TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Programmeerimise süvendatud algkursus ITI0140

2014

Teemad

- Moodulid (*ingl* modules)
- Erindid (*ingl* exceptions)

Kasulik viide

Docstring + muud konventsioonid

Google "google python style guide"

http://google-styleguide.googlecode.com/svn/trunk/pyguide.html

(Google) Docstring stiil

Igal funktsioonil **peab olema** docstring, välja arvatud kui:

1) funktsioon ei ole väljastpoolt nähtav (välja kutsutav väljastpoolt)

2) funktsioon on väga lühike

3) funktsioon on väga lihtne

Docstring peaks olema oma olemuselt **funktsiooni kasutusjuhend**, mitte kirjeldama implementatsiooni eripärasid (ära kirjuta, et kasutab X algoritmi, Y koodimustrit ja Z kavalust).

Moodulid

Mis on moodulid?

Moodulid on selliste **funktsioonide** (koos muu vajalikuga) **kogumikud**, mis täidavad mingit ühist ülesannet või sobivad kokku mingil muul põhjusel.

Näited: matemaatikafunktsioonide moodul (math), juhuarvude genereerimismoodul (random) jne.

Moodulid

Moodulid on sellised programmi komponendid, mida võib kasutada korduvalt erinevates projektides.

Moodulite funktsioonid on paindlikud ja kasulikud erinevate lähteandmetega.

Moodulid → Paketid

Igaüks võib teha oma mooduleid, neid jagada ja kasutada mitmetes projektides.

Mooduleid saab kombineerida komplektideks, mida nimetatakse pakettideks.

Pythoni jaoks on olemas palju võimsaid pakette (*ingl* package).

Vaata: https://pypi.python.org/pypi
(PyPI - the Python Package Index)

(There are currently 49477 packages here.)

Moodulite kasutamine

```
print("sin(pi) =", round(sin(pi), 3))
print("cos(pi) =", round(cos(pi), 3))
```

```
>>> Traceback (most recent call last):
    print("sin(pi) =", round(sin(pi), 3))
NameError: name 'sin' is not defined
```

Moodulite kasutamine

```
print("sin(pi) =", round(math.sin(math.pi), 3))
print("cos(pi) =", round(math.cos(math.pi), 3))
```

```
>>> Traceback (most recent call last):
    print("sin(pi) =", round(math.sin(math.pi), 3))
NameError: name 'math' is not defined
```

Moodulite kasutamine

```
import math

print("sin(pi) =", round(math.sin(math.pi), 3))
print("cos(pi) =", round(math.cos(math.pi), 3))
```

```
>>> sin(pi) = 0.0
cos(pi) = -1.0
```

Moodulite kasutamine

Saab lühendada imporditud mooduli nime
NB! Kasutada ainult väga pikkade nimede puhul

```
import math as m

print("sin(pi) =", round(m.sin(m.pi), 3))
print("cos(pi) =", round(m.cos(m.pi), 3))
```

```
>>> sin(pi) = 0.0
cos(pi) = -1.0
```

Moodulite kasutamine

Saab importida ainult teatud vajaminevaid asju (muutub teatud piirist mõttetuks)

```
from math import sin, cos, pi

print("sin(pi) =", round(sin(pi), 3))
print("cos(pi) =", round(cos(pi), 3))
```

```
>>> sin(pi) = 0.0
cos(pi) = -1.0
```

Moodulite näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
      x: numeric value

    Returns:
      The square root of x (e.g., square_root(4) = 2.0).
    """
    return math.sqrt(x)

print("Hello from gert.py!")
print("The square root of 4 is", square_root(4))
```

```
>>> Hello from gert.py!
The square root of 4 is 2.0
```

Moodulite näide: ivor.py

```
"""
Lecture 6 modules example

@author Ivor
"""
import math

def my_square_root(x):
    """
    Returns the square root of x.

