Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный технический университет им. Н. Э. Баумана»

Лабораторная работа №3

Отчёт о выполненной работы

Выполнил: студент группы СГН3-71

Волынов М.М.

Проверил: кандидат технических наук

Гапанюк Ю.Е.

Москва 2017

Задание на выполнение работы

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 реализовывать 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_4`
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 *одной строкой*
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`.
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.

Исходный код

Содержимое файла **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
gen = field(goods, 'title')
for i in range(4):
    print(i + 1, next(gen))

gen = field(goods, 'title', 'no_match')
for i in range(4):
    print(i + 1, next(gen))

gen = field(goods, 'title', 'price')
for i in range(4):
    print(i + 1, next(gen))

gen = field(goods, 'price')
for i in range(4):
    print(i + 1, next(gen))
```

Содержимое файла **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)

# Реализация задания 2
u = Unique(data1)
print(', '.join([str(i) for i in u]))

u = Unique(data2)
print(', '.join([str(i) for i in u]))

u = Unique(['a', 'A', 'b', 'B'])
print(', '.join([str(i) for i in u]))

u = Unique(['a', 'A', 'b', 'B'], ignore_case=True)
print(', '.join([str(i) for i in u]))

gen = gen_random(1, 3, 10)
print(', '.join([str(i) for i in gen]))
```

Содержимое файла **ex_3.py**

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

print(sorted(data, key=lambda x: abs(x)))
```

Содержимое файла **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]

# 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 должны 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 sorted([i for i in unique(field(arg, 'job-name'), ignore_case=True)])


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


@print_result
def f3(arg):
    return list(map(lambda x: '{} с опытом Python'.format(x), arg))


@print_result
def f4(arg):
    return list(['{}, зарплата {}'.format(l, r) for l, r in zip(arg, gen_random(100000, 200000, len(arg)))])


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

Содержимое файла **ctxmngrs.py**

```
from contextlib import contextmanager
import time


@contextmanager
def timer():
    start = time.time()
    yield
    end = time.time()
    print(end - start)
```

Содержимое файла **decorators.py**

```
def print_result(function):
    def wrapped(*args, **kwargs):
        print(function.__name__)
        result = function(*args, **kwargs)

        if isinstance(result, list):
            print(*result, sep='\n')
        elif isinstance(result, dict):
            print(*['{} = {}'.format(k, v) for k, v in result.items()], sep='\n')
        else:
            print(result)
        return result
    return wrapped
```

Содержимое файла **gens.py**

```
import random


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

    for item in items:
        if len(args) == 1:
            yield item[args[0]]
        else:
            result = {}
            for entry in args:
                value = item.get(entry)
                if value:
                    result[entry] = item[entry]
            if result:
                yield result


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

Содержимое файла **iterators.py**

```python
class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)

        if self.ignore_case:
            self.items = list(set([i.casefold() for i in items]))
        else:
            self.items = list(set(items))

        self.__index = 0
        self.__max = len(self.items)

    def __next__(self):
        if self.__index >= self.__max:
            raise StopIteration

        result = self.items[self.__index]
        self.__index += 1
        return result

    def __iter__(self):
        return self
```

Скриншоты работы

```
1 Ковер
2 Диван для отдыха
3 Стелаж
4 Вешалка для одежды
1 {'title': 'Ковер'}
2 {'title': 'Диван для отдыха'}
3 {'title': 'Стелаж'}
4 {'title': 'Вешалка для одежды'}
1 {'title': 'Ковер', 'price': 2000}
2 {'title': 'Диван для отдыха', 'price': 5300}
3 {'title': 'Стелаж', 'price': 7000}
4 {'title': 'Вешалка для одежды', 'price': 800}
1 2000
2 5300
3 7000
4 800
```

Рис 1. Вывод задания ex_1

```
1, 2
1, 2, 3
B, A, b, a
b, a
3, 3, 2, 3, 2, 1, 1, 1, 1, 2
```

Рис. 2. Вывод задания ex_2

```
[0, 1, -1, 4, -4, -30, 100, -100, 123]
```

Рис. 3. Вывод задания ex_3

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

Рис. 4. Вывод задания ex_4

```
5.500402450561523
```

Рис. 5. Вывод задания ex_5

```
. . .
энергетик литейного производства
энтомолог
юрисконсульт
юрисконсульт 2 категории
юрисконсульт. контрактный управляющий
юрист
юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
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 154970
программист / senior developer с опытом Python, зарплата 111068
программист 1с с опытом Python, зарплата 154061
программист c# с опытом Python, зарплата 145567
программист c++ с опытом Python, зарплата 194730
программист c++/c#/java с опытом Python, зарплата 170445
программист/ junior developer с опытом Python, зарплата 176803
программист/ технический специалист с опытом Python, зарплата 126431
программистр-разработчик  информационных систем с опытом Python, зарплата 121319
0.01700425148010254
```

Рис. 6. Вывод задания ex_6

Ссылка на репозиторий с исходным кодом:

https://github.com/StealthTech/lab_4