

Function, Class, Regular expressions & other tools

Sugarkhuu Radnaa

Py4Econ in Ulaanbaatar

py4econ@gmail.com

30 декабря 2021 г.

Week 3: Learning objectives

Get to know:

- ① Functions and Modules
- ② Class (OOP basics)
- ③ Regex
- ④ Additional
 - One-liner lambda
 - Exception handling
 - Debugging

Function

Function - not to repeat one action again and again

↓

```
1 def functionName(input1, input2, ...):  
2     """ this function is ... """  
3  
4     expression  
5  
6     return output(s)  
7  
8  
9 import numpy as np
```

Documenting a function is crucial!

Docstring prototype

```
13
14 def my_function(name, age):
15     """
16
17
18     Parameters
19     -----
20     name : str
21         name of person.
22     age : int
23         age of person.
24
25     Returns
26     -----
27     Grades
28
29     """
30
31
32 help(my_function)
33 print(my_function.__doc__)
34
```

Function arguments

- Positional (“input”)
- Keyword (“input=value”) - with a default value
- Optional positional (*args)
- Optional keyword (**kwargs)

Scope of variable

```
93
94 # variable scope
95
96 x = 20 # global by default
97
98 def my_func(ageby=5):
99     x = 5 # local by default
100
101     # global x # use global variable inside a function
102
103     x = x + ageby
104
105     return x
106
107
108
109
110
111
112
```

```
IPython 7.22.0 -- An enhanced Interactive Python.
In [1]: x = 20 # global by default
...:
...: def my_func(ageby=5):
...:     x = 5 # local by default
...:     # global x # use global variable inside a function
...:     x = x + ageby
...:     return x
...:
In [2]: my_func()
Out[2]: 10
In [3]: x
Out[3]: 20
```

```
93
94 # variable scope
95
96 x = 20 # global by default
97
98 def my_func(ageby=5):
99     x = 5 # local by default
100
101     global x # use global variable inside a function
102
103     x = x + ageby
104
105     return x
106
107
108
109
110
111
112
```

```
IPython 7.22.0 -- An enhanced Interactive Python.
In [1]: x = 20 # global by default
...:
...: def my_func(ageby=5):
...:     x = 5 # local by default
...:     global x # use global variable inside a function
...:     x = x + ageby
...:     return x
...:
In [2]: my_func()
Out[2]: 25
In [3]: x
Out[3]: 25
```

As your program gets longer, you may want

- to split it into several files for easier maintenance
- to use also a handy function that you have written in several programs without copying its definition into each program.

To support this, Python has a way to put definitions in a file and use them. Such a file is called a *module*; definitions from a module can be imported into other modules or script.

Def: Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

Important concepts:

- Object (instance)
- Method
- Inheritance (super, child classes)
- Setters and getters
- Variable accessibility – Public/Private/Protected

One liner and Lambda

Helpful for writing concise codes

```
257
258 # one liner
259 lst = [i if np.mod(i,2)==0 else 5 for i in range(0,10)]
260 # Output - [0, 5, 2, 5, 4, 5, 6, 5, 8, 5]
261
262 # lambda
263 func = lambda a : a + 10
264 func(5)
265 # Output - 15
266
267 my_list = [1, 5, 4, 6, 8, 11, 3, 12]
268 new_list = list(filter(lambda x: (x%2 == 0) , my_list))
269 # Output - [4, 6, 8, 12]
```

Exception handling

Exceptions occur from time to time in any sort of job. If you know what kind of errors potentially could occur and implement “remedy(ies)” on it beforehand

```
373  
374     try:  
375         # do main job  
376     except:  
377         # do another - Hereby, you are handling an exception!  
378
```

In IDEs (VSCode):

- Actions at breakpoints: continue, step over, step into, step out
- Additional actions at breakpoints - expression, hit count, log message
- Watch
- Exceptions

Hand inputted breakpoint:

- `import pdb, pdb.set_trace()`

Regular expressions

Search "pattern" from text and do an action Package "re":

- compile
- findall – find all matches
- match – match at the beginning of string
- search – first match anywhere in string
- split – split string into pieces at pattern points
- sub – replace match by user input
- group
- special characters

Regular expressions special characters

Regular Expressions Cheat Sheet by [DaveChild](#)

A quick reference guide for regular expressions (regex), including symbols, ranges, grouping, assertions and some sample patterns to get you started.

