Функциональное программирование

In [1]:
```python
def caller(func, params):
    return func(*params)


def printer(name, origin):
    print('I\'m {} of {}!'.format(name, origin))


caller(printer, ['Moana', 'Motunui'])
```

```
I'm Moana of Motunui!
```

In [2]:
```python
def get_multiplier():
    def inner(a, b):
        return a * b
    return inner


multiplier = get_multiplier()
multiplier(10, 11)
```

Out[2]: 110

In [3]:
```python
print(multiplier.__name__)
```

```
inner
```

In [4]:
```python
def get_multiplier(number):
    def inner(a):
        return a * number
    return inner


multiplier_by_2 = get_multiplier(2)
multiplier_by_2(10)
```

Out[4]: 20

map, filter, lambda

In [5]:
```python
def squarify(a):
    return a ** 2


list(map(squarify, range(5)))
```

Out[5]: [0, 1, 4, 9, 16]

```
In [6]:  squared_list = []

         for number in range(5):
             squared_list.append(squarify(number))

         print(squared_list)
```

```
[0, 1, 4, 9, 16]
```

```
In [7]:  def is_positive(a):
             return a > 0


         list(filter(is_positive, range(-2, 3)))
```

```
Out[7]:  [1, 2]
```

```
In [8]:  positive_list = []

         for number in range(-2, 3):
             if is_positive(number):
                 positive_list.append(number)

         print(positive_list)
```

```
[1, 2]
```

```
In [9]:  list(map(lambda x: x ** 2, range(5)))
```

```
Out[9]:  [0, 1, 4, 9, 16]
```

```
In [10]:  type(lambda x: x ** 2)
```

```
Out[10]:  function
```

```
In [11]:  list(filter(lambda x: x > 0, range(-2, 3)))
```

```
Out[11]:  [1, 2]
```

Написать функцию, которая превращает список чисел в список строк

```
In [12]:  def stringify_list(num_list):
              return list(map(str, num_list))


          stringify_list(range(10))
```

```
Out[12]:  ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

functools

```
In [13]:  from functools import reduce


          def multiply(a, b):
              return a * b


          reduce(multiply, [1, 2, 3, 4, 5])
```

Out[13]: 120

```
In [14]:  reduce(lambda x, y: x * y, range(1, 5))
```

Out[14]: 24

```
In [15]:  from functools import partial


          def greeter(person, greeting):
              return '{}, {}!'.format(greeting, person)


          hier = partial(greeter, greeting='Hi')
          helloer = partial(greeter, greeting='Hello')


          print(hier('brother'))
          print(helloer('sir'))
```

```
Hi, brother!
Hello, sir!
```

Списочные выражения

До этого момента мы с вами определяли списки стандартным способом, однако в питоне существует более красивая и лаконичная конструкция для создания списков и других коллекций.

```
In [16]:  square_list = []
          for number in range(10):
              square_list.append(number ** 2)

          print(square_list)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [17]:  square_list = [number ** 2 for number in range(10)]
          print(square_list)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [18]:  even_list = []
          for number in range(10):
              if number % 2 == 0:
                  even_list.append(number)

          print(even_list)
```

[0, 2, 4, 6, 8]

```
In [19]:  even_list = [num for num in range(10) if num % 2 == 0]

          print(even_list)
```

[0, 2, 4, 6, 8]

```
In [20]:  square_map = {number: number ** 2 for number in range(5)}

          print(square_map)
```

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

```
In [21]:  reminders_set = {num % 10 for num in range(100)}

          print(reminders_set)
```

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

```
In [22]:  print(type(number ** 2 for number in range(5)))
```

<class 'generator'>

```
In [23]:  num_list = range(7)
          squared_list = [x ** 2 for x in num_list]

          list(zip(num_list, squared_list))
```

Out[23]: [(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36)]

Функциональное программирование

- Функции — объекты первого класса
- map, filter, reduce, partial
- lambda — анонимные функции
- Списочные выражения