**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 «Системы обработки информации и управления»**

**Курс «Базовые компоненты интернет-технологий»**

Отчёт по лабораторной работе №3

Выполнил:

студент группы ИУ5-33Б

Александр Саргсян

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

2022 г.

**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 располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

**Текст программы:**

*field.py*

def field(items, \*args):  
 assert len(args) > 0  
 arr = []  
 if len(args) == 1:  
 for item in items:  
 for el in item:  
 if el == args[0] and item[el] is not None:  
 arr.append(item[el])  
 return arr  
 else:  
 for item in items:  
 for\_dict = dict()  
 for el in item:  
 for argument in args:  
 if el == argument and item[argument] is not None:  
 for\_dict[el] = item[el]  
 return for\_dict  
  
def main\_field():  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 res = field(goods, 'title')  
 for el in res:  
 print(el)  
 print(field(goods, 'title', 'price'))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_field()

*print\_result.py*

def print\_result(function):  
 def control(arr=[], \*args, \*\*kwargs):  
 print(function.\_\_name\_\_)  
 if len(arr) == 0:  
 result = function(\*args, \*\*kwargs)  
 else:  
 result = function(arr, \*args, \*\*kwargs)  
 if type(result) is list:  
 print('\n'.join(map(str, result)))  
 elif type(result) is dict:  
 for key, el in result.items():  
 print(f'{key} = {el}')  
 else:  
 print(result)  
 return result  
 return control  
  
@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\_result():  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_print\_result()

*sort.py*

def main\_sort():  
 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 a: abs(a), reverse=True)  
 print(result\_with\_lambda)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_sort()

*gen\_random.py*

from random import randint  
  
def gen\_random(count, first, last):  
 return (randint(first, last) for \_ in range(count))  
  
def main\_gen\_random():  
 numbers = gen\_random(4, 0, 5)  
 for el in numbers:  
 print(el, end=' ')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_gen\_random()

*unique.py*

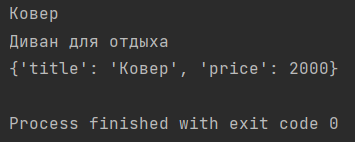
class Unique(object):  
 def \_\_init\_\_(self, items, ignore\_case=False, \*\*kwargs):  
 self.\_data = items  
 self.\_ignore\_case = ignore\_case  
 self.\_used\_data = set()  
 self.\_index = 0  
  
 def \_\_next\_\_(self):  
 if self.\_ignore\_case:  
 for count, el in enumerate(self.\_data):  
 if type(el) is str:  
 self.\_data[count] = el.lower()  
 while True:  
 if self.\_index >= len(self.\_data):  
 raise StopIteration  
 else:  
 current = self.\_data[self.\_index]  
 self.\_index += 1  
 if current not in self.\_used\_data:  
 self.\_used\_data.add(current)  
 return current  
  
 def \_\_iter\_\_(self):  
 return self  
  
  
def main\_unique():  
 data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
 ex = Unique(data, ignore\_case=True)  
 try:  
 while True:  
 print(ex.\_\_next\_\_())  
 except StopIteration:  
 print('Error "StopIteration"')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_unique()

*cm\_timer.py*

from time import time, sleep  
from contextlib import contextmanager  
  
class cm\_timer\_1:  
 def \_\_int\_\_(self):  
 self.\_start = 0  
 self.\_end = 0  
  
 def \_\_enter\_\_(self):  
 self.\_start = time()  
  
 def \_\_exit\_\_(self, the\_type, the\_value, the\_backing):  
 self.\_end = time()  
 print(f'Time of work: {self.\_end - self.\_start}')  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time()  
 yield None  
 end\_time = time()  
 print(f'Time of work: {end\_time - start\_time}')  
  
def main\_cm\_timer():  
 with cm\_timer\_1():  
 sleep(5.5)  
  
 with cm\_timer\_2():  
 sleep(5.5)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main\_cm\_timer()

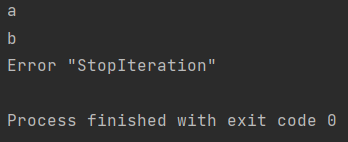
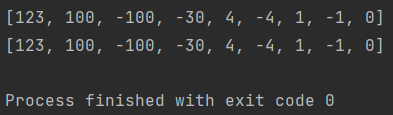
*process\_data.py*

from lab\_python\_fp.print\_result import print\_result  
from lab\_python\_fp.cm\_timer import cm\_timer\_1  
from lab\_python\_fp.gen\_random import gen\_random  
import json  
import sys  
  
try:  
 path = sys.argv[1]  
 print(path)  
except:  
 path = '../data\_light.json'  
  
with open(path, encoding='UTF-8') as f:  
 data = json.load(f)  
  
