**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 «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3

Выполнил:

студент группы ИУ5-33Б

Требуков Д.А.

Проверил:

Преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Москва, 2021 г.

***Задание:***

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.

***Текст программы:***

**field.py**

def field(items, \*args):  
 try:  
 assert len(args) > 0  
 except AssertionError:  
 print('Вы не указали ключи для вывода')  
 if len(args) == 1:  
 for arg in args:  
 for i in range(len(items)):  
 for j in items[i]:  
 if j == arg and items[i][j] is not None:  
 yield items[i][j]  
 else:  
 for i in range(len(items)):  
 d = {}  
 for arg in args:  
 for j in items[i]:  
 if j == arg:  
 d[j] = items[i][j]  
 yield d  
  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'}  
]  
#x = field(goods, 'title', 'color')  
#for i in range(len(goods)):  
# if i != len(goods) - 1:  
# print(next(x), end=', ')  
# else:  
# print(next(x))

**gen\_random.py**

import random  
  
def gen\_random(count, begin, end):  
 for x in range(count):  
 #print(random.randint(begin, end), end=' ')  
 yield random.randint(begin, end)  
  
  
  
#gen\_random(5, 1, 3)

**printresult.py**

import lab\_python\_fp.sort  
  
def printresult(func):  
 def f(data):  
 print('Имя функции {}'.format(func.\_\_name\_\_))  
 print('Результат функции:')  
 if type(func(data)) == list:  
 for i in func(data):  
 print(i)  
 elif type(func(data)) == dict:  
 for key, value in func(data).items():  
 print(key, ' = ', value)  
 else:  
 print(func(data))  
 return func(data)  
 return f  
  
  
@printresult  
def f1(data):  
 pass  
  
  
@printresult  
def f2(data):  
 pass  
  
  
@printresult  
def f3():  
 pass  
  
  
@printresult  
def f4():  
 pass  
"""  
@printresult  
def test\_1():  
 return 1  
  
  
@printresult  
def test\_2():  
 return 'iu5'  
  
  
@printresult  
def test\_3():  
 return {'a': 1, 'b': 2}  
  
  
@printresult  
def test\_4():  
 return [1, 2]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('!!!!!!!!')  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4() """

**processdata.py**

import lab\_python\_fp.sort  
import lab\_python\_fp.unique  
import lab\_python\_fp.timer  
import lab\_python\_fp.printresult  
import lab\_python\_fp.gen\_random  
import lab\_python\_fp.field  
import json  
import sys  
import os  
  
  
path = r'C:\Users\trebu\OneDrive\Рабочий стол\lab3\lab\_python\_fp\data\_light.json'  
with open(path, encoding='utf-8') as f:  
 data = json.load(f)  
  
@lab\_python\_fp.printresult.printresult  
def f1(arg):  
 return lab\_python\_fp.sort.sort(arg)  
  
  
@lab\_python\_fp.printresult.printresult  
def f2(arg):  
 return list(filter(filtration, arg))  
  
  
@lab\_python\_fp.printresult.printresult  
def f3(arg):  
 return list(map(lambda x: x + ' с опытом Python', arg))  
  
  
@lab\_python\_fp.printresult.printresult  
def f4(arg):  
 salary = [i for i in lab\_python\_fp.gen\_random.gen\_random(len(arg), 100000, 200000)]  
 return ['{}, зарплата {} руб.'.format(job, salary) for job, salary in zip(arg, salary)]  
  
  
def filtration(arg):  
 if arg[0:11] == 'Программист':  
 return True  
 else:  
 return False  
  
  
with lab\_python\_fp.timer.cm\_timer\_1():  
 f4(f3(f2(f1(data))))

**sort.py**

import lab\_python\_fp.unique  
import lab\_python\_fp.field  
  
def sort(data):  
 ndata = []  
 ndata = lab\_python\_fp.field.field(data, 'job-name')  
 ndata = list(lab\_python\_fp.unique.Unique(ndata, True))  
 ndata = sorted(ndata, reverse=True)  
 # for i in ndata:  
 # print(i, end=', ')  
 # print()  
 return list(ndata)  
  
  
#data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]  
#result = sorted(data, key = abs, reverse=True)  
#print(result)  
#resultLambda = sorted(data, key = lambda x: x if x > 0 else -x, reverse=True)  
#print(resultLambda)

**timer.py**

import time  
import contextlib  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time.time()  
  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print('time: ', time.time() - self.start\_time)  
  
  
@contextlib.contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 yield  
 print('time: ', time.time() - start\_time)  
  
  
with cm\_timer\_1():  
 time.sleep(0)  
  
  
with cm\_timer\_2():  
 time.sleep(0)

**unique.py**

class Unique(object):  
 def \_\_init\_\_(self, items, ignore\_case=False, \*\*kwargs):  
 self.seen = set()  
 self.items = items  
 self.ic = ignore\_case  
 self.kwargs = kwargs  
  
 def \_\_next\_\_(self):  
 it = iter(self.items)  
 while True:  
 try:  
 current = next(it)  
 except StopIteration:  
 raise  
 else:  
 if self.ic == True and isinstance(current, str):  
 if current.lower() not in self.seen:  
 self.seen.add(current.lower())  
 return current  
 elif current not in self.seen:  
 self.seen.add(current)  
 return current  
  
 def \_\_iter\_\_(self):  
 return self  
  
#data = ['A', 'a', 'c', 'A', 'b', 'B', 'a', 'B']  
#data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
#un = Unique(data)  
#for i in un:  
# print(i, end=' ')  
#print()