    Args:
      x: value to take square root of

    Returns:
      square root of x
    """
    return math.sqrt(x)

print("Hello from ivor.py!")
print("The square root of 9 is", my_square_root(9))
```

```
>>> Hello from ivor.py!
The square root of 9 is 3.0
```

Mooduli näide: module.py

```
"""
Square root module

@author gert
@author Ivor
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
      x: numeric value

    Returns:
      The square root of x (e.g., square_root(4) = 2.0).
    """
    return math.sqrt(x)

print("Hello from module.py!")
```

Mooduli näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import module

print("Hello from gert.py!")
print("The square root of 4 is", module.square_root(4))
```

Kasutame mooduli ruutjuure funktsiooni

```
>>> Hello from module.py!
Hello from gert.py!
The square root of 4 is 2.0
```

Programmi käivitades ilmneb midagi ebasoovitavat!

Mooduli näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import module

print("Hello from gert.py!")
print("The square root of 4 is", module.square_root(4))
```

Kasutame mooduli ruutjuure funktsiooni

```
>>> Hello from module.py!
Hello from gert.py!
The square root of 4 is 2.0
```

?

Mooduli näide: module.py

```
"""
Square root module

@author gert
@author Ivor
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
      x: numeric value

    Returns:
      The square root of x (e.g., square_root(4) = 2.0).
    """
    return math.sqrt(x)

if __name__ == "__main__":
    print("Hello from module.py!")
```

NB!

Mooduli näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import module

print("Hello from gert.py!")
print("The square root of 4 is", module.square_root(4))
```

```
>>> Hello from gert.py!
The square root of 4 is 2.0
```

Enam ei prindi “hello from module.py”

Mooduli näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import module

if __name__ == "__main__":
    print("Hello from gert.py!")
    print("The square root of 4 is", module.square_root(4))
```

Korrektne tava on alati kasutada __name__ == "__main__"

```
>>> Hello from gert.py!
The square root of 4 is 2.0
```

Mooduli näide: gert.py

```
"""
Lecture 6 modules example

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    print("The square root of 4 is", module.square_root(4))

if __name__ == "__main__":
    main()
```

Kasuta main() funktsiooni!

```
>>> Hello from gert.py!
The square root of 4 is 2.0
```

Erind (*ingl* exception)

Mis on erind?

Erind on programmi töö käigus tekkiv ootamatu olukord (viga).

See tähendab, et programmi lähtekood on **süntaktiliselt** korrektne, aga programmi käivitades tekib töö käigus viga.

Erind

Kõik erindid ei ole programmi töö jaoks tingimusteta fataalsed (*ingl* unconditionally fatal). See tähendab, et teatud juhtudel saab programm tööd jätkata kui eriolukord suudetakse lahendada.

Paljusid erindeid saab töödelda ja seda tegevust nimetatakse erinditöötluseks (*ingl* exception handling).

Erind

```
"""
Lecture 6 exception handling

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    print("The square root of -4 is", module.square_root(-4))

if __name__ == "__main__":
    main()
```

```
>>> Hello from gert.py!
Traceback (most recent call last):
  File "D:\Workspace2014\Python34\gert.py", line 13, in <module>
    main()
  File "D:\Workspace2014\Python34\gert.py", line 10, in main
    print(module.square_root(-4))
  File "D:\Workspace2014\Python34\module.py", line 18, in square_root
    return math.sqrt(x)
ValueError: math domain error
```

Mooduli näide: module.py

```
"""
Square root module

@author gert
@author Ivor
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
      x: numeric value

    Returns:
      The square root of x (e.g., square_root(4) = 2.0).
    """
    return math.sqrt(x)

if __name__ == "__main__":
    print("Hello from module.py!")
```

module.py

```
"""
Square root module

@author gert & ivor
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
        x: numeric value

    Returns:
        The square root of x (e.g., square_root(4) = 2).
        Returns -1 if math domain error occurs.
    """
    try:
        s = math.sqrt(x)
    except ValueError:
        return -1
    else:
        return s

if __name__ == "__main__":
    print("Hello from module.py!")
```

Ära unusta docstringi muutmata!

Erind

```
"""
Lecture 6 exception handling

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    print("The square root of -4 is", module.square_root(-4))

if __name__ == "__main__":
    main()
```

```
>>> Hello from gert.py!
The square root of -4 is -1
```

module.py

```
"""
Square root module

@author gert & ivor
"""
import math

def square_root(x):
    """
    Calculates the square root of x.

