MTAT.03.311
Loomuliku keele töötlus *Python*is
(6 EAP)

Karl-Oskar Masing

01.09.2015

Pythoni meeldetuletus

Miks Python?

- Paindlik
  - erinevad paradigmad
  - erinevat tüüpi rakendused
- Loetav
  - “Pseudokoodist Pythoni koodi saamine on taande küsimus”
- Kompaktne
  - funktsionaalne kood väheste ridadega
- Lihtne?

Miks Python?

- Sisseehitatud andmestruktuurid
- Palju standardteeke
- Töötab erinevatel operatsioonisüsteemidel
- Teegid eesti keelega töötamiseks

Õppimine

- Pythoni õpikud
  - Aivar Annamaa “Programmeerimise õpik”
    - https://programmeerimine.cs.ut.ee/
- Harjutamine
  - koduülesannete tegemine
- Teiste koodi lugemine
  - Stack Overflow

Interaktiivne keskkond vs fail

- Interaktiivne keskkond
  - funktsioonide ja arvutuste testimine
  - teekidega tutvumine
  - arvutamine (kalkulaatori asemel)
- Tekstifail (.py)
  - koduülesanded
  - miljoni dollari veebirakendus

Aritmeetika

** → * / // → +-

```
>>> 1 + 5  # liitmine
6
>>> 6 - 8  # lahutamine
-2
>>> 3 * 5  # korrutamine
15
>>> 2 ** 4  # astendamine
16
```

```
>>> 3 / 5  # jagamine
0
>>> 3.0 / 5
0.6
>>> 3.0 // 5  # jagamine + floor
0.0
>>> 12 % 5  # jääk
2
```

Loogika

```
>>> 2 == 2  # võrdne
True
>>> 2 != 2  # mittevõrdne
False
>>> 17 > 8  # suurem
True
```

```
>>> 17 < 8  # väiksem
False
>>> 3 >= 8  # suurem-võrdne
False
>>> 3 <= 8  # väiksem-võrdne
True
```

Loogika

```
>>> True and True
True
>>> True and False
False
>>> not True
False
```

```
>>> True or False
True
>>> False or False
False
>>> (1 < 5 and 1 == 2 or
... not False)
True
```

not → and → or

Muutujate väärtustamine

```
>>> joy = "Meeletu"
>>> joy
'Meeletu'
>>> x = 5.0
>>> x
5.0
```

x •= n ≈ x = x • n

```
>>> x += 5
>>> x
10.0
>>> x *= 2
>>> x
20.0
```

Muutujate nimetamine

- Sobib (esteetika, loetavus)
  - Ingliskeelne nimisõna
  - Ingliskeelne lühend
  - Matemaatiline/Algoritmiline tähis (õiges kontekstis)
- Esimene sümbol
  - Täht (frequency, pos, length, t)
  - Alakriips (_, _methods)
- Järgnevad sümbolid
  - tähed, numbrid, alakriipsud

Muutujate nimede semantika

- Nimi kirjeldab hoitavat väärtust
- Tava
  - mitmesõnaline_nimi_alakriipsude_ja_väiketähtedega
  - konstant või KONSTANT
  - _ (ebaoluline muutuja)
  - class_ (reserveeritud terminitest eristamiseks)
  - _local (faili-/mooduli-/klassi-siseseks kasutamiseks)

Kommentaarid

```
>>> # Kommentaar rea lõpuni
>>> print "Tere"  # Midagi võiks ikka kirjutada

>>> "Sõne kujul kommentaar, real muud olla ei tohi"

>>> """Mitmerealine
... kommentaar"""
```

Funktsioonid

```
>>> def tegevusega_seotud_nimi():
...     # Kood läheb siia
...     return “Tagastusväärtus“
```

Funktsioonid

```
>>> def output_zero():
...         return 0

>>> def contains_eesti(text):
...         return "eesti" in text.lower()
```

Funktsioonid

```
>>> def contains_exactly(text,pattern):
...         return pattern in text
```

- *return* käsuta funktsioon tagastab *None*

Tingimuslaused

```
>>> if kriteerium1_täidetud:
...     pass
...     elif kriteerium2_täidetud:
...     pass
...     else:
...     pass
```

Tingimuslaused

```
>>> def compare(a,b):
...         if a < b:
...             return -1
...         elif a > b:
...             return 1
...         else:
...             return 0
```

Tingimuslaused

```
>>> x = a if condition_met else b
```

(C-s, Javas jne condition_met ? a : b)

samaväärne kui

```
>>> if condition_met:
...     x = a
... else:
...     x = b
```

Konteinerid

- Elementide talletamine
- Elemendid võivad olla erinevat tüüpi

- Sisseehitatud konteinerid:
  - list (järjend)
  - tuple (ennik, vektor)
  - set (hulk)
  - dict (paisktabel, sõnaraamat)
  - string (sõne)

List

```
>>> list_ = list()
>>> list_ = []
>>> list_ = [1, 2, “dog”,[7, “cat”]]
```

- Elemendid järjestatult
- Elementide kättesaamine indeksi abil
  - Indeks algab nullist
- Muudetavate elementidega

