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»

Лабораторная работа №3-4

Выполнил студент группы ИУ5-35Б

Сулайманов Р. Б.

Москва

2022 г.

Полученное 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.

1) Задача 1 файл “field.py”:

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

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

2) 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. Пример:

gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например *2, 2, 3, 2, 1*

3) 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.

Пример:

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

Unique(data) будет последовательно возвращать только 1 и 2.

data = gen_random(10, 1, 3)

Unique(data) будет последовательно возвращать только 1, 2 и 3.

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

Unique(data) будет последовательно возвращать только a, A, b, B.

Unique(data, ignore_case=True) будет последовательно возвращать только a, b.

4) 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. 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:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

5) 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.

6) Задача 6 файл “cm_timmer.py”:

Необходимо написать контекстные менеджеры *cm_timer_1* и *cm_timer_2*, которые считают время работы блока кода и выводят его на экран. Пример:

with cm_timer_1():

sleep(5.5)

После завершения блока кода в консоль должно вывестись *time: 5.5* (реальное время может несколько отличаться).

cm_timer_1 и *cm_timer_2* реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

7) 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 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.

Текст программы и результаты работы функций

Программа состоит из нескольких файлов: main.py, data_light.json и файлы с реализацией функций.

1) Задача 1 файл “field.py”:

- Код функции:

```
def field(list_, *args):
    assert len(list_) != 0
    answer = []
    if len(args) != 1:
        for dictionary in list_:
            dict_ = {}
            for i in dictionary:
                for arg in args:
                    if i == arg:
                        dict_[arg] = dictionary[i]
            answer.append(dict_)
    else:
        for dictionary in list_:
            for i in dictionary:
                for arg in args:
                    if i == arg:
                        answer.append(dictionary[i])
    return answer
```

- Запуск функции:

```
#field
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]
print(field(goods, 'title', 'price'))
```

- Результат выполнения:

[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]

2) Задача 2 файл “gen_random.py”:

- Код функции:

```python
from random import randrange


def gen_random(num_count, begin, end):
    return [randrange(begin, end + 1) for _ in range(num_count)]
```

- Запуск функции:

```python
list_ = gen_random(5, -15, 15)
print(*list_)
```

- Результат выполнения:

-9 15 -6 -5 14

3) Задача 3 файл “unique.py”:

- Код функции:

```python
class Unique(object):
    def __init__(self, items, **kwargs):
        self.position = 0
        self.items = [str(i) for i in items]
        self.element = items[self.position]
        self.output = []
        try:
            self.ignore_case = kwargs["ignore_case"]
            if self.ignore_case:
                self.elements = [self.element.lower(), self.element.upper()]
            else:
                self.elements = [str(self.element)]
        except:
            self.ignore_case = False
            self.elements = [str(self.element)]
        self.output.append(self.element)

    def __next__(self):
        self.position += 1
        for i in range(self.position, len(self.items)):
            self.position = i
            if self.items[self.position] not in self.elements:
                if self.ignore_case:
                    self.element = self.items[self.position]
                    self.elements.extend([self.element.lower(),
self.element.upper()])
                    self.output.append(self.element)
                    return self.element
                else:
                    self.element = self.items[self.position]
                    self.elements.append(self.element)
                    self.output.append(self.element)
                    return self.element

    def __iter__(self):
        return self

    def filling(self):
        element = next(self)
        while element is not None:
            element = next(self)
        return self.output
```

- Запуск функции:

```
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
print(data)
print(*Unique(data).filling(), sep=', ')
```

- Результат выполнения:

a, A, b, B

4) Задача 4 файл “sort.py”:

- Код функций:

```
def sort_array(list_):
    return sorted(list_, key=abs, reverse=True)


def sort_array_lambda(list_):
    return sorted(list_, key=lambda n: abs(n), reverse=True)
```

- Запуск функций:

```
list_ = gen_random(5, -15, 15)
print(*list_)
list1 = sort_array(list_)
list2 = sort_array_lambda(list_)
print(*list1)
print(*list2)
```

- Результат выполнения:

15 14 -13 -10 3

15 14 -13 -10 3

15 14 -13 -10 3

5) Задача 5 файл “print_result.py”:

- Код функции:

```
def print_result(func):
    def wrapper(param):
        print(func.__name__)
        result = func(param)
        print(result)
        return result
    return wrapper
```

- Запуск функции:

```
@print_result
def test_1(num):
    return num

test_1(5)
```

- Результат выполнения:

test_1

5

6) Задача 6 файл “cm_timmer.py”:

- Код функций:

```
from time import perf_counter
from contextlib import contextmanager


class cm_timer_1:

    def __init__(self):
        pass

    def __enter__(self):
        self.begin = perf_counter()

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            print(perf_counter() - self.begin)


@contextmanager
def cm_timer_2():
    time_ = perf_counter()
    yield
    print(perf_counter() - time_)
```

- Запуск функций:

```
with cm_timer_1():
    sleep(5.5)

with cm_timer_2():
    sleep(3.5)
```

- Результат выполнения:

5.500734399975045

3.5004126999992877

7) Задача 7 файл “process_data.py”:

