Московский государственный технический университет им. Н.Э.Баумана

Кафедра «Системы обработки информации и управления»

Разработка интернет приложений

Отчет по Лабораторной работе №4.

Исполнитель: студент группы РТ5-51

Бгатцев А. В.

Москва – 2016

Задача 1 (ex_1.py)

Необходимо реализовать генераторы fieldи gen_random
Генератор fieldпоследовательно выдает значения ключей словарей массива

Исходный код:

ex_1.py

```
#!/usr/bin/env python3
from librip.genimport field
from librip.genimport gen_random

goods = [
    {'title': 'Ковер', 'price': None, 'color': 'green'},
    {'title': 'Дивандляотдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
    {'title': 'Вешалкадляодежды', 'price': 800, 'color': 'white'}
]

print([x for x in field(goods, 'price')])
print([x for x in field(goods, 'title', 'price')])
print([x for x in gen_random(1, 10, 7)])
# Реализациязадания 1
```

```
gen.py
import random
# Генераторвычлененияполейизмассивасловарей
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Дивандляотдыха', 'price': 5300, 'color': 'black'}
# ]
# field(goods, 'title') долженвыдавать 'Ковер', 'Дивандляотдыха'
# field(goods, 'title', 'price') долженвыдавать {'title': 'Ковер', 'price':
2000}, {'title': 'Дивандляотдыха', 'price': 5300}

deffield(items, *args):
assert len(args) >0
# Необходимореализоватьгенератор
for item in items:
for key in args:
try:
arr = item[key]
yield arr
except Exception:
yield None

# Генераторспискаслучайныхчисел
# Пример:
# gen_random(1, 3, 5) долженвыдатьпримерно 2, 2, 3, 2, 1
# Hint: реализациязанимает 2 строки
defgen_random(begin, end, num_count):
for i in range(num_count):
arr = random.randint(begin, end)
yield arr
# Необходимореализоватьгенератор
```

Результат работы:

[2000, 5300, 7000, 800]

['Ковер', 2000, 'Диван для отдыха', 5300, 'Стелаж', 7000, 'Вешалка для одежды', 800]

[6, 10, 4, 3, 1, 5, 2]

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

Исходный код:

Ex_2.py

```
#!/usr/bin/env python3
from librip.genimport gen_random
from librip.iteratorsimport Unique

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

# Реализациязадания 2

print(list(Unique(data1)))
print(list(Unique(data2)))
data = ['a', 'A', 'b', 'B']
print(list(Unique(data)))
data = ['a', 'A', 'b', 'B']
print(list(Unique(data, ignore_case=True)))
```

Iterators.py

```
# Итератордляудалениядубликатов
class Unique(object):

def __init__(self, items, **kwargs):
# Нужнореализоватьконструктор
        # Вкачествеключевогоаргумента, конструктордолженприниматьbool-
параметрignore_case,
        #
взависимостиотзначениякоторогобудутсчитатьсяодинаковыестрокивразномрегистре
        # Например: ignore_case = True, АбвиАБВразныестроки
        #           ignore_case = False, АбвиАБВодинаковыестроки,
однаизнихудалится
        # По-умолчаниюignore_case = False
if 'ignore_case' in kwargs.keys():
self.Ignore_case = kwargs['ignore_case']
else:
self.Ignore_case = False
self.Items = list(items)
self.Passed = set()

def __next__(self):
# Нужнореализовать __next__
while True:
if self.Index == len(self.Items) - 1:
raise StopIteration
self.Index += 1
a = str(self.Items[self.Index])
if self.Ignore_case:
                b = a
else:
```

```
                b = a.lower()
if b not in self.Passed:
self.Passed.add(b)
return a

def __iter__(self):
self.Index = -1
return self
```

Результат работы:

['1', '2']

['2', '1', '3']

['a', 'b']

['a', 'A', 'b', 'B']

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

Исходный код:

Ex_3.py

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

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

Результат работы:

[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

Исходный код:

Ex_4.py

```
from librip.decoratorsimport print_result

# Необходимовернореализоватьprint_result
```

```
# изаданиебудетвыполнено

@print_result
deftest_1():
return 1

@print_result
deftest_2():
return 'iu'

@print_result
deftest_3():
return {'a': 1, 'b': 2}


@print result
deftest_4():
return [1, 2]

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

decorators.py

