Advanced Python Programming Course

Lecture 3.

Lambda, map(), filter(), reduce(), zip()

Code formatting. Linters

Assoc. Prof. Kovalenko S.M.
Department of software engineering and intelligent management technologies,
NTU “KhPI”

Lambda

- The following terms may be used interchangeably depending on the programming language type and culture:

- `Anonymous functions`
- `Lambda functions`
- `Lambda expressions`
- `Lambda abstractions`
- `Lambda form`
- `Function literals`

Lambda example

```
lambda a, b: a + b
```

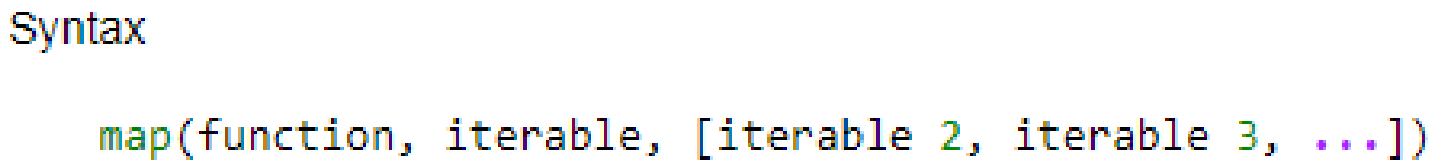
In the example above, the expression is composed of:

- The keyword: `lambda`
- A bound variables: `a` and `b`
- A body: `a + b`

Assoc. Prof. Svitlana Kovalenko,
dept. of Software Engineering and Intelligent Management Technologies, NTU “KhPI”

map()

`map` applies a function to all elements of an iterable object.



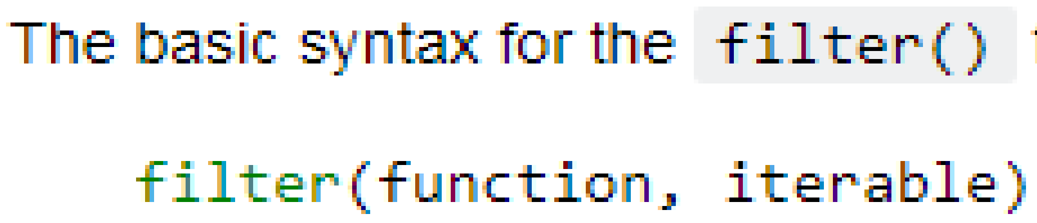
Syntax

```
map(function, iterable, [iterable 2, iterable 3, ...])
```

Assoc. Prof. Svitlana Kovalenko,
dept. of Software Engineering and Intelligent Management Technologies, NTU “KhPI”

filter()

- The Python built-in filter() function can be used to create a new iterator from an existing iterable that will efficiently filter out elements using a function that we provide



The basic syntax for the `filter()` function is:

```
filter(function, iterable)
```

reduce()

- `reduce()` is a function for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list.
- Python’s `reduce()` operates on any iterable and performs the following steps:
  - **Apply** a function to the first two items in an iterable and generate a partial result.
  - **Use** that partial result, together with the third item in the iterable, to generate another partial result.
  - **Repeat** the process until the iterable is exhausted and then return a single cumulative value.

Code formatting

PEP 8

- https://www.python.org/dev/peps/pep-0008/

- “Know when to be inconsistent - sometimes style guide recommendations just aren’t applicable. When in doubt, use your best judgment.”

Black

- *Black* is uncompromising the Python code formatter.
- Installing Black

```
pip install black
```

- To format Jupyter Notebooks, install with

```
pip install "black[jupyter]“
```

- Using

```
black {source_file_or_directory}
```

Code for testing `Black`

```
import math
def Some_Function(arg1, arg2 = 3):
    b=math.cos(math.pi)+10
    c=b*arg1
    return c/arg2
def Some_Function2(a1):
    return a1+a1*2

Some_Function(4,5)
Some_Function2(5)

>>> black black_test.py
```

