Министерство образования и науки 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)

Отчёт по лабораторной работе №3 по курсу «Разработка Интернет-Приложений»

Тема работы: "Функциональные возможности языка Python"

Выполнил: Брусов Никита, РТ5-51Б

Проверил:_____

15 октября 2020 г.

ЗАЧТЕНО / НЕ ЗАЧТЕНО_____

(подпись)

1. Цель лабораторной работы.

Изучение основных конструкций и возможностей ЯП Python при функциональном подходе.

2. Задание на лабораторную работу.

Для лабораторной работы №3 необходимо разработать консольное приложение, включающее в себя несколько модулей, представляющих из себя объекты, работающие в парадигме функционального программирования. Требуется создать:

1. Обработчик коллекции (генератор), работающий со списком словарей и выдающий те словари, в которых содержатся задаваемые в виде параметров вызова ключи;
2. Генератор случайных целых чисел;
3. Итератор, исключающий из исходной коллекции (списка или генератора) дубликаты;
4. Процедуру сортировки списка чисел по их абсолютному значению с помощью расширений коллекций (с использованием лямбда-выражения и без него);
5. Декоратор функции, предварительно оповещающей об исполняющейся процедуре и возвращающий её значения в удобном для чтения формате;
6. Контекстный менеджер в виде таймера (в двух исполнениях: как класс и как функция с использованием библиотеки “contextlib”);
7. Обработчик данных из файла в формате JSON, используя при этом созданные в предыдущих частях лабораторной работы элементы.

3. Ход выполнения лабораторной работы.

Первым делом создаём обработчик коллекции:

```
def field(dicts, *args):
    if len(args) > 1:
        for dictionary in dicts:
            for arg in args:
                if arg not in dictionary.keys():
                    break
             else:
                yield dictionary
    elif len(args) > 0:
        for dictionary in dicts:
            for dictKey in dictionary.keys():
                for arg in args:
                    if dictKey == arg:
                        yield dictionary.get(dictKey)
```

Далее – генератор случайных чисел:

```
def genRandom(amountOfNums, minNum, maxNum):
    for i in range(amountOfNums):
        yield random.randint(minNum, maxNum)
```

Итератор, пропускающий дубликаты:

```
class UniqueIterator:
    def __init__(self, **kwargs):
        self.currentIndex = 0
        self.usedElements = set()
        # Насколько правильно использован kwargs?
        self.ignoreCase = kwargs.get("ignoreCase")
        data = kwargs.get("data")
        if isinstance(data, GeneratorType):
            # Если на вход получен генератор, то необходимо создать коллекцию
            self.data = list()
            for generatorElement in data:
                self.data.append(generatorElement)
        else:
            self.data = data
        self.length = len(self.data)

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            if self.currentIndex >= self.length:
                raise StopIteration
            else:
                currentElement = self.data[self.currentIndex]
                self.currentIndex = self.currentIndex + 1
                if isinstance(currentElement, str) and self.ignoreCase == True:
                    condition = currentElement.lower()
                else:
                    condition = currentElement
                if (condition) not in self.usedElements:
                    self.usedElements.add(condition)
                    return currentElement
```

Сортировка списка:

```
generator = genRandom(20, -10, 10)
data = list()
for num in generator:
    data.append(num)
print(data)
# Насколько верна задумка?
# без лямбда-выражения
print("No lambda:")
print(list(i[1] for i in reversed(sorted(zip([abs(num) for num in data], data)))))
# с лямбда-выражением
print("Lambda:")
print(list(i[1] for i in reversed(sorted(zip(map(lambda x: abs(x), data), data)))))
```

Декоратор:

```
def printResult_decorator(funcToPrint):
    def decorating(*args, **kwargs):
        funcResult = funcToPrint(*args, **kwargs)
        print("Function {0} returns: ".format(funcToPrint.__name__), end = '')
        if isinstance(funcResult, list):
            print()
            for listElement in funcResult:
                print(listElement)
        elif isinstance(funcResult, dict):
            print()
            for dictElement in funcResult.items():
                print("Key = {0} -> Value = {1}".format(dictElement[0], dictElement[1]))
        else:
            print(funcResult)
        print("_" * 20)
        return funcResult
    return decorating
```

Таймер:

