Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №4

Выполнила:

студент группы ИУ5-52

Кучаева К.И.

Москва, 2017 г

iterators.py | ctxmngrs.py | decorators.py | gens.py

```python
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковые строки в разном регистре
        # По-умолчанию ignore_case = False
        self.ig_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 i in self.items:
            is_str = isinstance(i,str)
            if (not is_str) and (i not in self._s):
                self._s.add(i)
                return i
            elif is_str:
                if self.ig_case and (i.lower() not in self._s):
                    self._s.add(i.lower())
                    return i
                elif (not self.ig_case) and (i not in self._s):
                    self._s.add(i)
                    return i
        else: raise StopIteration()

    def __iter__(self):
        return self
```

ators.py | ctxmngrs.py | decorators.py | gens.py

```python
from datetime import datetime
# Здесь необходимо реализовать
# контекстный менеджер timer
# Он не принимает аргументов, после выполнения блока он должен выве
# Пример использования
# with timer():
#   sleep(5.5)
#
# После завершения блока должно вывестись в консоль примерно 5.5


class timer():
    def __enter__(self):
        self.start = datetime.today()

    def __exit__(self, exp_type, exp_value, exp_tr):
        print(datetime.today() - self.start)
```

rs.py × ctxmngrs.py × decorators.py × gens.py ×

```python
def print_result(func):
    def decor(*args):
        print("******", func.__name__, sep='\n')
        collect = func(*args)
        if isinstance(collect, list):
            print('\n'.join(map(str, collect)))
        elif isinstance(collect, dict):
            print('\n'.join(map(lambda k: str(k) + ' = ' + str(collect[k]),
                                collect)))
        else:
            print(collect)
        print("******")
        return collect

    return decor
```

ators.py × ctxmngrs.py × decorators.py × gens.py ×

```python
import random


# Генератор вычленения полей из массива словарей
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        for it in items:
            for key in args:
                elem = it.get(key)
                if elem is not None:
                    yield elem
    else:
        for it in items:
            res_dict = {}
            for key in args:
                elem = it.get(key)
                if elem is not None:
                    res_dict[key] = elem
            if len(res_dict)>0:
                yield res_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

```python
#!/usr/bin/env python3
from librip.gens import *

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'}
]

g1 = field(goods, 'title')
print(' '.join(map(str, g1)))
g2 = field(goods, 'title', 'price')
print(' '.join(map(str, g2)))
g3 = gen_random(1,5,4)
print(' '.join(map(str, g3)))

```

```
C:\Users\HP\AppData\Local\Programs\Python\Python35-32\python.exe K:/#1/inf/iu5web_labs/lab4/ex_1.py
Ковер Диван для отдыха Стелаж Вешалка для одежды
{'price': 2000, 'title': 'Ковер'} {'price': 5300, 'title': 'Диван для отдыха'} {'price': 7000, 'title': 'Стелаж'} {'price': 800, 'title': 'Вешалка для одежды'}
1 1 2 3

Process finished with exit code 0
```

ex_2.py

```python
#!/usr/bin/env python3
import ...

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

un = Unique(data1)
un2 = Unique(data2)
un3 = Unique(data3, ingnore_case=True)
un4 = Unique(data3)

print('list:', ' '.join(map(str,un)),'generator:', ' '.join(map(str, un2)), sep='\n')
print(' '.join(map(str, un3)))
print(' '.join(map(str, un4)))

```

ex_2

```
C:\Users\HP\AppData\Local\Programs\I
list:
1 2
generator:
3 1 2
A a b
A a b

Process finished with exit code 0
```

ex_3.py

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

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

```

ex_3

```
C:\Users\HP\AppData\Local\Programs\Python
[0, 1, -1, 4, -4, -30, 100, -100, 123]

Process finished with exit code 0
```

_3.py ex_4.py

```python
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()
```

ex_4

```
C:\Users\HP\AppData\Local\Programs\
******
test_1
1
******
******
test_2
iu
******
******
test_3
a = 1
b = 2
******
******
test_4
1
2
******

Process finished with exit code 0
```

ex_5.py

```
1  from time import sleep
2  from librip.ctxmngrs import timer
3
4  with timer():
5      sleep(5.5)
6
```

ex_5

```
C:\Users\HP\AppData\Local\Programs
0:00:05.501590

Process finished with exit code 0
```

ex_6.py

```
11
12      # Здесь необходимо в переменную path получить
13      # путь до файла, который был передан при запуске
14      path = sys.argv[1]
15
16      with open(path) as f:
17          data = json.load(f)
18
19
20      # Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
21      # Важно!
22      # Функции с 1 по 3 должны быть реализованы в одну строку
23      # В реализации функции 4 может быть до 3 строк
24      # При этом строки должны быть не длиннее 80 символов
25
26      @print_result
27      def f1(arg):
28          return list(unique(field(arg, 'job-name'), ignore_case=True))
29
30
31      @print_result
32      def f2(arg):
33          return list(filter(lambda x: re.match("^[п,П]рограммист", x) is not None, arg))
34
35
36      @print_result
37      def f3(arg):
38          return list(map(lambda x: x+" с опытом Python", arg))
39
```

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


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

```
f3
Программист с опытом Python
Программист C++/C#/Java с опытом Python
Программист 1С с опытом Python
Программистр-разработчик  информационных систем с опытом Python
Программист C++ с опытом Python
Программист/ Junior Developer с опытом Python
Программист / Senior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист C# с опытом Python
******
******
f4
Программист с опытом Python, зарплата 197237 руб
Программист C++/C#/Java с опытом Python, зарплата 141729 руб
Программист 1С с опытом Python, зарплата 192099 руб
Программистр-разработчик  информационных систем с опытом Python, зарплата 123726 руб
Программист C++ с опытом Python, зарплата 161139 руб
Программист/ Junior Developer с опытом Python, зарплата 137915 руб
Программист / Senior Developer с опытом Python, зарплата 113647 руб
Программист/ технический специалист с опытом Python, зарплата 181544 руб
Программист C# с опытом Python, зарплата 154840 руб
******
0:00:00.104112

Process finished with exit code 0
```