Московский Государственный Технический Университет им. Н.Э. Баумана

Разработка интернет-приложений

Отчёт по лабораторной работе №4

«Функциональные возможности»

Выполнил:

студент группы ИУ5-54

Ваняшкин Юрий

1. Цель работы

Задача 1 (ex_1.py)
Необходимо реализовать генераторы field и gen_random
Генератор field последовательно выдает значения ключей словарей массива.
Генератор gen_random последовательно выдает заданное количество случайных чисел в заданном диапазоне

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 визвращаемые 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.
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).
4. Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Используйте zip для обработки пары специальность — зарплата.
5.

2. Листинг программы

Модуль `ctxmngrs.py`

```
class timer:
time = 0
    def __enter__(self):
        self.time = time.time()
    def __exit__(self, exc_type, exc_val, exc_tb):
        print(time.time() - self.time)
        pass
```

Модуль `decorators.py`

```
def print_result(func):
    result = None
    def wrapper(*args, **kwargs):
        print(func.__name__)
        result = func(*args, **kwargs)
        if isinstance(result, list):
            for x in result:
                print(x)
        elif isinstance(result, dict):
            for key in result:
                print(key, " = ", result[key])
        else:
            print(result)
        return result
    return wrapper
```

Модуль `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 x in items:
            a = x.get(args[0])
            if a is not None:
                yield a
    else:
        for x in items:
            result = {}
            for i in args:
                a = x.get(i)
                if a is not None:
                    result.update({i: a})
            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):
    pass
    # Необходимо реализовать генератор
    for i in range(num_count):
        yield random.randint(begin, end)
```

Модуль `iterators.py`

```
class
Unique(object):
   _lst = []
   def __init__(self, items, ignore_case):
       # 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
       assert len(items) > 0
       self._lst = list(items)
       self.ignore_case = ignore_case
       self.returned = list()
   def __next__(self):
       for x in self._lst:
           if (self.ignore_case & isinstance(x, str)):
               if x.lower() not in self.returned:
                   self.returned.append(x.lower())
                   return x
           else:
               if x not in self.returned:
                   self.returned.append(x)
                   return x
       raise StopIteration()
   def __iter__(self):
       return self
```

Модуль `ex_1.py`

```
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'}
    ]
    a = field(goods, 'title', 'price', 'empty_key', 'color')
    for x in a:
        print(x)
    data2 = gen_random(1, 3, 10)
    for x in data2:
        print(x)
```

Модуль `ex_2.py`

```
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)
res = [x for x in Unique(data1, False)]
print(res)
res = [x for x in Unique(list(data2), False)]
print(res)
```

Модуль `ex_3.py`

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

Модуль `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()
```

Модуль `ex_5.py`

```
from time import sleep
from librip.ctxmngrs import timer
with timer():
    sleep(2)
```

Модуль `ex_6.py`

```
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.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
def f1(arg):
    return sorted((x for x in unique(list(field(data, "job-name")), True)))
@print_result
def f2(arg):
    return list(filter(lambda x: x[0:11] == "Программист", arg))
@print_result
def f3(arg):
    return list(map(lambda x: x + " с опытом Python", arg))
@print_result
def f4(arg):
    salaries = list(gen_random(100000, 200000, len(arg)))
    for name, salary in zip(arg, salaries):
        print(name, ", зарплата ", salary)
with timer():
    f4(f3(f2(f1(data))))
```