**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «ПиКЯП»

Отчет по лабораторной работе №3

«Функциональные возможности языка Python.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34Б |  | преподаватель каф. ИУ5 |
| Сафронов Андрей |  | Нардид А. Н. |
|  |  |  |

Москва, 2024 г.

**Цель лабораторной работы:** изучение возможностей функционального программирования в языке Python.

**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.

Код программы:

main.py

from lab\_python\_fp.field import task1  
from lab\_python\_fp.gen\_random import task2  
from lab\_python\_fp.unique import task3  
from lab\_python\_fp.sort import task4  
from lab\_python\_fp.print\_result import task5  
from lab\_python\_fp.cm\_timer import task6  
from lab\_python\_fp.process\_data import task7  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 task7()

cm\_timer.py

import time  
import contextlib  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.timer = time.time()  
  
 def \_\_exit\_\_(self, \*args):  
 print(time.time() - self.timer)  
  
  
@contextlib.contextmanager  
def cm\_timer\_2():  
 timer = time.time()  
 yield  
 print(time.time() - timer)  
  
  
def task6():  
 with cm\_timer\_1():  
 time.sleep(0.5)  
 with cm\_timer\_2():  
 time.sleep(0.5)

field.py

def field(items, \*args):  
 assert len(args) > 0  
 result = []  
 if len(args) == 1:  
 for dct in items:  
 if args[0] in dct and dct[args[0]] is not None:  
 result.append(dct[args[0]])  
 else:  
 for dct in items:  
 temp = dict()  
 for key in args:  
 if key in dct and dct[key] is not None:  
 temp[key] = dct[key]  
 result.append(temp)  
 return result  
  
  
def task1():  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 print(field(goods, 'title'))  
 print(field(goods, 'title', 'price'))

gen\_random.py

import random  
  
  
def gen\_random(num\_count, begin, end):  
 return [random.randint(begin, end) for \_ in range(num\_count)]  
  
  
def task2():  
 print(\*gen\_random(5, 1, 3))

print\_result.py

import functools  
  
  
def print\_result(func):  
 @functools.wraps(func)  
 def wrapper(\*args,\*\*kwargs):  
 print(f'Function: {func.\_\_name\_\_}')  
 result = func(\*args,\*\*kwargs)  
 if type(result) is list:  
 for i in result:  
 print(i)  
 elif type(result) is dict:  
 for key, value in result.items():  
 print(f'{key} = {value}')  
 else:  
 print(result)  
 print()  
 return result  
  
 return wrapper  
  
  
@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 task5():  
 print('!!!!!!!!')  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4()

process\_data.py

import json  
import sys  
from lab\_python\_fp.field import field  
from lab\_python\_fp.gen\_random import gen\_random  
from lab\_python\_fp.unique import Unique  
from lab\_python\_fp.print\_result import print\_result  
from lab\_python\_fp.cm\_timer import cm\_timer\_1  
  
  
@print\_result  
def f1(arg):  
 result = sorted(Unique(field(arg, 'job-name'), ignore\_case=True), key=lambda s: s.lower())  
 return result  
  
  
@print\_result  
def f2(arg):  
 result = list(filter(lambda s: s.lower().startswith('программист'), arg))  
 return result  
  
  
@print\_result  
def f3(arg):  
 result = list(map(lambda s: s + ' с опытом работы Python', arg))  
 return result  
  
  
@print\_result  
def f4(arg):  
 salaries = gen\_random(len(arg), 100\_000, 200\_000)  
 result = []  
 for job, salary in zip(arg, salaries):  
 result.append(job+f', зарплата {salary} руб.')  
 return result  
  
  
def task7():  
 path = 'data\_light.json'  
 with open(path, encoding='utf8') as f:  
 data = json.load(f)  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

sort.py

def sort(lst, reverse=False):  
 return sorted(lst, key=abs, reverse=reverse)  
  
  
def sort\_with\_lambda(lst, reverse=False):  
 return sorted(lst, key=lambda x: -x if x < 0 else x, reverse=reverse)  
  
  
def task4():  
 data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
 print(sort(data, True))  
 print(sort\_with\_lambda(data, True))

unique.py

from lab\_python\_fp.gen\_random import gen\_random  
  
  
class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.data = items  
 self.itr = iter(self.data)  
 self.repeat = []  
 self.ignore\_case = False  
 if 'ignore\_case' in kwargs:  
 self.ignore\_case = kwargs['ignore\_case']  
 pass  
  
 def \_\_next\_\_(self):  
 nxt = next(self.itr)  
 while nxt in self.repeat or type(nxt) is str and self.ignore\_case and nxt.lower() in self.repeat:  
 nxt = next(self.itr)  
 else:  
 if type(nxt) is str and self.ignore\_case:  
 self.repeat.append(nxt.lower())  
 else:  
 self.repeat.append(nxt)  
 return nxt  
  
 def \_\_iter\_\_(self):  
 return self  
  
  
def task3():  
 data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
 print(\*Unique(data))  
 data = gen\_random(10, 1, 3)  
 print(\*Unique(data))  
 print(\*Unique(('aABbcC'[i] for i in range(6)), ignore\_case=True))

Результаты:

