

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python»

Выполнил:
Плотников Ф.С. ИУ5-51

Проверил:
Гапанюк Ю. Е.
Балашов А. М.

Москва, 2021 г.

Цель работы:

Изучение возможностей функционального программирования в языке Python.

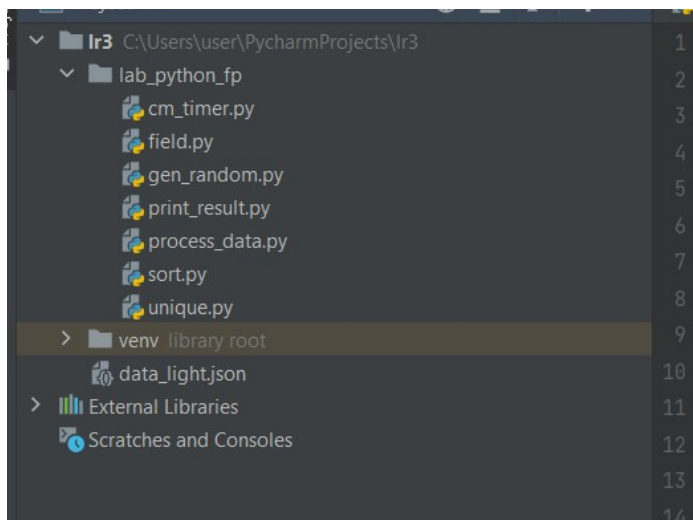
Задание:

Задание лабораторной работы состоит из решения нескольких задач. Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fr. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Выполнение:

Был создан пакет lab_python_fr, который содержит в себе модули задач. Некоторые задачи используют другие задачи как внешние модули через import.



Задача 1:

Необходимо реализовать генератор field. Генератор field последовательно выдаст значения ключей словаря.

Реализация:

```
def zero_args_test(func):
    def testing(_, *args):
        if len(args) > 0:
            return func(_, *args)
        else:
            return ('Нет аргументов, вызов { } невозможен'.format(func.__name__))
    return testing

@zero_args_test
def field(items, *args):
    request = list()
    if len(args) == 1:
        for each_dict in items:
            if args[0] in each_dict:
                request.append(each_dict[args[0]])
    else:
        for each_dict in items:
```

```

temp_dict = {}
for key in each_dict:
    if (key in args):
        temp_dict[key] = each_dict[key]
request.append(temp_dict)
return request

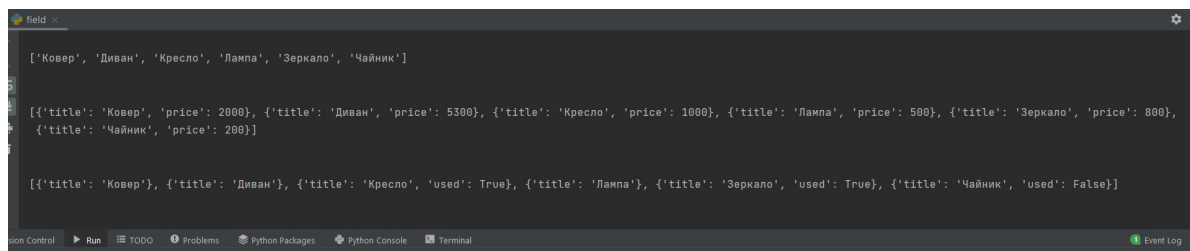
def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'Зеленый'},
        {'title': 'Диван', 'price': 5300, 'color': 'Желтый'},
        {'title': 'Кресло', 'price': 1000, 'color': 'Красное', 'used': True},
        {'title': 'Лампа', 'price': 500, 'power': '50w'},
        {'title': 'Зеркало', 'price': 800, 'used': True},
        {'title': 'Чайник', 'price': 200, 'power': '50w', 'used': False},
    ]

    print(field(goods, '\n\n'))
    print(field(goods, 'title'), '\n\n')
    print(field(goods, 'title', 'price'), '\n\n')
    print(field(goods, 'title', 'used'), '\n\n')

if __name__ == "__main__":
    main()

```

Запуск:



```

field
[ 'Ковер', 'Диван', 'Кресло', 'Лампа', 'Зеркало', 'Чайник' ]

[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван', 'price': 5300}, {'title': 'Кресло', 'price': 1000}, {'title': 'Лампа', 'price': 500}, {'title': 'Зеркало', 'price': 800}, {'title': 'Чайник', 'price': 200}]

[{'title': 'Ковер'}, {'title': 'Диван'}, {'title': 'Кресло', 'used': True}, {'title': 'Лампа'}, {'title': 'Зеркало', 'used': True}, {'title': 'Чайник', 'used': False}]

```

Задача 2:

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.

