Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ	**«Информатики и систем управления»**
КАФЕДРА	ИУ5

Дисциплина «Разработка интернет-приложений»

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

Студент	группы ИУ5-52Б	Гришин Илья
Преподаватель		Гапанюк Ю.Е.

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.

1. Задание

Задание лабораторной работы состоит из решения нескольких задач.

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

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

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

Задача 2 (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.

Задача 3 (файл unique.py)

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

Задача 4 (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.

Задача 5 (файл print_result.py)

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 выводить в столбик через знак равенства.

Задача 6 (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.

Задача 7 (файл process_data.py)

В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.

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.

Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр.

Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые содержат слово “программист”.

Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python”.

Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

2. Текст программ

field.py

```
goods = [
    {'title': 'Ковер', 'price': 3000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стул', 'price': None, 'color': 'green'},
    {'title': None, 'price': None, 'color': 'green'},
]


def field(items,*args):
    assert len(args) > 0, 'Не переданы аргументы полей словаря'
    if len(args) == 1:
        for i in range(len(items)):
            if (items[i].get(args[0])!=None):
                yield items[i].get(args[0])
    else:
        for i in range(len(items)):
            d={}
            for j in range(len(args)):
                if (items[i].get(args[j])!=None):
                    d.update({args[j]: items[i].get(args[j])})
            if (d!={}):
                yield d

def main():
    f=field(goods, 'title')
    for i in f:
        print(i, end='; ')
    print('\n',end='')
    f=field(goods, 'title', 'price')
    for i in f:
        print(i, end='; ')

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

gen_random.py

```
import random

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randint(begin, end)

def main():
    gen = gen_random(5, 1, 3)
    for i in gen:
```

```
        print(i, end=' ')
    print('\n',end='')
    gen = gen_random(10, 0, 10)
    for i in gen:
        print(i, end=' ')

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

unique.py

```
from gen_random import gen_random


class Unique(object):
    def __init__(self, items, **kwargs):
        self.set_unique = set()
        self.items = items
        if len(kwargs) != 0:
            self.ignore_case = kwargs
        else:
            self.ignore_case = False
    def __iter__(self):
            return self
    def __next__(self):
        while True:
            for item in self.items:
                temp_item = item
                if (temp_item not in self.set_unique) and not(self.ignore_case an
d temp_item.swapcase() in self.set_unique):
                    self.set_unique.add(temp_item)
                    return temp_item
            else:
                raise StopIteration



def main():
    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(data1)
    itr1 = Unique(data1)
    for i1 in itr1:
        print(i1, end=' ')
    print('\n', end='')

    data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print(data2)
    itr2 = Unique(data2)
    for i2 in itr2:
        print(i2, end=' ')
```

```
    print('\n', end='')

    print(data2)
    itr3 = Unique(data2, ignor_case=True)
    for i3 in itr3:
        print(i3, end=' ')
    print('\n', end='')

    data3 = gen_random(5, 1, 3)
    for i4 in data3:
        print(i4, end=' ')
    print('\n', end='')
    itr4 = Unique(data3)
    for i5 in itr4:
        print(i5, end=' ')


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

sort.py

```
def main():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result_with_lambda)


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

print_result.py

```
def print_result(func):
    def decorated_func(*args):
        print(func.__name__)
        result = func(*args)
        if (type(result)==list):
            for i in result:
                print(i)
        else:
            if (type(result)==dict):
                for i in result:
                    print(i, '-->', result.get(i))
            else:
                print(result)
```

```
        return result
    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()
```

cm_timer.py

```
import time
from contextlib import contextmanager

class cm_timer_1:
    def __init__(self):
        self.begin_time = time.time()
    def __enter__(self):
        pass
    def __exit__(self, exc_type, exc_val, exc_tb):
        print('time: ', time.time() - self.begin_time)


@contextmanager
def cm_timer_2():
    begin_time = time.time()
```

```
    yield 1
    print('time: ', time.time() - begin_time)


def main():
    with cm_timer_1():
        time.sleep(5.5)

    with cm_timer_2():
        time.sleep(2)


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

process_data.py

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


path = 'data_light.json'


with open(path, encoding='utf-8') as f:
    data = json.load(f)


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


@print_result
def f2(arg):
    return filter(lambda x: re.search('Программист', x) or re.search('программист
', x), arg)


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


@print_result
```

```python
def f4(arg):
    return list(map(lambda x: x + ", зарплата " + str(*gen_random(1, 100000, 200000)) + " руб", arg))


def main():
    with cm_timer_1():
        f4(f3(f2(f1(data))))


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

3. Результаты

field.py

```
c:\2\rip\lab3\lab_python_fp>field.py
Ковер;  Диван для отдыха;  Стул;
{'title': 'Ковер', 'price': 3000};  {'title': 'Диван для отдыха', 'price': 5300}
;  {'title': 'Стул'};
```

gen_random.py

