Московский государственный технический университет им. Н. Э. Баумана
(МГТУ им. Н. Э. Баумана)

Кафедра «Системы обработки информации и управления» (ИУ5)

Лабораторная работа №2–3

По дисциплине: «Парадигмы и конструкции языков программирования»

«Функциональные возможности языка Python»

Выполнила: Холодова К. А.,
студентка группы ИУ5-32Б

Проверил: Гапанюк Ю. Е.

г. Москва 2023 г.

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python

Задание:

1) Файл field.py
2) Файл gen_random.py
3) Файл unique.py
4) Файл sotrt.py
5) Файл print_result.py
6) Файл cm_timer.py
7) Файл process_data.py

Код программы

файл field.py

```
# Необходимо реализовать генератор field. Генератор field последовательно
выдает значения ключей словаря

def field(items, *args):
    assert len(args) > 0

    for item in items:
        if len(args) == 1:
            key = args[0]
            if key in item and item[key] is not None:
                yield item[key]
        else:
            filtered_item = {key: item[key] for key in args if key in item and
item[key] is not None}
            if filtered_item:
                yield filtered_item


# пример использования
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]

# проверка для одного аргумента
for title in field(goods, 'title'):
    print(title)

# проверка для двух аргументов
for item in field(goods, 'title', 'price'):
    print(item)
```

файл gen_random.py

```
# Необходимо реализовать генератор gen_random(количество, минимум, максимум),
# который последовательно выдает заданное количество случайных чисел в заданном
диапазоне от минимума до максимума, включая границы диапазона

import random

# Пример:
# gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел в диапазоне от 1
до 3, например 2, 2, 3, 2, 1

def gen_random(count, minimum, maximum):
    for _ in range(count):
        yield random.randint(minimum, maximum)

for num in gen_random(5, 1, 3):
    print(num)
```

файл unique.py

```
# Пример:

# data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
# Unique(data) будет последовательно возвращать только 1 и 2

# data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
# Unique(data) будет последовательно возвращать только a, A, b, B

class Unique(object):
    def __init__(self, items, ignore_case=False, **kwargs):
        self.seen = set()
        self.ignore_case = ignore_case
        self.items = items
        self.index = 0

    def __next__(self):
        while self.index < len(self.items):
            item = self.items[self.index]
            self.index += 1
            key = item.lower() if self.ignore_case and isinstance(item, str)
else item
            if key not in self.seen:
                self.seen.add(key)
                return item
        raise StopIteration()

    def __iter__(self):
        return self

# пример использования
```

```
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_iterator1 = Unique(data1)
for item in unique_iterator1:
    print(item)

data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_iterator2 = Unique(data2, ignore_case=False)
for item in unique_iterator2:
    print(item)
```

файл sort.py

```
# Дан массив 1, содержащий положительные и отрицательные числа
# Необходимо одной строкой кода вывести на экран массив 2, которые содержит
значения массива 1, отсортированные по модулю в порядке убывания
# Сортировку необходимо осуществлять с помощью функции sorted. Пример:

# data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
# Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

# Необходимо решить задачу двумя способами:

# С использованием lambda-функции
# Без использования lambda-функции

# with L
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
print(data)
sorted_data = sorted(data, key=lambda x: abs(x), reverse=True)
print(sorted_data)


# without L
data = [2, -10, 20, 160, -120, 123, 1, 0, -1, -4]
print(data)
sorted_data = sorted(data, key=abs, reverse=True)
print(sorted_data)
```

файл print_result.py

```
# Необходимо реализовать декоратор print_result, который выводит на экран
результат выполнения функции

def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(func.__name__)
        if isinstance(result, list):
            for item in result:
                print(item)
        elif isinstance(result, dict):
```

```
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)
        return result
    return wrapper

# примеры использования декоратора

@print_result
def test_1():
    return 1

@print_result
def test_2():
    return 'iu5'

@print_result
def test_3():
    return {'a': 1, 'b': 2}

@print_result
def test_4():
    return [1, 2]

if __name__ == '__main__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

файл cm_timer.py

```
# Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран. Пример:

# with cm_timer_1():
#     sleep(5.5)

# После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться)
# cm_timer_1 и cm_timer_2 реализуют одинаковую функциональность,
# но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib)

import time

# with Context Manager 1
```

```
class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"time: {elapsed_time}")

# with Context Manager 2

from contextlib import contextmanager

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"time: {elapsed_time}")

with cm_timer_1():
    time.sleep(5.5)

with cm_timer_2():
    time.sleep(5.5)
```

файл process_data.py

```
import json
import sys
import time
from contextlib import contextmanager

path = "C:/Users/holod/Desktop/bmstu/учеба/5 семестр/пикяп/лабы/лаба 2-
3/lab_python_fp/data_light.json"

with open(path, encoding="utf8") as f:
    data = json.load(f)

@contextmanager
def cm_timer_1():
    start_time = time.time()
    yield
    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"time: {elapsed_time}")

def print_result(func):
```

```
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(result)
        return result
    return wrapper

@print_result
def f1(arg):
    return sorted(list(set([job['job-name'].lower() for job in arg])))

@print_result
def f2(arg):
    return list(filter(lambda job: job.startswith('программист'), arg))

@print_result
def f3(arg):
    return list(map(lambda job: job + ', с опытом Python', arg))

@print_result
def f4(arg):
    salary = [str(i) for i in range(100000, 200001)]
    return list(zip(arg, salary))

if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Результат выполнения:

1) field.py

```
Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
```

2) gen_random.py

```
1
1
3
2
3
```

3) unique.py

```
1
2
a
A
b
B
```

4) sort.py

```
[4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[2, -10, 20, 160, -120, 123, 1, 0, -1, -4]
[160, 123, -120, 20, -10, -4, 2, 1, -1, 0]
```

5) print_result.py

```
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

6) cm_timer.py

```
time: 5.50092339515686
time: 5.515078067779541
```

7) process_data.py

```
['программист', 'программист / senior developer', 'программист 1с', 'программист с#', 'программист с++', 'программист с++/с#/java', 'программист/ junior developer', 'программист
/ технический специалист', 'программистр-разработчик  информационных систем']
['программист, с опытом Python', 'программист / senior developer, с опытом Python', 'программист 1с, с опытом Python', 'программист с#, с опытом Python', 'программист с++, с опы
том Python', 'программист с++/с#/java, с опытом Python', 'программист/ junior developer, с опытом Python', 'программист/ технический специалист, с опытом Python', 'программистр-
разработчик  информационных систем, с опытом Python']
[('программист, с опытом Python', '100000'), ('программист / senior developer, с опытом Python', '100001'), ('программист 1с, с опытом Python', '100002'), ('программист с#, с оп
ытом Python', '100003'), ('программист с++, с опытом Python', '100004'), ('программист с++/с#/java, с опытом Python', '100005'), ('программист/ junior developer, с опытом Python
', '100006'), ('программист/ технический специалист, с опытом Python', '100007'), ('программистр-разработчик  информационных систем, с опытом Python', '100008')]
time: 0.1258530616760254
```