List - elementide kättesaamine

```
>>> list_ = [1, 2, “dog”,[7, “cat”]]
>>> list_[0]
1
>>> list_[1]
2
>>> list_[3]
[7, ‘cat‘]
>>> list_[-1]
[7,’cat’]
```

```
>>> list_[-2]
‘dog‘

>>> list_[3][0]
7
>>> list_[3][1]
‘cat‘
```

List - väärtuste seadmine

```
>>> list_ = []
>>> list_[0] = 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
>>> list_ = [1,2,3]
>>> list_[0] = 2
>>> list_
[2, 2, 3]
>>> list_[2] = [7, 8]
```

```
>>> list_
[2, 2, [7, 8]]
>>> list_[2][0] = ‘dog‘
>>> list_
[2, 2, [‘dog‘,8]]
```

List - lõikamine

```
>>> list_ = [1,2,3,4,5]
>>> list_[:]  # koopia tervest listist
[1, 2, 3, 4, 5]
>>> list_[:3]
[1, 2, 3]
>>> list_[3:]
[4, 5]
>>> list_[1:4]
[2, 3, 4]
```

```
>>> list_[::-1]
[5, 4, 3, 2, 1]

>>> list_[1:3] = [“dog”,”cat”,”horse”]
>>> list_
[1, ”dog”, “cat”, “horse”, 5]
```

List - kasulikke meetodeid

len(list_) - list_ elementide arv

list_.index(k) - k esimese esinemise indeks

list_.count(k) - k esinemiste arv

list_.sort() - sorteerib kohapeal

sorted(list_) - tagastab uue listi sorteeritult

list_.append(k) - lisab elemendi k lõppu

list_.extend(L) - lisab listi L elemendid lõppu

list_.insert(k,idx) - lisab k kohale idx, lükkab järgmisi edasi (aeglane)

list_.pop() - tagastab ja eemaldab viimase elemendi

list_.pop(i) - eemaldab i-nda elemendi (aeglane)

Tuple

```
>>> tuple_ = tuple()
>>> tuple_ = ()  # suht mõttetu
>>> tuple_ = (1, 2, “dog”,[7, “cat”])
```

- Elemendid järjestatult
- Elementide kättesaamine indeksi abil
  - Sarnane listiga
- Elemente ei saa muuta
  - Vajadusel uus ennik teha

Sõne

```
>>> string_ = str()
>>> string_ = “”
>>> string_ = “Imeilus”
```

- Enniku omadused, erinevad meetodid
  - pm tähtede ennik

Sõne

```
>>> "Eesel vaatas vihast vasikat"
'Eesel vaatas vihast vasikat'
>>> 'Eesel vaatas vihast vasikat'
'Eesel vaatas vihast vasikat'
```

```
>>> """Eesel vaatas vihast vasikat
... mitu rida"""
'Eesel vaatas vihast vasikat\nmitu rida'
>>> "Eesel vaatas vihast vasikat" \
... "mitu rida"
'Eesel vaatas vihast vasikatmitu rida'
```

Sõne - kasulikke meetodeid

string_.(r)find(s) - esimese/viimase *s* leiu indeks

string_.count(s) - *s* alamsõnede arv

string_.lower() - string_ läbiva väiketähega

string_.strip() - eemaldab whitespace'i mõlemalt poolt

string_.replace(s1,s2) - asendab kõik s1 -> s2

Sõne - eriti oluline

```
>>> string_ = “Polla on ilus koer”
>>> split_string_ = string_.split(" ")
>>> split_string_
['Polla', 'on', 'ilus', 'koer']
>>> ','.join(split_string_)
'Polla,on,ilus,koer'
```

Set

```
>>> set_ = set()
>>> set_ = {1,2,3}
```

- Elemendid suvalises järjestuses
- Elementide kättesaamine hulgaoperatsioonide ning iteraatori abil
- Muudetavad elemendid

Set - operatsioonid

```
>>> A = {1,2,3,4,5}
>>> B = {2,3,5,6}
>>> A | B  # ühend
set([1, 2, 3, 4, 5, 6])
>>> A & B  # ühisosa
set([2, 3, 5])
```

```
>>> A - B  # vahe
set([1, 4])
>>> A <= B  # alamhulk
False
>>> A >= B  # ülemhulk
False
```

Dict

```
>>> dict_ = dict()
>>> dict_ = {}
>>> dict_ = {‘a‘: 1, ’b’: 2, ‘c’: 3}
```

- Elemendid suvalises järjekorras
- Elementide kättesaamine võtmete abil
- Elemendid muudetavad

Dict - operatsioonid

```
>>> dict_ = {}
>>> dict_[‘a’] = 1
>>> dict_
{'a': 1}
>>> dict_[‘a‘]
1
>>> dict_[‘b’] = 2
```

```
>>> dict_.keys()
['a', 'b']
>>> dict_.values()
[1, 2]
```

Defaultdict

```
>>> from collections import defaultdict
>>> dict_ = defaultdict(int)
>>> dict_ = defaultdict(list)
>>> dict_ = defaultdict(lambda: defaultdict(dict))
```

