**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «ПиКЯП (Парадигмы и конструкции языков программирования)»

Отчет по лабораторной работе №3-4

«**Функциональные возможности языка Python**»

*Засчитывается за 2 выполненных лабораторных работы*

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| Студент группы ИУ5-31Б |  | Преподаватель каф. ИУ5 |
| Нагдасёв Д.М. |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2024 г

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

**Код программы:**

1. Файл **field.py**

def field(items, \*args):

    assert len(args) > 0, "Необходимо передать хотя бы одно поле."

    for item in items:

        if len(args) == 1:

            value = item.get(args[0])

            if value:

                yield value

        else:

            result = {}

            for key in args:

                value = item.get(key)

                if value:

                    result[key] = value

            if result:

                yield result

goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},

    {'title': 'Диван для отдыха', 'color': 'black'}]

print(list(field(goods, 'title')))

print(list(field(goods, 'title', 'price')))

print(list(field(goods, 'title', 'price', 'color')))

2. Файл **gen\_random.py**

from random import randint

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

    for \_ in range(num\_count):

        yield randint(begin, end)

print(list(gen\_random(5, 1, 10)))

3. Файл **unique.py**

import random

class Unique(object):

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

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

        self.seen = set()

        self.items = iter(items)

    def \_\_next\_\_(self):

        for item in self.items:

            comparison\_item = item.lower() if self.ignore\_case and isinstance(item, str) else item

            if comparison\_item not in self.seen:

                self.seen.add(comparison\_item)

                return item

        raise StopIteration

    def \_\_iter\_\_(self):

        return self

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

for unique\_item in Unique(data):

    print(unique\_item, end=" ")

print('\n')

def gen\_random(count, start, end):

    for \_ in range(count):

        yield random.randint(start, end)

data = gen\_random(10, 1, 3)

for unique\_item in Unique(data):

    print(unique\_item, end=" ")

print('\n')

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

for unique\_item in Unique(data):

    print(unique\_item, end=" ")

print('\n')

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

    print(unique\_item, end=" ")

4. Файл **sort.py**

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

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

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

    print(sorted(data, key=lambda x: abs(x), reverse=True))

5. Файл **print\_result.py**

def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

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

        func\_name = func.\_\_name\_\_

        print(func\_name)

        if isinstance(result, list):

            for item in result:

                print(item)

        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\_\_':

    test\_1()

    test\_2()

    test\_3()

    test\_4()

6. Файл **cm\_timer.py**

import time

from contextlib import contextmanager

class cm\_timer\_1:

    def \_\_enter\_\_(self):

        self.start\_time = time.time()

        return self

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

        end\_time = time.time()

        elapsed\_time = end\_time - self.start\_time

        print(f"time: {elapsed\_time:.2f}")

with cm\_timer\_1(): # comment for right work №7

    time.sleep(5.2)

@contextmanager

def cm\_timer\_2():

    start\_time = time.time()

    try:

        yield

    finally:

        end\_time = time.time()

        elapsed\_time = end\_time - start\_time

        print(f"time: {elapsed\_time:.2f}")

with cm\_timer\_2():

    time.sleep(5.2)

7. Файл **process\_data.py**

import json

import random

from cm\_timer import cm\_timer\_1

from print\_result import print\_result

path = 'lab2/data\_light.json'

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

    data = json.load(f)

@print\_result

def f1(arg):

    return sorted(set(job['job-name'].strip() for job in arg), key=str.casefold)

@print\_result

def f2(arg):

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

@print\_result

def f3(arg):

    return list(map(lambda job: f"{job} с опытом Python", arg))

@print\_result

def f4(arg):

    salaries = [random.randint(100000, 200000) for \_ in arg]

    return [f"{job}, зарплата {salary} руб." for job, salary in zip(arg, salaries)]

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

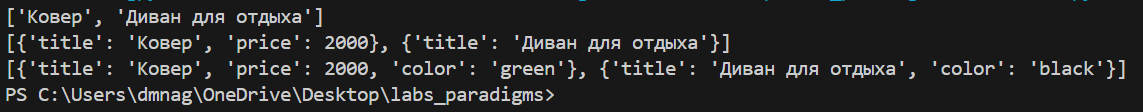
    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

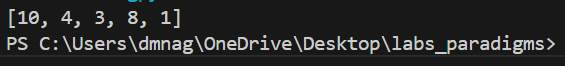
Файл **data\_light.json** выдаётся в условии задачи.

**Результаты выполнения программы**

1. Файл **field.py**



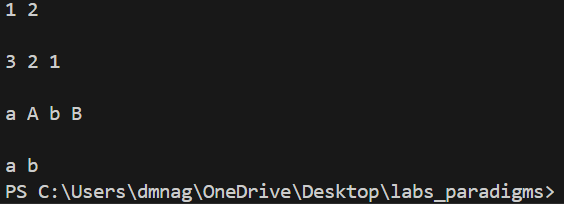
2. Файл **gen\_random.py**



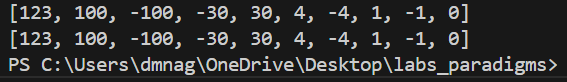




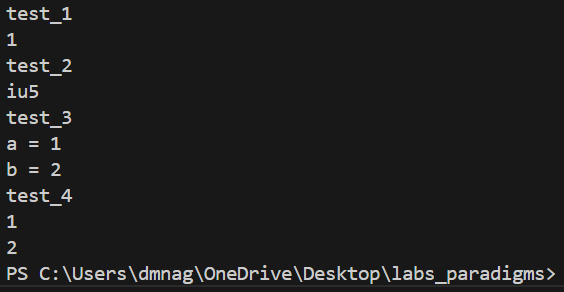
3. Файл **unique.py**



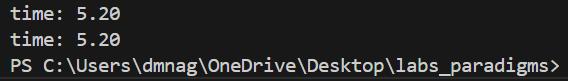
4. Файл **sort.py**



5. Файл **print\_result.py**



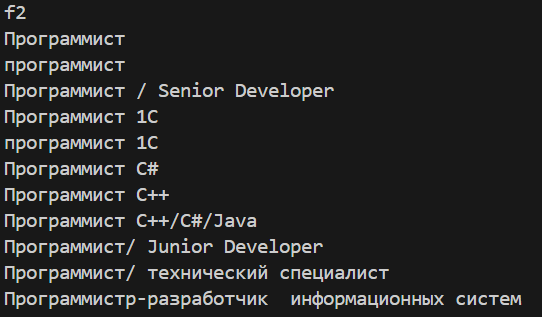
6. Файл **cm\_timer.py**



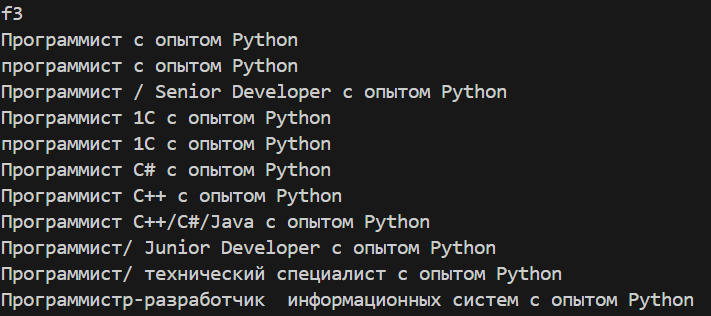
7. Файл **process\_data.py**

Функция **f1()** просто выводит содержимое файла **data\_light.json**.

Функция **f2(f1()):**



Функция **f3(f2(f1())):**



Функция **f4(f3(f2(f1()))):**

