Politechnika Świętokrzyska

Laboratorium

Programowanie w języku Python 2

Ćwiczenie 1

Programowanie funkcyjne
List comprehension

dr inż Robert Kazała

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie z funkcyjnym stylem programowania w języku Python. Poznanie dostępnych funkcji umozliwiających programowanie funkcyjne. Zapoznanie z mechanizmem list comprehension.

1. Programowanie funkcyjne

Przykład 1.1

```
square = lambda x: x * x
x = [1, 2, 3, 4, 5]
print(list(map(lambda num: num * num, x)))
```

Przykład 1.2

```
x = [1, 2, 3, 4, 5]

def square(num):
    return num*num

print(list(map(square, x)))
```

Przykład 1.3

```
x = [1, 2, 3, 4, 5]
print(list(map(lambda num: num * num, x)))
```

Przykład 1.4

```
product = 1
x = [1, 2, 3, 4]
for num in x:
    product = product * num
```

Przykład 1.5

```
from functools import reduce
  product = reduce((lambda x, y: x * y), [1, 2, 3, 4])
```

Przykład 1.6

```
from functools import reduce
lis = [ 1 , 3, 5, 6, 2 ]
output = reduce(lambda a,b : a if a > b else b, lis)
```

Przykład 1.7

```
x = range(-5, 5)
new_list = []
for num in x:
    if num < 0:
        new_list.append(num)
```

Przykład 1.8

```
x = range(-5, 5)
all_less_than_zero = list(filter(lambda num: num < 0, x))
```

Przykład 1.9

```
a = ("John", "Charles", "Mike")
b = ("Jenny", "Christy", "Monica", "Vicky")
x = zip(a, b)

print(*x)
```

Przykład 1.10

```
import functools, operator
functools.reduce(operator.add, [1, 2, 3, 4], 0)

sum([1, 2, 3, 4])
```

Przykład 1.11

```
import functools
product = 1
for i in [1, 2, 3]:
    product *= i
```

Przykład 1.12

```
product = functools.reduce(operator.mul, [1, 2, 3], 1)
```

2. Comprehension list

Przykład 2.1

```
print([x * x for x in [1, 2, 3, 4]])
```

Przykład 2.2

```
x = range(-5, 5)
all_less_than_zero = list(filter(lambda num: num < 0, x))
```

```
print(all_less_than_zero)

x = range(-5, 5)
all_less_than_zero = [num for num in x if num < 0]
```

Przykład 2.3

```
from pylab import *
a=zeros((10,8))
a[::2,:]=1
b=zeros(a.shape)
for (y,x) in [(y,x) for y in range(a.shape[0]) for x in
range(a.shape[1]) if a[y,x]>0.5]:
    print(y,x)
    b[y,x]=a[y,x]

imshow(b,interpolation='none',cmap=cm.gray)
```

Przykład 2.4

```
transposed = []
matrix = [[1, 2, 3, 4], [4, 5, 6, 8]]

for i in range(len(matrix[0])):
    transposed_row = []

    for row in matrix:
        transposed_row.append(row[i])
    transposed.append(transposed_row)

print(transposed)
```

Przykład 2.5

```
matrix = [[1, 2], [3,4], [5,6], [7,8]]
transpose = [[row[i] for row in matrix] for i in range(2)]
print (transpose)
```

Przykład 2.6

```
x = range(-5, 5)
all_less_than_zero = list(map(lambda num: num * num,
list(filter(lambda num: num < 0, x))))
```

Przykład 2.7

```
x = range(-5, 5)
all_less_than_zero = [num * num for num in x if num < 0]
```

Przykład 2.8

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
[(x if x != "apple" else "orange") for x in fruits if 'a' in x and
'e' in x or 'k' in x]
```

Przykład 2.9

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
[f[0].upper()+f[1:] for f in fruits]

[f.capitalize() for f in fruits]
```

Przykład 2.10

```
DIAL_CODES = [     (86, 'China'),
    (91, 'India'),
    (1, 'United States'),
    (62, 'Indonesia'),
    (55, 'Brazil'),
    (92, 'Pakistan'),
    (880, 'Bangladesh'),
    (234, 'Nigeria'),
    (7, 'Russia'),
    (81, 'Japan'),
    ]
country_code = {country: code for code, country in DIAL_CODES}
country_code


{code: country.upper() for country, code in country_code.items()
if code < 66}
```

Przykład 2.11

```
square_dict = dict()
for num in range(1, 11):
    square_dict[num] = num*num
print(square_dict)

square_dict = {num: num*num for num in range(1, 11)}
print(square_dict)
```

Literatura

https://docs.python.org/3/howto/functional.html

https://kite.com/blog/python/functional-programming/

https://stackabuse.com/functional-programming-in-python/

https://julien.danjou.info/python-and-functional-programming/

https://en.wikipedia.org/wiki/List_comprehension

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

https://realpython.com/list-comprehension-python/

https://www.geeksforgeeks.org/comprehensions-in-python/

https://www.python-course.eu/lambda.php

https://www.python-course.eu/list_comprehension.php

Zadania

1. Uruchomić, przeanalizować, omówić efekt działania wszystkich przykładów z instrukcji.
2. Do każdego przykładu z instrukcji wykorzystującego programowanie funkcyjne utworzyć własny przykład wykorzystujący zaprezentowany w nim mechanizm.
3. Do każdego przykładu z instrukcji wykorzystującego list comprehension utworzyć własny przykład wykorzystujący zaprezentowany w nim mechanizm.
4. Napisać w sposób funkcyjny i korzystając z list comprehension następujące wyrażenia:
    - obliczające sumę elementów nieparzystych z wybranego zakresu,
    - wyliczające sumę elementów, których wartości spełniają zadany warunek,
    - zliczające wystąpienia zadanych wyrazów w tekście,
    - obliczające wartości poszczególnych pozycji faktury i sumę faktury, na której podana jest dowolna ilość pozycji obejmujących cenę, ilość, rabat. Pozycje faktury zapisane są na liście [[10, 3, 0.02],[22, 10, 0.05],[7, 8, 0.03]].