



Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №3

по курсу «РИП»

Исполнитель: ИУ5-51, Черепанов Е.

Преподаватель: Гапанюк Ю. Е.

Москва, 2018 г.

Задание.

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.

Код и результаты выполнения.

ctxmgrs.py:

```
import time
```

```
# Здесь необходимо реализовать
```

```
# контекстный менеджер timer
```

```
# Он не принимает аргументов, после выполнения блока он должен вывести время выполнения в секундах
```

```
class timer:
```

```
    def __enter__(self):
```

```
        self.start = time.time()
```

```
    def __exit__(self, exc_type, exc_val, exc_tb):
```

```
        ti = (time.time()) - self.start
```

```
        print(ti)
```

decorators.py:

```
# Здесь необходимо реализовать декоратор, print_result который принимает на вход функцию,
```

```
# вызывает её, печатает в консоль имя функции, печатает результат и возвращает значение
```

```
# Если функция вернула список (list), то значения должны выводиться в столбик
```

```
# Если функция вернула словарь (dict), то ключи и значения должны выводиться в столбик через знак равно
```

```
# РЕАЛИЗАЦИЯ
```

```
def print_result(func):
```

```
    def decorated_func(*args):
```

```
        if len(args) == 0:
```

```
            result = func()
```

```
        else:
```

```
            result = func(args[0])
```

```
        print(func.__name__)
```

```
        if type(result) == list:
```

```
            for i in result:
```

```
                print(i)
```

```
        elif type(result) == dict:
```

```
            for key in result:
```

```
                print(str(key) + " = " + str(result[key]))
```

```
        else:
```

```
            print(result)
```

```
        return result
```

```
    return decorated_func
```

gens.py:

```
import random
```

Генератор вычленения полей из массива словарей

```
def field(arr, *args):
    assert len(args) > 0
    # Необходимо реализовать генератор
    for el in arr: # где el - словарь
        slovar = {}
        for arg in args:
            if (arg in el.keys()) and (len(args) == 1):
                yield el[arg] # генератор выдает только значения полей
            elif arg in el is not None:
                slovar[arg] = el[arg] # формируем новый словарь,
                # где пропускаем элементы равные None
        if len(slovar) > 0 and len(args) > 1:
            yield slovar
```

Генератор списка случайных чисел

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

iterators.py:

```
from types import GeneratorType
```

Итератор для удаления дубликатов

```
class Unique(object):
```

```
    def __init__(self, items, ignore_case=False, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковые строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ разные строки
        # ignore_case = False, Абв и АБВ одинаковые строки, одна из них удалится
        # По-умолчанию ignore_case = False
        self.unique_items = []
        self.ignore_case = ignore_case
        self.items = iter(items)
```

```
    def __next__(self):
        # Нужно реализовать __next__
```

```
        while True:
```

```
            item = self.items.__next__()
            compare_item = None
```

```
            if self.ignore_case and type(item) is str:
                compare_item = item.lower()
            else:
                compare_item = item
```

```
            if compare_item not in self.unique_items:
                self.unique_items.append(compare_item)
            return item
```

```
    def __iter__(self):
```

```
return self
```

ex_1.py:

```
from librip.gens import *
```

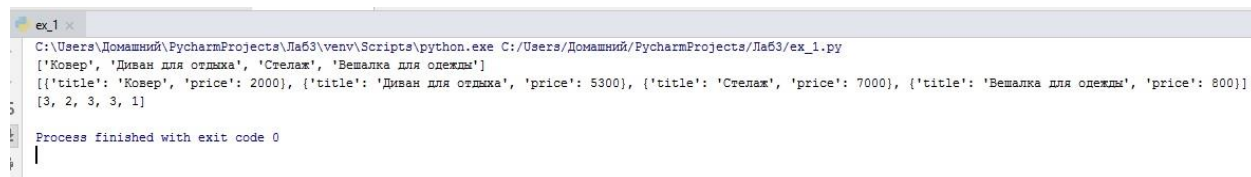
```
goods = [  
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},  
    {'title': 'Стелаж', 'price': 7000, 'color': 'white'},  
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}  
]
```

Реализация задания 1 (генераторы field и gen_random)

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

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

Результат:



```
ex_1 x  
C:\Users\Домашний\PycharmProjects\Ла63\venv\Scripts\python.exe C:/Users/Домашний/PycharmProjects/Ла63/ex_1.py  
['Ковер', 'Диван для отдыха', 'Стелаж', 'Вешалка для одежды']  
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стелаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800}]  
[3, 2, 3, 3, 1]  
Process finished with exit code 0
```

ex_2.py:

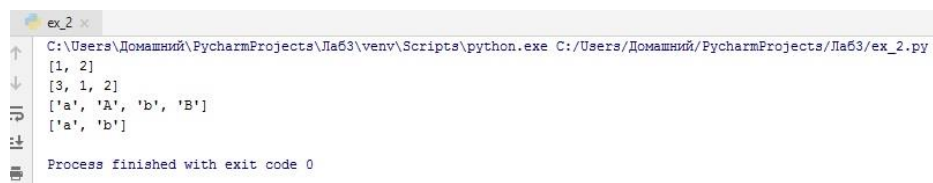
```
from librip.gens import gen_random  
from librip.iterators import Unique
```

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

Реализация задания 2

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

Результат:



```
ex_2 x  
C:\Users\Домашний\PycharmProjects\Ла63\venv\Scripts\python.exe C:/Users/Домашний/PycharmProjects/Ла63/ex_2.py  
[1, 2]  
[3, 1, 2]  
['a', 'A', 'b', 'B']  
['a', 'b']  
Process finished with exit code 0
```

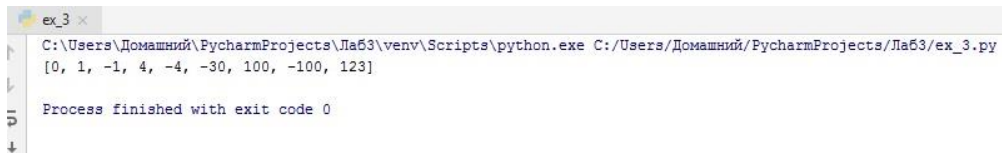
ex_3.py:

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

Реализация задания 3

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

Результат:



```
ex_3 x
C:\Users\Домашний\PycharmProjects\Лаб3\venv\Scripts\python.exe C:\Users\Домашний\PycharmProjects\Лаб3\ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]
Process finished with exit code 0
```

ex_4.py:

```
from librip.decorators import print_result
```

Необходимо верно реализовать print_result
и задание будет выполнено

```
@print_result # test_1=print_result(test_1)
def test_1():
    return 1
```

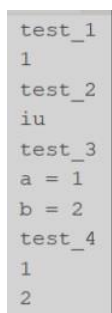
```
@print_result
def test_2():
    return 'iu'
```

```
@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:



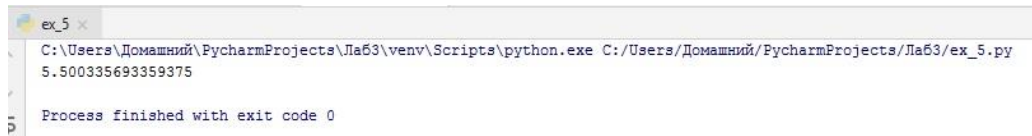
```
test_1
1
test_2
iu
test_3
a = 1
b = 2
test_4
1
2
```

ex_5.py:

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

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

Результат:



```
ex_5 ×  
C:\Users\Домашний\PycharmProjects\Лаб3\venv\Scripts\python.exe C:/Users/Домашний/PycharmProjects/Лаб3/ex_5.py  
5.500335693359375  
Process finished with exit code 0
```

ex_6.py:

```
import json  
import sys  
from librip.ctxmgrs import timer  
from librip.decorators import print_result  
from librip.gens import field, gen_random  
from librip.iterators import Unique
```

```
path = "data_light_cp1251.json"
```

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

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

```
#отсортированный список профессий без повторений
```

```
@print_result
```

```
def f1(arg):
```

```
    return list(Unique(list(field(arg, "job-name")), ignore_case=True))
```

```
#специальности, связанные с программированием
```

```
@print_result
```

```
def f2(arg):
```

```
    return list(filter(lambda s: "программист" in s[0:12], arg))
```

```
#все программисты должны быть знакомы с Python
```

```
@print_result
```

```
def f3(arg): # map(func, arr)
```

```
    return list(map(lambda s: s + " с опытом Python", arg))
```

```
#сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей
```

```
@print_result
```

```
def f4(arg):
```

```
    Sal = gen_random(100000, 200000, len(arg))
```

```
    return list(map(lambda s: '{} зарплата {} руб.'.format(  
        s[0], s[1]), zip(arg, Sal)))
```

```
with timer():
```

```
    f4(f3(f2(f1(data))))
```

Результат:

ip:

ex 6

Секретарь судебного заседания в аппарате мирового судьи Железнодорожного судебного района города Ростова-на-Дону
варщик зефира
варщик мармеладных изделий
Оператор склада
Специалист по электромеханическим испытаниям аппаратуры бортовых космических систем
Заведующий музеем в д.Копорье
Документовед
Специалист по испытаниям на электромагнитную совместимость аппаратуры бортовых космических систем
Менеджер (в промышленности)
f2
f3
f4
0.09106111526489258

Process finished with exit code 0