    Args:
        x: numeric value

    Returns:
        The square root of x (e.g., square_root(4) = 2).

    Raises:
        ValueError: In case x < 0
    """
    try:
        s = math.sqrt(x)
    except ValueError as e:
        print("module.square_root(" + str(x) + "):", e)
        raise
    else:
        return s

if __name__ == "__main__":
    print("Hello from module.py!")
```

Kui funktsioon tõstatab erindeid, siis kirjuta see docstringi!

Veateade ja erindi uuesti tõstatamine

Erind

```
"""
Lecture 6 exception handling

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    print("The square root of -4 is", module.square_root(-4))

if __name__ == "__main__":
    main()
```

OK, aga vähe kasu!

```
>>> Hello from gert.py!
Traceback (most recent call last):
  File "D:\Workspace2014\Python34\gert.py", line 20, in <module>
module.square_root(-4): math domain error
    main()
  File "D:\Workspace2014\Python34\gert.py", line 16, in main
    print("The square root of -4 is", module.square_root(-4))
  File "D:\Workspace2014\Python34\module.py", line 22, in square_root
    s = math.sqrt(x)
ValueError: math domain error
```

Erind

```
"""
Lecture 6 exception handling

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    try:
        result = module.square_root(-4)
    except:
        # TODO: Figure out what to do in case of an exception
        print("square_root() exception")
    else:
        print("The square root of -4 is", result)


if __name__ == "__main__":
    main()

>>> Hello from gert.py!
module.square_root(-4): math domain error
square_root() exception
```

Erind

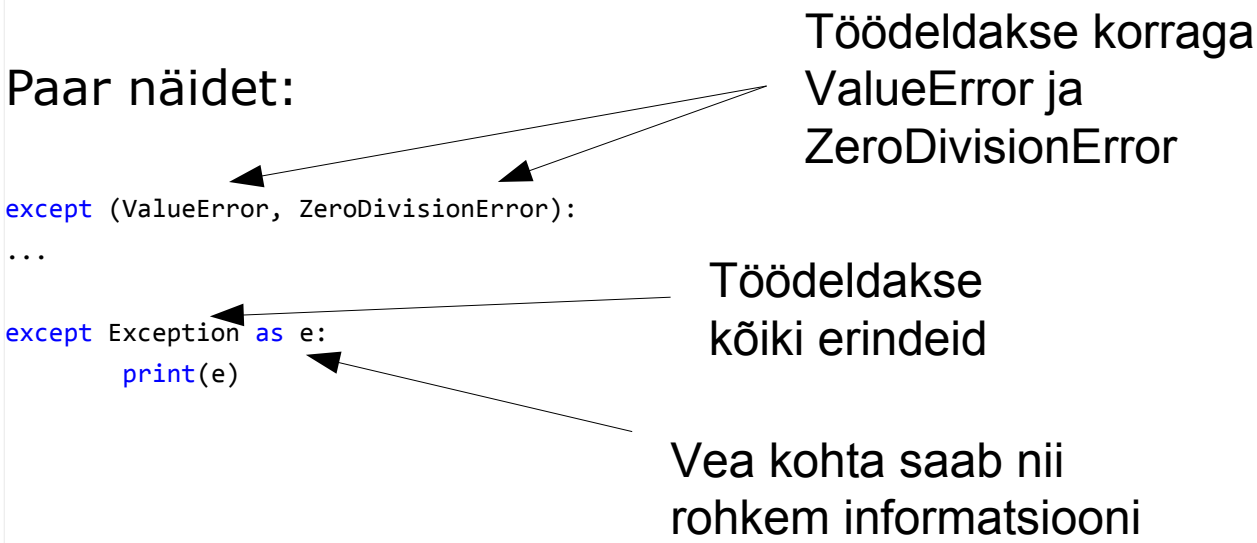
```
"""
Lecture 6 exception handling