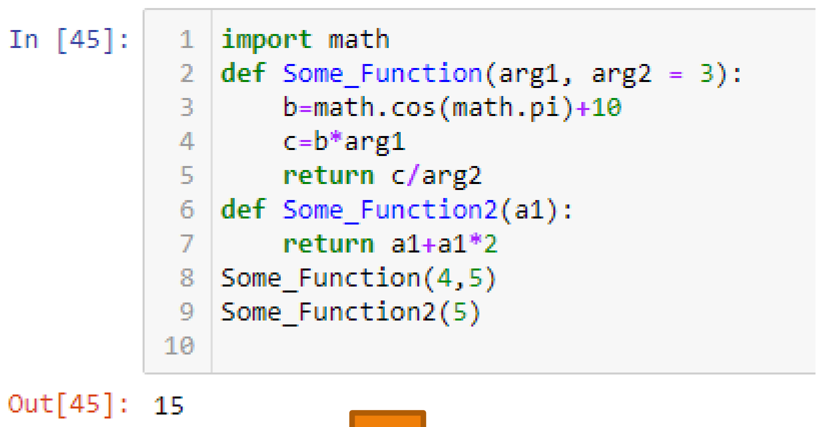
Assoc. Prof. Svitlana Kovalenko,
dept. of Software Engineering and Intelligent Management Technologies, NTU “KhPI”

Code for testing `Black`. Result

```
import math


def Some_Function(arg1, arg2=3):
    b = math.cos(math.pi) + 10
    c = b * arg1
    return c / arg2


def Some_Function2(a1):
    return a1 + a1 * 2


Some_Function(4, 5)
Some_Function2(5)
```

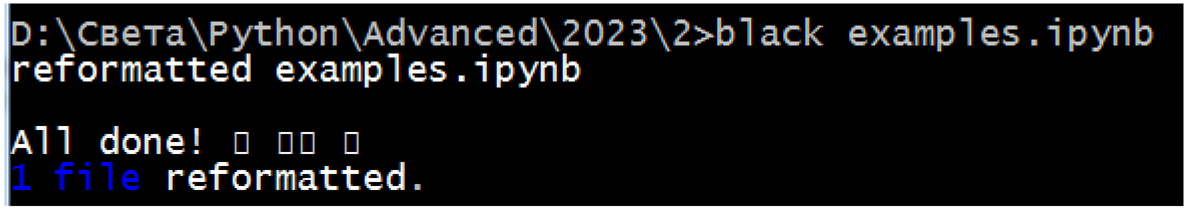
Assoc. Prof. Svitlana Kovalenko,
dept. of Software Engineering and Intelligent Management Technologies, NTU “KhPI”

`Black` test with Jupyter notebooks

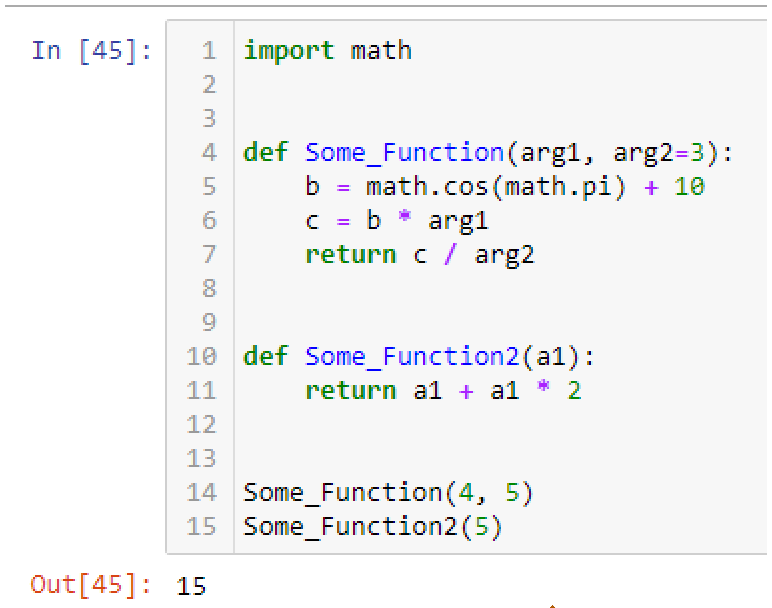


```
In [45]:
import math
def Some_Function(arg1, arg2 = 3):
    b=math.cos(math.pi)+10
    c=b*arg1
    return c/arg2
def Some_Function2(a1):
    return a1+a1*2
Some_Function(4,5)
Some_Function2(5)

Out[45]: 15
```

```
>>> black black_test.ipynb
```



```
D:\Света\Python\Advanced\2023\2>black examples.ipynb
reformatted examples.ipynb

All done! ✨ 🍰 ✨
1 file reformatted.
```



```
In [45]:
import math


def Some_Function(arg1, arg2=3):
    b = math.cos(math.pi) + 10
    c = b * arg1
    return c / arg2


def Some_Function2(a1):
    return a1 + a1 * 2


Some_Function(4, 5)
Some_Function2(5)

Out[45]: 15
```