```
c:\2\rip\lab3\lab_python_fp>gen_random.py
3 1 1 1 2
10 1 6 4 8 0 2 2 4 9
```

unique.py

```
c:\2\rip\lab3\lab_python_fp>unique.py
[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
1 2
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
a A b B
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
a b
1 1 2 1 3
```

sort.py

```
c:\2\rip\lab3\lab_python_fp>sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

print_result.py

```
c:\2\rip\lab3\lab_python_fp>print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a --> 1
b --> 2
test_4
1
2
```

cm_timer.py

```
c:\2\rip\lab3\lab_python_fp>cm_timer.py
time:  5.506808280944824
time:  2.0001022815704346
```

process_data.py

```
c:\2\rip\lab3>process_data.py
f1
<lab_python_fp.unique.Unique object at 0x01CB38B0>
f2
<filter object at 0x01CB38F8>
f3
Системный программист (C, Linux) с опытом Python
Веб-программист с опытом Python
Программист с опытом Python
Программист C++/C#/Java с опытом Python
1С программист с опытом Python
программист с опытом Python
Инженер-программист ККТ с опытом Python
инженер - программист с опытом Python
Инженер-программист (Клинский филиал) с опытом Python
Инженер-программист (Орехово-Зуевский филиал) с опытом Python
Ведущий программист с опытом Python
Программист 1С с опытом Python
Программистр-разработчик  информационных систем с опытом Python
Инженер - программист АСУ ТП с опытом Python
инженер-программист с опытом Python
Программист C++ с опытом Python
Программист/ Junior Developer с опытом Python
Программист / Senior Developer с опытом Python
Инженер-электронщик (программист АСУ ТП) с опытом Python
Старший программист с опытом Python
Web-программист с опытом Python
Веб - программист (PHP, JS) / Web разработчик с опытом Python
Программист/ технический специалист с опытом Python
программист 1С с опытом Python
Программист C# с опытом Python
Инженер-программист 1 категории с опытом Python
Ведущий инженер-программист с опытом Python
Инженер-программист САПОУ (java) с опытом Python
Помощник веб-программиста с опытом Python
веб-программист с опытом Python
педагог программист с опытом Python
Инженер-программист ПЛИС с опытом Python
Инженер-программист с опытом Python
f4
Системный программист (C, Linux) с опытом Python, зарплата 150759 руб
Веб-программист с опытом Python, зарплата 159716 руб
Программист с опытом Python, зарплата 196719 руб
Программист C++/C#/Java с опытом Python, зарплата 180934 руб
1С программист с опытом Python, зарплата 180532 руб
программист с опытом Python, зарплата 160764 руб
Инженер-программист ККТ с опытом Python, зарплата 146681 руб
инженер - программист с опытом Python, зарплата 182435 руб
Инженер-программист (Клинский филиал) с опытом Python, зарплата 173614 руб
Инженер-программист (Орехово-Зуевский филиал) с опытом Python, зарплата 138234 р
уб
Ведущий программист с опытом Python, зарплата 136227 руб
Программист 1С с опытом Python, зарплата 145053 руб
```

```
Программистр-разработчик  информационных систем с опытом Python, зарплата 170207
 руб
Инженер - программист АСУ ТП с опытом Python, зарплата 195875 руб
инженер-программист с опытом Python, зарплата 107845 руб
Программист C++ с опытом Python, зарплата 121703 руб
Программист/ Junior Developer с опытом Python, зарплата 144676 руб
Программист / Senior Developer с опытом Python, зарплата 125821 руб
Инженер-электронщик (программист АСУ ТП) с опытом Python, зарплата 158128 руб
Старший программист с опытом Python, зарплата 108844 руб
Web-программист с опытом Python, зарплата 146926 руб
Веб - программист (PHP, JS) / Web разработчик с опытом Python, зарплата 177160 р
уб
Программист/ технический специалист с опытом Python, зарплата 158742 руб
программист 1С с опытом Python, зарплата 109344 руб
Программист C# с опытом Python, зарплата 136115 руб
Инженер-программист 1 категории с опытом Python, зарплата 109044 руб
Ведущий инженер-программист с опытом Python, зарплата 128522 руб
Инженер-программист САПОУ (java) с опытом Python, зарплата 128757 руб
Помощник веб-программиста с опытом Python, зарплата 149102 руб
веб-программист с опытом Python, зарплата 197180 руб
педагог программист с опытом Python, зарплата 121388 руб
Инженер-программист ПЛИС с опытом Python, зарплата 153363 руб
Инженер-программист с опытом Python, зарплата 123423 руб
time:  0.03556060791015625
```