Реализация:

```

from random import randrange

def get_random(number, min, max):
    return [randrange(min, max+1) for i in range(number)]

def main():
    print(get_random(10, 1, 5))

if __name__ == "__main__":
    main()

```

Запуск:

```
gen_random x
C:\Users\user\PycharmProjects\lr3\venv\Scripts\python.exe C:/Users/user/PycharmProjects/lr3/lab_python_fp/gen_random.py
[4, 5, 5, 1, 1, 4, 5, 1, 4, 5]

Process finished with exit code 0
```

Задача 3:

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

Реализация:

```
from gen_random import get_random

class Unique(object):

    def __init__(self, items, **kwargs):
        self.unicList = list()
        self.data = items
        self.index = 0
        if 'ignore_case' in kwargs.keys() and kwargs['ignore_case'] == True:
            self.ignore_case = True
        else:
            self.ignore_case = False

    def __next__(self):
        while True:
            if self.index < len(self.data):
                if self.ignore_case == False:
                    current = self.data[self.index]
                    self.index += 1
                    if not current in self.unicList:
                        self.unicList.append(current)
                        return current
                else:
                    current = self.data[self.index]
                    self.index += 1
                    if isinstance(current, str):
                        current = current.lower()
                    if not current in self.unicList:
                        self.unicList.append(current)
                        return current
            else:
                raise StopIteration

    def __iter__(self):
        return self

def main():
    data = ['a', 'A', 'b', 'B', 'c', 'C', 'C', 'A', 'D']
    random_nums = get_random(10, 1, 7)
    print(data)
    for i in Unique(data, ignore_case=True):
        print(i, end='\t')
    print("\n")

    print(random_nums)
    for i in Unique(random_nums):
        print(i, end='\t')
```

```
if __name__ == "__main__":  
    main()
```

Запуск:

```
unique x  
C:\Users\user\PycharmProjects\lr3\venv\Scripts\python.exe C:/Users/user/PycharmProjects/lr3/lab_python_fp/unique.py  
['a', 'A', 'b', 'B', 'c', 'C', 'C', 'A', 'D']  
a b c d  
  
[3, 6, 4, 6, 1, 3, 5, 6, 3, 7]  
3 6 4 1 5 7  
Process finished with exit code 0
```

Задача 4:

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.

Реализация:

```
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
```

```
def main():  
    result = sorted(data, key=abs, reverse=True)  
    print(result)  
  
    result_with_lambda = (lambda mass: sorted(mass, key=abs, reverse=True))(data)  
    print(result_with_lambda)  
  
if __name__ == "__main__":  
    main()
```

Запуск:

```
C:\Users\user\PycharmProjects\lr3\venv\Scripts\python.exe  
C:/Users/user/PycharmProjects/lr3/lab_python_fp/sort.py  
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]  
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
```

Process finished with exit code 0

Задача 5:

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

Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.

Если функция вернула список (`list`), то значения элементов списка должны выводиться в столбик.

Если функция вернула словарь (`dict`), то ключи и значения должны выводиться в столбик через знак равенства.

Реализация:

```
def print_result(func):
    def decorated_func(*args, **kwargs):
        print(func.__name__)
        if isinstance(func(*args, **kwargs), dict):
            for key in func(*args, **kwargs):
                print(key, '=', func(*args, **kwargs)[key])
        elif isinstance(func(*args, **kwargs), list):
            for item in func(*args, **kwargs):
                print(item)
        else:
            print(func(*args, **kwargs))
        return func(*args, **kwargs)

    return decorated_func

@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]

def main():
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

```
if __name__ == '__main__':  
    main()
```

Запуск:

C:\Users\user\PycharmProjects\lr3\venv\Scripts\python.exe

C:/Users/user/PycharmProjects/lr3/lab_python_fp/print_result.py

!!!!!!!

test_1

1

test_2

iu5

test_3

a = 1

b = 2

test_4

1

2

Process finished with exit code 0

Задача 6:

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

Реализация:

