**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

**Отчёт по Лабораторной работе №3**

**“Функциональные возможности языка Python”**

Отчёт

(вид документа)

Листы А4

(вид носителя)

9

(количество листов)

|  |  |
| --- | --- |
| Исполнитель: | Студент группы ИУ5-54Б  Самойлов А.М |

Москва - 2020

**Цель работы**: Изучения функциональных возможностей языка 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 располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

### Задача 1 (файл field.py)

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

### 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.

### 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.

### 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.

### 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.

### 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.

### 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](https://github.com/iu5team/iu5web-fall-2020/tree/master/labs/lab3/data/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.

## Листинг программы

* field.py

def field(items, \*args):  
 assert len(args) > 0, **'No Args'** if len(args) > 1:  
 for item in items:  
 dict = {}  
 for arg in args:  
 if (arg in item) and (item[arg] is not None):  
 dict[arg] = item[arg]  
 if len(dict.keys()) > 0:  
 yield dict  
 elif len(args) == 1:  
 for item in items:  
 for arg in args:  
 if (arg in item) and (item[arg] is not None):  
 yield item[arg]  
  
def main():  
 print(list(field(goods, **'title'**, **'price'**)))  
 print(list(field(goods, **'color'**)))  
 print(list(field(goods, **'title'**)))  
  
goods = [  
 {**'title'**: **'Ковер'**, **'price'**: 2000, **'color'**: **'green'**},  
 {**'title'**: **'Диван для отдыха'**, **'color'**: **'black'**},  
 {**'title'**: None, **'price'**: None},  
 ]  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

* gen\_random.py

import random  
  
*# функция рандомайзер*def gen\_random(num\_count, begin, end):  
 result = [] *# лист для результата* if begin > end: temp = begin; begin = end; end = temp; *# проверка границ интервала (корректирование)* for i in range(0, num\_count):  
 result.append(random.randint(begin, end))  
 return result  
  
def main():  
 print(gen\_random(10, 0, 12))  
 print(gen\_random(15, 0, 1))  
 print(gen\_random(5, -5, 0))  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

* unique.py

class Unique():  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 ignore = kwargs[**"ignore"**] if **"ignore"** in kwargs else False  
 self.uniqueItems = set()  
 for i in items:  
 if isinstance(i, str) and ignore:  
 self.uniqueItems.add(i.lower())  
 else:  
 self.uniqueItems.add(i)  
  
 def \_\_next\_\_(self):  
 if len(self.uniqueItems) > 0:  
 return self.uniqueItems.pop()  
 raise StopIteration()  
  
 def \_\_iter\_\_(self):  
 return self  
  
def main():  
 print(**"**\033**[33mlist1**\033**[0m: "**, list1)  
 print(**"**\033**[36mUnique list1**\033**[0m: "**, sorted(list(Unique(list1))), **"**\n**"**)  
  
 print(**"**\033**[33mlist2**\033**[0m: "**, list2)  
 print(**"**\033**[36mUnique list2**\033**[0m: "**, sorted(list(Unique(list2, ignore=False))), **"**\n**"**)  
 print(**"**\033**[33mlist2**\033**[0m: "**, list2)  
 print(**"**\033**[36mUnique list2**\033**[0m(**\033**[31mignore = True**\033**[0m): "**, sorted(list(Unique(list2, ignore=True))), **"**\n**"**)  
  
 print(**"**\033**[33mlist3**\033**[0m: "**, list3)  
 print(**"**\033**[36mUnique list3**\033**[0m: "**, list(Unique(list3, ignore=False)), **"**\n**"**)  
 print(**"**\033**[33mlist3**\033**[0m: "**, list3)  
 print(**"**\033**[36mUnique list3**\033**[0m(**\033**[31mignore = True**\033**[0m): "**, list(Unique(list3, ignore=True)),  
 **"**\n**"**)  
  
  
list1 = [1,2,3,4,4,3,2,4,5,3,6,7,8]  
list2 = [**'a'**,**'B'**,**'c'**,**'d'**,**'c'**,**'e'**,**'a'**,**'B'**,**'C'**,**'A'**,**'b'**]  
list3 = [**'a'**,**'B'**,1,**'c'**,**'d'**,3,**'c'**,**'e'**,3,**'a'**,**'B'**,2,**'C'**,5,**'A'**,5,**'b'**]  
  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

* sort.py

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
def 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)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

* cm\_timer.py

from time import time, sleep  
from contextlib import contextmanager  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start = time()  
 def \_\_exit\_\_(self, type, value, traceback):  
 self.end = time()  
 print(**f"**\033**[33mЗатраченное время**\033**[0m:** \033**[36m**{(self.end - self.start)}\033**[0m"**)  
  
