МГТУ им. Н.Э. Баумана
Факультет «Информатика и системы управления»

ДИСЦИПЛИНА:
«Разработка интернет-приложений»

Отчет по лабораторной работе №2
«Python. Функциональные возможности»

Выполнил:
Студент 3 курса
Факультет ИУ
Группа ИУ5-51Б
Ерохин И.А.
Преподаватель:
Гапанюк Ю.Е.

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 запрограммировать 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_2
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, с помощью кода в *одну строку* 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
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
test_1()
```

На консоль выведется:

```
test_1
1
```

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.

Исходный код данного задания

Задача 1

```
import random

#    Генератор вычленения полей из массива словарей def field(arr, *args):
     assert len(args) > 0
#       Необходимо реализовать генератор for el in arr: # где el - словарь
        slovar = {} for
        arg in args:
            if (arg in el.keys()) and (len(args) == 1):
                yield el[arg]  # генератор выдает только значения полей
            elif arg in el is not None:
                slovar[arg] = el[arg]  # формируем новый словарь,
#                   где пропускаем элементы равные None if len(slovar) > 0 and len(args) > 1:
            yield slovar


#    Генератор списка случайных чисел
def gen_random(begin, end, num_count):
#       Необходимо реализовать генератор for i in range(num_count):
        yield random.randint(begin, end)
```

Задача 2

```
from types import GeneratorType
from librip.ex1 import gen_random


#    Итератор для удаления дубликатов
     class Unique(object):
     def __init__(self, items, ignore_case=False, **kwargs):
#           Нужно реализовать конструктор
#           В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
#           в зависимости от значения которого будут считаться одинаковые строки в разном регистре
#           Например: ignore_case = True, Абв и АБВ разные строки
#                     ignore_case = False, Абв и АБВ одинаковые строки, одна из них удалится
#           По-умолчанию ignore_case = False
        self.unique_items = []
```

```
        self.ignore_case = ignore_case
        self.items = iter(items)

    def __next__(self):
#           Нужно реализовать __next__ while True:
            item = self.items.__next__()
            compare_item = None

            if self.ignore_case and type(item) is str:
                compare_item = item.lower()
            else:
                compare_item = item

            if compare_item not in self.unique_items:
                self.unique_items.append(compare_item)
                return item

    def __iter__(self):

        return self
```

Задача 3

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

Задача 4

```
def print_result(func):
    def decorated_func(*args):
        if len(args) == 0:
            result = func()
        else:
            result = func(args[0])
        print(func.__name__)
        if type(result) == list:
            for i in result:
                print(i)
        elif type(result) == dict:
            for key in result:
                print(str(key) + " = " + str(result[key]))
        else:
            print(result)
        return result

    return decorated_func
```

Задача 5

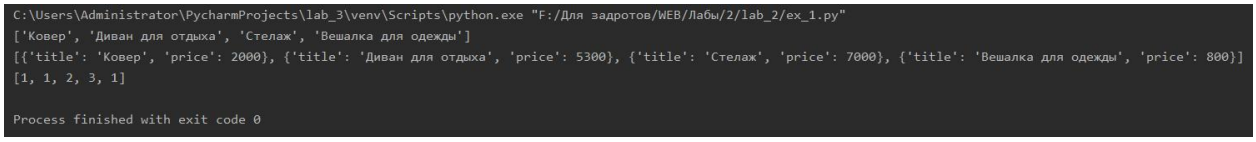
```
import time


#   Здесь необходимо реализовать
#   контекстный менеджер timer
#   Он не принимает аргументов, после выполнения блока он должен вывести время выполнения в секундах

class timer:
```

```
    def __enter__(self):
        self.start = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        ti = (time.time()) - self.start
        print(ti)
```

Задача 6

```
 #!/usr/bin/env python3
import json
from librip.ex5 import timer
from librip.ex4 import print_result
from librip.ex1 import field, gen_random
from librip.ex2 import Unique

#     path = "data_light_cp1251.json"

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

with open("data_light_cp1251.json") as read_file:
    data = json.load(read_file)


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


@print_result
def f2(arg):
    return list(filter(lambda s: "программист" in s, arg))


@print_result
def f3(arg): # map(func, arr)
    return list(map(lambda s: s + " с опытом Python", arg))


@print_result
def f4(arg):
    prof = gen_random(100000, 200000, len(arg))
    return list(map(lambda s: '{}, зарплата {} руб.'.format(s[0], s[1]), zip(arg, prof)))


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

Скриншоты с результатами выполнения



```
C:\Users\Administrator\PycharmProjects\lab_3\venv\Scripts\python.exe "F:/Для задротов/WEB/Лабы/2/lab_2/ex_2.py"
[1, 2]
[2, 3, 1]
['a', 'A', 'b', 'B']
['a', 'b']

Process finished with exit code 0
```

```
C:\Users\Administrator\PycharmProjects\lab_3\venv\Scripts\python.exe "F:/Для задротов/WEB/Лабы/2/lab_2/ex_3.py"
[0, 1, -1, 4, -4, -30, 100, -100, 123]

Process finished with exit code 0
```

```
C:\Users\Administrator\PycharmProjects\lab_3\venv\Scripts\python.exe "F:/Для задротов/WEB/Лабы/2/lab_2/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
```

```
C:\Users\Administrator\PycharmProjects\lab_3\venv\Scripts\python.exe "F:/Для задротов/WEB/Лабы/2/lab_2/ex_5.py"
0.503803014755249

Process finished with exit code 0
```

```
f4
Системный программист (C, Linux) с опытом Python, зарплата 182265 руб.
Веб-программист с опытом Python, зарплата 195658 руб.
1С программист с опытом Python, зарплата 138879 руб.
Инженер-программист ККТ с опытом Python, зарплата 180170 руб.
инженер - программист с опытом Python, зарплата 135724 руб.
Инженер-программист (Клинский филиал) с опытом Python, зарплата 128555 руб.
Инженер-программист (Орехово-Зуевский филиал) с опытом Python, зарплата 126879 руб.
Ведущий программист с опытом Python, зарплата 172130 руб.
Инженер - программист АСУ ТП с опытом Python, зарплата 113696 руб.
инженер-программист с опытом Python, зарплата 176943 руб.
Инженер-электронщик (программист АСУ ТП) с опытом Python, зарплата 189151 руб.
Старший программист с опытом Python, зарплата 192530 руб.
Web-программист с опытом Python, зарплата 172565 руб.
Веб - программист (PHP, JS) / Web разработчик с опытом Python, зарплата 171463 руб.
Инженер-программист 1 категории с опытом Python, зарплата 148682 руб.
Ведущий инженер-программист с опытом Python, зарплата 194084 руб.
Инженер-программист САПОУ (java) с опытом Python, зарплата 112349 руб.
Помощник веб-программиста с опытом Python, зарплата 175340 руб.
педагог программист с опытом Python, зарплата 199581 руб.
Инженер-программист ПЛИС с опытом Python, зарплата 102533 руб.
0.0937964916229248
```