Московский государственный технический университет им. Н.Э.Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю: _____ Согласовано: _____

"___"_____2016 г. "___"_____2016 г.

«Введение в python»

Отчет по лабораторной работе №4

(вид документа)

писчая бумага формата А4

(вид носителя)

_____

(количество листов)

Исполнитель: студент группы РТ5-51

_____ Попков В.Е.

"_____"_____ 2016 г.

Москва – 2016

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 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

Код:

```
import random
def field(f, *args):
    assert (len(args) > 0)

    # Необходимо реализовать генератор
    for i in f:
            v = {}
            if (len(args)==1):
                if i[args[0]]!=None:
                    yield i[args[0]]
            else:
                for ar in args:
                    if i[ar] !=None:
                        v[ar]=i[ar]
                yield v

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

Результат:

```
C:\Python35-32\python.exe C:/RRR/ex-lab4-master/ex_1.py
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стеллаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800}]
[4, 5, 2, 3, 4]

Process finished with exit code 0
```

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

---

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

Код:

```python
from collections import Counter
# Итератор для удаления дубликатов
class Unique(object):
    ignore_case = False
    i = -1
    buff = []

    def __init__(self, items, **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.ignore_case = False
        if (len(kwargs) > 0 and kwargs['ignore_case'] != None) and (type(kwargs['ignore_case']) == bool):
            self.ignore_case = kwargs['ignore_case']
            if(self.ignore_case):
                self.arr = list(Counter(map(lambda x: x.lower(),items)))
        else:
            self.arr = list(Counter(items))
    def __next__(self):
        while self.i < len(self.arr)-1:
            self.i += 1
            return self.arr[self.i]
        raise StopIteration()


    def __iter__(self):
        return self
```

Результат:

```python
#!/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, 7, 10)
d2 = ['f','F','fa','d']
print(list(Unique(data1)))

print(list(Unique(list(data2))))

print(list(Unique(d2, ignore_case=True)))
```

```
x_2
  C:\Python35-32\python.exe C:/RRR/ex-lab4-master/ex_2.py
  [1, 2]
  [1, 2, 4, 5, 6, 7]
  ['d', 'fa', 'f']

  Process finished with exit code 0
```

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]

Код и результат:

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

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3
print(sorted(data,key=lambda m:abs(m)))
```

```
ex_3
ebugger  Console
  C:\Python35-32\python.exe "C:\Program Files (x86)\JetBrains\PyCharm 2016.2.3\helpers
  pydev debugger: process 2668 is connecting

  Connected to pydev debugger (build 162.1967.10)
  [0, 1, -1, 4, -4, -30, 100, -100, 123]

  Process finished with exit code 0
```

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
```

Декоратор должен располагаться в librip/ decorators .py

Код декоратора

```python
def print_result(decor):
    def obertka(*args):
        print(decor.__name__)
        if len(args) > 0:
            d = decor(args[0])
        else:
            d = decor()
        if (type(d) == list):
            print("\n".join([str(x) for x in d]))
        elif (type(d) == dict):
            print("\n".join([str(x) + " = " + str(d[x]) for x in d]))
        else:
            print(d)
        return(d)
    return obertka
```

Выполнение ex_4.py

```python
test_1()
test_2()
test_3()
test_4()
```

```
_4
C:\Python35-32\python.exe C:/RRR/ex-lab4-master/ex_4.py
test_1
1
test_2
iu
test_3
b = 2
a = 1
test_4
1
2

Process finished with exit code 0
```

Задача 5 (ex_5.py)

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

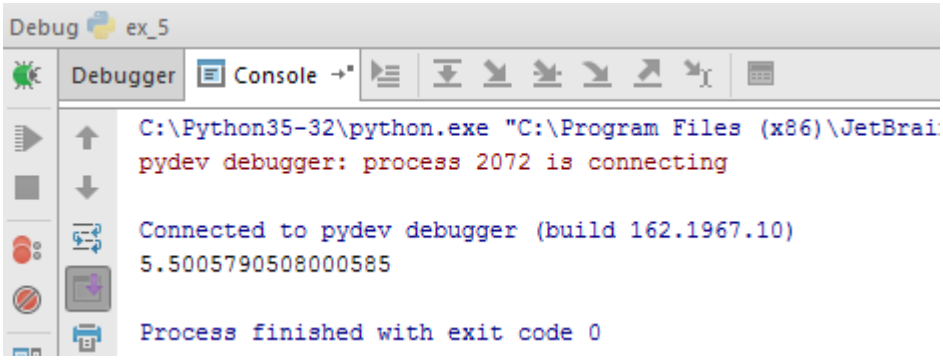
```
with timer():
        sleep(5.5)
```

После завершения блока должно вывестись в консоль примерно 5.5

Код менеджера

