**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3

«Функциональные возможности языка Python».

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| Дьячков М. Ю. ИУ5-51 |  | Гапанюк Ю. Е. |
|  |  | Балашов А. М. |

Москва, 2021 г.

**Цель работы:**

Изучение возможностей функционального программирования в языке Python.

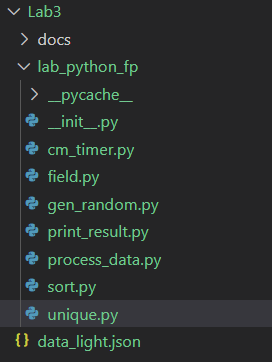
**Задание:**

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.

**Выполнение:**

Был создан пакет lab\_python\_fp, который содержит в себе модули задач. Некоторые задачи используют другие задачи как внешние модули через import.



**Задача 1:**

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

Реализация:

def zero\_args\_test(func):

    def testing(\_, \*args):

        if len(args) > 0:

            return func(\_, \*args)

        else:

            return('Нет аргументов, вызов {} невозможен'.format(func.\_\_name\_\_))

    return testing

@zero\_args\_test

def field(items, \*args):

    request = list()

    if len(args) == 1:

        for each\_dict in items:

            if args[0] in each\_dict:

                request.append(each\_dict[args[0]])

    else:

        for each\_dict in items:

            temp\_dict = {}

            for key in each\_dict:

                if (key in args):

                    temp\_dict[key]=each\_dict[key]

            request.append(temp\_dict)

    return request

def main():

    goods = [

        {'title': 'Ковер',   'price': 2000,  'color': 'Зеленый'},

        {'title': 'Диван',   'price': 5300,  'color': 'Желтый'},

        {'title': 'Кресло',  'price': 1000,  'color': 'Красное', 'used': True},

        {'title': 'Лампа',   'price': 500,   'power':'50w'},

        {'title': 'Зеркало', 'price': 800,   'used': True},

        {'title': 'Чайник',  'price': 200,   'power':'50w',      'used': False},

    ]

    print(field(goods), '\n\n')

    print(field(goods, 'title'), '\n\n')

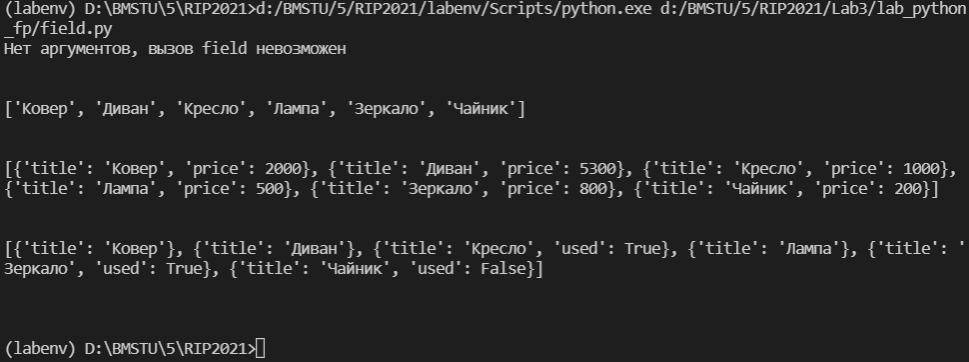
    print(field(goods, 'title', 'price'), '\n\n')

    print(field(goods, 'title', 'used'), '\n\n')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



**Задача 2:**

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.

Реализация:

from random import randrange

def get\_random(number, min, max):

    return [randrange(min, max+1) for i in range(number)]

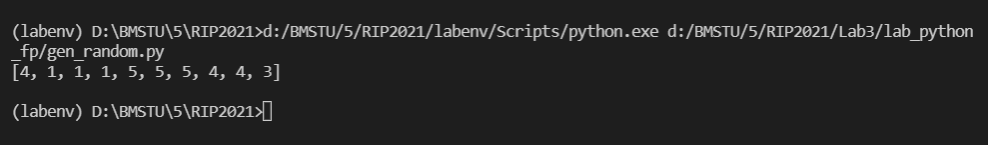
def main():

    print(get\_random(10, 1, 5))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



