Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №3

по курсу

Разработка Интернет Приложений

Выполнила:
Костян Алина
ИУ5-53

Проверил:
Гапанюк Ю.Е.

Москва, 2018

Задание

Важно выполнять все задачи последовательно. С 1 по 5 задачу формируется модуль librip, с помощью которого будет выполняться задание 6 на реальных данных из жизни. Весь вывод на экран (даже в столбик) необходимо запрограммировать одной строкой.

Подготовительный этап

1. Зайти на github.com и выполнить fork проекта с заготовленной структурой https://github.com/iu5team/ex-lab4
2. Переименовать репозиторий в lab_3
3. Выполнить git clone проекта из вашего репозитория

Задача 1 (ex_1.py)

Необходимо реализовать генераторы field и gen_random
Генератор field последовательно выдает значения ключей словарей массива
Пример:

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

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
Пример:
gen_random(1, 3, 5)должен выдать 5 чисел от 1 до 3, т.е. примерно 2, 2, 3, 2, 1

В ex_1.py нужно вывести на экран то, что они выдают, с помощью кода в *одну строку*
Генераторы должны располагаться в librip/gen.py

Задача 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.

Пример:
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
Unique(data) будет последовательно возвращать только 1 и 2

data = gen_random(1, 3, 10)
unique(gen_random(1, 3, 10))будет последовательно возвращать только 1, 2 и 3

data = [‘a’, ‘A’, ‘b’, ‘B’]
Unique(data) будет последовательно возвращать только a, A, b, B

data = [‘a’, ‘A’, ‘b’, ‘B’]
Unique(data, ignore_case=True) будет последовательно возвращать только a, b

В ex_2.py нужно вывести на экран то, что они выдают *одной строкой*. **Важно** продемонстрировать работу как с массивами, так и с генераторами (gen_random).
Итератор должен располагаться в librip/iterators.py

Задача 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
Пример:
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
Вывод: [0, 1, -1, 4, -4, -30, 100, -100, 123]

Задача 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
Пример:

```
@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()
```

На консоль выведется:

```
test_1
1
test_2
iu
test_3
a = 1
b = 2
test_4
1
2
```

Декоратор должен располагаться в librip/decorators.py

Задача 5 (ex_5.py)

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран
Пример:

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

После завершения блока должно вывестись в консоль примерно 5.5

Задача 6 (ex_6.py)
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 (ссылку на полную версию размером ~ 1 Гб. в формате xml можно найти в файле README.md).
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.

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.

Исходный код

Файл: gens.py

```
import random

def field(items, *args):
    assert len(args) > 0
    for item in items:
        for j in args:
            if item[j] is not None:
                yield (item[j])


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

Файл: ex_1.py

```
from librip.gens import field
from librip.gens import 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'}
]

print(list(field(goods, 'title', 'price', 'color')))

print(list(gen_random(1,3,5)))
```

Файл: Iterators.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items)
        self.unique=[]
        if len(kwargs) == 0:
            self.ignore_case = False
        else:
            self.ignore_case = kwargs.values()
        pass

    def __next__(self):
        while True:
            item = self.items.__next__()
            if type(item) is str and self.ignore_case:
                myitem =item.lower()
            else:
                myitem = item
            if myitem not in self.unique:
                self.unique.append(myitem)
                return myitem
        pass

    def __iter__(self):
        return self
```

Файл: 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', 'c', 'C']
# Реализация задания 2
print(list(Unique(data1)))

print(list(Unique(data2)))

print(list(Unique(data3)))

print(list(Unique(data3, ignore_case = True)))
```

Файл: ex_3.py

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3

print(sorted(data, key = abs))
```

Файл: decorators.py

```
def print_result(func, *arg):
    def decorated_func(*arg):
        result = func(*arg)
        print(func.__name__)
        if type(result) is dict:
            for m, n in result.items():
                print('{} = {}'.format(m, n))
        elif type(result) is list:
            for i in result:
                print(i)
        else:
            print(result)
        return result
    return decorated_func
```