  
@print\_result  
def f1(arg):  
 return sorted(list(set([el['job-name'] for el in arg])), key=lambda a: a.lower())  
  
@print\_result  
def f2(arg):  
 return list(filter(lambda s: (s.split())[0].lower() == 'программист', arg))  
  
@print\_result  
def f3(arg):  
 return list(map(lambda k: k + ' с опытом Python', arg))  
  
  
@print\_result  
def f4(arg):  
 return list(zip(arg, ['зарплата ' + str(el) + ' руб.' for el in gen\_random(len(arg), 100000, 200000)]))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

**Примеры выполнения:**

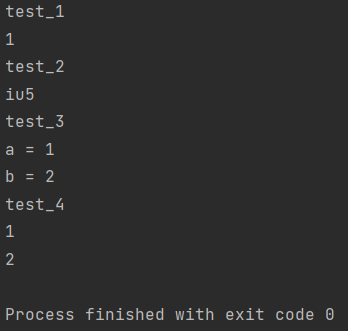
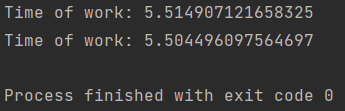
**** gen\_random.py

field.py

****

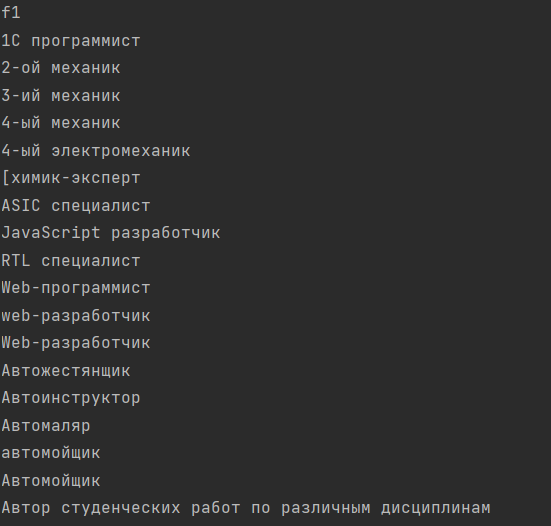
sort.py

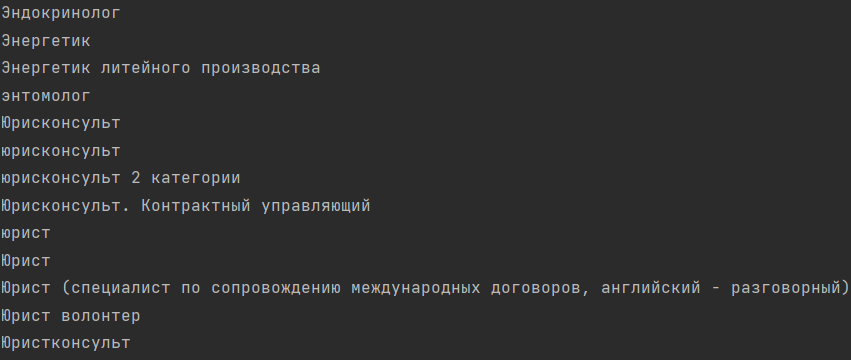
unique.py

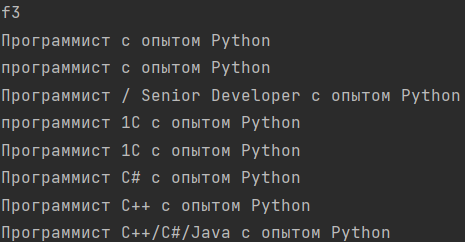
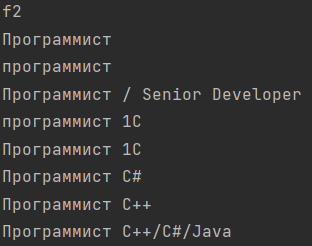
****

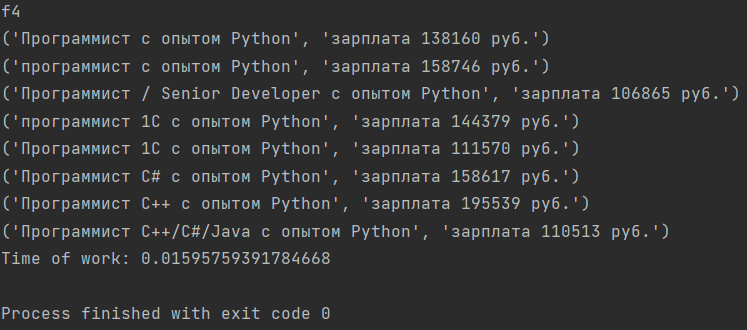
cm\_timer.py

print\_result.py

****

**…**

****

**** process\_data.py