Московский Государственный Технический Университет им. Н.Э. Баумана

Отчет по лабораторной работе № 3-4 по курсу Базовые компоненты интернет-технологий

" Функциональные возможности языка Python."

Исполнитель:

Фруктин А.Е. РТ5-31Б

Проверил:

Гапанюк Юрий Евгеньевич

Москва, 2022

Задание

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.
Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Код

cm_timer.py:

```
from contextlib import contextmanager
from time import time, sleep

class cm_timer_1:
    def __init__(self):
        self.start_time = time()
    def __enter__(self):
        return self
    def __exit__(self, exc_type, exc_val, exc_tb):
        if exc_type is not None:
            print(exc_type, exc_val, exc_tb)
        self.end_time = time()
        print(f'Время работы блока (class) {self.end_time - self.start_time:.5f} сек.')

@contextmanager
def cm_timer():
    start_time = time()
    yield 1
    print(f'Время работы блока (function) {time() - start_time:.5f} сек.')



# with cm_timer():
#     sleep(1.5)

# with cm_timer_1():
#     sleep(1.5)
```

field.py:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
```

```
    {'title': 'Диван для отдыха', 'color': 'black'}
]

def field(items, *args):
    assert len(args) > 0
    result = []
    for item in items:
        dict_result = {}
        if len(args) == 1 and item[args[0]] is not None:
            result.append(item[args[0]])
        else:
            for key in args:
                if key in item and item[key] is not None:
                    dict_result[key] = item[key]
        if len(dict_result) > 0:
            result.append(dict_result)
    return result


# print(field(goods, 'title'))
# print(field(goods, 'title', 'price'))
# print(field(goods, 'title', 'price', 'color'))
# print(field(goods, 'title', 'color'))
```

gen_random.py:

```
import random

def gen_random(num_count, begin, end):
    result = [random.randint(begin, end) for i in range(num_count)]
    return result

# print(gen_random(5, 1, 3))
# print(gen_random(5, 5, 10))
```

print_result.py:

```

def print_result(func):
    def wrapper(*args, **kwargs):
        print(func.__name__)
        result = func(*args, **kwargs)
        if (isinstance(result, list)):
            print(*result, sep = '\n')
        elif (isinstance(result, dict)):
            for key, value in result.items():
                print(key, value, sep = ' = ')
        else:
            print(result)
        return result
    return wrapper

@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()
    print('!!!!!!!!')
```

sort.py:

```
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key = lambda x: -abs(x))
    print(result_with_lambda)
```

unique.py:

```
#from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = list(items)
        self.ignore_case = kwargs.get('ignore_case', False)
        self.unique_items = set()

    def __iter__(self):
        return self

    def __next__(self):
        if len(self.items) == 0:
```

```
            raise StopIteration
        item = self.items.pop(0)
        if self.ignore_case:
            item = item.lower()
        if item not in self.unique_items:
            self.unique_items.add(item)
            return item
        else:
            return self.__next__()


# data = [1,1,1,2,2,3,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
# print(*Unique(data))
# data = ['a', 'A', 'B', 'a', 'b']
# print(*Unique(data, ignore_case=True))
# print(*Unique(data))
```

process_data.py:

```
import json
import sys
from cm_timer import cm_timer_1
from gen_random import gen_random
from print_result import print_result
from field import field
from unique import Unique

path = 'lab3_4/lab_python_fp/data_light.json'
with open(path, encoding='UTF8') as f:
    data = json.load(f)

@print_result
def f1(arg):
    return sorted(Unique(field(arg, 'job-name'), ignore_case=True))
    # return (Unique(sorted(field(arg, 'job-name'), key=len), ignore_case=True))


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


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


@print_result
def f4(arg):
    return list(map(lambda x: x + ', зарплата ' + str(gen_random(1, 100000,
200000)) + ' руб.', arg))
```

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

Результаты выполнения программы

Основной результат:

```
f4
программист с опытом Python, зарплата [173711] руб.
программист / senior developer с опытом Python, зарплата [169781] руб.
программист 1с с опытом Python, зарплата [170528] руб.
программист c# с опытом Python, зарплата [102465] руб.
программист c++ с опытом Python, зарплата [173728] руб.
программист c++/c#/java с опытом Python, зарплата [126172] руб.
программист/ junior developer с опытом Python, зарплата [172091] руб.
программист/ технический специалист с опытом Python, зарплата [129074] руб.
программистр-разработчик  информационных систем с опытом Python, зарплата [137212] руб.
Время работы блока (class) 0.25429 сек.
```

Промежуточные функции:

```
f2
программист
программист / senior developer
программист 1с
программист c#
программист c++
программист c++/c#/java
программист/ junior developer
программист/ технический специалист
программистр-разработчик  информационных систем
f3
программист с опытом Python
программист / senior developer с опытом Python
программист 1с с опытом Python
программист c# с опытом Python
программист c++ с опытом Python
```

```
монтажник-сантехник (ов, вк)
монтажник-сборщик рекламных конструкций
монтажники жбк (керченский мост)
монтажники металлоконструкций
монтажники технологического оборудования
монтажники труб пнд
монтер пути
монтеры пути и бригадиры пути
монтёр пути
мотальщица
моторист
моторист (машинист)
моторист, дизелист
музыкальный руководитель
наборщик
```