|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Гонов Марат ИУ5-35Б

Парадигмы и конструкции языков программирования

**ОТЧЁТ ПО**

**Лабораторной работе №4-5**

Москва

2023

**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.

* field.py: Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.
* 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.
* unique.py: Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
* 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.
* print\_result.py: Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции.
* 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.
* process\_data.py: 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.

**Текст программы.**

**field.py**

def field(items, \*args):

    assert len(args) > 0

    c\_i = len(items)

    c\_a = len(args)

    for i in range(c\_i):

        for j in range(c\_a):

            if args[j] in items[i] and args[j] is not None:

                yield items[i][args[j]]

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

    goods = [

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

        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}

    ]

    for ii in field(goods, 'title'):

        print(ii)

    print('\n')

    for jj in field(goods, 'title', 'price'):

        print(jj)

**gen\_random.py**

import random

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

    for i in range(num\_count):

        yield random.randint(begin, end)

def main():

    for num in gen\_random(5, 1, 3):

       print(num)

**unique.py**

from gen\_random import gen\_random

class Unique(object):

    def \_\_init\_\_(self, items, ignore\_case=False, \*\*kwargs):

        self.ignore\_case = ignore\_case

        self.items = items

        self.index = 0

        self.unique\_list = []

        self.seen = set()

    def \_\_next\_\_(self):

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

            item = self.items[self.index]

            self.index += 1

            if self.ignore\_case:

                item = item.lower()

            if item not in self.seen:

                self.seen.add(item)

                return item

        raise StopIteration()

    def \_\_iter\_\_(self):

        return self

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

    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

    unique\_data1 = Unique(data1)

    for item in unique\_data1:

        print(item)

    print('\n')

    data2 = []

    for num in gen\_random(5, 1, 3):

        data2.append(num)

    unique\_data2 = Unique(data2)

    for item in unique\_data2:

        print(item)

    print('\n')

    data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    unique\_data3 = Unique(data3, ignore\_case=True)

    for item in unique\_data3:

        print(item)

    print('\n')

    data4 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    unique\_data4 = Unique(data3)

    for item in unique\_data4:

        print(item)

**sort.py**

import math

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

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

    result = sorted(data, key=abs)

    print(result)

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

    result\_with\_lambda = sorted(data, key=lambda x: math.sqrt(x\*\*2))

    print(result\_with\_lambda)

**print\_result.py**

import inspect

def print\_result(func):

    def wrapper(\*args):

        print("Имя функции:", func.\_\_name\_\_)

        result = func(\*args)

        if isinstance(result, list) or inspect.isgenerator(result):

            for el in result:

                print(el)

        elif isinstance(result, dict) :

            for key in result:

                print(key, "=", result[key])

        else:

            print(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 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()

        print("Время выполнения кода:", end\_time - self.start\_time, "секунд")

@contextmanager

def cm\_timer\_2():

    start\_time = time.time()

    yield

    end\_time = time.time()

    print("Время выполнения кода:", end\_time - start\_time, "секунд")

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

    with cm\_timer\_1():

        time.sleep(5.5)

    with cm\_timer\_2():

        time.sleep(5.5)

**process\_data.py**

import json

from field import field

from gen\_random import gen\_random

from unique import Unique

from print\_result import print\_result

from cm\_timer import cm\_timer\_1

path = r"C:\Windows\System32\pikap\_3sem\Lab 4-5\data\_light.json"

with open(path, 'r', encoding='utf-8') as f:  # Specify 'utf-8' encoding here

    data = json.load(f)

@print\_result

def f1(arg):

    return sorted(Unique(list(field(data, 'job-name')), True))

@print\_result

def f2(arg):

    return list(filter(lambda x: 'программист' in x, field(data, 'job-name')))

@print\_result

def f3(arg):

    return list(map(lambda x: x + ' с опытом Python', filter(lambda x: 'программист' in x, field(data, 'job-name'))))

@print\_result

def f4(arg):

    for job in list(filter(lambda x: 'программист' in x, field(data, 'job-name'))):

        salary = list(gen\_random(1, 100000,200000))

        yield job + " " + str(salary[0])

