**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-51 |  | преподаватель каф. ИУ5 |
| Кочкин С.А. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

г. 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**

**ctxmngrs.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 выводить 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)

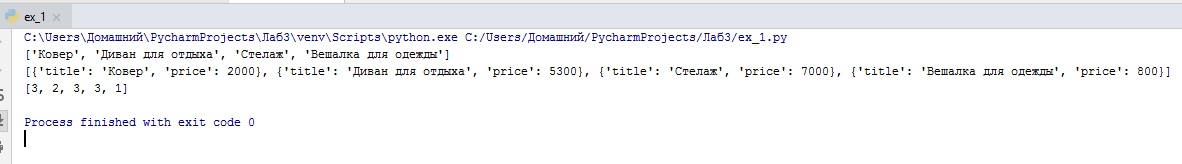
**iterators.py:**

**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, Aбв и АБВ разные строки  
 # ignore\_case = False, Aбв 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)))

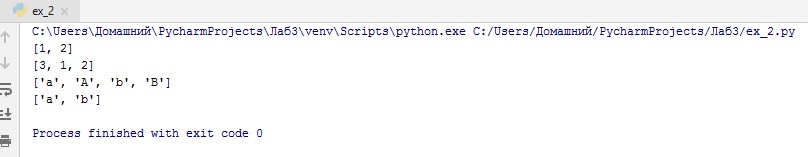
**Результат:**



**ex\_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**)))

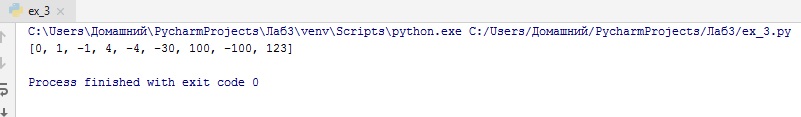
**Результат:**

****

**ex\_3.py:**

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

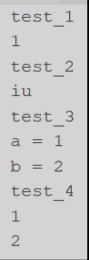
**Результат:**

****

**ex\_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()

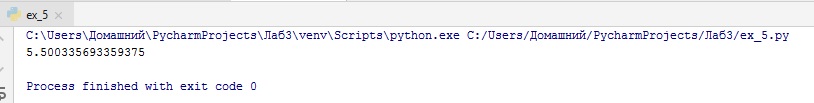
**Результат:**

****

**ex\_5.py:**

**from** time **import** sleep  
**from** librip.ctxmngrs **import** timer  
  
**with** timer():  
 sleep(5.5)

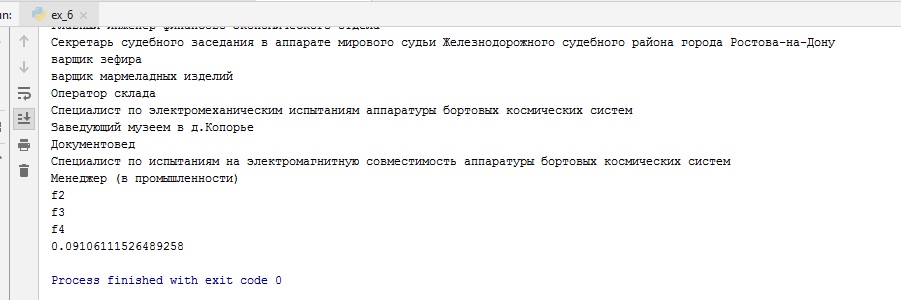
**Результат:**

****

**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  
  
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:**

****