3Московский государственный технический университет имени Н.Э. Баумана

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет РТ Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по лабораторной работе № 3 по курсу**

**Базовые компоненты**

"Функциональные возможности языка Python"

10

 (количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-31Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Билалов А.К. |
|  |  | “1” декабря 2022 г. |
|  |  |  |
| Проверил |  |  |
| Доцент кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю.Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва 2022

# Оглавление

[Оглавление 2](#_Toc113796368)

[Описание задания 3](#_Toc113796369)

[Текст программы 4](#_Toc113796370)

[Результаты выполнения программы 9](#_Toc113796371)

# Описание задания

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

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.

# Текст программы

field.py:

def field(arrset, \*args):

    assert len(args) > 0

    result = []

    for seti in arrset:

        args\_in\_seti = [arg for arg in args if arg in seti]

        result.append({arg: seti[arg] for arg in args\_in\_seti} if len(args) > 1 else seti[args\_in\_seti[0]] if len(args\_in\_seti) > 0 else None)

    while None in result:

        result.remove(None)

    while {} in result:

        result.remove({})

    return result

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

    workers = [

        {'name': 'Иван', 'surname': 'Иванов', 'age': 25, 'position': 'водитель'},

        {'name': 'Петр', 'surname': 'Петров', 'age': 35, 'position': 'инженер'},

        {'name': 'Алексей', 'surname': 'Алексеев', 'age': 18, 'position': 'стажер'},

    ]

    print(field(workers, 'age'))

    print(field(workers, 'name'))

    print(field(workers, 'name', 'position'))

    print(field(workers, 0, 2))

    print(field(workers, 0))

gen\_random.py:

import random

# Пример:

# gen\_random(5, 1, 3) должен выдать выдать 5 случайных чисел

# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1

# Hint: типовая реализация занимает 2 строки

def gen\_random(num\_count, begin, end):

    return [random.randint(begin, end) for i in range(num\_count)]

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

    print(gen\_random(5, 1, 3))

    print(gen\_random(5, 1, 3))

print\_result.py

def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

        print(func.\_\_name\_\_)

        result = func(\*args, \*\*kwargs)

        if isinstance(result, list):

            print(\*result, sep='\n')

        elif isinstance(result, dict):

            for key, value in result.items():

                print(f'{key} = {value}')

        else:

            print(result)

        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]

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

    print('!!!!!!!!')

    test\_1()

    test\_2()

    test\_3()

    test\_4()

cm\_timer.py:

import contextlib

import datetime

import time

class cm\_timer\_1:

    def \_\_enter\_\_(self):

        self.start = datetime.datetime.now()

        return self

    def \_\_exit\_\_(self, \*args):

        self.finish = datetime.datetime.now()

        print("time: ", self.finish - self.start)

@contextlib.contextmanager

def cm\_timer\_2():

    start = datetime.datetime.now()

    yield

    finish = datetime.datetime.now()

    print("time: ", finish - start)

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

    with cm\_timer\_1():

        time.sleep(5.5)

    with cm\_timer\_2():

        time.sleep(5.5)

sort.py:

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

if \_\_name\_\_ == '\_\_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)

unique.py:

# Итератор для удаления дубликатов

class Unique(object):

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

        ignore\_case = kwargs.get('ignore\_case', False)

        self.items = []

        self.index = 0

        mod\_elems = []

        for item in items:

            mod\_elem = None

            if ignore\_case:

                mod\_elem = item.lower() if isinstance(item, str) else item

                if mod\_elem in mod\_elems:

                    continue

                mod\_elems.append(mod\_elem)

                self.items.append(item)

            else:

                if item in self.items:

                    continue

                self.items.append(item)

    def \_\_next\_\_(self):

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

            item = self.items[self.index]

            self.index += 1

            return item

        else:

            self.index = 0

            raise StopIteration

    def \_\_iter\_\_(self):

        return self

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

    for i in Unique([1, 2, 3, 1, 2, 3]):

        print(i, end=" ")

    print()

    for i in Unique(["a", "A", "B", "b"], ignore\_case=True):

        print(i, end=" ")

    print()

process\_data.py:

import json

import sys

from lab\_python\_fp.cm\_timer import cm\_timer\_1

from lab\_python\_fp.print\_result import print\_result

from lab\_python\_fp.field import field

from lab\_python\_fp.gen\_random import gen\_random

from lab\_python\_fp.unique import Unique

import os

path = "lab3\\lab\_python\_fp\\data\_light.json"

with open(path, encoding='utf-8') as f:

    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`

# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку

# В реализации функции f4 может быть до 3 строк

@print\_result

def f1(arg):

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

@print\_result

def f2(arg):

    return list(filter(lambda x: x.lower().startswith('программист'), arg))

@print\_result

def f3(arg):

    return list(map(lambda x: x + ' с опытом Python', arg))

@print\_result

def f4(arg):

    return [f'{i[0]}, зарплата {i[1]}' for i in list(zip(arg, gen\_random(len(arg), 100000, 200000)))]

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

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

# Результаты выполнения программы















