Московский государственный технический университет
им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Отчёт по лабораторной работе №3
по курсу «Разработка интернет-приложений»
Функциональные возможности языка Python

Выполнил:
студент группы ИУ5-51Б
Головацкий А. Д.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Подпись и дата:
24.11.2021

Подпись и дата:
24.11.2021

Москва, 2021 г.

Описание задания.

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.

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.

Текст программы.

```
def field(items, *args):
    assert len(args) > 0, 'at least one argument was expected'
    result = []

    for item in items:
        temp = {}
        for arg in args:
            value = item.get(arg)
            if value is not None:
                temp.update({arg: value})
        if temp:
            if len(args) == 1:
                yield temp[args[0]]
            else:
                yield temp
```

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.

Текст программы.

```
import random


def gen_random(num_count, begin, end):
    return (random.randint(begin, end) for i in range(num_count))
```

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.

Текст программы.

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.elements = {}
        ignore_case = kwargs.get('ignore_case')

        if ignore_case is None:
            ignore_case = False
        elif not isinstance(ignore_case, bool):
            raise Exception(
                "ignore_case parameter is not Boolean")

        for item in items:
            if ignore_case is True and isinstance(item, str):
                item = item.lower()

            self.elements[item] = item

    def __next__(self):
        return next(self.elements)

    def __iter__(self):
        return iter(self.elements)

    def print_unique(self):
        print("[", end="")
        print(*self.elements, sep=", ", end="")
        print("]")
```

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.

Необходимо решить задачу двумя способами:

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

Текст программы.

```
def sort(data):
    return [sorted(data, reverse=True, key=abs), sorted(data, reverse=True,
key=lambda x: -x if (x < 0) else x)]
```

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.

Текст программы.

```
def print_result(func):
    def wrapper(*args):
        print(f'\n\u001b[1;34;4m{func.__name__}\u001b[0m\n')

        result = func(*args)

        if isinstance(result, list):
            for elem in result:
                print(elem)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        elif result:
            print(result)

        return result
    return wrapper
```

Задача 6 (файлы cm_timer1.py и cm_timer2.py)

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

Текст программы.

```
import time


class cm_timer1:
    def __enter__(self):
        self.start_time = time.perf_counter()
        return self.start_time

    def __exit__(self, type, value, traceback):
        elapsed_time = time.perf_counter() - self.start_time
        self.start_time = None

        print(f'elapsed time: {elapsed_time} seconds')
```

```
from contextlib import contextmanager
import time


@contextmanager
def cm_timer2():
    try:
        start_time = time.perf_counter()
        yield start_time

    finally:
        elapsed_time = time.perf_counter() - start_time
        print(f'Elapsed time: {elapsed_time} seconds')
```

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.

```
import json
from lab3.lab_python_fp.cm_timer1 import cm_timer1
from lab3.lab_python_fp.print_result import print_result
from lab3.lab_python_fp.unique import Unique
from lab3.lab_python_fp.gen_random import gen_random

path = r"D:\home\RIP\lab3\data_light.json"

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


@print_result
def f1(data):
    return sorted(Unique((data[x]["job-name"] for x, _ in enumerate(data)),
ignoreCase=True), key=lambda x: x.lower())


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


@print_result
```

```python
def f3(data):
    return list(map(lambda x: x + " с опытом Python", data))


@print_result
def f4(data):
    sal_gen = gen_random(len(data), 100000, 200000)
    return list(zip(data, sal_gen))


if __name__ == '__main__':
    with cm_timer1():
        f4(f3(f2(f1(data))))
```

Файл main.py:

```
from time import sleep

from lab3.lab_python_fp.field import field
from lab3.lab_python_fp.gen_random import gen_random
from lab3.lab_python_fp.unique import Unique
from lab3.lab_python_fp.sort import sort
from lab3.lab_python_fp.print_result import print_result
from lab3.lab_python_fp.cm_timer1 import cm_timer1
from lab3.lab_python_fp.cm_timer2 import cm_timer2


@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]


def main():

    # task 1
    print('-----------------------------------------task 1----------------------
----------------\n')
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    field_gen = field(goods, 'title')
    for i in field_gen:
        print(i)
    field_gen = field(goods, 'title', 'price')
    for i in field_gen:
        print(i)

    # task 2
    print('-----------------------------------------task 2----------------------
----------------\n')
    random_gen = gen_random(5, 1, 3)
    for i in random_gen:
        print(i)

    # task 3
    print('-----------------------------------------task 3----------------------
----------------\n')
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    Unique(data).print_unique()
    data = gen_random(10, 1, 3)
    Unique(data).print_unique()
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    Unique(data).print_unique()
    Unique(data, ignore_case=True).print_unique()

    # task 4
```

```
    print('-----------------------------------------task 4----------------------
----------------------\n')
    data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
    print(sort(data))

    # task 5
    print('-----------------------------------------task 5----------------------
----------------------\n')
    test_1()
    test_2()
    test_3()
    test_4()

    # task 6
    print('-----------------------------------------task 6----------------------
----------------------\n')
    with cm_timer1():
        sleep(2)
    with cm_timer2():
        sleep(2)

    # task 7
    print('-----------------------------------------task 7----------------------
----------------------\n')


if __name__ == "__main__":
    main()
```

Экранные формы с примерами выполнения программы.

```
C:\Users\Andrew\AppData\Local\Programs\Python\Python38\python.exe D:/home/RIPLabs/lab3/main.py
----------------------------------------task 1----------------------------------------

Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
----------------------------------------task 2----------------------------------------

3
1
2
2
2
----------------------------------------task 3----------------------------------------

[1, 2]
[3, 2, 1]
[a, A, b, B]
[a, b]
----------------------------------------task 4----------------------------------------

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

```
----------------------------------------task 5----------------------------------------


test_1

1

test_2

iu5

test_3

a = 1
b = 2

test_4

1
2
----------------------------------------task 6----------------------------------------

elapsed time: 2.0046294000000002 seconds
Elapsed time: 2.0150366 seconds
```

Task 7:

```
Энергетик литейного производства
энтомолог
Юрисконсульт
юрисконсульт
юрисконсульт 2 категории
Юрисконсульт. Контрактный управляющий
Юрист
юрист
Юрист (специалист по сопровождению международных договоров, английский - разговорный)
Юрист волонтер
Юристконсульт

f2

программист
программист 1С

f3

программист с опытом Python
программист 1С с опытом Python

f4

('программист с опытом Python', 150673)
('программист 1С с опытом Python', 106344)
elapsed time: 0.08219530000000003 seconds
```