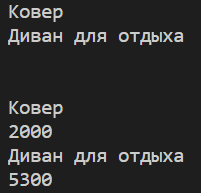
if \_\_name\_\_ == '\_\_main\_\_':

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

Экранные формы:

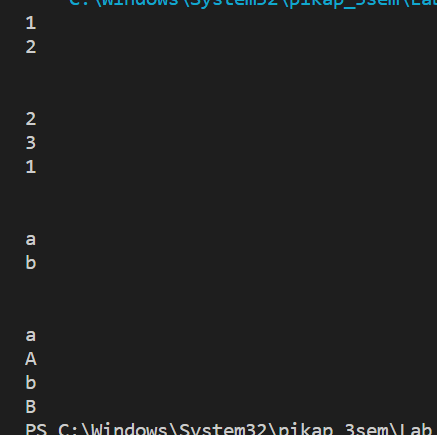
**field.py**



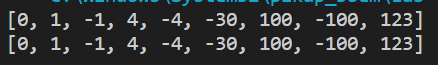
**gen\_random.py**



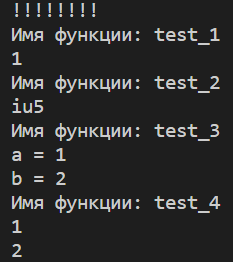
**unique.py**

****

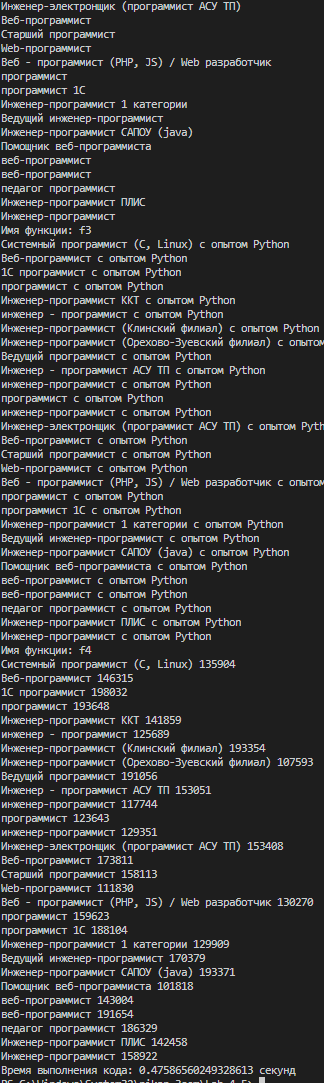
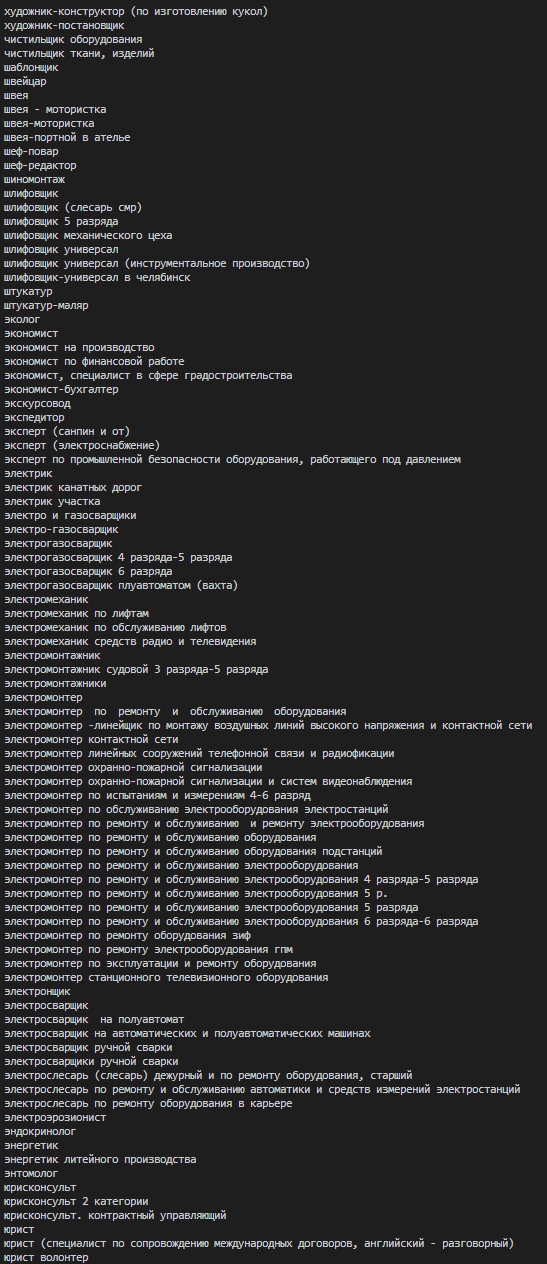
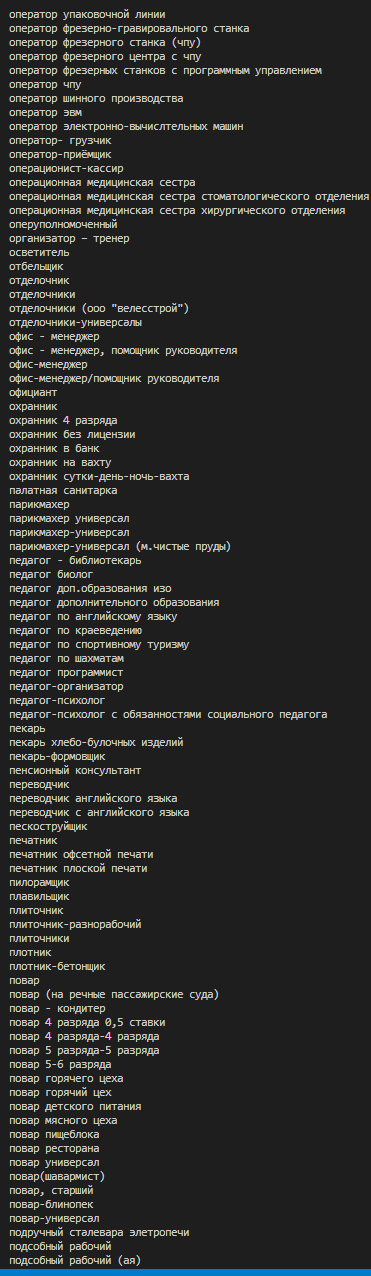
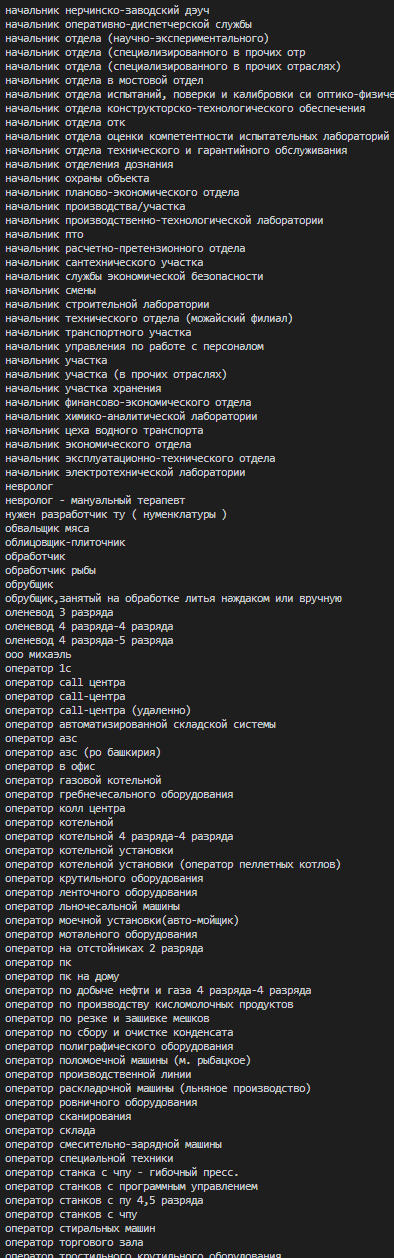
**sort.py**



**print\_result.py**



**cm\_timer.py  
**

**process\_data.py  
**