@author gert
"""
import module

def main():
    print("Hello from gert.py!")
    try:
        result = module.square_root("blah")
    except:
        # TODO: Figure out what to do in case of an exception
        print("square_root() exception")
    else:
        print("The square root of -4 is", module.square_root(result))


if __name__ == "__main__":
    main()

>>> Hello from gert.py!
square_root() exception
```

Sisendiks sõne!

Erind

Erindeid on palju (`ValueError`, `TypeError`, `ZeroDivisionError`, `IndexError` jne). Neid on võimalik töödelda eraldi, kombineerida kokku või töödelda kõiki korraga.

Paar näidet:



Erind

Programmeerija peab ise otsustama kuidas peaks programm erindite puhul käituma, aga **erindeid ei tohi <u>üldjuhul</u> kasutada programmi töö juhtimiseks** (*ingl* control flow). Vastasel juhul on koodi raskem lugeda ja see võib tekitada arusaamatusi.

(lisainfo google "principle of least astonishment")

Lisainfo: google "python exception"

https://docs.python.org/3/tutorial/errors.html

Faili kirjutamine

Kirjutamine = "w" ((re)write)
Lõppu lisamine = "a" (append)

Reavahetus

```
f = open('filename.txt', 'w')
f.write("Testing testing 1, 2, 3!\n")
f.writelines(["First", "second", "third"])
f.writelines(["Hello\n", "there\n", "hello\n"])
f.close()
```

```
Testing testing 1, 2, 3!
FirstsecondthirdHello
there
hello
```

Ülesanne

The Hound of the Baskervilles: **http://www.gutenberg.org/cache/epub/3070/pg3070.txt**

Tekst on vaja jaotada sõnadeks. Sõnaks loeme antud juhul kõik järjest asetsevad tähed, mille kohta Pythoni **isalpha**() funktsioon vastab tõeselt. Sõnade ümber asetsevad muud sümbolid. Sõnadeks jaotamine ei peaks olema tõstutundlik. Leitud sõnade pealt on vaja teksti kohta tekitada statistika.

Lahendus peab koosnema **kahest (2)** failist: **moodul**, mis sisaldab teksti töötlevaid ja statistilisi funktsioone, ja **peaprogramm**, mis seda moodulit kasutab.

Moodulis peavad olema esindatud järgmised funktsioonid:

1) **process_file**(file) - võtab argumendiks avatud faili ja tagastab teksti sõnadeks jaotamise tulemusena tekkiva järjendi (st kõik sõnad tekstist nende leidmise järjekorras, sh kordused)

2) **count_words**(lst) - võtab argumendiks sõnade järjendi ja tagastab sõnastiku, kus võti on sõna ja väärtus on selle sõna esinemiste arv

3) **find_top_words**(dct, n) - võtab argumendiks *count_words()* tagastatava sõnastiku ja tagastab sõnastiku, kus võti on sõna pikkus ja väärtus on n sagedasemalt esinenud sõna vastava pikkusega ja esinemiste arvuga (st omakorda kuni n pikkused järjendid ennikutest või sõnastikud)

4) **print_top_words**(dct, file) - võtab vastu *find_top_words()* tagastatava sõnastiku ja trükib tulemustest viisakalt vormindatud tabeli faili, sõnade pikkuse järgi kasvavalt, sõnade esinemise sageduse järgi kahanevalt, sama sageduse korral tähestiku järjekorras

Faili avamine ja sulgemine võivad olla peaprogrammis. Failifunktsioonide kasutamisel peab olema tehtud **erinditöötlus**, st näiteks olematu faili avamine peaks trükkima vastavasisulise teate aga mitte Pythoni vea (Traceback).

Kasutada docstringi ja korrektset ülesehitust (main() jne)

Ülesanne

print_top_words() väljundi näide:

length	word	count
12	baskervilles	1
9	gutenberg	1
6	arthur	1
6	project	1
5	conan	1
5	doyle	1
5	hound	1
3	the	2
2	by	1
2	of	1
1	s	1