Файл: ctxmngrs.py

```
import time

class timer:
    def __init__(self):
        pass

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

    def __exit__(self, type, value, traceback):
        print(time.clock() - self.time)
```

Файл: 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

path =
"C:\\Users\\LinaLT\\PycharmProjects\\Lab3\\data_light_cp1251.j
son"

# Здесь необходимо в переменную path получить
# путь до файла, который был передан при запуске

with open(path) as f:
    data = json.load(f)


# Далее необходимо реализовать все функции по заданию, заменив
`raise NotImplemented`
# Важно!
# Функции с 1 по 3 дожны быть реализованы в одну строку
# В реализации функции 4 может быть до 3 строк
# При этом строки должны быть не длиннее 80 символов

@print_result
def f1(arg):
    return list(Unique(sorted(list(field(arg, "job-name"))),
ignore_case=True))


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


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


@print_result
def f4(arg):
    salaries = list(gen_random(100000, 200000,
len(list(arg))))
    return list(zip(list(arg), list(map(lambda x: "зарплата "
+ str(x) + " руб.", salaries))))

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

Результаты

ex_1

```
C:\Users\LinaLT\PycharmProjects\Lab3\venv\Scripts\python.exe C:/Users/LinaLT/PycharmProjects/Lab3/ex_1.py
['Ковер', 2000, 'green', 'Диван для отдыха', 5300, 'black', 'Стелаж', 7000, 'white', 'Вешалка для одежды', 800, 'white']
[2, 2, 2, 2, 2]

Process finished with exit code 0
```

ex_2

```
C:\Users\LinaLT\PycharmProjects\Lab3\venv\Scripts\python.exe C:/Users/LinaLT/PycharmProjects/Lab3/ex_2.py
[1, 2]
[2, 1, 3]
['a', 'A', 'b', 'B', 'c', 'C']
['a', 'b', 'c']

Process finished with exit code 0
```

ex_3

```
C:\Users\LinaLT\PycharmProjects\Lab3\venv\Scripts\python.exe C:/Users/LinaLT/PycharmProjects/Lab3/ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]

Process finished with exit code 0
```