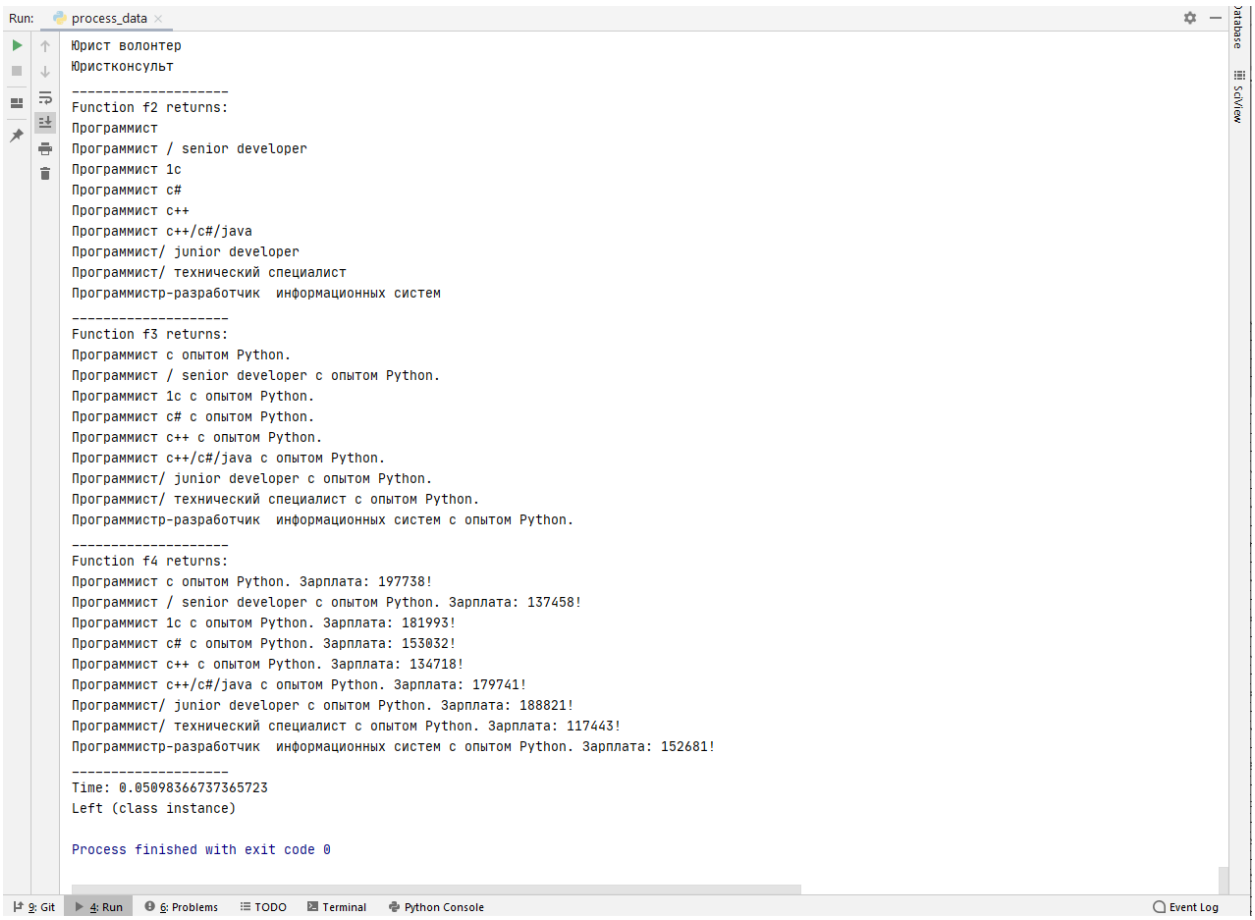
```
class Timer_CM_Class:
    def __init__(self):
        print("Created (class instance)")
        self.timer = None


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


    def __exit__(self, exc_type, exc_val, exc_tb):
        self.timer = time.time() - self.timer
        print("Time: {0}".format(self.timer))
        print("Left (class instance)")


@contextmanager
def Timer_CM_Lib():
    print("Entered (func with contextlib)")
    timer = time.time()
    [i for i in genRandom(80000, -10, 10)]
    yield time.time() - timer
    print("Left (func with contextlib)")
```

Обработка данных:

```
import json
from lab_python_fp.print_result import printResult_decorator
from lab_python_fp.unique import UniqueIterator
from lab_python_fp.cm_timer import Timer_CM_Class
from lab_python_fp.field import field
from lab_python_fp.gen_random import genRandom

@printResult_decorator
def f1(dataFile):
    # Правомерно ли с точки зрения задания модифицировать исходный регистр?
    return sorted(vocation.capitalize() for vocation in UniqueIterator(data = field(dataFile, "job-name"), ignoreCase = True))


@printResult_decorator
def f2(sortedData):
    return list(filter(lambda x : "Программист" in x, sortedData))


@printResult_decorator
def f3(filteredData):
    return list(map(lambda x : x + " с опытом Python.", filteredData))


@printResult_decorator
def f4(modifiedData):
    return list(str(info[0]) + " Зарплата: " + str(info[1]) + "!" for info in zip(modifiedData,genRandom(len(modifiedData), 100000, 200000)))


if __name__ == "__main__":
    data = list()
    with open("d:\\study\\5_Семестр\\РИП\\Lab3\\data_light.json", 'r', encoding = 'utf8') as data_file:
        data = json.load(data_file)
    with Timer_CM_Class():
        f4(f3(f2(f1(data))))
```

4. Результаты работы

Поскольку размер коллекции составляет тысячи записей, скриншоты с демонстрацией выведения списка профессий без дубликатов (состоящего так же из нескольких тысяч элементов) не прилагаются. Результаты последующей обработки:


Run:
process_data
Юрист волонтер
Юристконсульт

Function f2 returns:
Программист
Программист / senior developer
Программист 1с
Программист c#
Программист c++
Программист c++/c#/java
Программист/ junior developer
Программист/ технический специалист
Программистр-разработчик информационных систем

Function f3 returns:
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.

Function f4 returns:
Программист с опытом Python. Зарплата: 197738!
Программист / senior developer с опытом Python. Зарплата: 137458!
Программист 1с с опытом Python. Зарплата: 181993!
Программист c# с опытом Python. Зарплата: 153032!
Программист c++ с опытом Python. Зарплата: 134718!
Программист c++/c#/java с опытом Python. Зарплата: 179741!
Программист/ junior developer с опытом Python. Зарплата: 188821!
Программист/ технический специалист с опытом Python. Зарплата: 117443!
Программистр-разработчик информационных систем с опытом Python. Зарплата: 152681!

Time: 0.05098366737365723
Left (class instance)
Process finished with exit code 0
Database
SciView
9: Git
4: Run
6: Problems
TODO
Terminal
Python Console
Event Log