TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Programmeerimise süvendatud algkursus ITI0140

2014

Teemad

- Sõne (ingl *string*)
- Järjend (ingl *list*)
- Funktsioon (ingl *function*)
- Ennik või n-korteež (ingl *tuple*)
- Ülesanne

Sõne (ingl *string*)

Milleks kasutatakse?

Tekst-tüüpi muutujate esitamiseks

```
tervitus = 'Tere!'

tervitus = "Tere!"
```

Operatsioonid sõnedega

```
"Tere" + ", Gert!"
Tere, Gert!

len("Tere")
4

" tervitus ".strip()
tervitus

"terekest".replace("e","i").upper()
TIRIKIST
```

Operatsioonid sõnedega

Sõne elementide (tähemärkide) ligipääs ja lõiked (ingl *slice*)

```
tervitus = "Tere!"
print(tervitus[1])
e


print(tervitus[1:3])
er
# s.t [1..3) ehk viimast ei võeta


print(tervitus[1:-1])
ere
# -1 tähendab, et lõpust üks vähem
```

Operatsioonid sõnedega

```
tervitus = "Tere!"
 print(tervitus[0:5:1])
Tere!
# [0..5) ja iga “esimene” element

print(tervitus[::1])
Tere!

print(tervitus[::2])
Tr!
# Iga teine element

# PS! Lõike saab võtta väga kavalalt – googelda
"python slice notation"
```

Tähed sõnes

Igal tähel on vastavuses täisarv:

```
tervitus = "Tere!"

 print(tervitus[1])
e

print(ord(tervitus[1]))
101

print(ord(tervitus[2]))
114

print(chr(ord(tervitus[1]) + 1))
f
```

Vt. veel: https://en.wikipedia.org/wiki/ASCII

Vaata lisaks!

Pythoni dokumentatsioon aitab alati!

http://docs.python.org/3/library/string.html

Või lihtsalt googelda “string python”

PS! Pane tähele, millise Pythoni versiooni dokumentatsiooni vaatad

Järjend (ingl *list*)

Milleks kasutatakse?

Loetelu esitamiseks

Järjenditega realiseeritakse järjekordi (ingl *queue*) ja pinusid (ingl *stack*)

```
tyhi = []

teadlased = ["Turing", "Dijkstra",
"Knuth", "Kolmogorov"]
```

Operatsioonid järjenditega

Kuvamine:

```
print(teadlased)
["Turing", "Dijkstra", "Knuth", "Kolmogorov"]
```

Elementide arv (järjendi pikkus):

```
len(teadlased)
4
```

Kuulumise kontroll:

```
if "Dijkstra" in teadlased:
    print("Dijkstra on teadlane!")
Dijkstra on teadlane!
```

Operatsioonid järjenditega

Järjendi ühe elemendi või elementide lõigu (ingl *slice*) kuvamine:

```
print(teadlased[0])
Turing
# NB! Vastuseks on sõne

print(teadlased[0:2])
["Turing", "Dijkstra"]
# NB! Vastuseks on järjend
# s.t [0..2) ehk viimast ei võeta
```

Operatsioonid järjenditega

Minimaalse või maksimaalse elemendi leidmine:

```
min([42, 12, 99, 66])
>> 12

max([42, 12, 99, 66])
>> 99
```

Operatsioonid järjenditega

Järjendite liitmine:

```
[42, 12] + [99, 66]
[42, 12, 99, 66]
```

NB! Järjendi elementide järjekord on tähtis:

```
[42, 12] == [12, 42]
False

[42, 12] == [42, 12]
True
```

Järjendite sisseehitatud funktsioonid

Elementide lisamine lõppu:

```
l = [2, 4, 6]
l.append(8)
print(l)
[2, 4, 6, 8]
```

Järjendite sisseehitatud funktsioonid

Elementide lisamine määratud kohta:

```
l = [2, 4, 6]
l.insert(1, 8)
print(l)
[2, 8, 4, 6]
# Indekseerimine algab arvust null!
```

Järjendite sisseehitatud funktsioonid

Antud väärtusega (NB!) elemendi eemaldamine

```
l = [2, 4, 6, 8]
l.remove(4)
print(l)
>> [2, 6, 8]
```

Järjendite sisseehitatud funktsioonid

Elemendi eemaldamine:

```
l = [2, 4, 6, 8]
l.pop(2)
print(l)
[2, 4, 8]
```

Ja see pole veel kõik...

Pythoni reference aitab alati!

https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Või lihtsalt googelda “list python”

PS! Pane tähele, millise Pythoni versiooni dokumentatsiooni vaatad

Järjendiga for-tsükkel

Indeksiga variant:

```
l = [2, 4, 6]
for i in range(len(l)):
    print("Element indeksiga ", i, " väärtus on ", l[i])
Element indeksiga 0 väärtus on 2
Element indeksiga 1 väärtus on 4
Element indeksiga 2 väärtus on 6
```

Järjendiga for-tsükkel

Indeksita variant:

```
l = [2, 4, 6]
for i in l:
    print(i)
2
4
6
```

Järjendiga for-tsükkel

Enumerate variant:

```
l = [2, 4, 6]
for i, val in enumerate(l):
    print(i, " : ", val)
0 : 2
1 : 4
2 : 6
```

Funktsioon

Milleks kasutatakse funktsioone?

Meie kontekstis – alamülesande lahendamiseks

```
def alam():
    print("Kutsuti välja funktsioon alam()!")
```

Funktsioon

```
def alam():
    print("Kutsuti välja funktsioon alam()!")

print("Peaprogramm alustab tööd")
alam()
print("Peaprogramm lõpetab")
```

```
Peaprogramm alustab tööd
Kutsuti välja funktsioon alam()!
Peaprogramm lõpetab
```

Argumentidega funktsioon

Argumendid sarnanevad matemaatikast tuntud funktsiooni parameetritele:

```
def ruumala(a, b, c):
    print("Risttahuka ruumala on", a*b*c, "kuupmeetrit.")


ruumala(2, 3, 4)
Risttahuka ruumala on 24 kuupmeetrit.
```

Väärtusega funktsioon

Funktsioon tagastab väärtuse muutujana:

```
def ruumala(a, b, c):
    return a*b*c

x = ruumala(2, 3, 4)
y = ruumala(3, 4, 5)
print(x, y, x + y)
24 60 84
```

Esimesed ohukohad

```
def ruumala(a, b, c):
    return a*b*c

print(ruumala("oops", 3, 4))
?

oopsoopsoopsoopsoopsoopsoopsoopsoopsoopsoopsoops
```

Erinditest (ingl *exception*) räägime hiljem!

Lisainfo

Pythoni reference aitab alati!

http://docs.python.org/3/tutorial/controlflow.html#defining-functions

Või lihtsalt googelda “defining functions python”

PS! Pane tähele, millise Pythoni versiooni dokumentatsiooni vaatad

Ennikud

Ennikud on peaaegu samad nagu järjendid, aga neid ei saa peale loomist muuta (*immutable type*):

```
t = (1, "esimene", 345.43, "tekst")
# NB! t = (1, )


print(t[0])
1

t[0] = 123
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    u[0] = 123
TypeError: 'tuple' object does not support item
assignment
```

Ülesanne

Ülesanne on üleval (kohe) aine kodulehel

https://courses.cs.ttu.ee/pages/ITI0140