- Код функций:

```
from time import sleep
from lab_python_fp.gen_random import gen_random
from lab_python_fp.sort import sort_array, sort_array_lambda
from lab_python_fp.field import field
```

```python
from lab_python_fp.print_result import print_result
from lab_python_fp.unique import Unique
from lab_python_fp.cm_timer import cm_timer_1, cm_timer_2
from lab_python_fp.print_result import print_result

@print_result
def f1(data):
    return Unique(field(data, "job-name")).filling()

@print_result
def f2(data):
    return list(filter(lambda param: param.startswith("программист"), data))

@print_result
def f3(data):
    return list(map(lambda add: add + " с опытом Python", data))

@print_result
def f4(data):
    res = list(zip(data, gen_random(len(data), 100000, 200000)))
    return [i[0] + ", зарплата " + str(i[-1]) + " руб." for i in res]
```

- Запуск функции:

```python
path = "data_light.json"

with open(path, "r", encoding="utf-8") as f:
    data = json.load(f)

with cm_timer_1():
    f4(f3(f2(f1(data))))
```

- Результат выполнения:

f1

['Администратор на телефоне', 'Медицинская сестра', 'Охранник сутки-день-ночь-вахта', …, аппаратуры бортовых космических систем', 'Инженер-программист', 'Менеджер (в промышленности)']

f2

['программист', 'программист 1С']

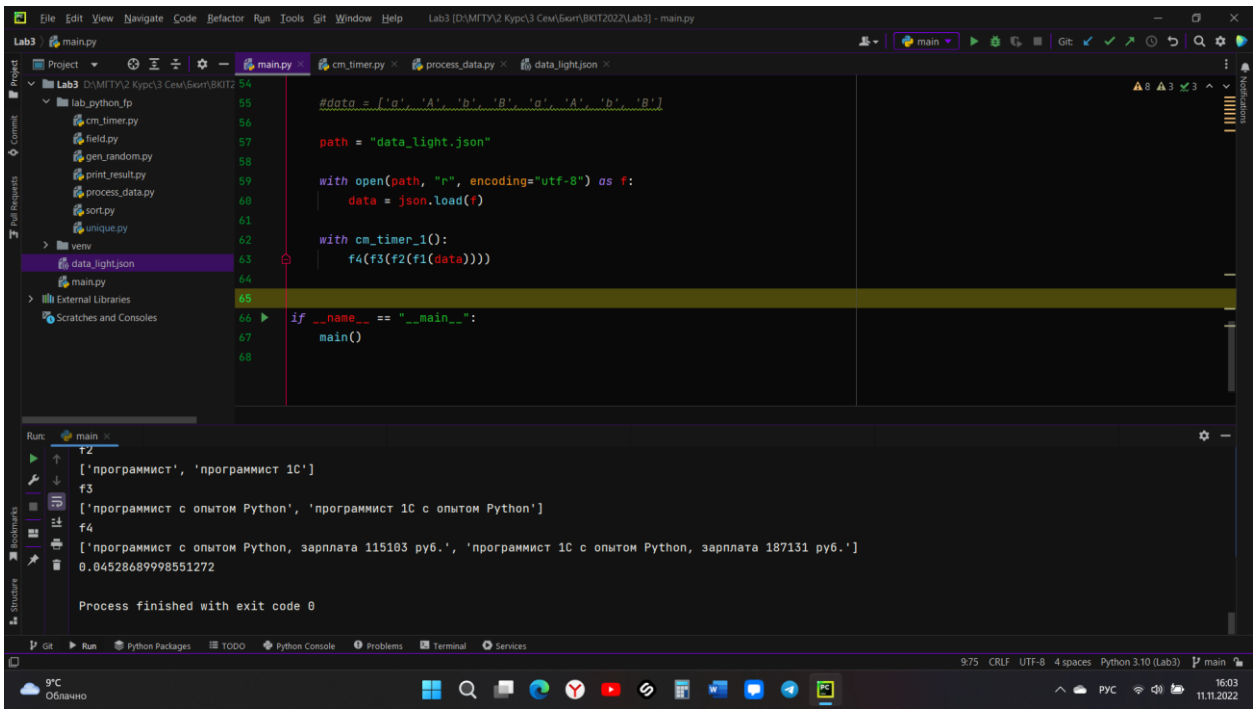
f3

['программист с опытом Python', 'программист 1С с опытом Python']

f4

['программист с опытом Python, зарплата 115103 руб.', 'программист 1С с опытом Python, зарплата 187131 руб.']

0.04528689998551272


File Edit View Navigate Code Refactor Run Tools Git Window Help
Lab3 [D:\МГТУ\2 Курс\3 Сем\Бэкп\ВКІТ2022\Lab3] - main.py
Project
Lab3 D:\МГТУ\2 Курс\3 Сем\Бэкп\ВКІТ2
lab_python_fp
cm_timer.py
field.py
gen_random.py
print_result.py
process_data.py
sort.py
unique.py
venv
data_light.json
main.py
External Libraries
Scratches and Consoles
main.py
cm_timer.py
process_data.py
data_light.json
#data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
path = "data_light.json"
with open(path, "r", encoding="utf-8") as f:
data = json.load(f)
with cm_timer_1():
f4(f3(f2(f1(data))))
if __name__ == "__main__":
main()
Run: main
f2
['программист', 'программист 1С']
f3
['программист с опытом Python', 'программист 1С с опытом Python']
f4
['программист с опытом Python, зарплата 115103 руб.', 'программист 1С с опытом Python, зарплата 187131 руб.']
0.04528689998551272
Process finished with exit code 0
Git Run Python Packages TODO Python Console Problems Terminal Services
9:75 CRLF UTF-8 4 spaces Python 3.10 (Lab3) main
9°C
Облачно
РУС
16:03
11.11.2022