Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»

Кафедра «Автоматизированные системы обработки информации и управления»

Отчёт по лабораторной работе №3

«Python. Функциональные возможности»

По курсу «Разработка интернет приложений»

Выполнил:

Студент группы РТ5-51

Непочатый Е.В.

Москва 2017

Задание

Важно выполнять все задачи последовательно. 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 реализовывать одной строкой.

Подготовительный этап

1. Зайти на github.com и выполнить `fork` проекта с заготовленной структурой https://github.com/iu5team/ex-lab4 2. Переименовать репозиторий в `lab_4`

3. Выполнить `git clone` проекта из вашего репозитория

Задача 1 (`ex_1.py`)

Необходимо реализовать генераторы `field` и `gen_random`

Генератор `field` последовательно выдает значения ключей словарей массива

Пример:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
 ]
```

`field(goods, 'title')` должен выдавать `'Ковер', 'Диван для отдыха'`

`field(goods, 'title', 'price')` должен выдавать `{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}`

1. В качестве первого аргумента генератор принимает `list`, дальше через `*args` генератор принимает неограниченное кол-во аргументов.
2. Если передан один аргумент, генератор последовательно выдает только значения полей, если поле равно `None`, то элемент пропускается
3. Если передано несколько аргументов, то последовательно выдаются словари, если поле равно `None`, то оно пропускается, если все поля `None`, то пропускается целиком весь элемент

Генератор `gen_random` последовательно выдает заданное количество случайных чисел в заданном диапазоне

Пример:

`gen_random(1, 3, 5)` должен выдать 5 чисел от 1 до 3, т.е. примерно `2, 2, 3, 2, 1`

В `ex_1.py` нужно вывести на экран то, что они выдают *одной строкой*

Генераторы должны располагаться в `librip/gen.py`

Задача 2 (ex_2.py)

Необходимо реализовать итератор, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты. Конструктор итератора также принимает на вход именной bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False . Итератор **не должен модифицировать** возвращаемые значения.

Пример:

```
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

Unique(data) будет последовательно возвращать только 1 и 2

```
data = gen_random(1, 3, 10)
```

unique(gen_random(1, 3, 10)) будет последовательно возвращать только 1, 2 и 3

```
 data = ['a', 'A', 'b',
'B']
```

Unique(data) будет последовательно возвращать только a, A, b, B

```
data = ['a', 'A', 'b', 'B']
```

Unique(data, ignore_case=True) будет последовательно возвращать только a, b

В ex_2.py нужно вывести на экран то, что они выдают *одной строкой* . **Важно** продемонстрировать работу как с массивами, так и с генераторами (gen_random).

Итератор должен располагаться в librip/iterators.py

Задача 3 (ex_3.py)

Дан массив с положительными и отрицательными числами. Необходимо одной строкой вывести на экран массив, отсортированный по модулю. Сортировку осуществлять с помощью функции sorted Пример:

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

Вывод: [0, 1, -1, 4, -4, -30, 100, -100, 123]

Задача 4 (ex_4.py)

Необходимо реализовать декоратор print_result , который выводит на экран результат выполнения функции. Файл ex_4.py **не нужно** изменять.

Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции, печатать результат и возвращать 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

Пример:

```
@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()
```

На консоль выведется:

```
test_1
1
test_
2 iu
test_
3 a =
1 b =
2
test_
4
1
2
```

Декоратор должен располагаться в `librip/decorators.py`

Задача 5 (`ex_5.py`)

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран Пример: `with timer():     sleep(5.5)`

После завершения блока должно вывестись в консоль примерно 5.5

Задача 6 (`ex_6.py`)

Мы написали все инструменты для работы с данными. Применим их на реальном примере, который мог возникнуть в жизни. В репозитории находится файл `data_light.json` . Он содержит облегченный список вакансий в России в формате `json` (ссылку на полную версию размером ~ 1 Гб. в формате `xml` можно найти в файле `README.md`).

Структура данных представляет собой массив словарей с множеством полей: название работы, место, уровень зарплаты и т.д.

В `ex_6.py` дано 4 функции. В конце каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора `@print_result` печатается результат, а контекстный менеджер `timer` выводит время работы цепочки функций.