ex_4

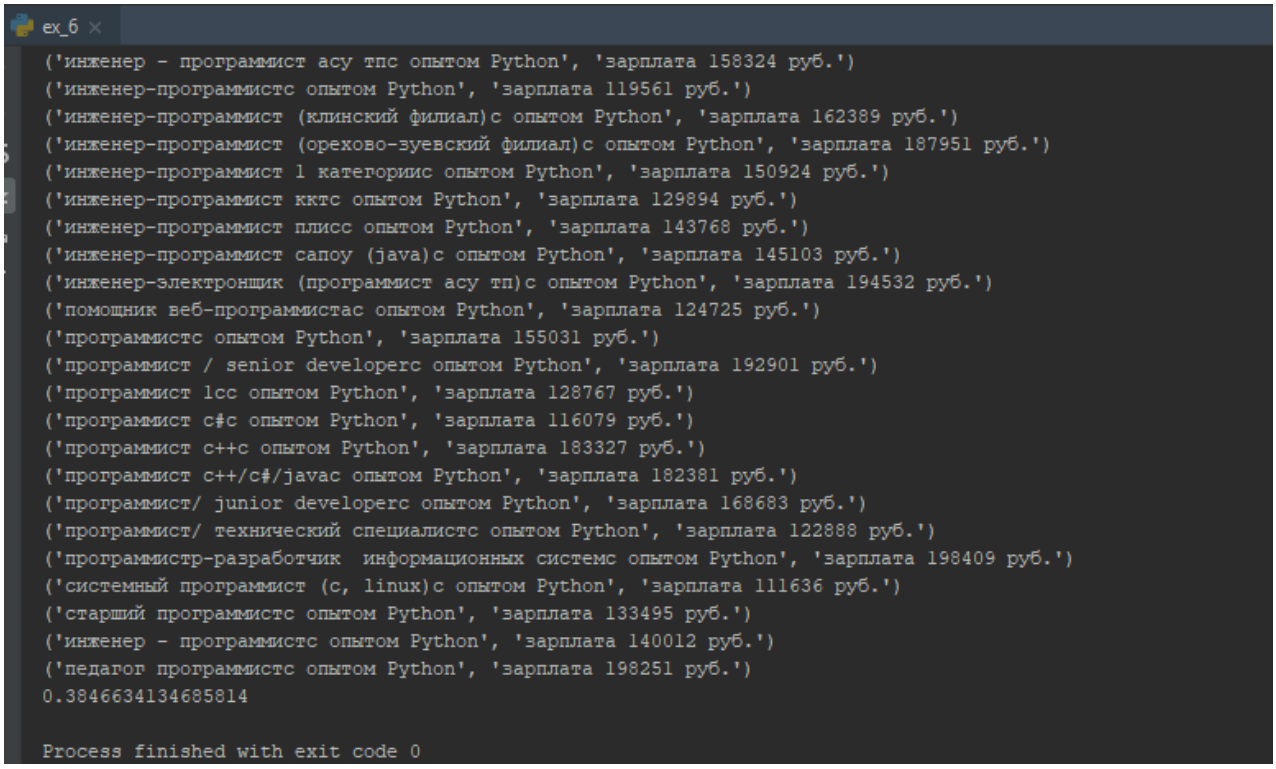
```
C:\Users\LinaLT\PycharmProjects\Lab3\venv\Scripts\python.exe C:/Users/LinaLT/PycharmProjects/Lab3/ex_4.py
test_1
1
test_2
iu
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0
```

ex_5

```
C:\Users\LinaLT\PycharmProjects\Lab3\venv\Scripts\python.exe C:/Users/LinaLT/PycharmProjects/Lab3/ex_5.py
5.512763421575502

Process finished with exit code 0
```


ex_6
('инженер - программист асу тпс опытом Python', 'зарплата 158324 руб.')
('инженер-программистс опытом Python', 'зарплата 119561 руб.')
('инженер-программист (клинский филиал)с опытом Python', 'зарплата 162389 руб.')
('инженер-программист (орехово-зуевский филиал)с опытом Python', 'зарплата 187951 руб.')
('инженер-программист 1 категориис опытом Python', 'зарплата 150924 руб.')
('инженер-программист кктс опытом Python', 'зарплата 129894 руб.')
('инженер-программист плисс опытом Python', 'зарплата 143768 руб.')
('инженер-программист сапоу (java)с опытом Python', 'зарплата 145103 руб.')
('инженер-электронщик (программист асу тп)с опытом Python', 'зарплата 194532 руб.')
('помощник веб-программистас опытом Python', 'зарплата 124725 руб.')
('программистс опытом Python', 'зарплата 155031 руб.')
('программист / senior developerc опытом Python', 'зарплата 192901 руб.')
('программист 1сс опытом Python', 'зарплата 128767 руб.')
('программист c#c опытом Python', 'зарплата 116079 руб.')
('программист c++c опытом Python', 'зарплата 183327 руб.')
('программист c++/c#/javac опытом Python', 'зарплата 182381 руб.')
('программист/ junior developerc опытом Python', 'зарплата 168683 руб.')
('программист/ технический специалистс опытом Python', 'зарплата 122888 руб.')
('программистр-разработчик информационных системс опытом Python', 'зарплата 198409 руб.')
('системный программист (с, linux)с опытом Python', 'зарплата 111636 руб.')
('старший программистс опытом Python', 'зарплата 133495 руб.')
('инженер - программистс опытом Python', 'зарплата 140012 руб.')
('педагог программистс опытом Python', 'зарплата 198251 руб.')
0.3846634134685814
Process finished with exit code 0