Anchors

| | |
|--------------------|---|
| <code>^</code> | Start of string, or start of line in multi-line pattern |
| <code>\A</code> | Start of string |
| <code>\$</code> | End of string, or end of line in multi-line pattern |
| <code>\Z</code> | End of string |
| <code>\b</code> | Word boundary |
| <code>\B</code> | Not word boundary |
| <code>\<</code> | Start of word |
| <code>\></code> | End of word |

Character Classes

| | |
|-----------------|-------------------|
| <code>\c</code> | Control character |
| <code>\s</code> | White space |
| <code>\S</code> | Not white space |
| <code>\d</code> | Digit |
| <code>\D</code> | Not digit |
| <code>\w</code> | Word |
| <code>\W</code> | Not word |
| <code>\x</code> | Hexadecimal digit |
| <code>\O</code> | Octal digit |

POSIX

| | |
|-----------------------|--------------------|
| <code>[upper:]</code> | Upper case letters |
|-----------------------|--------------------|

Quantifiers

| | | | |
|----------------|-----------|--------------------|-----------|
| <code>*</code> | 0 or more | <code>{3}</code> | Exactly 3 |
| <code>+</code> | 1 or more | <code>{3,}</code> | 3 or more |
| <code>?</code> | 0 or 1 | <code>{3,5}</code> | 3, 4 or 5 |

Add a `?` to a quantifier to make it ungreedy.

Escape Sequences

| | |
|-----------------|----------------------------|
| <code>\</code> | Escape following character |
| <code>\Q</code> | Begin literal sequence |
| <code>\E</code> | End literal sequence |

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

Common Metacharacters

| | | | |
|-------------------|-------------------|----------------|-----------------|
| <code>^</code> | <code>[</code> | <code>.</code> | <code>\$</code> |
| <code>{</code> | <code>*</code> | <code>(</code> | <code>\</code> |
| <code>+</code> | <code>)</code> | <code> </code> | <code>?</code> |
| <code><</code> | <code>></code> | | |

The escape character is usually `\`.

Special Characters

| | |
|-----------------|----------|
| <code>\n</code> | New line |
|-----------------|----------|

Groups and Ranges

| | |
|---------------------|---|
| <code>.</code> | Any character except new line (<code>\n</code>) |
| <code>(a b)</code> | a or b |
| <code>(...)</code> | Group |
| <code>(?...)</code> | Passive (non-capturing) group |
| <code>[abc]</code> | Range (a or b or c) |
| <code>[^abc]</code> | Not (a or b or c) |
| <code>[a-q]</code> | Lower case letter from a to q |
| <code>[A-Q]</code> | Upper case letter from A to Q |
| <code>[0-7]</code> | Digit from 0 to 7 |
| <code>\x</code> | Group/subpattern number "x" |

Ranges are inclusive.

Pattern Modifiers

| | |
|------------------|--|
| <code>g</code> | Global match |
| <code>i *</code> | Case-insensitive |
| <code>m *</code> | Multiple lines |
| <code>s *</code> | Treat string as single line |
| <code>x *</code> | Allow comments and whitespace in pattern |
| <code>e *</code> | Evaluate replacement |
| <code>U *</code> | Ungreedy pattern |
| <code>*</code> | PCRE modifier |

String Replacement

Additional reading: Why is “class” useful?