- Sarnane tavalisele *dict*'ile
- Kui vastavat võtit sõnastikus pole, on võtmele vastava väärtuse vaikeväärtus andmetüübi vaikeväärtus

Dict vs defaultdict

```
>>> def get_freq_table(words):
…         freq_table = dict()
…         for word in words:
…             if word in freq_table:
…                 freq_table[word] += 1
…             else:
…                 freq_table[word] = 1
```

```
>>> from collections import defaultdict
>>> def get_freq_table(words):
…         freq_table = defaultdict(int)
…         for word in words:
…             freq_table[word] += 1
```

Konverteerimine

```
>>> list_ = [1,2,3,1,4,6,2]
>>> set(list_)
set([1, 2, 3, 4, 6])
>>> list(set(list_))
[1, 2, 3, 4, 6]
>>> tuple(list_)
(1, 2, 3, 1, 4, 6, 2)
>>> tuple(set(list_))
(1, 2, 3, 4, 6)
```

```
>>> list_ = [['a',1],['b',2],['c',3]]
>>> dict(list_)
{'a': 1, 'c': 3, 'b': 2}
>>> list_ = [('a',1),('b',2),('c',3)]
>>> dict(list_)
{'a': 1, 'c': 3, 'b': 2}
>>> list_ = (('a',1),('b',2),('c',3))
>>> dict(list_)
{'a': 1, 'c': 3, 'b': 2}
```

Tsüklid - while

```
>>> while condition_met:
…     pass
```

OHT LÕPMATUKS TSÜKLIKS!

```
>>> idx = 0
>>> while idx < 10:
…       parse_sentence(idx)
```

Iteraator

- Objekt, mis võimaldab ükshaaval andmestruktuurist elemente pärida
- Saab läbida ühe korra
- Meetod *next()*, mis tagastab järgmise elemendi
- Itereeritav objekt tagastab iteraatori *__iter__()* meetodi abil
- Itereeritav ei tohi olla enda iteraator

Tsüklid - for

```
>>> for x in iterable:
…       process(x)
```

- Muutuja *x* väärtustatakse igal iteratsioonil ühe *iterable*'i poolt tagastatud väärtusega

- Pärast tsükli lõppu on *x* väärtuseks viimane tagastatud väärtus

Tsüklid - for

```
>>> for element in [1,2,3,4,5]:
...     print element,
1 2 3 4 5

>>> for i in range(5):
...     print i,
0 1 2 3 4

>>> range(1,6)
[1, 2, 3, 4, 5]
```

Tsüklid - for

```
>>> list_ = ["dog","cat","duck","sheep"]
>>> range(len(list_))
[0, 1, 2, 3]

>>> def find_index(element,list_):
...       for i in range(len(list_)):
...         if list_[i] == element:
...           return i
```

Tsüklid - for

```
>>> enumerate(list_)
[(0, 'dog'), (1, 'cat'), (2, 'duck'), (3, 'sheep')]

>>> def find_index(element,list_):
...       for i, candidate_item in list_:
...         if element == candidate_item:
...           print i
```

Generaator

- Nagu iteraator, kuid
  - Ei tööta eksisteerival andmestruktuuril
    - Ei vaja mälu andmestiku hoidmiseks
  - Genereerib igal päringul järgmise elemendi
  - Otstarbekas ühekordselt kasutatavate andmete korral
    - Teeb genereerimiseks tööd

xrange(n) - range(n)’i vaste laisa väärtustamisega

Generaator

```
>>> def my_xrange(n):
…         i = 0
…         while i < n:
…             yield i
…             i += 1
…
>>> range_ = my_xrange(10)
>>> for i in range_:
…         print i,
0 1 2 3 4 5 6 7 8 9
```

```
>>> (x*x for x in range(10))
<generator object <genexpr> at
0x7f34a9691640>


>>> squares = (x*x for x in range(10))
>>> for square in squares:
...         print square,
...
0 1 4 9 16 25 36 49 64 81
```

Set comprehension

$$\left\{ x^2 : x \in [0, 10) \, , 2 \mid x \right\}$$

```
>>> {x**2 for x in range(10) if x % 2 == 0}
set([0, 16, 4, 64, 36])
```

List comprehension

```
>>> squares = []
>>> for i in range(10):
…         squares.append(i*i)
```

Aeglane ja lohisev

```
>>> map(lambda x: x*x, range(10))
```

Aeglane ja kole

```
>>> [x*x for x in range(10)]
```

Kiire ja ilus

List comprehension

[f(var) for var in iterable if f’(var)]

```
>>> [str(x) for x in range(10)]
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

>>> [[x] for x in range(10) if x % 2 == 0]
[[0], [2], [4], [6], [8]]
```

Dict comprehension

```
>>> {x:y for x,y in enumerate(range(5,10))}
{0: 5, 1: 6, 2: 7, 3: 8, 4: 9}
```

Import

```
>>> from math import *  # ohtlik
>>> log(10)
2.302585092994046

>>> from math import log
>>> log(10)
2.302585092994046

>>> import math
>>> math.log(10)
2.302585092994046
```

KASUTADA AINULT LAPSEVANEMA LOAL

ENAMASTI OK

KASUTADA JULGESTI