**Задача 3:**

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

Реализация:

from gen\_random import get\_random

class Unique(object):

    def \_\_init\_\_(self, items, \*\*kwargs):

        self.unicList = list()

        self.data = items

        self.index = 0

        if 'ignore\_case' in kwargs.keys() and kwargs['ignore\_case'] == True:

            self.ignore\_case = True

        else:

            self.ignore\_case = False

    def \_\_next\_\_(self):

        while True:

            if self.index < len(self.data):

                if self.ignore\_case == False:

                    current = self.data[self.index]

                    self.index += 1

                    if not current in self.unicList:

                        self.unicList.append(current)

                        return current

                else:

                    current = self.data[self.index]

                    self.index += 1

                    if isinstance(current, str):

                        current = current.lower()

                    if not current in self.unicList:

                        self.unicList.append(current)

                        return current

            else:

                raise StopIteration

    def \_\_iter\_\_(self):

        return self

def main():

    data = ['a','A','b', 'B', 'c', 'C', 'C', 'A', 'D']

    random\_nums = get\_random(10, 1, 7)

    print(data)

    for i in Unique(data, ignore\_case=True):

        print(i, end='\t')

    print('\n')

    print(random\_nums)

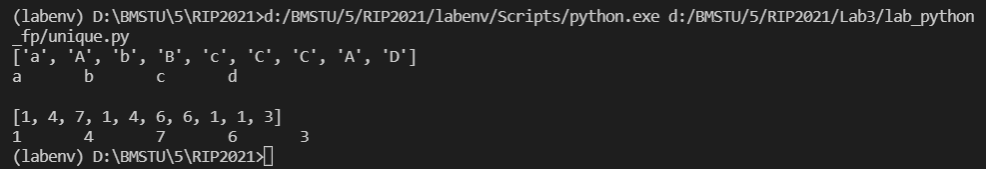
    for i in Unique(random\_nums):

        print(i, end='\t')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



**Задача 4:**

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. Пример:

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Необходимо решить задачу двумя способами:

* С использованием lambda-функции.
* Без использования lambda-функции.

Реализация:

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

def main():

    result = sorted(data, key=abs, reverse=True)

    print(result)

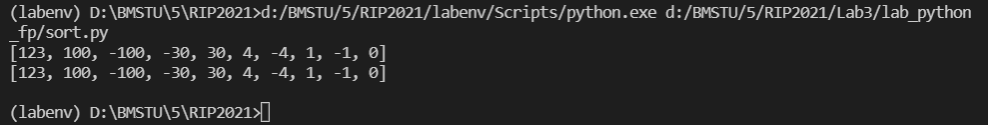
    result\_with\_lambda = (lambda mass : sorted(mass, key=abs, reverse=True))(data)

    print (result\_with\_lambda)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



**Задача 5:**

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.

Реализация:

def print\_result(func):

    def decorated\_func(\*args, \*\*kwargs):

        print(func.\_\_name\_\_)

        if isinstance(func(\*args, \*\*kwargs), dict):

            for key in func(\*args,\*\*kwargs):

                print(key, '=', func(\*args, \*\*kwargs)[key])

        elif isinstance(func(\*args, \*\*kwargs), list):

            for item in func(\*args, \*\*kwargs):

                print(item)

        else:

            print(func(\*args, \*\*kwargs))

        return func(\*args, \*\*kwargs)

    return decorated\_func

@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()

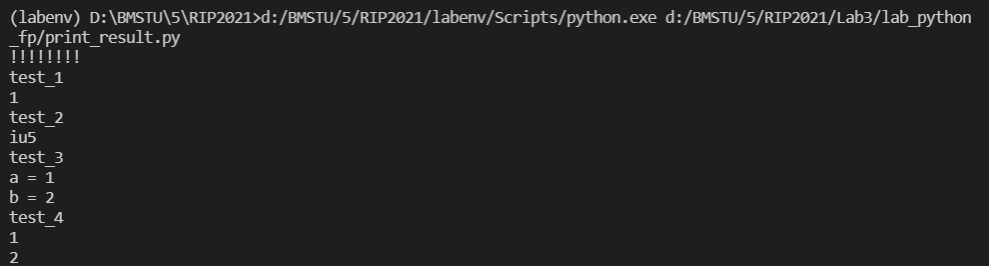
    test\_3()

    test\_4()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Запуск:



**Задача 6:**

Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время работы блока кода и выводят его на экран.

