Московский государственный технический университет им. Н.Э. Баумана

Лабораторная работа №4 по дисциплине “Разработка интернет приложений”

ИСПОЛНИТЕЛЬ:

студент группы РТ5-51
Чечелев Д.С.

"__"_____2017 г.

Москва - 2017

Файл ex_1.py:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from librip.generators import field, random_generator

goods = [
    {'title': None, 'price': None, 'color': None},
    {'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', 'price')))
print(list(random_generator(1, 3, 10)))
```

Результат:
['Ковёр', 'Диван для отдыха', 'Стеллаж', 'Вешалка для одежды']
[{'title': 'Ковёр', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стеллаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800}, {'price': 7000}]
[1, 2, 3, 3, 2, 1, 2, 3, 3, 2]

Файл ex_2.py (в конце бесконечный генератор для проверки):

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from librip.generators import random_generator
from librip.generators import infinite
from librip.iterators import Unique
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
data2 = random_generator(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)))
count = 0
for x in Unique(infinite()):
    if count < 20:
        print(x)
        count += 1
    else:
        break
```

Результат:

```
Ignore case =  False
['Aab', 'aAB', 'BBB', 'bbb']
Ignore case =  True
['a', 'b']
Ignore case =  True
[1, 2, 3]
Ignore case =  True
[1, 2]
Ignore case =  None
84
367
811
529
73
580
604
319
757
895
567
387
565
90
410
154
357
337
255
948
```

Файл ex_3.py:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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
a = 1
b = 2
test_4
1
2

Файл ex_5.py:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from time import sleep
from librip.timr import timer

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

Результат:
5.506459978985263

Файл ex_6.py:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import json
import sys
from librip.timr import timer
from librip.decorators import print_result
from librip.generators import field, random_generator
from librip.iterators import 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 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(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(random_generator(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:
Ignore case = True
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
…
…тут огромный список профессий…
…
Электрослесарь по ремонту оборудования в карьере
Электроэрозионист
Эндокринолог
Энергетик
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 119095 руб
Программист / Senior Developer с опытом Python, зарплата 129912 руб
Программист 1С с опытом Python, зарплата 151184 руб
Программист C# с опытом Python, зарплата 103066 руб
Программист C++ с опытом Python, зарплата 186584 руб
Программист C++/C#/Java с опытом Python, зарплата 154650 руб
Программист/ Junior Developer с опытом Python, зарплата 127595 руб
Программист/ технический специалист с опытом Python, зарплата 157292 руб
Программистр-разработчик информационных систем с опытом Python, зарплата 149523 руб
0.06353686495529558

Файл timr.py:

```python
import time


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

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

Файл decorators.py:

```python
def print_result(func_to_decorate):
    def decorated_func(*args):
        res = func_to_decorate(*args)
        print(func_to_decorate.__name__)
        if type(res) in (str,int):
            print(res)
        if type(res) is list:
            [print(i) for i in res]
        if type(res) is dict:
            for y, z in res.items():
                print('{} = {}'.format(y, z))
        return res

    return decorated_func
```

Файл generators.py:

```python
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


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


def infinite():
    i=0
    while 1:
        i = random.randint(1, 1000)
        yield i
```

Файл iterators.py:

```python
class Unique():
    """Итератор для удаления дубликатов"""
    def __init__(self, input, **kwargs):
        self.item = iter(input)
        self.lst = []
        self.ignore_case = kwargs.get('ignore_case')
        print("Ignore case = ", self.ignore_case)

    def __next__(self):
        if self.ignore_case:
            buffer = next(self.item)
            buf_str = str(buffer)
            buf_str = buf_str.lower()
            while buf_str in self.lst:
                buffer = next(self.item)
                buf_str = str(buffer).lower()
            self.lst.append(buf_str)
            return buffer
        else:
            buffer = next(self.item)
            buf_str = str(buffer)
            while buf_str in self.lst:
                buffer = next(self.item)
                buf_str = str(buffer)
            self.lst.append(buf_str)
            return buffer

    def __iter__(self):
        return self
```