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

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Разработка интернет приложений»

Отчет по лабораторной работе №4

«Python. Функциональные возможности»

Выполнил:
студент группы ИУ5-52

Злобина С.В.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2017 г.

1 Задание

1. (ex_1.py) Необходимо реализовать генераторы field и gen_random. Генератор field последовательно выдает значения ключей словарей массива.
   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.

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.

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.

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.

5. (ex_5.py) Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран.

6. 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. 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.

2 Листинг

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):
  assert len(args) > 0
  if len(args) == 1:
    for item in items:
      try:
        yield item[args[0]]
      except KeyError:
        continue
  else:
    dict = {}
    for item in items:
      for arg in args:
        try:
          dict[arg] = item[arg]
        except KeyError:
          continue
      if not dict:
        continue
      else:
        yield dict
# 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 _ in range(num_count):
    yield random.randint(begin, end)
```

ex_1.py

```
#!/usr/bin/env python3
from librip.gens import field
from librip.gens import gen_random
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'}
]
goods2 = [
   {'color': 'green'},
   {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
   {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
   {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
]
for i in field(goods2, 'price'):
   print(i, end=', ')
print()
for i in gen_random(4, 7, 5):
   print(i, end=', ')
```

iterators.py

 # Итератор для удаления дубликатов

```
class Unique(object):
   def __init__(self, items, **kwargs):
      self._values_set = set()
      if isinstance(items, list):
         self._items_gen = (elem for elem in items)
      else:
         self._items_gen = items
      self.ignore_case = kwargs.get('ignore_case', False)
   def __next__(self):
      for item in self._items_gen:
         is_str = isinstance(item, str)
         if (not is_str) and (item not in self._values_set):
            self._values_set.add(item)
            return item
         elif is_str:
            if self.ignore_case and (item.lower() not in self._values_set):
               self._values_set.add(item.lower())
               return item
            if (not self.ignore_case) and (item not in self._values_set):
               self._values_set.add(item)
               return item
      else:
         raise StopIteration()
   def __iter__(self):
      return self
```

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, 3, 10)
data3 = ['a', 'A', 'B', 'b']
data4 = ['a', 'A', 2, 6, 2, 'B', 'b']
for i in Unique(data1):  # со списком из чисел
  print(i, end=', ')
print()
for i in Unique(data2):  # с генератором чисел
  print(i, end=', ')
print()
for i in Unique(data3):  # со списком строк, не игнорируя регистр
  print(i, end=', ')
print()
for i in Unique(data3, ignore_case=True):  # со списком строк, игнорируя
регистр
  print(i, end=', ')
print()
for i in Unique(data4, ignore_case=True):  # со смешанным списком строк и
чисел, не игнорируя регистр
  print(i, end=', ')
```

ex_3.py

```
#!/usr/bin/env python3
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
print(sorted(data, key=lambda x: abs(x)))
```

decorators.py

```
def print_result(function_to_decorate):
  def decorated_by_print_result(*args):
    print(function_to_decorate.__name__)
    returned_value = function_to_decorate(*args)
    if isinstance(returned_value, list):
      print('\n'.join(map(str, returned_value)))
    elif isinstance(returned_value, dict):
      print('\n'.join(map(lambda k: str(k) + ' = ' + str(returned_value[k]),
returned_value)))
    else:
      print(returned_value)
    return returned_value
  return decorated_by_print_result
```

ex_4.py

```
from librip.decorators import print_result
# Необходимо верно реализовать print_result
# и задание будет выполнено
@print_result
def decor_test1():
  return 1
@print_result
def decor_test2():
  return 'iu'
@print_result
def decor_test3():
  return {'a': 1, 'b': 2}
@print_result
def decor_test4():
  return [1, 2]
decor_test1()
decor_test2()
decor_test3()
decor_test4()
```

ctxnmgrs.py

```
import time
class timer:
  def __enter__(self):
    self._start_time = time.time()
  def __exit__(self, exc_type, exc_val, exc_tb):
    print('Elapsed time: {} sec'.format(time.time() - self._start_time))
# 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
```

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
import re
path = sys.argv[1]
# 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
def f1(arg):
  return list(unique(field(arg, 'job-name'), ignore_case=True))
@print_result
def f2(arg):
  return list(filter(lambda x: re.match("^[п,П]рограммист", x) is not None,
arg))
@print_result
def f3(arg):
  return list(map(lambda x: x + " с опытом Python", arg))
@print_result
def f4(arg):
  salary_list = list(zip(arg, list(gen_random(100000, 200000, len(arg)))))
  return list(map(lambda x: x[0] + ", зарплата " + str(x[1]) + " руб",
salary_list))
with timer():
  f4(f3(f2(f1(data))))
```

Результат

ex_1.py

```
{'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стелаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800},
4, 4, 5, 5, 4,
```

ex_2.py

```
1, 2,
2, 1, 3,
a, A, B, b,
a, B,
a, 2, 6, B,
```

ex_3.py

```
[0, 1, -1, 4, -4, -30, 100, -100, 123]
```

ex_4.py

```
decor_test1
1
decor_test2
iu
decor_test3
a = 1
b = 2
decor_test4
1
2
```

ex_5.py

```
Elapsed time: 5.50490665435791 sec
```

ex_6.py

```
f1
1C 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
```

```
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 108463 руб
Программист / Senior Developer с опытом Python, зарплата 101386 руб
Программист 1С с опытом Python, зарплата 132603 руб
Программист C# с опытом Python, зарплата 119395 руб
Программист C++ с опытом Python, зарплата 150701 руб
Программист C++/C#/Java с опытом Python, зарплата 154381 руб
Программист/ Junior Developer с опытом Python, зарплата 192846 руб
Программист/ технический специалист с опытом Python, зарплата 156599 руб
Программистр-разработчик  информационных систем с опытом Python, зарплата 126145 руб
Elapsed time: 0.026276350021362305 sec
```