Согласовано:
Гапанюк Ю.Е.

"__"_____2016 г.

Утверждаю:
Гапанюк Ю.Е.

«__»__

Лабораторная работа №4 по курсу

Разработка интернет приложений

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-53
Семенова Е.В.

_____

"21" октября 2016 г.

Москва - 2016

Задание

Задача 1 (ex_1.py)

Необходимо реализовать генераторы field и gen_random.

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 в озвращаемые значения.

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.

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.

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.

Iterators.py

```
# Итератор для удаления дубликатов
class Unique(object):
	def __init__(self, items, **kwargs):

		self.ignore_case = kwargs.get('ignore_case', False)

		self.items = iter(items)

		self.unique_items = set()


		self.index = 0

	def __next__(self):


		for x in self.items:
			if not x is None:
				origin = x
				if self.ignore_case == True:
					x = str(x).lower()
				if not x in self.unique_items:
					self.unique_items.add(x)
					return origin

		raise StopIteration

	def __iter__(self):
		return self
```

decorators.py

```
def print_result(func):
	def wrapper(*args, **kwargs):

		result = func(*args, **kwargs)

		print(func.__name__)
		if type(result) == dict:
			for i in result.keys():
				print(i, " = ", result[i])
```

```
            elif type(result) == list:
                for i in result:
                    print(i)
            elif type(result) == tuple:
                if len(result) == 1:
                    print(result[0])
            else:
                print(result)
            return result
        return wrapper
```

gen.py

```
import random

# Генератор вычленения полей из массива словарей
def field(items, *args):
    assert len(args) > 0

    for n in items:
        if len(args) == 1:
            if not n.get(args[0]) is None:
                yield n.get(args[0]);
        else:
            new_items = {};
            for x in args:
                if not n[x] is None:
                    new_items[x] = n[x];
            yield new_items;

# Генератор списка случайных чисел
def gen_random(begin, end, num_count):
    a = 0;
    while a < num_count:
        yield begin + round(random.random()*(end - begin))
        a += 1
```

ctxmgrs.py

```
# контекстный менеджер timer
import time

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

ex_1.py

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

```
from librip.gen import field
from librip.gen import 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'}
]

for x in field(goods, 'title'):
        print(x, end = " ")

for x in field(goods, 'title', 'price'):
        print(x, end = " ")

for x in field(goods, 'title', 'price', 'color'):
        print(x, end = " ")

for x in gen_random(3, 5, 20):
        print(x, end = " ")
```

Результат выполения:

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'} 3 4 5 4 3 3 3 4 5 5 4 3 5 4 4 3 3 5 3 4

ex_2.py

```
#!/usr/bin/env python3
from librip.gen 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"]
data4 = ["Nna", "nna", None]

# Реализация задания 2

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

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

for i in Unique(data3, ignore_case = True):
        print(i, end = " ")
```

```
#print()

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

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

Результат выполнения:

```
1 2 2 3 1 a Nna Nna nna
```

ex_3.py

```
#!/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)))
```

Результат выполнения:

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

ex_4.py

```
from librip.decorators import print_result

@print_result
def test_1(a):
    return a

@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(1)
test_2()
test_3()
test_4()
```

Результат выполнения:

```
test_1
1
```

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

ex_5.py

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

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

Результат выполнения:

```
5.505624771118164
```

ex_6.py

```
#!/usr/bin/env python3
import json
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gen import field, gen_random
from librip.iterators import Unique as unique

path = sys.argv[1]

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

@print_result
def f1(arg):
    return sorted([y for y in unique(field(data, "job-name"), ignore_case = True)], key = lambda x:
x.lower())

@print_result
def f2(arg):
    return list(filter(lambda x: x[0:11].lower() == "программист", arg))

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

@print_result
def f4(arg):
    return [x + ", зарплата " + str(y) + " руб." for x, y in zip(arg, gen_random(100000, 200000,
len(arg)))]

with timer():
```

```
f4(f3(f2(f1(data))))
```

Результат выполнения:

```
…
f4
Программист с опытом Python, зарплата 152823 руб.
Программист / Senior Developer с опытом Python, зарплата 184503 руб.
Программист 1С с опытом Python, зарплата 105960 руб.
Программист C# с опытом Python, зарплата 163268 руб.
Программист C++ с опытом Python, зарплата 120242 руб.
Программист C++/C#/Java с опытом Python, зарплата 105318 руб.
Программист/ Junior Developer с опытом Python, зарплата 171521 руб.
Программист/ технический специалист с опытом Python, зарплата 185949 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 129750 руб.
0.10032367706298828
```