@contextmanager  
def cm\_timer\_2() -> None:  
 begin = time()  
 yield  
 calcTime = time() - begin  
 print(**f"**\033**[33mЗатраченное время**\033**[0m:** \033**[36m**{calcTime}\033**[0m"**)  
  
def main():  
 with cm\_timer\_1():  
 testList = [x for x in range(100\_000)]  
  
 with cm\_timer\_1():  
 sleep(1)  
  
 with cm\_timer\_2():  
 testList = [x for x in range(100\_000\_000)]  
  
 with cm\_timer\_2():  
 sleep(1.2)  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

* print\_result.py

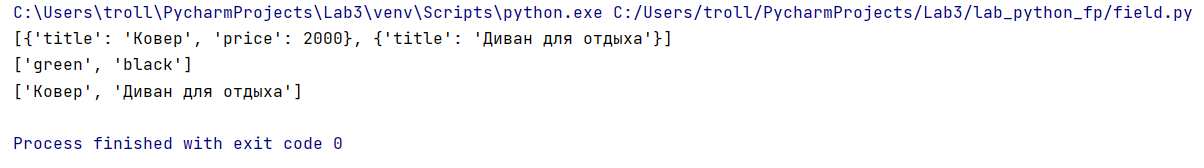
def print\_result(func):  
 def decoratedFunc(\*args, \*\*kwargs):  
 result = func(\*args, \*\*kwargs)  
 print(**"**\033**[31m"** + func.\_\_name\_\_ + **"**\033**[0m"**)  
 if isinstance(result, dict):  
 for key in result.keys():  
 print(**"{} = {}"**.format(key, result[key]))  
 elif isinstance(result, list):  
 for value in result:  
 print(value)  
 else:  
 print(result)  
 return result  
 return decoratedFunc  
  
  
@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(**'!!!!!!!!'**)  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4()  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

* process\_file.py

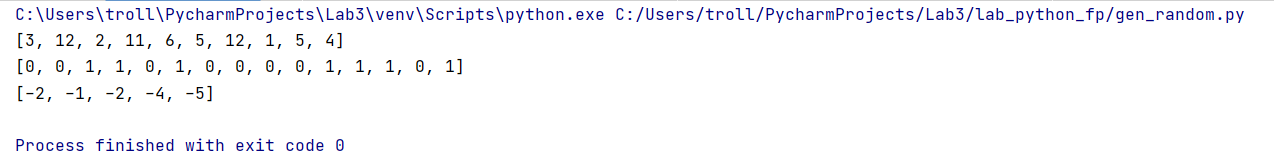
from lab\_python\_fp import print\_result, unique, gen\_random, field, cm\_timer  
import json  
  
*# Путь к json файлу*path = **"C:/Users/troll/PycharmProjects/Lab3/data\_light.json"***# парсим файл, получется список словарей*with open(path, encoding=**'utf-8'**) as f, cm\_timer.cm\_timer\_1():  
 data = json.load(f)  
  
  
@print\_result.print\_result  
def f1(arg):  
 *# Используем ранее разработанные функции, возващаем отсортированный набор названий вакансий* return sorted(set(i for i in unique.Unique(field.field(data, **'job-name'**), ignore=False)))  
  
  
@print\_result.print\_result  
def f2(arg):  
 *# отбираем только те названия, которые начинаются со слова "программист"* return list(filter(lambda x: x.startswith(**"Программист"**), arg))  
  
  
@print\_result.print\_result  
def f3(arg):  
 *# используя map дописываем "с опытом работы" к каждой вакансии* return list(map(lambda x: x + **" с опытом Python"**, arg))  
  
  
@print\_result.print\_result  
def f4(arg):  
 *# дописываем ЗП к каждой вакансии* return list(i + **f", зарплата** \033**[34m**{gen\_random.gen\_random(1,100000,200000)[0]}\033**[0m руб."** for i in arg)  
  
  
def main():  
 *# вызываем функции с таймером* with cm\_timer.cm\_timer\_1():  
 f4(f3(f2(f1(data))))  
  
