Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ 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

Выполнила:

студент группы ИУ5-53Б

Латыпова К.Н.

Москва, 2020 г.

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.
   - Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.
   - При реализации необходимо использовать конструкцию **kwargs.
   - Итератор должен поддерживать работу как со списками, так и с генераторами.
   - Итератор не должен модифицировать возвращаемые значения.

   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.

- 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.

2. Текст программы

main.py:

```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows,
actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

unique.py:

```
from lab_python_fp.gen_random import gen_random
class Unique(object):
    def __init__(self, items, **kwargs):
        self.unique_items=[]
        self.items=iter(items)
        if 'ignore_case' not in kwargs:
            self.ignore_case=False
        else:
            self.ignore_case=kwargs['ignore_case']
    def __next__(self):
        while True:
            item=self.items.__next__()
            compare_item=None
            if self.ignore_case and type(item) is str:
                compare_item=item.lower()
            else:
                compare_item=item
            if compare_item not in self.unique_items:
                self.unique_items.append(compare_item)
                return item
    def __iter__(self):
        return self
```

```
a=[1,1,1,1,1,2,2,2,2,2]
b=['a','A','b', 'B', 'a', 'A', 'c', 'C']
c=gen_random(5,3,7)
print(list(Unique(a)))
print(list(Unique(b)))
print(list(Unique(b, ignore_case=True)))
print(list(Unique(c)))
```

sort.py:

```
data = [4, -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 data: abs(data),
reverse=True)
    print("Результат с лямбда", result_with_lambda)
```

process_data.py:

```
import json
import sys
from lab_python_fp.gen_random import gen_random
from lab_python_fp.print_result import print_result
from lab_python_fp.unique import Unique
from lab_python_fp.cm_timer import cm_timer_1, cm_timer_2
from lab_python_fp.field import field
import time

path =""

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

@print_result
def f1(arg):
    return sorted(list(Unique(field(arg, 'job-name'), ignore_case=True)),
key=lambda x: str.casefold(x))


@print_result
def f2(arg):
    return list(filter(lambda x: "программист" in x.lower(), arg))


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


@print_result
def f4(arg):
    return dict(zip(arg, gen_random(len(arg), 100000, 200000)))


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

```
    with cm_timer_2():
        f4(f3(f2(f1(data))))
```

print_result.py:

```
def print_result(func, *arg):
    def decor(*arg):
        print(func.__name__)
        a=func(*arg)
        if type(a) is list:
            for k  in a:
                print(k)
        elif type(a) is dict:
            for k, v in a.items():
                print(k,'=',v)
        else:
            print(a)
        return a
    return decor

@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__':
    test_1()
    test_2()
    test_3()
    test_4()
```

gen_random.py:

```
import random

def gen_random(a,b,c):
    for _ in range(a):
        yield random.randint(b, c)

print(list(gen_random(5,1,3)))
```

field.py:

```
def field(items, *args):
    assert len(args)>0
    if len(args)==1:
        for a in items:
```

```
            for b in args:
                if b in a:
                    yield a[b]
    else:
        for a in items:
            c={}
            for b in args:
                if b in a:
                    c[b]=a[b]
            if len(c.keys())>0:
                yield c

goods=[
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха','price': 5300, 'color': 'black'}
    ]

print(list(field(goods, 'title')))
print(list(field(goods, 'title', 'price')))
```

cm_timer.py:

```
import time

class cm_timer_1:

    def __enter__(self):
        self.time = time.time()

    def __exit__(self, value, key, traceback):
        print(time.time()-self.time)

class cm_timer_2:
    def __init__(self):
        self._start_time = None

    def __enter__(self):
        self._start_time = time.perf_counter()

    def __exit__(self, value, key, traceback):
        elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None
        print(f"Elapsed time: {elapsed_time:0.4f} seconds")
with cm_timer_1():
    time.sleep(5.5)
with cm_timer_2():
    time.sleep(3.5)
```

3. Экранные формы с примерами выполнения программы:

main.py:

```
Hi, PyCharm

Process finished with exit code 0
```

unique.py:

```
[2, 2, 3, 1, 2]
[1, 2]
['a', 'A', 'b', 'B', 'c', 'C']
['a', 'b', 'c']
[7, 4, 5, 6]

Process finished with exit code 0
```

sort.py:

```
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Результат с лямбда [123, 100, -100, -30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

process_data.py:

```
[2, 1, 3, 3, 1]
[1, 2]
['a', 'A', 'b', 'B', 'c', 'C']
['a', 'b', 'c']
[3, 6, 5, 7]
5.5005810260772705
Elapsed time: 3.4999 seconds
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]
```

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
```

gen_random.py:

```
[3, 2, 1, 2, 1]

Process finished with exit code 0
```

field.py:

```
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]

Process finished with exit code 0
```

cm_timer.py:

```
5.5008015632629395
Elapsed time: 3.5005 seconds

Process finished with exit code 0
```