```
# Здесь необходимо реализовать декоратор, print_result который принимает на
вход функцию,
# вызывает её, печатает в консоль имя функции, печатает результат и
возвращает 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

defprint_result(func):
defdecorated(*args):
print(func.__name__)
if len(args) >0:
            res = func(args[0])
else:
            res = func()
if type(res) is list:
print("\n".join(map(str, res)))
eliftype(res) is dict:
print("\n".join(["{}={}".format(str(x), str(res[x])) for x in res]))
else:
print(res)
return res
return decorated
```

Результат работы:

```
test_1
1
test_2
iu
test_3
a=1
b=2
test_4
1
2
```

Задача 5 (ex_5.py)

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран
После завершения блока должно вывестись в консоль примерно 5.5

Исходный код:

Ex_5.py

```
fromtime import sleep
from librip.ctxmngrsimport timer

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

ctxmngrs.py

```
# Здесьнеобходимореализовать
# контекстныйменеджер timer
# Оннепринимаетаргументов,
послевыполненияблокаондолженвывестивремявыполнениявсекундах

import time


class timer:
def __enter__(self):
self.start = time.time()

def __exit__(self, exp_type, exp_value, traceback):
print("Времяисполнения: " + str(time.time() - self.start))
```

Результат работы:

Время исполнения: 5.50780987739563

Задача 6 (ex_6.py)

В ex_6.py дано 4 функции. В конце каждая функция вызывается, принимая на вход результат работы
предыдущей. За счет декоратора @print_resultпечатается 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).
4. Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей иприсоединить её к названию специальности.

Исходный код:

Ex_6.py

```
#!/usr/bin/env python3
import json
import sys
from librip.ctxmngrsimport timer
from librip.decoratorsimport print_result
from librip.genimport field, gen_random
from librip.iteratorsimport Unique as unique

# Здесьнеобходимовпеременную path получить
# путьдофайла, которыйбылпереданпризапуске

#assert len(sys.argv) > 0
#path = sys.argv[0]
path = "data_light.json"

with open(path, encoding="utf8") as f:
    data = json.load(f)


# Далеенеобходимореализоватьвсефункциипозаданию, заменив `raise
NotImplemented`
# Важно!
# Функциис 1 по 3 дожныбытьреализованыводнустроку
# Вреализациифункции 4 можетбытьдо 3 строк
# Приэтомстрокидолжныбытьнедлиннее 80 символов

@print_result
deff1(arg):
return list(unique(list(field(arg, "job-name")), ignore_case=True))


@print_result
deff2(arg):
return list(filter(lambda _: "Программист" in _, arg))


@print_result
deff3(arg):
return list(map(lambda x: x + " сопытом Python", arg))


@print_result
deff4(arg):
return list(map(lambda x: "{}, зарплата {} руб.".format(x[0], x[1]),
zip(arg, gen_random(100000, 200000, len(arg)))))

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

Результат работы:

```
…
Программист C++
Программист/ JuniorDeveloper
Программист / SeniorDeveloper
Программист/ технический специалист
Программист C#
```

f3
Программист с опытом Python
Программист C++/C#/Java с опытом Python
Программист 1C с опытом Python
Программистр-разработчик информационных систем с опытом Python
Программист C++ с опытом Python
Программист/ JuniorDeveloper с опытом Python
Программист / Senior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист C# с опытом Python
f4
Программист с опытом Python, зарплата 128546 руб.
Программист C++/C#/Java с опытом Python, зарплата 119246 руб.
Программист 1C с опытом Python, зарплата 177771 руб.
Программистр-разработчик информационных систем с опытом Python, зарплата 120525 руб.
Программист C++ с опытом Python, зарплата 102029 руб.
Программист/ JuniorDeveloper с опытом Python, зарплата 154593 руб.
Программист / SeniorDeveloper с опытом Python, зарплата 111524 руб.
Программист/ технический специалист с опытом Python, зарплата 175802 руб.
Программист C# с опытом Python, зарплата 118795 руб.
Времяисполнения: 0.031199932098388672