Задача реализовать все 4 функции по заданию, ничего не изменяя в файле-шаблоне. Функции `f1-f3` должны быть реализованы в 1 строку, функция `f4` может состоять максимум из 3 строк.

Что функции должны делать:

1. Функция `f1` должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна **игнорировать регистр**. Используйте наработки из предыдущих заданий.
2. Функция `f2` должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Иными словами нужно получить все специальности, связанные с программированием. Для фильтрации используйте функцию `filter`.
3. Функция `f3` должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: *Программист C# с опытом Python*. Для модификации используйте функцию `map`.
4. Функция `f4` должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: *Программист C# с опытом Python, зарплата 137287 руб.* Используйте `zip` для обработки пары специальность — зарплата.

Задание 1

```python
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for i in items:
            a = i.get(args[0])
            if a:
                yield a
    else:
        for i in items:
            a = {key: i[key] for key in args if i.get(key)}
            if a:
                yield a
```

```python
def gen_random(begin, end, num_count):
    assert begin < end and num_count > 0
    for i in range(num_count):
        yield random.randrange(begin, end)
```

```python
print(*field(goods, 'title'), sep=', ')
print(*field(goods, 'title', 'price'), sep=', ')
print(*field(goods, 'title', 'price', 'color'), sep=', ')

print(*gen_random(1, 20, 5), sep=', ')
```

Скриншоты

```
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}
{'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'}
```

Задание 2

```python
class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        self.__prev = None
        self.items = list(filter(self.__check_unique, items))
        self.len = len(self.items)
        self.ind = 0

    def __check_unique(self, x):
        if self.ignore_case:
            if isinstance(x, str) and isinstance(self.__prev, str):
                if x.casefold() == self.__prev.casefold():
                    return False
        else:
            if x == self.__prev:
                return False

        self.__prev = x
        return True

    def __next__(self):
        if self.ind == self.len:
            raise StopIteration

        self.ind += 1
        return self.items[self.ind - 1]

    def __iter__(self):
        return self
```

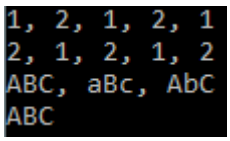
```
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1]
data2 = gen_random(1, 3, 10)
data3 = ['ABC', 'aBc', 'AbC']

# Реализация задания 2
uni1 = Unique(data1)
print(*uni1, sep=', ')

uni2 = Unique(data2)
print(*uni2, sep=', ')

uni31 = Unique(data3)
uni32 = Unique(data3, ignore_case=True)
print(*uni31, sep=', ')
print(*uni32, sep=', ')
```

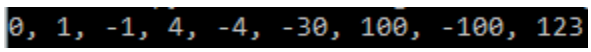
Скриншоты



Задание 3

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
sort = sorted(data, key=abs)  # key=lambda x: abs(x)
print(*sort, sep=', ')
```

Скриншоты



Задание 4

```
def print_result(func):
    def dec_func(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)

        if isinstance(res, list):
            for x in res:
                print(x)

        elif isinstance(res, dict):
            for key, val in res.items():
                print('{} = {}'.format(key, val))

        else:
            print(res)

        return res

    return dec_func
```

Скриншоты

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

Задание 5

```
class timer:
    def __init__(self):
        self.time = 0

    def __enter__(self):
        self.time = time.clock()

    def __exit__(self, exc_type, exc_val, exc_tb):
        print(time.clock() - self.time)
```

Скриншоты

```
5.49988824780208
```

Задание 6

```
path = None  # 'data_light_cp1251.json'

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

if len(sys.argv) < 2:
    path = input('Введите имя файла: ')
else:
    path = sys.argv[1]

with open(path) 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 list(unique(sorted(field(arg, 'job-name'), key=str.casefold),
ignore_case=True))


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


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

```python
@print_result
def f4(arg):
    salaries = list(gen_random(100000, 200001, len(arg)))
    return list(map(lambda x: x[0] + ', зарплата ' + str(x[1]) + ' руб', zip(arg,
salaries)))


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

Скриншоты

<Тут очень много>