Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе 3-4.

Выполнил:

студент группы ИУ5-33Б

Иванов Николай

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Нардид А.Н.

Подпись и дата:

Москва, 2023 г.

Текст задания

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.

Задание 1

Задача 1 (файл field.py)

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

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

`field(goods, 'title')` должен выдавать `'Ковер', 'Диван для отдыха'`

`field(goods, 'title', 'price')` должен выдавать `{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}`

- 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.

Task1.py

```python
def func():
    goods = [{'title': 'ковер', 'price': 1500, 'color': 'red'},
             {'title': 'диван', 'price': 2000, 'color': 'red'}]
    print(str(list(field(goods, 'title')))[1:-1])
    print(str(list(field(goods, 'title', 'price'))))[1:-1])
```

```python
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for item in items:
            if item[args[0]] is not None:
                yield item[args[0]]
    else:
        for item in items:
            res = {}
            for key in args:
                if key in item and item[key] is not None:
                    res[key] = item[key]
            yield res
```

Результат

```
Task 1


'ковер', 'диван'
{'title': 'ковер', 'price': 1500}, {'title': 'диван', 'price': 2000}
```

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`

task2.py

```python
import random
def func():
    print(str(list(field(4,7,17)))[1:-1])


def field(cnt , min , max):
    ans = []
    for i in range(cnt):
        ans.append(random.randint(min, max))
    yield ans
```

Результат

```
Task 2


[16, 15, 8, 10]
```

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.

task3.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.it = iter(items)
        self.ignore_case = kwargs.get('ignore_case', False)
        self.res = set()
    def __next__(self):
        while True:
            item = next(self.it)
            item_ = item.lower() if self.ignore_case else item
            if item_ not in self.res:
                self.res.add(item_)
                return item_
    def __iter__(self):
        return self

def func():
    data = [1,4,5,7, 'int', 'char*', 1,4,7, 'char*']
    data_ = Unique(data)
    print(list(data_))
```

Результат

```
Task 3


[1, 4, 5, 7, 'int', 'char*']
```

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.

task4.py

```
def func():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    res = sorted(data, key = abs, reverse=True)
    print(res)
    res_ = sorted(data, key = lambda x: abs(x), reverse=True)
    print(res_)
```

Результат

```
Task 4

[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

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

Task5.py

```
def func():
    test1()
    test2()
    test3()
    test4()


def print_result(func):
    def wrapper():
```

```
        res = func()
        if isinstance(res, list):
            for item in res:
                print(item)
        elif isinstance(res, dict):
            for key, value in res.items():
                print(f'{key} = {value}')
        else:
            print(res)
        return res

    return wrapper


@print_result
def test1():
    return 1


@print_result
def test2():
    return 'iu5'


@print_result
def test3():
    return {'a': 1, 'b': 2}


@print_result
def test4():
    return [1, 2]
```

Результат

```
Task 5


1
iu5
a = 1
b = 2
1
2
```

Задача 6 (файл 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. 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).

task6.py

```
import time
from contextlib import contextmanager

def func():
    with timer1():
        time.sleep(5.5)
    with timer2():
        time.sleep(5.5)
class timer1():
    def __enter__(self):
        self.start = time.time()
    def __exit__(self, exc_type, exc_val, exc_tb):
        time_ = time.time() - self.start
        print(f'time = {time_}')

@contextmanager
def timer2():
    start = time.time()
    yield
    time_ = time.time() - start
    print(f'time = {time_}')
```

Результат

```
Task 6

time = 5.503463983535767
time = 5.5012900829315186
```

Задача 7.

Работа с json файлом.

json_test.py

```
import json
from plotly.graph_objs import Scattergeo, Layout
from plotly import offline

def func():
    filename = 'data/eq_data_1_day_m1.geojson'
    with open(filename) as f:
        try:
            all_eq_data = json.load(f)
        except:
            pass
    all_eq_dicts = all_eq_data['features']
    mags, lons, lats, texts = [], [], [], []
    for eq_dict in all_eq_dicts:
```

```
        mag = eq_dict['properties']['mag']
        lon = eq_dict['geometry']['coordinates'][0]
        lat = eq_dict['geometry']['coordinates'][1]
        title = eq_dict['properties']['title']
        mags.append(mag)
        lons.append(lon)
        lats.append(lat)
        texts.append(title)
    print(len(all_eq_dicts))
    print(mags[:10])
    print(lons[:5])
    print(lats[:5])
    data =[{
        'type' : 'scattergeo',
        'lon'  : lons,
        'lat' : lats,
        'text': texts,
        'marker' : {
            'size' : [5*mag for mag in mags],
            'color': mags,
            'colorscale': 'Viridis',
            'reversescale': True,
            'colorbar': {'title': 'Magnitude'},
        },
    }]
    my_layout = Layout(title='Earthquakes')
    fig = {'data' : data , 'layout' : my_layout }
    offline.plot(fig, filename='earthquakes.html')
    readable_file = 'data/readable_eq_data.json'
    with open(readable_file, 'w') as f :
        json.dump(all_eq_data, f, indent=4)
```

Результат