Московский государственный технический университет им. Н.Э.Баумана

Отчет по лабораторной работе №4 по курсу Разработка интернет приложений

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-52 _____

(подпись)

Матвеева П.Р.

"__"_____2016 г.

Москва, МГТУ - 2016

Оглавление

1. Описание задания лабораторной работы.

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 о дной 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). П ример: 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. Модули.

Librip

ctxmngrs.py

```
import time
class timer:
    def __enter__(self):
        self.t = time.clock()
    def __exit__(self, exp_type, exp_value, traceback):
        print(time.clock()-self.t)
```

decorators.py

```
def print_result(func):
    def decorated_func(*args):
        print(func.__name__)
        if len(args)>0:
            res=func(args[0])
        else:
            res=func()
        if type(res) is list:
            print("\n".join(map(str,res)))
        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
```

gens.py

```
import random


# Генератор вычленения полей из массива словарей
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, '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', 'price': 5300}

def field(items, *args):
    assert len(args) > 0, 'No args'
    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])
            else:
                continue
            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


    # Генератор списка случайных чисел
# Пример:
# gen_random(1, 3, 5) должен выдать примерно 2, 2, 3, 2, 1
# Hint: реализация занимает 2 строки


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.ignore_case = 0
        self.items = iter(items) if isinstance(items, list) else items
        if 'ignore_case' in kwargs:
            self.ignore_case = kwargs['ignore_case']
        self.lst=[]
        pass

    def __next__(self):
        while True:
            el = next(self.items)
            if el == 'Юрист':
                print()
            if self.ignore_case:
                if el.lower().strip() not in self.lst:
                    self.lst.append(el.lower().strip())
                    return el
            elif el not in self.lst:
                    self.lst.append(el)
                    return el
        raise StopIteration

    def __iter__(self):

        return self
```

ex_1.py

```
#!/usr/bin/env python3
from librip.gens import field
from librip.gens import gen_random
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': None, 'price': 7000, 'color': 'white'},
    {'title': None, 'price': None, 'color': 'white'}
]

# Реализация задания 1
g = field(goods, 'title', 'price')
for i in g:
    print(i, end=" ")

print()
num = gen_random (1,5,3)
for i in num:
    print(i, end=" ")
```

ex_2.py

```python
#!/usr/bin/env python3

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

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
data2 = gen_random(1, 3, 10)
data3 = ['a', 'A', 'b', 'B']
data4 = ['юрист', 'Юрист ']

# Реализация задания 2

u = Unique(data1)
for i in u:
    print(i, end=" ")
print()

u = Unique(data2)
for i in u:
    print(i, end=" ")
print()

u = Unique(data3)
for i in u:
    print(i, end=" ")
print()

u = Unique(data4, ignore_case = True)
for i in u:
    print(i, end=" ")
```

ex_3.py

```python
#!/usr/bin/env python3

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3

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

ex_4.py

```python
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()
```

ex_5.py

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

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

ex_6.py

```
#!/usr/bin/env python3
import json
import sys
#import codecs
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
print(sys.getdefaultencoding())
#sys.stdout = codecs.getwriter('utf8')(sys.stdout.buffer, 'replace')
#sys.stderr = codecs.getwriter('utf8')(sys.stderr.buffer, 'replace')

path = sys.argv[1]

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

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


# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Важно!
# Функции с 1 по 3 дожны 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 sorted(unique([i for i in 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))


@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))))
```

3. Результаты работы.

```
"C:\Program Files (x86)\Python35-32\python.exe" C:/Users/hp/PycharmProjects/lab4/ex_1.py
{'title': 'Ковер', 'price': 2000} {'title': 'Диван для отдыха', 'price': 5300} {'price': 7000}
5 3 5
Process finished with exit code 0
```

```
"C:\Program Files (x86)\Python35-32\python.exe" C:/Users/hp/PycharmProjects/lab4/ex_2.py
1 2
2 3 1
a A b B
юрист
Process finished with exit code 0
```

```
"C:\Program Files (x86)\Python35-32\python.exe" C:/Users/hp/PycharmProjects/lab4/ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]
Process finished with exit code 0
```

```
"C:\Program Files (x86)\Python35-32\python.exe" C:/Users/hp/PycharmProjects/lab4/ex_4.py
test_1
1
test_2
iu
test_3
a=1
b=2
test_4
1
2
```

```
"C:\Program Files (x86)\Python35-32\python.exe" C:/Users/hp/PycharmProjects/lab4/ex_5.py
5.50036044054209

Process finished with exit code 0
```