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 == 'True':
                if isinstance(a, str):
                    a = a.lower()
            if 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

```
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
[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
f2
1с программист
web-программист
веб - программист (php, js) / web разработчик
веб-программист
ведущий инженер-программист
ведущий программист
инженер - программист
инженер - программист асу тп
инженер-программист
инженер-программист (клинский филиал)
инженер-программист (орехово-зуевский филиал)
инженер-программист 1 категории
инженер-программист ккт
инженер-программист плис
инженер-программист сапоу (java)
инженер-электронщик (программист асу тп)
педагог программист
помощник веб-программиста
программист
программист / senior developer
программист 1с
программист c#
программист c++
программист c++/c#/java
программист/ junior developer
программист/ технический специалист
программистр-разработчик  информационных систем
системный программист (c, linux)
старший программист
```

```
f3
1с программист с опытом Python
web-программист с опытом Python
веб - программист (php, js) / web разработчик с опытом Python
веб-программист с опытом Python
ведущий инженер-программист с опытом Python
ведущий программист с опытом Python
инженер - программист с опытом Python
инженер - программист асу тп с опытом Python
инженер-программист с опытом Python
инженер-программист (клинский филиал) с опытом Python
инженер-программист (орехово-зуевский филиал) с опытом Python
инженер-программист 1 категории с опытом Python
инженер-программист ккт с опытом Python
инженер-программист плис с опытом Python
инженер-программист сапоу (java) с опытом Python
инженер-электронщик (программист асу тп) с опытом Python
педагог программист с опытом Python
помощник веб-программиста с опытом Python
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
системный программист (c, linux) с опытом Python
старший программист с опытом Python
```

```
f4
1с программист с опытом Python, зарплата 113349 руб.
web-программист с опытом Python, зарплата 146244 руб.
веб - программист (php, js) / web разработчик с опытом Python, зарплата 184398 руб.
веб-программист с опытом Python, зарплата 153511 руб.
ведущий инженер-программист с опытом Python, зарплата 192321 руб.
ведущий программист с опытом Python, зарплата 180649 руб.
инженер - программист с опытом Python, зарплата 112577 руб.
инженер - программист асу тп с опытом Python, зарплата 135372 руб.
инженер-программист с опытом Python, зарплата 112088 руб.
инженер-программист (клинский филиал) с опытом Python, зарплата 135355 руб.
инженер-программист (орехово-зуевский филиал) с опытом Python, зарплата 180174 руб.
инженер-программист 1 категории с опытом Python, зарплата 119350 руб.
инженер-программист ккт с опытом Python, зарплата 155012 руб.
инженер-программист плис с опытом Python, зарплата 139813 руб.
инженер-программист сапоу (java) с опытом Python, зарплата 167600 руб.
инженер-электронщик (программист асу тп) с опытом Python, зарплата 182234 руб.
педагог программист с опытом Python, зарплата 152147 руб.
помощник веб-программиста с опытом Python, зарплата 159801 руб.
программист с опытом Python, зарплата 183417 руб.
программист / senior developer с опытом Python, зарплата 106118 руб.
программист 1с с опытом Python, зарплата 109977 руб.
программист c# с опытом Python, зарплата 173057 руб.
программист c++ с опытом Python, зарплата 176139 руб.
программист c++/c#/java с опытом Python, зарплата 150050 руб.
программист/ junior developer с опытом Python, зарплата 148357 руб.
программист/ технический специалист с опытом Python, зарплата 102157 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 156784 руб.
системный программист (с  linux) с опытом Python  зарплата 130449 руб.
старший программист с опытом Python, зарплата 171394 руб.
0:00:00.088063

Process finished with exit code 0
```