkkijonon välilyönnin kohdalta, mutta metodille voidaan antaa parametrina joku muu merkki, jonka kohdasta jako tehdään.
- Jaossa käytetty merkki ei tule mukaan mihinkään osaan.

Merkkijonon jakaminen, esimerkkejä

```
>>> teksti = "Pitempi teksti, joka sisaltaa monta eri sanaa."
>>> osat = teksti.split()
>>> print osat
['Pitempi', 'teksti,', 'joka', 'sisaltaa', 'monta', 'eri',
'sanaa.']
>>> sanarivi = "kirja=book"
>>> kaannokset = sanarivi.split("=")
>>> print kaannokset
['kirja', 'book']
```

Merkkijonojen vertailu

- Merkkijonojen sisältöjä voi verrata toisiinsa vertailuoperaattoreilla ==, !=, <=, >=, < ja >.
- Kun katsotaan, onko toinen merkkijono suurempi kuin toinen, verrataan merkkejä keskenään merkkijonojen alusta lähtien.
- Järjestyksen määrää kirjainten arvo käytetyssä merkkikoodausjärjestelmässä – mitä lukuarvoa kukin kirjain vastaa.
- Käytännössä koodit noudattavat muuten aakkosjärjestystä, mutta isot kirjaimet ovat ennen pieniä ja skandinaaviset aakkoset eivät ole keskenään oikeassa järjestyksessä.

Esimerkkejä merkkijonojen vertailuista

```
>>> nimi1 = "matti"
>>> nimi2 = "teppo"
>>> nimi1 == nimi2
False
>>>
>>> nimi1 < nimi2
True
>>>
>>> nimi3 = "Teppo"
>>> nimi2 == nimi3
False
>>>
>>> nimi3 < nimi2
True
```

Esimerkkejä merkkijonojen vertailuista, jatkoa

```
>>> nimi1 = "matti"
>>> nimi4 = "matilda"
>>> nimi1 < nimi4
False
```

Esimerkki: lämpötilamuunnos

- Seuraava esimerkkiohjelma muuttaa käyttäjän antamia lämpötiloja fahrenheit-asteista celsius-asteiksi tai päinvastoin käyttäjän valinnan mukaan.
- Merkkijonojen vertailulla selvitetään, kumpaan suuntaan käyttäjä haluaa tehdä muunnoksen ja sen mukaan kutsutaan oikeaa funktiota.
- Lisäksi vertailun avulla selvitetään, haluaako käyttäjä tehdä uuden muunnoksen.
- Käyttäjän syötteet muutetaan kokonaan isoiksi tai pieniksi kirjaimiksi, jotta niiden ero ei vaikuttaisi vertailun tulokseen.

Lämpötilamuunnos: koodi

```
def muunna_celsiuksiksi(fahr_asteet):
    celsius_asteet = (fahr_asteet - 32) * 5.0 / 9.0
    return celsius_asteet


def muunna_fahrenheiteiksi(cel_asteet):
    fahrenheit_asteet = 9.0/5.0 * cel_asteet + 32
    return fahrenheit_asteet
```

Lämpötilamuunnos: koodi jatkuu

```
def main():
    jatko = "kylla"
    while jatko != "ei":
        rivi = raw_input("Anna lampotilan yksikko (C/F): ")
        yksikko = rivi.upper()
        if yksikko == "C":
            rivi = raw_input("Lampotila celsius-asteina: ")
            asteet = float(rivi)
            fahrenheit = muunna_fahrenheiteiksi(asteet)
            print asteet, "C on", fahrenheit, "F."
        elif yksikko == "F":
            rivi = raw_input("Lampotila fahrenheiteina: ")
            asteet = float(rivi)
            celsius = muunna_celsiuksiksi(asteet)
            print asteet, "F on", celsius, "C."
```

Lämpötilamuunnos: koodi jatkuu

```
        else:
            print "Virheellinen yksikko, oikea on C tai F"
        rivi = raw_input("Haluatko jatkaa (kylla/ei)? ")
        jatko = rivi.lower()

main()
```

Syötteessä useita lukuja samalla rivillä

- Seuraavassa esimerkissä käyttäjä antaa useita lukuja samalla rivillä.
- Luvut on erotettu toisistaan välilyönnillä.
- Rivi jaetaan ensin osiin `split`-metodilla. Tämän jälkeen luvut ovat listassa, mutta merkkijonoina, ei lukuina.
- Listassa olevat merkkijonot voidaan muuttaa desimaaliluvuiksi tyypinmuunnoksella.
- Ohjelma laskee rivillä olevien lukujen summan ja tulostaa sen.

Lukuja samalla rivillä, koodi

```
def main():
    print "Anna luvut samalla rivilla,"
    print "erota toisistaan valilyonnilla."
    lukurivi = raw_input()
    luvut_tekstina = lukurivi.split()
    summa = 0.0
    for luku in luvut_tekstina:
        summa += float(luku)
    print "Lukujen summa on %.2f" % (summa)


main()
```