Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка интернет приложений»

Отчет по лабораторной работе №2

«Python. Функциональные возможности»

Выполнил:
студент группы ИУ5-52
Заровная Н.А.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2017 г.

1 Задание

1. (ex_1.py) Необходимо реализовать генераторы field и gen_random. Генератор field последовательно выдает значения ключей словарей массива.

1) В качестве первого аргумента генератор принимает list, дальше через *args генератор принимает неограниченное кол-во аргументов.
2) Если передан один аргумент, генератор последовательно выдает только значения полей, если поле равно None, то элемент пропускается
3) Если передано несколько аргументов, то последовательно выдаются словари, если поле равно None, то оно пропускается, если все поля None, то пропускается целиком весь элемент.

Генератор gen_random последовательно выдает заданное количество случайных чисел в заданном диапазоне.

2. (ex_2.py) Необходимо реализовать итератор, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты. Конструктор итератора также принимает на вход именной bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False. Итератор не должен модифицировать возвращаемые значения.

3. (ex_3.py) Дан массив с положительными и отрицательными числами. Необходимо одной строкой вывести на экран массив, отсортированный по модулю. Сортировку осуществлять с помощью функции sorted.

4. (ex_4.py) Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции. Файл ex_4.py не нужно изменять. Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции, печатать результат и возвращать значение. Если функция вернула список (list), то значения должны выводиться в столбик. Если функция вернула словарь (dict), то ключи и значения должны выводить в столбик через знак равно.

5. (ex_5.py) Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран.

6. Мы написали все инструменты для работы с данными. Применим их на реальном примере, который мог возникнуть в жизни. В репозитории находится файл data_light.json . Он содержит облегченный список вакансий в России в формате json. Структура данных представляет собой массив словарей с множеством полей: название работы, место, уровень зарплаты и т.д. В ex_6.py дано 4 функции. В конце каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер timer выводит время работы цепочки функций. Задача реализовать все 4 функции по заданию, ничего не изменяя в файле-шаблоне. Функции f1-f3 должны быть реализованы в 1 строку, функция f4 может состоять максимум из 3 строк.
Что функции должны делать:
1. Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих заданий.
2. Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Иными словами нужно получить все специальности, связанные с программированием. Для фильтрации используйте функцию filter.

3. 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.
4. 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.

2 Листинг

gens.py

```
import random


# Генератор вычленения полей из массива словарей
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for i in items:
            for key in args:
                a = i.get(key)
                if a is not None:
                    yield a
    else:
        for i in items:
            dict = {}
            for key in args:
                a = i.get(key)
                if a is not None:
                    dict[key] = a
            if len(dict) > 0:
                yield dict


# Генератор списка случайных чисел
def gen_random(begin, end, num_count):
    for i in range(num_count):
        yield random.randint(begin, end)
```

ex_1.py

```
from librip.gens import field, gen_random

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
]

# Реализация задания 1
print(' '.join(map(str, field(goods, 'title'))))
print(' '.join(map(str, field(goods, 'title', 'price'))))
print(', '.join(map(str, gen_random(1, 3, 5))))
```

iterators.py

```
# Итератор для удаления дубликатов
class Unique(object):
```

```
    def __init__(self, items, **kwargs):
        # По-умолчанию ignore case = False
        self.ignore_case = kwargs.get('ignore_case', 'False')
        if isinstance(items, list):
            self.items = (x for x in items)
        else:
            self.items = items
        self._s = set()

def __next__(self):
    for a in self.items:
        if self.ignore_case and isinstance(a, str):
            if a.lower() not in map(lambda s: s.lower(), self._s):
                self._s.add(a)
                return a
        elif a not in self._s:
            self._s.add(a)
            return a
    else:
        raise StopIteration

def __iter__(self):
    return self
```

ex_2.py

```
from librip.gens import gen_random
from librip.iterators import Unique

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
data2 = gen_random(1, 3, 10)
data3 = ['a', 'A', 'b', 'B']

# Реализация задания 2
print('list: ', ' '.join(map(str, Unique(data1))))
print('random: ', ' '.join(map(str, Unique(data2))))
print('list ignore_case=False: ', ' '.join(map(str, Unique(data3))))
print('list ignore_case=True: ', ' '.join(map(str, Unique(data3,
ignore_case='True'))))
```

ex_3.py

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
print(sorted(data, key=lambda x: abs(x)))
```

decorators.py

```
# декоратор, который принимает на вход функцию
def print_result(func):
    def wrapper(*args):
        print(func.__name__)
        a = func(*args)
        if isinstance(a, list):
            print('\n'.join(map(str, a)))
        elif isinstance(a, dict):
            for key, value in a.items():
                print(key, '=', value)
        else:
            print(a)
        return a
    return wrapper
```

ex_4.py

```
from librip.decorators import print_result


@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu'


@print_result
def test_3():
    return {'a': 1, 'b': 2}


@print_result
def test_4():
    return [1, 2]


test_1()
test_2()
test_3()
test_4()
```

ctxnmgrs.py

```
from datetime import datetime


class timer():
    def __enter__(self):
        self.now = datetime.now()

    def __exit__(self, exp_type, exp_value, traceback):
        print(datetime.now() - self.now)
```

ex_5.py

```
from time import sleep
from librip.ctxmngrs import timer

with timer():
    sleep(5.5)
```

ex_6.py

```
import json
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gens import field, gen_random
from librip.iterators import Unique as unique


path = sys.argv[1]
with open(path) as f:
    data = json.load(f)
```

```
@print_result
def f1(arg):
    return list(sorted(unique(field(arg, 'job-name'), ignore_case='True')))


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

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


@print_result
def f4(arg):
    g = gen_random(100000, 200000, len(arg))
    return list(map(lambda y: y[0] + y[1], list(zip(arg, list(map(lambda x:
', зарплата ' + x + ' руб.', map(str, g)))))))


with timer():
   f4(f3(f2(f1(data))))
```

Результат

ex_1.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_1.py
Ковер Диван для отдыха Стелаж Вешалка для одежды
{'title': 'Ковер', 'price': 2000} {'title': 'Диван для отдыха', 'price': 5300} {'title': 'Стелаж', 'price': 7000} {'title': 'Вешалка для одежды', 'price': 800}
2, 1, 2, 2, 1
```

ex_2.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_2.py
list:  1 2
random:  3 2 1
list ignore_case=False:  aAA Aaa bBB Bbb
list ignore_case=True:  aAA bBB
```

ex_3.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]
```

ex_4.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_4.py
test_1
1
test_2
iu
test_3
a = 1
b = 2
test_4
1
2
```

ex_5.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_5.py
0:00:05.500887
```

ex_6.py

```
C:\Python37\python.exe C:/PyCharmProgects/lab4/ex_6.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
[химик-эксперт
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор

электромонтер станционного телевизионного оборудования
электросварщик
энтомолог
юрисконсульт 2 категории
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
```

```
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 149930 руб.
Программист / Senior Developer с опытом Python, зарплата 155150 руб.
Программист 1С с опытом Python, зарплата 171116 руб.
Программист C# с опытом Python, зарплата 173754 руб.
Программист C++ с опытом Python, зарплата 182333 руб.
Программист C++/C#/Java с опытом Python, зарплата 110009 руб.
Программист/ Junior Developer с опытом Python, зарплата 117011 руб.
Программист/ технический специалист с опытом Python, зарплата 109186 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 147738 руб.
0:00:11.156959
```