Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления (ИУ5)

О т ч е т

по лабораторной работе №3

Функциональные возможности языка Python

Дисциплина: Разработка Интернет-Приложений

Студент гр. ИУ5-53Б _____ Терентьев В.О.

(Подпись, дата) (Фамилия И.О.)

Преподаватель _____ Гапанюк Ю.Е.

(Подпись, дата) (Фамилия И.О.)

Москва, 2020

1. Описание задания

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.

1.1. Задача 1 (файл field.py)

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

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количество аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

1.2. 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.

1.3. 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.

1.4. 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. Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

1.5. 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.

1.6. 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.
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.

1.7. 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

2.1. lab_python_fp\field.py

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

```
# field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
# field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000
}, {'title': 'Диван для отдыха', 'price': 5300}

def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for item in items:
            if args[0] in item:
                yield item[args[0]]
    else:
        for item in items:
            d = {arg: item[arg] for arg in args if arg in item}
            if len(d) > 0:
                yield d

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

    print([x for x in field(goods, "title")],
        [x for x in field(goods, "title", "price")],
        sep="\n")
```

2.2. lab_python_fp\gen_random.py

```
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
# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randrange(begin, end + 1)

if __name__ == '__main__':
    print([x for x in gen_random(5, 1, 3)])
```

2.3. lab_python_fp\unique.py

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-
параметр ignore_case,
```

```
        # в зависимости от значения которого будут считаться одинаковыми строки в
 разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из к
оторых удалится
        # По-умолчанию ignore_case = False
        self.data = iter(items)
        self.uniq = set()
        if 'ignore_case' not in kwargs:
            self.ignore_case = False
        else:
            self.ignore_case = kwargs['ignore_case']

    def __next__(self):
        if self.ignore_case:
            try:
                current = self.data.__next__().lower()
            except Exception:
                current = self.data.__next__()
        else:
            current = self.data.__next__()
        if current not in self.uniq:
            self.uniq.add(current)
            return current
        else:
            return self.__next__()

    def __iter__(self):
        return self

if __name__ == '__main__':
    from gen_random import gen_random

    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

    for i in Unique(data):
        print(i)

    data = gen_random(10, 1, 3)
    print('---')
    for i in Unique(data):
        print(i)

    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print('---')
    for i in Unique(data):
        print(i)

    print('---')
    for i in Unique(data, ignore_case=True):
        print(i)
```

2.4. lab_python_fp\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), reverse=True)
    print(result_with_lambda)
```

2.5. lab_python_fp\print_result.py

```
def print_result(func_to_decorate):

    def decorated_func(*args):
        print(func_to_decorate.__name__)
        rs = func_to_decorate(*args)
        if type(rs) == list:
            for item in rs:
                print(item)
        elif type(rs) == dict:
            for i, k in rs.items():
                print(i, '=', k)
        else:
            print(rs)
        return rs

    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]


if __name__ == '__main__':
```

```
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

2.6. lab_python_fp\cm_timer.py

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

class cm_timer_1:

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

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            print(time() - self.t)

@contextmanager
def cm_timer_2():
    t = time()
    yield
    print(time() - t)

if __name__ == '__main__':
    with cm_timer_1():
        sleep(5.5)

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

2.7. lab_python_fp\process_data.py

```
import json
import sys
from field import field
from gen_random import gen_random
from unique import Unique
from print_result import print_result
from cm_timer import cm_timer_1
# Сделаем другие необходимые импорты

path = "C:\\Users\\webma\\iCloudDrive\\MGTU\\5\\1 DIA\\github\\DIA\\lab3\\data_li
ght.json"
```

```
# Необходимо в переменную path сохранить путь к файлу, который был передан при за
пуске сценария

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

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemen
ted`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    return sorted(Unique([x for x in field(arg, "job-name")], ignore_case=True))


@print_result
def f2(arg):
    return list(filter(lambda x: x.lower().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('{0}, зарплата {1} руб.'.format(x[0], x[1]) for x in zip(arg, gen
_random(len(arg), 100000, 200000)))


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

3. Экранные формы с примерами выполнения программы

3.1. lab_python_fp\field.py

```
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3> & C:/Use
n.exe "c:/Users/webma/iCloudDrive/MGTU/5/1 DIA/github/DIA/lab3/lab_
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

3.2. lab_python_fp\gen_random.py

```
PS C:\Users\webma\i
n.exe "c:/Users/web
[3, 3, 3, 2, 1]
PS C:\Users\webma\i
```

3.3. lab_python_fp\unique.py

```
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3> &
n.exe "c:/Users/webma/iCloudDrive/MGTU/5/1 DIA/github/DIA/lab
1
2
---
1
3
2
---
a
A
b
B
---
a
b
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

3.4. lab_python_fp\sort.py

```
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3> &
n.exe "c:/Users/webma/iCloudDrive/MGTU/5/1 DIA/github/DIA/lab
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

3.5. lab_python_fp\print_result.py

```
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3> &
n.exe "c:/Users/webma/iCloudDrive/MGTU/5/1 DIA/github/DIA/lab
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

3.6. lab_python_fp\cm_timer.py

```
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3> &
n.exe "c:/Users/webma/iCloudDrive/MGTU/5/1 DIA/github/DIA/lab
5.507290601730347
5.5098536014556885
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

3.7. lab_python_fp\process_data.py

```
юрист волонтер
юристконсульт
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
программист c++/c#/java с опытом Python
программист/ junior developer с опытом Python
программист/ технический специалист с опытом Python
программистр-разработчик  информационных систем с опытом Python
f4
программист с опытом Python, зарплата 199375 руб.
программист / senior developer с опытом Python, зарплата 181033 руб.
программист 1с с опытом Python, зарплата 122383 руб.
программист c# с опытом Python, зарплата 109027 руб.
программист c++ с опытом Python, зарплата 145334 руб.
программист c++/c#/java с опытом Python, зарплата 150450 руб.
программист/ junior developer с опытом Python, зарплата 103081 руб.
программист/ технический специалист с опытом Python, зарплата 124173 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 193027 руб.
2.17500638961792
PS C:\Users\webma\iCloudDrive\MGTU\5\1 DIA\github\DIA\lab3>
```

4. Ссылка на репозиторий

https://github.com/iYroglif/DIA