

Отчет по лабораторной работе № 3 по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Хижняков Вадим Максимович

Москва, МГТУ – 2021

Задание:

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете

lab_python_fr. Решение каждой задачи должно располагаться 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 выводиться 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 С# 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 С# 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
    if len(args) == 1:
        for item in items:
            if args[0] in item.keys():
                yield item[args[0]]
    else:
        for item in items:
            res = dict()
            for key in args:
                if key in item.keys():
                    res[key] = item[key]
            yield res

if __name__ == '__main__':
    goods = [
        {'title': 'Ковёр', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]

    for val in field(goods, 'title', 'price'):
        print(val)
```

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
# 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

class Unique(object):
    def __init__(self, items, **kwargs):
        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 += 1
                cased = current.lower() if self.ignore_case else current
                if cased not in self.used_elements:
                    self.used_elements.add(cased)
                    return current

if __name__ == '__main__':
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    print('===== IGNORE CASE')
    print('=====')
    for val in Unique(data, ignore_case=True):
        print(val)

    print('===== DONT IGNORE CASE')
    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:

```
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 CmTimer:  
  
    def __init__(self):  
        self.start_time = None  
        self.end_time = None  
  
    def __enter__(self):  
        self.start_time = time.time()  
  
    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():  
    start_time = time.time()  
    yield  
    end_time = time.time()  
    print('time: {}'.format(end_time - start_time))  
  
if __name__ == '__main__':  
    with CmTimer():  
        time.sleep(1.0)  
  
    with cm_timer():  
        time.sleep(1.0)
```

Файл process_data.py (в каталоге lab_python_fp):

```
import json  
import os  
from pathlib import Path  
from cm_timer import CmTimer  
from print_result import print_result  
from gen_random import gen_random  
  
# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске  
# сценария  
# 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

path = Path(os.getcwd(), 'lab3', 'mocks', 'data_light.json')

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

@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 CmTimer():
        f4(f3(f2(f1([el['job-name'] for el in data]))))

```

Пример выполнения программы:

>  Run echo "~~~~~ print_result ~~~~~"

✓  Run python ./lab3/print_result.py

```
1 ▶ Run python ./lab3/print_result.py
7 =====
8 test_1
9 1
10 test_2
11 iu5
12 test_3
13 a = 1
14 b = 2
15 test_4
16 1
17 2
```

>  Run echo "~~~~~ sort ~~~~~"

✓  Run python ./lab3/sort.py

```
1 ▶ Run python ./lab3/sort.py
7 [123, 100, -100, -30, 4, -4, 1, -1, 0]
8 [123, 100, -100, -30, 4, -4, 1, -1, 0]
```

>  Run echo "~~~~~ unique ~~~~~"

✓  Run python ./lab3/unique.py

```
1 ▶ Run python ./lab3/unique.py
7 ===== IGNORE CASE =====
8 a
9 b
10 ===== DONT IGNORE CASE =====
11 a
12 A
13 b
14 B
```

✓ ✓ Run python ./lab3/cm_timer.py

```
1 ▶ Run python ./lab3/cm_timer.py
7 time: 1.0011119842529297
8 time: 1.0003855228424072
```

> ✓ Run echo "~~~~~ field ~~~~~"

✓ ✓ Run python ./lab3/field.py

```
1 ▶ Run python ./lab3/field.py
7 {'title': 'Ковер', 'price': 2000}
8 {'title': 'Диван для отдыха', 'price': 5300}
```

> ✓ Run echo "~~~~~ gen_random ~~~~~"

✓ ✓ Run python ./lab3/gen_random.py


```
1 ▶ Run python ./lab3/gen_random.py
7 1
8 3
9 2
10 2
11 3
```

Run python ./lab3/process_data.py

```
1 ▶ Run python ./lab3/process_data.py
7 f1
8 1с программист
9 2-ой механик
10 3-ий механик
11 4-ый механик
12 4-ый электромеханик
13 [химик-эксперт
14 asic специалист
15 javascript разработчик
16 rtl специалист
17 web-программист
18 web-разработчик
19 автожестящик
20 автоинструктор
21 автомаляр
22 автомойщик
23 автор студенческих работ по различным дисциплинам
24 автослесарь
25 автослесарь - моторист
26 автоэлектрик
27 агент
28 агент банка
29 агент нпф
30 агент по гос. закупкам недвижимости
31 агент по недвижимости
32 агент по недвижимости (стажер)
33 агент по недвижимости / риэлтор
34 агент по привлечению юридических лиц
35 агент по продажам (интернет, тв, телефония) в пао ростелеком в населенных пунктах амурской области: г. благовещенск, г. белогорск, г. свободный, г. шимановск, г. зей, г. тында
36 агент торговый
37 агрегатчик-топливник komatsu
38 агроном
39 агроном по защите растений
40 агроном-полевод
41 агрохимик почвовед
42 администратор
43 администратор (удаленно)
```

```
43 администратор (удаленно)
44 администратор active directory
45 администратор в парикмахерский салон
46 администратор зала (предприятий общественного питания)
47 администратор кофейни
48 администратор на ресепшен
49 администратор на телефоне
50 администратор по информационной безопасности
51 администратор ресторана
52 администратор сайта
53 администратор ярмарок выходного дня
54 администратор-кассир
55 аккомпаниатор на 0,5 ст.
56 аккумуляторщик 4 разряда
57 акушерка
58 акушерка в родильное отделение
59 акушерка женской консультации
60 акушерка лысогогорская врачебная амбулатория
61 акушерка фап
62 акушерка, ао
63 акушерка, вп
64 альпинист промышленный
65 аналитик
66 анестезиолог - реаниматолог
67 анестезиолог-реаниматолог
68 анестезиолог-реаниматолог детский
69 аниматор
70 антенщик-мачтовик 4 разряда
71 аппаратчик обработки зерна
72 аппаратчик обработки зерна 5 разряда
73 аппаратчик пастеризации
74 аппаратчик установки опытного производства
75 аппаратчик химводочистки
76 арматурщик
77 арматурщик кузовного цеха
78 арматурщики
79 артист (кукловод) театра кукол
80 артист оркестра
81 артист отдела социально - культурной деятельности районного цнц
```

```
2283 программист-разработчик информационных систем
2284 f3
2285 программист с опытом Python
2286 программист / senior developer с опытом Python
2287 программист 1с с опытом Python
2288 программист с# с опытом Python
2289 программист с++ с опытом Python
2290 программист с++/с#/java с опытом Python
2291 программист/ junior developer с опытом Python
2292 программист/ технический специалист с опытом Python
2293 программист-разработчик информационных систем с опытом Python
2294 f4
2295 программист с опытом Python, зарплата 117687 руб.
2296 программист / senior developer с опытом Python, зарплата 175163 руб.
2297 программист 1с с опытом Python, зарплата 197386 руб.
2298 программист с# с опытом Python, зарплата 150456 руб.
2299 программист с++ с опытом Python, зарплата 191515 руб.
2300 программист с++/с#/java с опытом Python, зарплата 139508 руб.
2301 программист/ junior developer с опытом Python, зарплата 190032 руб.
2302 программист/ технический специалист с опытом Python, зарплата 144911 руб.
2303 программист-разработчик информационных систем с опытом Python, зарплата 148533 руб.
2304 time: 0.013310670852661133
```

>  Post Run actions/checkout@v2