Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторной работе №4

«Python. Функциональные возможности»

Выполнил:
студент группы ИУ5-53
Гатауллин И. И.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2017 г.

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
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.
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 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
@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). 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.

Код программы

ctxmngrs.py

```
# Здесь необходимо реализовать
# контекстный менеджер timer
# Он не принимает аргументов, после выполнения блока он должен вывести время выполнения в секундах
# Пример использования
# with timer():
#   sleep(5.5)
#
# После завершения блока должно вывестись в консоль примерно 5.5

import contextlib
import time

@contextlib.contextmanager
def timer():
    t = time.time()
    yield #вып функция которая работает с контекст мессенджером (sleep 5.5)
    print(time.time() - t)
```

decorators.py

```
def print_result(func_to_decorate):
    def decorated_func(*args, **kwargs):
        print(func_to_decorate.__name__)

        func_to_decorate_res = func_to_decorate(*args, **kwargs)
        if type(func_to_decorate_res) == list:
            for i in func_to_decorate_res:
```

```
                print(i)
        elif type(func_to_decorate_res) == dict:
            for i in func_to_decorate_res.keys():
                print('{} = {}'.format(i, func_to_decorate_res[i]))
        else:
            print(func_to_decorate_res)
        return func_to_decorate_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):
    # Необходимо реализовать генератор
    assert len(args) > 0 #он проверяет длину массива аргументов (тайтл,
прайс)

    if len(args) == 1: #выдает только значение по ключу
        for item in items:
            if args[0] in item and item[args[0]]:
                yield item[args[0]]  #вывод
    else:
        for item in items:
            result = {}
            for arg in args:
                if arg in item and item[arg]:
                    result[arg] = item[arg]
            if (result): #если словарь result не пустой в результате то мы
вернем
                yield result



# 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 count in range(num_count):
        yield random.randint(begin, end)
    # Необходимо реализовать генератор
```

iterators.py

```
# Итератор для удаления дубликатов


class Unique(object):
    def __init__(self, items, **kwargs): #**kwargs - произвольное количество
именованных аргументов
        # 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 = kwargs.get('ignore_case', False) #пытаемся достать
ignore_case, если ничего нет то False (можешь почитать методы словарей)

        self.items = self._uniq_list(items)
        self.index = 0
        self.length = len(self.items)

    def _uniq_list(self, lst):
        cheker = {}
        result = []
        if self.ignore_case: #если тру, можно написать ==True
            for i in lst:
                if str(i).lower() not in cheker:
                    cheker[str(i).lower()] = 'exist'
                    result.append(i)
        else: #если ==False
            for i in lst:
                if i not in cheker:
                    cheker[i] = 0
                    result.append(i)

        return result

    def __next__(self):
        if self.index == self.length:
            raise StopIteration
        self.index += 1
        return self.items[self.index - 1] #нам нужно вернуть 0-ой элемент


    def __iter__(self):
        return self
```

ex_1.py

```
#!/usr/bin/env python3
import librip.gens as gens


def print_list(lst):
    print(' ,'.join(map(str, lst))) # map -функ которая применяет функцию
str(к строке) к lst, join (соединяет только строки) ставит запятые и выводит


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'},
    {'title': None, 'price': None, 'color': None},
    {'color': 'white'},
]

res_list = []

for item in gens.field(goods, 'title', 'price'): # генератор возвращает в
item нужный элемент
    res_list.append(item)
print_list(res_list) #вот тут скобочик появятся, я от них избавлялся _list
```

```
print_list([i for i in gens.gen_random(1, 3, 5)])
```

ex_2.py

```
#!/usr/bin/env python3
from librip.gens import gen_random
from librip.iterators import Unique

data1 = [1, 1, 2, 1, 1, 4, 5, 2, 2, 2]
data2 = gen_random(1, 3, 10)

un1 = Unique(data1)
un2 = Unique(data2)
print(' ,'.join(map(str, [i for i in un1])))
print(' ,'.join(map(str, [i for i in un2])))

# Реализация задания 2
```