Реализация:

from time import sleep, time

from contextlib import contextmanager

class cm\_timer\_1():

    def \_\_init\_\_(self):

        self.timer = time()

    def \_\_enter\_\_(self):

        pass

    def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):

        self.timer = time() - self.timer

        print("time: {0:0.1f}".format(self.timer))

@contextmanager

def cm\_timer\_2():

    t = time()

    yield

    t = time() - t

    print("time: {0:0.1f}".format(t))

def main():

    with cm\_timer\_1():

        sleep(3.5)

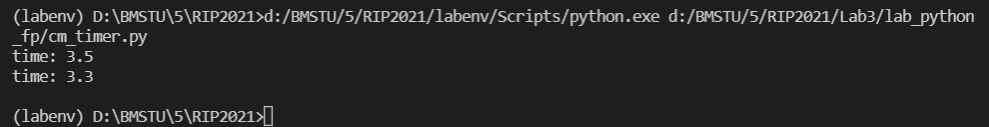
    with cm\_timer\_2():

        sleep(3.3)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



**Задача 7:**

* 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.

Реализация:

import json

from cm\_timer import cm\_timer\_1, cm\_timer\_2

from field import field

from unique import Unique

from print\_result import print\_result

from gen\_random import get\_random

@print\_result

def f1(data):

    return sorted(Unique(field(data,'job-name'), ignore\_case=True))

@print\_result

def f2(data):

    return list(filter(lambda d: 'программист' in d[:11], data))

@print\_result

def f3(data):

    return list(map(lambda x: x + " с опытом Python", data))

@print\_result

def f4(data):

    return list(zip(data, list(map(lambda x: "Зарплата " + x + " руб",map(str,(get\_random(len(data), 100000, 200000)))))))

def main():

    with open('D:\\BMSTU\\5\\RIP2021\\Lab3\\data\_light.json', 'r', encoding='utf8') as jft:

        data = json.load(jft)

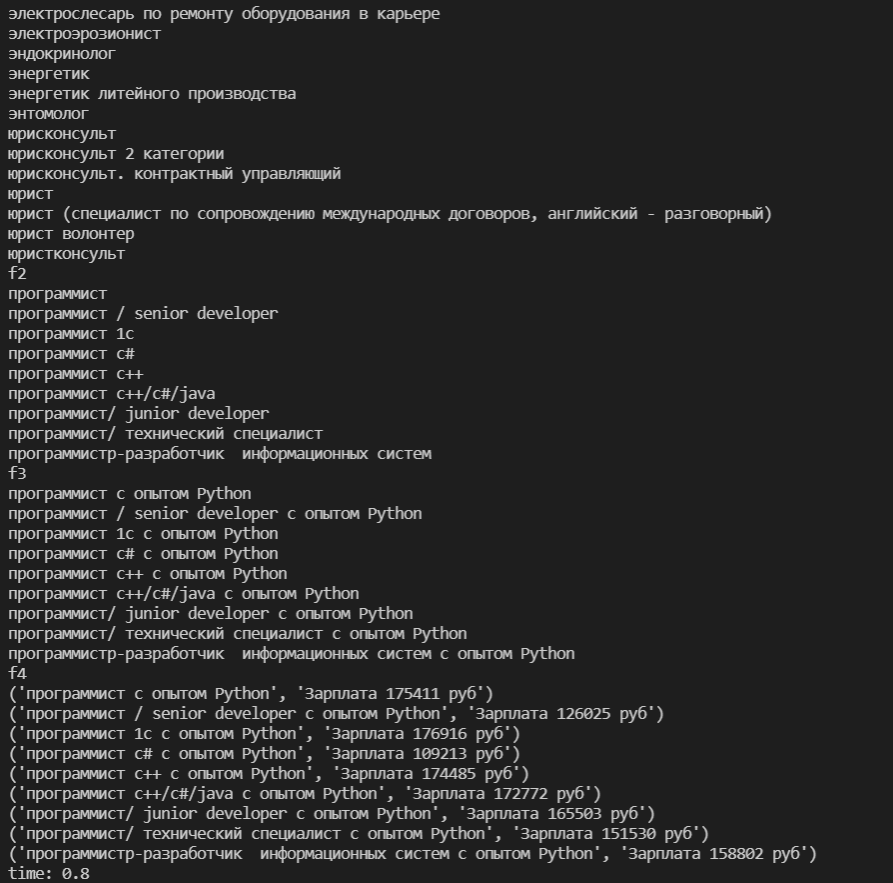
    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Запуск:



\*для первой функции не вышло заснять весь вывод по причине того, что он просто огромен, однако свою работу функция выполняет, ваканскии выводятся без повторений.