```python
import time
class timer:
    t=0
    def __enter__(self):
        self.t = time.clock()
    def __exit__(self, exp_type, exp_value, traceback):
        print(time.clock() - self.t)
# После завершения блока должно вывестись в консоль примерно 5.5
```

Результат:



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.

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

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*

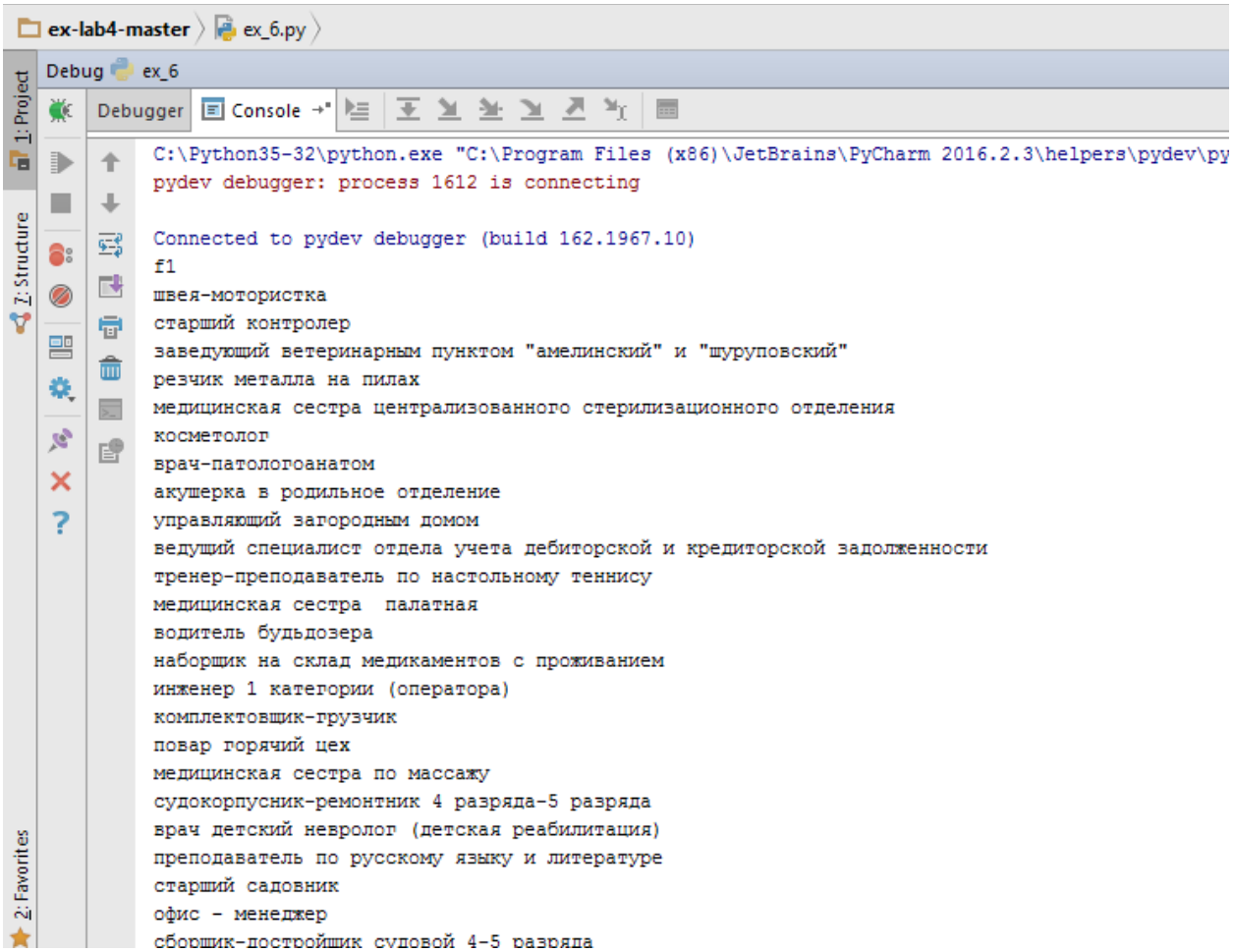
4. Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: *Программист C# с опытом Python, зарплата 137287 руб.*

Код:

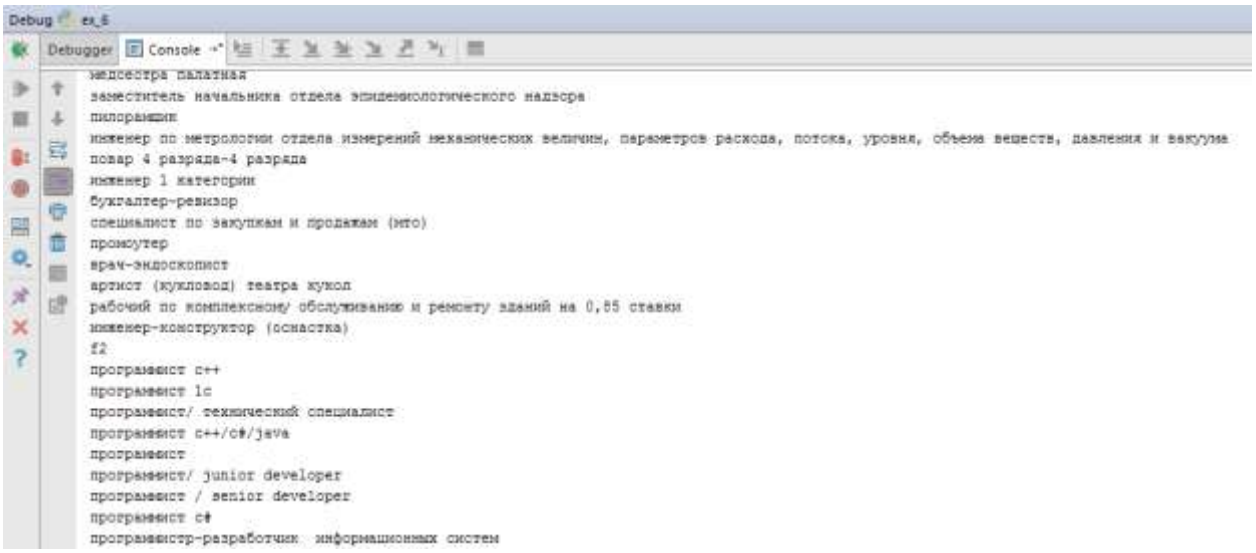
```
import json
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gen import field, gen_random
from librip.iterators import Unique as unique
...
path = 'data_light.json'
...
with open(path, encoding="utf8") as f:
    data = json.load(f)
...
@print_result
def f1(arg):
    return list(unique(list(field(arg, "job-name")), ignore_case=True))
@print_result
def f2(arg):
    return list(filter(lambda x: str(x).startswith('программист'), arg))
@print_result
def f3(arg):
    return list(map(lambda x: "{} с опытом Python".format(x), arg))
@print_result
def f4(arg):
    return list(map(lambda x: "{}, зарплата {} руб.".format(*x), zip(arg, gen_random(100000, 200000, len(arg)))))
with timer():
    f4(f3(f2(f1(data))))
```

Результат

Начало блока f1



конец f1 и блок f2



блоки f3, f4 и время

```
f3
программист c++ с опытом Python
программист 1с с опытом Python
программист/ технический специалист с опытом Python
программист c++/c#/java с опытом Python
программист с опытом Python
программист/ junior developer с опытом Python
программист / senior developer с опытом Python
программист c# с опытом Python
программистр-разработчик  информационных систем с опытом Python
f4
программист c++ с опытом Python, зарплата 128694 руб.
программист 1с с опытом Python, зарплата 105681 руб.
программист/ технический специалист с опытом Python, зарплата 179846 руб.
программист c++/c#/java с опытом Python, зарплата 197399 руб.
программист с опытом Python, зарплата 141832 руб.
программист/ junior developer с опытом Python, зарплата 124410 руб.
программист / senior developer с опытом Python, зарплата 150256 руб.
программист c# с опытом Python, зарплата 107778 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 113101 руб.
0.1256107128298034

Process finished with exit code 0
```