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))) #key-опциональный параметр, в
который передаешь функ., которая возвращает объект сортировки
#lambda x аналог def ###(x): return abs(x)
```

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

path = 'data_light_cp1251.json'

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

with open(path) 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 #декорирование
def f1(arg):
    uni = Unique([i for i in field(arg, 'job-name')], ignore_case=True)
    return sorted([i for i in uni])


@print_result
def f2(arg):
    return list(filter(lambda x: x.startswith('программист') or
x.startswith('Программист'), arg))


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


@print_result
def f4(arg):
    salary = [x for x in gen_random(100000, 200000, len(arg))]
    return [ '{} , зарплата {} руб'.format(job, sal) for job, sal in zip(arg,
salary)] #лист из пар


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

Результаты программы

f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
[химик-эксперт
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г. Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында
Агент торговый
Агроном-полевод
Администратор
Администратор (удаленно)
Администратор Active Directory
Администратор в парикмахерский салон
Администратор зала (предприятий общественного питания)
Администратор кофейни
Администратор на ресепшен
Администратор на телефоне
Администратор по информационной безопасности
Администратор ресторана
Администратор сайта
Администратор ярмарок выходного дня
Администратор-кассир
Аккомпаниатор на 0,5 ст.
Акушерка
Акушерка Лысогорская врачебная амбулатория
Акушерка ФАП
Акушерка женской консультации
Акушерка, АО

[illegible]
Бригадир в животноводстве
Бригадир технического обслуживания газоиспользующего оборудования
Бригадир, производитель работ
Брокер коммерческой недвижимости
Брошировщик
Бухгалтер
Бухгалтер (по заработной плате)
Бухгалтер 2 категории
Бухгалтер на группу "Обработка первичной документации"
Бухгалтер по МТП и ГСМ
Бухгалтер по ведению первичной документации
Бухгалтер по заработной плате
Бухгалтер по начислению заработной платы
Бухгалтер по расчету заработной платы
Бухгалтер по расчету калькуляции
Бухгалтер, ведущий
Бухгалтер,Ведущий бухгалтер
Бухгалтер-кассир 1 категории
Бухгалтер-материалист
Бухгалтер-ревизор
Бухгалтер-экономист
ВОСПИТАТЕЛЬ В ЧАСТНЫЙ ДЕТСКИЙ САД
ВРАЧ АНЕСТЕЗИОЛОГ РЕАНИМАТОЛОГ
Вальцовщик
Вахта
Вахтер
Вахтёр
Веб - программист (PHP, JS) / Web разработчик
Веб-программист
Ведущий библиотекарь отдела книгохранения
Ведущий бухгалтер
Ведущий инженер
Ведущий инженер Лаборатории испытаний и экспертизы
Ведущий инженер Отдела оценки компетентности испытательных лабораторий и физико-химических измерений (Сергиево-Посадский филиал)
Ведущий инженер Отдела эксплуатации (Сергиево-Посадский филиал)
Ведущий инженер в сметно-договорной отдел
Ведущий инженер внедрения КИП и А
Ведущий инженер конструктор ОКБ
Ведущий инженер по метрологии Лаборатории аттестации методик выполнения измерений (Сергиево-Посадский филиал)
Ведущий инженер по метрологии Отдела испытаний, поверки и калибровки СИ механических и геометрических величин (Сергиево-Посадский филиал)
Ведущий инженер по метрологии Отдела испытаний, поверки и калибровки СИ оптико-физических, теплотехнических, температурных величин, давления и вакуума, физико-химического состава и свойств веществ (Сергиево-Посадский филиал)
Ведущий инженер по метрологии отдела радиоакустических измерений
Ведущий инженер-конструктор
Ведущий инженер-программист
Ведущий инженер-технолог
Ведущий инженер-технолог (химик)

Слесарь-электромонтажник
Сменный мастер
Сменный техник-технолог
Сметчик
Снабженец
Сотрудник на подработку в офис
Социальный педагог
Социальный работник
Спасатель 2 категории
Специалист
Специалист 1 категории (класса)
Специалист 1 разряда финансового отдела
Специалист 2 категории отдела организации конкурсных процедур
Специалист в сфере закупок
Специалист группы технической поддержки
Специалист договорного отдела
Специалист коммерческого отдела
Специалист контакт-центра
Специалист мониторинга и контроля розничных продаж (Центральный аппарат)
Специалист отдела автоматизации
Специалист отдела аналитики и мониторинга
Специалист отдела информатизации и информационной безопасности, место работы г. Новый Уренгой
Специалист отдела испытаний продукции
Специалист отдела конкурентных процедур
Специалист отдела организации продаж и обслуживания
Специалист отдела оценки кредитоспособности
Специалист отдела продаж государственным заказчикам
Специалист отдела продаж спецодежды
Специалист отдела сопутствующих товаров и услуг(РО Башкирия)
Специалист планово-экономического сектора
Специалист по автоматизации бизнес-процессов
Специалист по влагозащите
Специалист по водоподготовке и очистным сооружениям
Специалист по выдаче займов
Специалист по выдаче займов сельским хозяйствам
Специалист по государственным закупкам
Специалист по закупкам
Специалист по закупкам и продажам (МТО)
Специалист по испытаниям на электромагнитную совместимость аппаратуры бортовых космических систем
Специалист по кадрам
Специалист по кадрам (Можайский филиал)
Специалист по кадровому делопроизводству - секретарь
Специалист по кредитным услугам г. Новый Оскол
Специалист по кредитным услугам пгт Томаровка
Специалист по монтажным работам
Специалист по мультимедиа
Специалист по неразрушающему контролю (НК)
Специалист по обработке жалоб
Специалист по обслуживанию контрольно-кассовой техники

физик-эксперт
формовщик
фрезеровщик
фтизиатрия
химик
художник-постановщик
швея - мотористка
шиномонтаж
шлифовщик 5 разряда
шлифовщик механического цеха
эколог
электромонтер -линейщик по монтажу воздушных линий высокого напряжения и контактной сети
электромонтер по испытаниям и измерениям 4-6 разряд
электромонтер станционного телевизионного оборудования
электросварщик
энтомолог
юрисконсульт 2 категории
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/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 C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/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 191117 руб
Программист / Senior Developer с опытом Python , зарплата 193791 руб
Программист 1С с опытом Python , зарплата 186235 руб
Программист C# с опытом Python , зарплата 109890 руб
Программист C++ с опытом Python , зарплата 144433 руб
Программист C++/C#/Java с опытом Python , зарплата 112253 руб
Программист/ Junior Developer с опытом Python , зарплата 169760 руб
Программист/ технический специалист с опытом Python , зарплата 151773 руб
Программистр-разработчик информационных систем с опытом Python , зарплата 183736 руб
0.04679989814758301