Отчет по лабораторной работе №3
по курсу «Разработка интернет-приложений»
«Функциональные возможности в Python»

Выполнила:

Попова Марина, ИУ5-51

_____

Преподаватель:

Гапанюк Ю.Е.

_____

2017 г.

1) Задание лабораторной работы.

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 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

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.

2)Код программы

librip:

gens.py:

```
import random


def field(items, *args):
    assert len(args) > 0, 'No arguments'
    i = 0
    if len(args) == 1:
        while i < len(items):
            if items[i].get(args[0]) is not None:
                yield items[i].get(args[0])
            i += 1
    else:
        while i < len(items):
            d = {}
            for el in args:
                if items[i].get(el) is not None:
                    d[el] = items[i].get(el)
            if len(d) != 0:
                yield d
            i += 1


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

iterators.py

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items) if isinstance(items, list) else items  #
проходим по элементам
        self.ignore_case = kwargs.get('ignore_case', False)  # По-умолчанию
ignore_case = False
        self.lst = set()

    def __next__(self):
        while True:
            el = next(self.items)
            if self.ignore_case:
                if el.lower() not in self.lst:
                    self.lst.add(el.lower())
                    return el
            elif el not in self.lst:
                self.lst.add(el)
                return el

    def __iter__(self):
        return self
```

decorators.py

```
def print_result(func):
    def decorated_func(*args, **kwargs): #decorator
```

```
        print(func.__name__) #вывод имени функции  # test_1
        res=func(*args, **kwargs)
        if type(res) is list:
            print("\n".join(map(str,res))) #преобразование в str
        elif type(res) is dict:
            print('\n'.join([str(x)+"="+str(res[x]) for x in res]))
        else:
            print(res)
        return res
    return decorated_func
```

ctxmngrs.py

```
class timer:
    """Блок with делает неявный вызов методов __enter__ и __exit__ у объекта.
Такой объект называется менеджером контекста."""
    def __enter__(self):
        self.t = time.time()
    def __exit__(self, exp_type, exp_value, traceback):
        print('time of block is',time.time()-self.t)
```

lab4:

ex_1.py

```
from librip.gens import field, gen_random

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
g = field(goods, 'title', 'color', 'price')
for i in g:
    print(i, end="\n")
num = gen_random(1,5,10)
for i in num:
    print(i, end=" ")
```

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/ex_1.py
{'title': 'Ковер', 'color': 'green', 'price': 2000}
{'title': 'Диван для отдыха', 'color': 'black', 'price': 5300}
{'title': 'Стелаж', 'color': 'white', 'price': 7000}
{'title': 'Вешалка для одежды', 'color': 'white', 'price': 800}
4 4 1 4 2 5 2 1 2 4
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','P','k','M','A','p','m']

# Реализация задания 2
print('Before Unique',data1)
u1 = Unique(data1)
print('After Unique', end = ' ')
for i in u1:
    print(i, end=" ")
print('\n\nC ГЕНЕРАТОРОМ\nBefore Unique')
d2=[]
for i in data2:
    d2.append(i)
    print(i, end=" ")
print('\nAfter Unique')
u2 = Unique(d2)
for k in u2:
    print(k, end=" ")
print()


print('Before Unique',data3)
u3 = Unique(data3)
for i in u3:
    print(i, end=" ")
print()

print('Без повторений по регистру:')
u4 = Unique(data3, ignore_case = True)
for i in u4:
    print(i, end=" ")
```

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/ex_2.py
Before Unique [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
After Unique 1 2
C ГЕНЕРАТОРОМ
Before Unique
3 3 2 2 3 1 2 2 2 1
After Unique
3 2 1
Before Unique ['a', 'P', 'k', 'M', 'A', 'p', 'm']
a P k M A p m
Без повторений по регистру:
a P k M
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=abs))
```

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/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
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()
```

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/ex_4.py
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

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

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

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/ex_5.py
time of block is 5.503028154373169

Process finished with exit code 0
```

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 as unique

path = sys.argv[1]

#with open("data_light_cp1251.json", encoding="cp1251") as f:
#    data = json.load(f)

with open(path, encoding="cp1251") as f:
    data = json.load(f)

@print_result
def f1(arg):
    return sorted(unique(field(arg, 'job-name'), ignore_case=1), key=lambda
x: x.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))  # arg
уменьшилось по предыдущему фильтру

@print_result
def f4(arg):
    s = list(gen_random(100000, 200000, len(arg)))
    return list('{}, зарплата {} руб.'.format(arg, s) for arg, s in zip(arg,
s))

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

Результаты:

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/marinapopova/lab4/ex_6.py /Users/marinapopova/lab4/data_light_cp1251.json
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
автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
```

```
f2
Программист
Программист / Senior Developer
Программист 1С
Программист С#
Программист С++
Программист С++/С#/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 С# с опытом Python
Программист С++ с опытом Python
Программист С++/С#/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 178169 руб.
Программист / Senior Developer с опытом Python, зарплата 190295 руб.
Программист 1С с опытом Python, зарплата 170009 руб.
Программист С# с опытом Python, зарплата 117143 руб.
Программист С++ с опытом Python, зарплата 178169 руб.
Программист С++/С#/Java с опытом Python, зарплата 199265 руб.
Программист/ Junior Developer с опытом Python, зарплата 187459 руб.
Программист/ технический специалист с опытом Python, зарплата 163027 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 171615 руб.
time of block is 0.030148983001708984
```