Московский Государственный Технический Университет им. Баумана

Отчет по лабораторной работе №3-4
«Функциональные возможности языка Python»
По дисциплине «Базовые компоненты интернет технологий»

Выполнила:
Студентка группы ИУ5-32Б
Алцыбеева Маргарита
20.11.2022

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Ю. Е.

Москва, 2022 г.

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.

Задача 1 (файл field.py)

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

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.

Задача 3 (файл unique.py)

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

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.

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

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.

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:

Файл field.py

```
def field(items, *args):
    assert len(args) > 0
    ans = []
    buff ={}
    for item in items:
        for key in args:
            if item.get(key) is not None:
                buff[key] = item.get(key)
                if len(args) == 1:
                    ans.append(buff[key])
        if len(args) != 1:
            ans.append(buff.copy())
            buff.clear()
    return ans


def main():
    goods = [
        {'title': 'Ковёр', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
        {'title': 'Картина', 'price': 1000},
        {'gg': '7'}
    ]

    print(field(goods, 'title'))
    print(type(field(goods, 'title')[1]))


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

Файл get_random.py

```
from random import randint

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield randint(begin, end)

def main():
    print(*(gen_random(5,1,6)), end=' ')

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

Файл unique.py

```
from get_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        self.used_elements = set()
        self.items = list(items)
        self.index = 0

    def __next__(self):
        while True:
            if self.index >= len(self.items):
                raise StopIteration
```

```
            else:
                current = str(self.items[self.index])
                self.index = self.index + 1
                if self.ignore_case:
                    if current.lower() not in self.used_elements:
                        self.used_elements.add(current.lower())
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current

    def __iter__(self):
        return self


def main():
    data = [1, 1, 1, 1, 2, 2, 5, 5, 5, 5, 5]
    for i in Unique(data):
        print(i, end=' ')
    print()

    data = ['a', 'A', 'a', 'A', 'b', 'B', 'b', 'B', 'c', 'c', 'c', 'C', 'C']
    for i in Unique(data):
        print(i, end=' ')
    print()

    data = ['a', 'A', 'a', 'A', 'b', 'B', 'b', 'B', 'c', 'c', 'c', 'C', 'C']
    for i in Unique(data, ignore_case=True):
        print(i, end=' ')
    print()

    data = gen_random(10, 1, 4)
    for i in Unique(data):
        print(i, end=' ')
    print()

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

Файл sort.py

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

if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda i: abs(i), reverse=True)
    print(result_with_lambda)
```

Файл print_result.py

```
def print_result(func):
    def decorated(*args, **kwargs):
        print(func.__name__)
        result = func(*args, **kwargs)
        if type(result) == list:
            for i in result:
                print(i)
        elif type(result) == dict:
            for key in result.keys():
                print(key, ' = ', result[key])
        else:
```

```
            print(result)
        return result
    return decorated


@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():
    test_1()
    test_2()
    test_3()
    test_4()

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

Файл cm_timer.py

```
from  time import sleep, perf_counter
from contextlib import contextmanager

class cm_timer1():
    def __init__(self):
        self.time_start = 0
        self.time_end = 0

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

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.time_end = perf_counter()
        print('time: {}'.format(self.time_end - self.time_start))

@contextmanager
def cm_timer2():
    time_start = perf_counter()
    yield
    time_end = perf_counter()
    print('time: {}'.format(time_end - time_start))

def main():
    with cm_timer1():
        sleep(2.5)
    with cm_timer2():
        sleep(2.5)
```

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

Файл procces_data.py

```
import json
import sys
from lab_python_fp.unique import Unique
from lab_python_fp.cm_timer import cm_timer1
from lab_python_fp.get_random import gen_random
from lab_python_fp.field import field
from lab_python_fp.print_result import print_result

path = 'D:/lab3_4/data_light.json'

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


@print_result
def f1(arg):
    return sorted(Unique(field(arg, 'job-name'), ignore_case=True))
    #return sorted(Unique(['программист','программист ссс','программист ббб',
'Программист'], ignore_case=True))
    #return field(arg, 'job-name')

@print_result
def f2(arg):
    return list(filter(lambda i: i[:11].lower() == 'программист', arg))

@print_result
def f3(arg):
    return list(map(lambda j: j + ' с опытом Python', arg))

@print_result
def f4(arg):
    return list(zip(arg, gen_random(len(arg), 100000, 200000)))


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

Примеры работы программы (без f1):

f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик информационных систем
f3
Программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1С с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программистр-разработчик информационных систем с опытом Python
f4
('Программист с опытом Python', 189258)
('Программист / Senior Developer с опытом Python', 114104)
('Программист 1С с опытом Python', 111518)
('Программист C# с опытом Python', 138715)
('Программист C++ с опытом Python', 184908)
('Программист C++/C#/Java с опытом Python', 156518)
('Программист/ Junior Developer с опытом Python', 168413)
('Программист/ технический специалист с опытом Python', 190990)
('Программистр-разработчик информационных систем с опытом Python', 178851)
time: 0.04344530000000002