```
from time import sleep, time  
from contextlib import contextmanager  
  
class cm_timer_1():  
    def __init__(self):  
        self.timer = time()  
  
    def __enter__(self):  
        pass  
  
    def __exit__(self, exp_type, exp_value, traceback):  
        self.timer = time() - self.timer  
        print("time: {0:0.1f}".format(self.timer))  
  
@contextmanager  
def cm_timer_2():  
    t = time()  
    yield  
    t = time() - t  
    print("time: {0:0.1f}".format(t))  
  
def main():  
    with cm_timer_1():  
        sleep(3.5)  
  
    with cm_timer_2():  
        sleep(3.3)
```

```
if __name__ == "__main__":  
    main()
```

Запуск:

C:\Users\user\PycharmProjects\lr3\venv\Scripts\python.exe

C:/Users/user/PycharmProjects/lr3/lab_python_fp/cm_timer.py

time: 3.5

time: 3.3

Process finished with exit code 0

Задача 7:

- В файле data_light.json содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер cm_timer_1 выводит время работы цепочки функций.
- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
- Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: Программист C# с опытом Python. Для модификации используйте функцию map.
- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист C# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Реализация:

```
import json  
from cm_timer import cm_timer_1, cm_timer_2  
from field import field  
from unique import Unique  
from print_result import print_result  
from gen_random import get_random
```

```
@print_result  
def f1(data):
```



```

return sorted(Unique(field(data,'job-name'), ignore_case=True))

@print_result
def f2(data):
    return list(filter(lambda d: 'программист' in d[:11], data))

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

@print_result
def f4(data):
    return list(zip(data, list(map(lambda x: "Зарплата " + x + " руб",map(str,(get_random(len(data), 100000, 200000)))))))

def main():
    with open('C:\\Users\\User\\PycharmProjects\\lr3\\data_light.json', 'r', encoding='utf8') as jft:
        data = json.load(jft)
    with cm_timer_1():
        f4(f3(f2(f1(data))))

if __name__ == "__main__":
    main()

```

Запуск:

The screenshot shows the PyCharm IDE interface. The top toolbar includes buttons for Run, Debug, and other development actions. The Run console at the bottom displays the output of the script, showing a list of 20 job titles. The status bar at the very bottom indicates the file encoding is UTF-8 and the Python version is 3.9.

```

C:\Users\User\PycharmProjects\lr3\venv\Scripts\python.exe C:/Users/user/PycharmProjects/lr3/lab_python_fp/process_data.py
f1
1с программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
asic специалист
javascript разработчик
rtl специалист
web-программист
web-разработчик
автожестящик
автоинструктор
автомаляр
автомоищик
автор студенческих работ по различным дисциплинам
автослесарь
автослесарь - моторист
автоэлектрик
агент
агент банка
агент нпф
агент по гос. закупкам недвижимости
агент по недвижимости
агент по недвижимости (стажер)
агент по недвижимости / риэлтор
агент по привлечению юридических лиц
агент по продажам (интернет, тв, телефония) в пао ростелеком в населенных пунктах амурской области: г. благовещенск, г. белогорск, г. свободный, г. шимановск, г. зейя, г. тында

```

f2

программист

программист / senior developer

программист 1с

программист с#

программист c++

программист с++/c#/java
программист/ junior developer
программист/ технический специалист
программист-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 с++ с опытом Python
программист с++/c#/java с опытом Python
программист/ junior developer с опытом Python
программист/ технический специалист с опытом Python
программист-разработчик информационных систем с опытом Python
f4

('программист с опытом Python', 'Зарплата 177749 руб')
('программист / senior developer с опытом Python', 'Зарплата 166100 руб')
('программист 1с с опытом Python', 'Зарплата 121933 руб')
('программист c# с опытом Python', 'Зарплата 105845 руб')
('программист с++ с опытом Python', 'Зарплата 163178 руб')
('программист с++/c#/java с опытом Python', 'Зарплата 177672 руб')
('программист/ junior developer с опытом Python', 'Зарплата 153767 руб')
('программист/ технический специалист с опытом Python', 'Зарплата 117684 руб')
('программист-разработчик информационных систем с опытом Python', 'Зарплата 176596 руб')
time: 0.2

Process finished with exit code 0

