Московский государственный технический университет им. Н.Э. Баумана

Лабораторная работа №4 по дисциплине “Разработка интернет приложений”

ИСПОЛНИТЕЛЬ:

студент группы РТ5-51
Коростелёв В.М.

"__"_____2016 г.

Москва - 2016

Файл ex_1.py:

```
#!/usr/bin/env python3
from librip.gen import field, gen_random

goods = [
    {'title': 'Ковёр', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стеллаж', 'price': 7000, 'color': 'white'},
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'},
    {'title': None, 'price': 7000},
    {'title': None, 'price': None}
]

# Реализация задания 1
print(list(field(goods, 'title')))
print(list(field(goods, 'title', 'price1')))
print(list(gen_random(1, 3, 10)))
```

Результат:

```
['Ковёр', 'Диван для отдыха', 'Стеллаж', 'Вешалка для одежды']
[{'title': 'Ковёр'}, {'title': 'Диван для отдыха'}, {'title': 'Стеллаж'}, {'title': 'Вешалка для одежды'}]
[2, 3, 2, 2, 3, 3, 1, 2, 3, 3]
```

Файл ex_2.py:

```
#!/usr/bin/env python3
from librip.gen 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']
data4 = ['Aab', 'aAB', 'BBB', 'bbb', 'bbb']

# Реализация задания 2

print(list(Unique(data4, ignore_case=False)))
print(list(Unique(data3, ignore_case=True)))
print(list(Unique(data2, ignore_case=True)))
print(list(Unique(data1, ignore_case=True)))
```

Результат:

```
['Aab', 'aAB', 'BBB', 'bbb']
['a', 'b']
[2, 1, 3]
[1, 2]
```

Файл ex_3.py:

```
#!/usr/bin/env python3

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3
print(sorted(data, key=abs))
```

Результат:

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

Файл ex_4.py:

```
from librip.decorators import print_result

# Необходимо верно реализовать print_result
# и задание будет выполнено

@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()
```

Результат:

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

Файл ex_5.py:

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

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

Результат:

```
5.500229973923499
```

Файл ex_6.py:

```
#!/usr/bin/env python3
import json
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gen import field, gen_random
from librip.iterators import Unique as unique

path = r"...\data_light.json"

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

with open(path,encoding="utf-8") 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 sorted(unique(field(arg, 'job-name'), ignore_case=True), key=str.lower)
```

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


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


@print_result
def f4(arg):
    salary = list(gen_random(100000, 200000, len(arg)))
    return list('{}, зарплата {} руб'.format(x, y) for x, y in zip(arg, salary))


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

Результат:

```
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 164904 руб
Программист / Senior Developer с опытом Python, зарплата 144216 руб
Программист 1С с опытом Python, зарплата 176897 руб
Программист C# с опытом Python, зарплата 111416 руб
Программист C++ с опытом Python, зарплата 159298 руб
Программист C++/C#/Java с опытом Python, зарплата 129450 руб
Программист/ Junior Developer с опытом Python, зарплата 193850 руб
Программист/ технический специалист с опытом Python, зарплата 169262 руб
Программистр-разработчик  информационных систем с опытом Python, зарплата 168139 руб
12.785966938430777
```

Файл ctxmngrs.py:

```
import time


class timer:
    def __enter__(self):
        time.clock()
```

```
    def __exit__(self, exp_type, exp_value, traceback):
        print(time.clock())
```

Файл decorators.py:

```
def print_result(func_to_decorate):
    def decorated_func(*args):
        res = func_to_decorate(*args)
        print(func_to_decorate.__name__)
        if type(res) is str or type(res) is int:
            print(res)
        if type(res) is list:
            list(map(lambda x: print(x), res))
        if type(res) is dict:
            for k, v in res.items():
                print('{} = {}'.format(k, v))
        return res


    return decorated_func
```

Файл gen.py:

```
import random


def field(items, *args):
    assert len(args) > 0
    for item in items:
        if len(args) == 1:
            if item.get(args[0]) is None:
                continue
            yield item[args[0]]
        else:
            dictionary = {}
            for name in args:
                if item.get(name) is None:
                    continue
                dictionary[name] = item.get(name)
            if dictionary:
                yield dictionary
            else:
                continue


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

Файл iterators.py:

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = list(items)
        self.index = -1
        self.lst = []
        self.len = len(self.items)
        self.ignore_case = kwargs.get('ignore_case')

    def __next__(self):
        self.index += 1
        if self.index == self.len:
            raise StopIteration
        buffer = self.items[self.index]
        buf_str = str(buffer)
        if self.ignore_case:
            buf_str = buf_str.lower()
            while buf_str in self.lst:
                self.index += 1
                if self.index == self.len:
                    raise StopIteration
                buffer = self.items[self.index]
                buf_str = str(buffer).lower()
            self.lst.append(buf_str)
            return buffer
        else:
            while buffer in self.lst:
                self.index += 1
                if self.index == self.len:
                    raise StopIteration
                buffer = self.items[self.index]
            self.lst.append(buffer)
            return buffer

    def __iter__(self):
        return self
```