Please read most upvoted answer by "dantiston":
[https://stackoverflow.com/questions/33072570/
when-should-i-be-using-classes-in-python](https://stackoverflow.com/questions/33072570/when-should-i-be-using-classes-in-python)

Homework

- 1 Task 1
- 2 Task 2
- 3 Task 3
- 4 Task 4
- 5 Task 5
- 6 Task 6

Deadline: 31 December, 2021

Note: Create a github repo from the start and populate it with your results

Task 1: Function arguments

Jupyter notebook дээр зөвхөн `args`, зөвхөн `args=value`, зөвхөн `*args`, зөвхөн `**kwargs` аргументуудтай (4 тусдаа функц) болон эдгээрийн холимог (2 функц) байгуулж, ажиллуулж үзүүл. Нийт 6 функц.

Task 2: Class

Create a bank deposit class which you can

- withdraw money
- deposit
- check the balance

Show on Jupyter notebook how it works

Examples:

- <https://www.engineeringbigdata.com/python-atm-code-for-account-balance-withdraw-and-deposit-functions/>
- <https://www.geeksforgeeks.org/python-program-to-create-bankaccount-class-with-deposit-withdraw-function/>
- <https://www.vtupulse.com/python-programs/python-program-using-classes-and-objects-to-deposit-and-withdraw-money-in-a-bank-account/>
- Tkinter GUI - <https://www.youtube.com/watch?v=SF-enJWjekY&list=PLtMugc7g4GapTtbhzODIjw7FJK-xJEBEE&index=16>

Task 3: Exception

- 1 Generate array of 1000 random integers in numpy and create an array with only the negative even numbers.
- 2 Loop one by one through this array . . .
- 3 . . .when negative odd, then raise “odd error” and continue to next loop,
- 4 . . .when even but positive, raise “sign error” and continue to next loop . . .
- 5 . . .when “negative even”, then append your list.

Task 4: Debugging

- Debugging гэж юу вэ?
- Breakpoint – ийн ямар 3 төрөл (log message г.м) Vscode дээр байдаг вэ?
- Debug-ийн step into, step over, stop out – ийн ялгааг тайлбарла

Task 5: Regex

Өөрөө зохиох эсвэл бэлэн текст интернетээс олж regex-ийн дараах үйлдлүүдийг ашиглан текстүүд гаргаж авах жишээнүүд үзүүл (нэг бүр дээр нь бус хамтад нь ашиглаж болно. Гэхдээ доорх бүх тэмдгээс ядаж 1 удаа ашиглаарай)

- `[a-zA-Z0-9]`, `[a-z]`, `[A-Z]`, `[0-9]`
- `\d`, `\D`, `\w`, `\W`, `\s`
- `^`, `$`, `?`, `*`, `+`, `.`
- `{m,n}`, `{,n}`, `{m,}`, `{n}`
- Look behind, Look ahead, Negative look behind, Negative look ahead

Task 6: Module

- Өгсөн тоон list-ний нийлбэр, ялгавар, үржвэрийг олдог 3 тусдаа функц бүхий модуль файл үүсгэ.
- Дээрх модульд “main” гэдэг функц нэм. Уг функц нь “жишээ” list-ний хувьд дээрх 3 функцийг ажиллуулан үр дүн гарч буйг хэвлэн гаргаж харуулдаг байна.
- `if __name__ == '__main__':` дотор `main()` функцийг ажиллуулна (`do something`-ийн оронд байршин)
- `python “yourModuleName”.py` гэж terminal дээр уншуулахад гарах үр дүн юу вэ? (screenshot байхад болно)
- Өөр файл дотроос “`import yourModuleName`” гэж импортлоход өмнөх хэсгийн үр дүнгүүд хэвлэгдэж гарахгүй байгаа. Яагаад?

Check out on (answer of Fooz) "`if __name__ == '__main__':`":
<https://stackoverflow.com/questions/419163/what-does-if-name-main-do>

Thank you!