Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



## Отчет по Лабораторной работе 4

**По курсу**

**“Разработка интернет-приложений”**

Преподаватель Гапанюк Ю.Е. Студентка группы ИУ5-52 Петрова И.В.

Москва, 2017г.

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

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.

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.

1. 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.

1. 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.
2. 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.

# Исходный код

## gens.py

import random

def field(lst, \*args): assert len(args) > 0 for i in lst:

if len(args) == 1:

if args[0] in i: yield i[args[0]]

elif len(args) > 1: result = {} for k in args:

if k in i:

result[k] = i[k] if result != {}:

yield result

def gen\_random(start, stop, number): for i in range(number):

yield random.randrange(start, stop)

## iterators.py

class Unique:

data = [] used = [] i = 0

ignore\_case = False

def init (self, data, ignore\_case=False): self.data = [x for x in data] self.ignore\_case = ignore\_case

def next (self):

if self.i >= len(self.data): raise StopIteration

if isinstance(self.data[self.i], str) and self.ignore\_case: if self.data[self.i].lower() in self.used:

while (self.data[self.i].lower()) in self.used: self.i += 1

if self.i >= len(self.data): raise StopIteration

self.used.append(self.data[self.i]) self.i += 1

return self.data[self.i - 1] else:

self.used.append(self.data[self.i].lower()) self.i += 1

return self.data[self.i - 1]

else:

if self.data[self.i] in self.used:

while (self.data[self.i]) in self.used: self.i += 1

if self.i >= len(self.data): raise StopIteration

self.used.append(self.data[self.i]) self.i += 1

return self.data[self.i - 1] else:

self.used.append(self.data[self.i]) self.i += 1

return self.data[self.i - 1]

def iter (self): return self

## decorators.py

def print\_result(func): def decorated(\*args):

print(func. name ) result = func(\*args)

if isinstance(result, dict): for k, v in result.items():

print(str(k) + '=' + str(v)) elif isinstance(result, list):

for a in result: print(a)

else:

print(result) return result

return decorated

## ctxmngrs.py

import time

class timer:

def enter (self): self.t = time.clock()

def exit (self, exc\_type, exc\_val, exc\_tb): print(time.clock()-self.t)

## ex\_1.py

#!/usr/bin/env python3

from librip.gens import field, gen\_random

goods = [

{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},

{'title': 'Стелаж', 'price': 7000, 'color': 'white'},

{'price': 800}

]

# Реализация задания 1

for k in field(goods, 'title', 'color'): print(k)

lst = []

for x in gen\_random(4, 9, 6): lst.append(x)

print(lst)

lst2 = [x for x in gen\_random(4, 9, 6)] print(lst2)

## ex\_2.py

#!/usr/bin/env python3

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, 6, 10)

data3 = ["brauberg", "Brauberg", "PaperMate", "papermate", "Faber- Castell", "faber-castell"]

# Реализация задания 2

print([x for x in Unique(data3, False)]) print([x for x in Unique(data1, False)]) print([x for x in Unique(data2, False)])

## ex\_3.py

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

# Реализация задания 3

print(sorted(data, key=abs))

## ex\_4.py

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

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

path = "data\_light\_cp1251.json"

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

@print\_result def f1(arg):

return sorted(unique([x["job-name"] for x in arg], ignore\_case=True), key=lambda x: x.lower())

@print\_result def f2(arg):

return list(filter(lambda x: x.lower()[:11] == 'программист', arg))

@print\_result def f3(arg):

return list(map(lambda x: x + ' с опытом Python', arg))

@print\_result def f4(arg):

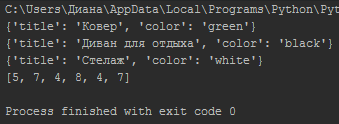
return [k[0] + ', зарплата ' + str(k[1]) for k in list(zip(arg, [x for x in gen\_random(100000, 200000, len(arg))]))]

with timer():

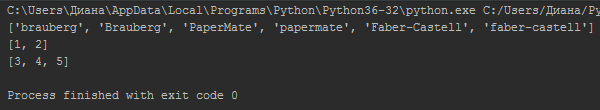
f4(f3(f2(f1(data))))

# Скриншоты с результатами выполнения

ex\_1.py



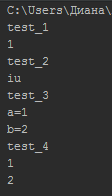
ex\_2.py



ex\_3.py



ex\_4.py



ex\_5.py



ex\_6.py f2

Программист

Программист / Senior Developer Программист 1С

Программист 1С программист 1С Программист C# Программист С++

Программист С++/С#/Java

Программист/ Junior Developer

Программист/ технический специалист

Программистр-разработчик информационных систем f3

Программист с опытом Python

Программист / Senior Developer с опытом Python Программист 1С с опытом Python

Программист 1С с опытом Python программист 1С с опытом Python Программист C# с опытом Python Программист С++ с опытом Python

Программист С++/С#/Java с опытом Python

Программист/ Junior Developer с опытом Python

Программист/ технический специалист с опытом Python

Программистр-разработчик информационных систем с опытом Python f4

Программист с опытом Python, зарплата 125345

Программист / Senior Developer с опытом Python, зарплата 134361

Программист 1С с опытом Python, зарплата 119257 Программист 1С с опытом Python, зарплата 142451 программист 1С с опытом Python, зарплата 133809 Программист C# с опытом Python, зарплата 104052 Программист С++ с опытом Python, зарплата 197995

Программист С++/С#/Java с опытом Python, зарплата 138702

Программист/ Junior Developer с опытом Python, зарплата 177223

Программист/ технический специалист с опытом Python, зарплата 160491

Программистр-разработчик информационных систем с опытом Python, зарплата 117517

0.12734693076163806

Process finished with exit code 0