*# точка входа*if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

# Результат в консоли

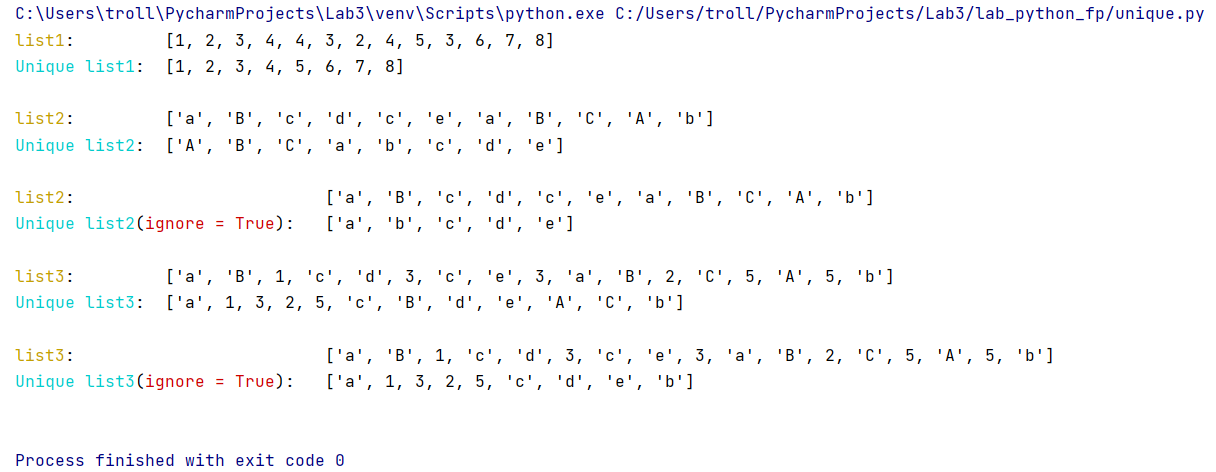
* field.py



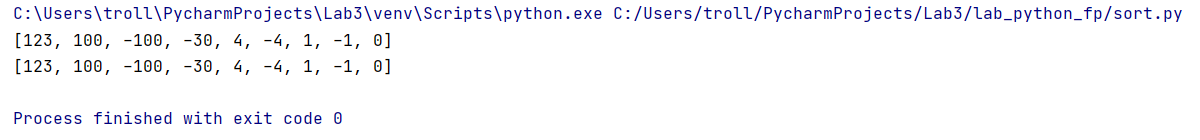
* gen\_random.py



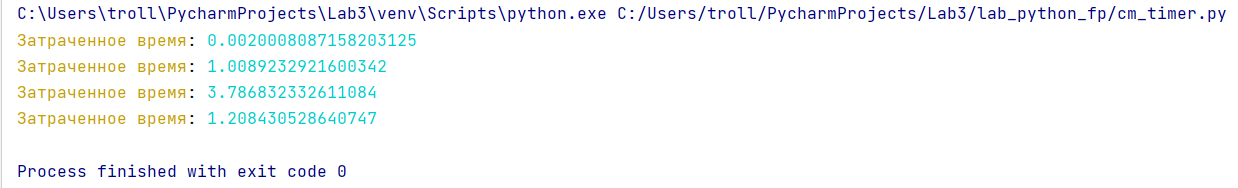
* unique.py



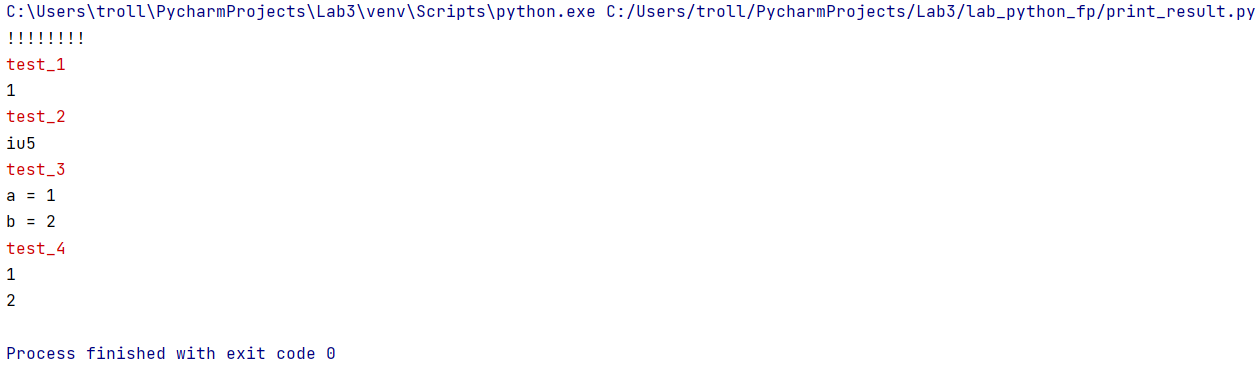
* sort.py



* cm\_timer.py



* print\_result.py



* process\_file.py

Перед этим выводится время чтения JSON и список всех вакансий до тех, что на букву «Ю»

