Отчет по лабораторной работе № 3 по курсу

"Разработка Интернет-Приложений"

Выполнила:

Студентка группы

ИУ5-55Б

Зубарева А. М.

Москва, МГТУ – 2021

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

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

- Задание 2
  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.
- Задание 3

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.

- Задание 4

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.

- Задание 5

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.

- Задание 6

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

- Задание 7

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.

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:

```
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, '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', 'price': 5300}

def field(items, *args):
    assert len(args) > 0
```

```
    if len(args) == 1:
        for item in items:
            if args[0] in item.keys():
                print(item[args[0]])
    else:
        for item in items:
            res = dict()
            for key in args:
                if key in item.keys():
                    res[key] = item[key]
            print(res)


goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

field(goods, 'title', 'price')
field(goods, 'title')
```

Файл gen_random.py:

```
import random


def gen_random(num_count, begin, end):
    for item in range(num_count):
        yield random.randint(begin, end)


if __name__ == '__main__':
    for i in gen_random(5, 1, 3):
        print(i)
```

Файл unique.py :

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-
параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми
строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна
из которых удалится
        # По-умолчанию ignore_case = False
        self.used_elements = set()
        self.data = items
        self.index = 0

        if 'ignore_case' not in kwargs:
            self.ignore_case = False
        else:
            self.ignore_case = kwargs['ignore_case']

    def __iter__(self):
        return self
```

```
    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[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


data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
for val in Unique(data):
    print(val)
print('======================================')
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for val in Unique(data, ignore_case=True):
    print(val)
print('======================================')
for val in Unique(data):
    print(val)
```

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:

```
# Здесь должна быть реализация декоратора
def print_result(func):
    def wrapper(*args):

        out = func(*args)
        print(func.__name__)
        if isinstance(out, list):
            for val in out:
                print(val)
            return out
        elif isinstance(out, dict):
            for key, val in out.items():
                print('{} = {}'.format(key, val))
            return out
        else:
```

```
            print(out)
            return out

    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()
```

Файл cm_timer.py:

```
import time
from contextlib import contextmanager


class Cm_timer_1:

    def __init__(self):
        self.start_time = None
        self.end_time = None

    def __enter__(self):
        self.start_time = time.time()

        # Должен возвращаться значимый объект
        # например, открытый файл

    def __exit__(self, exp_type, exp_value, traceback):
        self.end_time = time.time()
        print('time: {}'.format(self.end_time - self.start_time))


@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    print('time: {}'.format(end_time - start_time))
```

```
if __name__ == '__main__':
    with Cm_timer_1():
        time.sleep(1.0)

    with cm_timer_2():
        time.sleep(1.0)
```

Файл process_data.py (в каталоге lab_python_fp):

```
imp import json
import sys
import cm_timer
from print_result import print_result
from gen_random import gen_random
# Сделаем другие необходимые импорты

path = '.\data_light.json'

# Необходимо в переменную path сохранить путь к файлу, который был передан
при запуске сценария

with open(path, encoding='utf8') as f:
    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise
NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк


@print_result
def f1(arg):
    return sorted(set([val.lower() for val in arg]), key=str.lower)


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


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


@print_result
def f4(arg):
    temp = list(zip(arg, [(', зарплата '+str(el) + ' руб.') for el in
list(gen_random(len(arg), 100000, 200000))]))
    return [(el[0]+el[1]) for el in temp]


if __name__ == '__main__':
    with cm_timer.Cm_timer_1():
        f4(f3(f2(f1([el['job-name'] for el in data]))))
```

Экранные формы с примерами выполнения программы:

```
C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python field.py
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
Ковер
Диван для отдыха
```

Командная строка

```
C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python unique.py
1
2
======================================
a
b
======================================
a
A
b
B

C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

```
C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python cm_timer.py
time: 1.0046381950378418
time: 1.0037727355957031
```

```
C:\BMSTU\5semestr\РИП\ui5web-fall-2021-tasks\lab3\lab_python_fp>python process_data.py
f1
1с программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
asic специалист
javascript разработчик
rtl специалист
web-программист
web-разработчик
автожестянщик
автоинструктор
автомаляр
автомойщик
автор студенческих работ по различным дисциплинам
автослесарь
автослесарь - моторист
автоэлектрик
агент
агент банка
агент нпф
test_3
```

```
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, зарплата 136827 руб.
программист / senior developer с опытом Python, зарплата 138264 руб.
программист 1с с опытом Python, зарплата 196310 руб.
программист c# с опытом Python, зарплата 109128 руб.
программист c++ с опытом Python, зарплата 120511 руб.
программист c++/c#/java с опытом Python, зарплата 152780 руб.
программист/ junior developer с опытом Python, зарплата 197435 руб.
программист/ технический специалист с опытом Python, зарплата 126459 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 130429 руб.
time: 0.9140422344207764
```