Московский государственный технический университет им. Н.Э. Баумана

Отчет по лабораторной работе № 4 по курсу разработка интернет-приложений

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-71Ц _____
(подпись)

Баглай П.С

"__"_____2017 г.

ПРОВЕРИЛ:

Гапанюк Ю.Е _____
(подпись)

"__"_____2017 г.

Москва – 2017 г.

Порядок работы

Задача 1 (ex_1.py)

Необходимо реализовать генераторы fieldи gen_random
Генератор fieldпоследовательно 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 дляотдыха'
field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price':2000},
{'title': 'Диван для отдыха'}

1. В качестве первого аргумента генератор принимает list, дальше через *argsгенератор принимает неограниченное кол-во аргументов.
2. Если передан один аргумент, генератор последовательно выдает только значения полей, если полеравно
   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_randomпоследовательно 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 1и 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.

Задача 5 (ex_5.py)

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран .

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 xmlможно 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-f3должны 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 f2должна 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 f3должна 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 f4должна 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 zipдля обработки пары специальность — зарплата.

Листинг программы

1. ex_1.py: использование *args

```
#!/usr/bin/env python3
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'}
]

g1 = field(goods, 'title')
print(' '.join(map(str, g1)))
g2 = field(goods, 'title', 'price')
print(' '.join(map(str, g2)))
g3 = gen_random(1,5,4)
print(' '.join(map(str, g3)))
```

2. ex_2.py: ипользование **kwargs, поддержка работы как со списками, так и с генераторами

```
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']

un = Unique(data1)
un2 = Unique(data2)
un3 = Unique(data3, ingnore_case=True)
un4 = Unique(data3)

print('list:', ' '.join(map(str,un)),'generator:', ' '.join(map(str, un2)),
sep='\n')
print(' '.join(map(str, un3)))
print(' '.join(map(str, un4)))
```

3.ex_3.py: использование lambda-выражения

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

4. ex_4.py: внутри декоратора печать должна быть реализована в одну строчку. Печать словарей и массивов должна выполняться в столбик

```
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()
```

5. ex_5.py

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

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

6. ex_6.py соблюдения кол-ва строк и использование функций, указанных в задании

```
#!/usr/bin/env python3
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
import re

path = r"data_light_cp1251.json"



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



@print_result
def f1(arg):
    return list(unique(field(arg, 'job-name'), ignore_case=True))


@print_result
def f2(arg):
    return list(filter(lambda x: re.match("^[п,П]программист", x) is not
None, arg))


@print_result
def f3(arg):
    return list(map(lambda x: x+" опыт Python", arg))


@print_result
def f4(arg):
    li = list(zip(arg, list(gen_random(100000, 200000, len(arg)))))
    return list(map(lambda x: x[0]+",зарплата "+str(x[1])+" руб", li))


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

Результаты работы программы

1.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}
5 5 4 1


Process finished with exit code 0
```

2.ex_2.py

```
list:
1 2
generator:
2 1 3
A a b
A a b

Process finished with exit code 0
```

3.ex_3.py

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

Process finished with exit code 0
```

4.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
```

5.ex_5.py

```
0:00:05.507395

Process finished with exit code 0
```

6.ex_6.py

```
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
```