МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА

Факультет Информатика и системы управления

Кафедра Системы обработки информации и управления (ИУ5)

Разработка интернет-приложений

Отчет по лабораторной работе №3

Выполнил: Торжков Максим Сергеевич

Группа: ИУ5-51Б

Преподаватель: Гапанюк Юрий Евгеньевич

Дата: 28.10.19

Подпись:

Москва, 2020 г.

Описание задания:

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово аргументов.

- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

Задача 2 (файл gen_random.py)

Необходимо реализовать генератор gen_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона

Задача 3 (файл unique.py)

- Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.
- При реализации необходимо использовать конструкцию **kwargs.
- Итератор должен поддерживать работу как со списками, так и с генераторами.

- Итератор не должен модифицировать возвращаемые значения.

Задача 4 (файл sort.py)

Дан массив 1, содержащий положительные и отрицательные числа. Необходимо **одной строкой кода** вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
- Если функция вернула словарь (dict), то ключи и значения должны выводить в столбик через знак равенства.

Задача 6 (файл cm_timer.py)

Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран.

cm_timer_1 и cm_timer_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами.

Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- В файле data_light.json содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер cm_timer_1 выводит время работы цепочки функций.

- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
- 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.
- 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.

Текст программы и результаты:

Файл field.py:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'foo': 'bar'},
    {'title': None}
]

def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        for obj in items:
            if args[0] in obj and obj[args[0]] is not None:
                yield obj[args[0]]
    else:
        for obj in items:
            res = {}
            for prop in args:
                if prop in obj and obj[prop] is not None:
                    res[prop] = obj[prop]
            if len(res) > 0:
                yield res


f = field(goods, 'title', 'price')
#f = field(goods, 'color')
```

```
print(next(f))
print(next(f))
#print(next(f))
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/field.py
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}

Process finished with exit code 0
```

Файл gen_random.py:

```
import random

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randrange(begin, end + 1)

f = gen_random(10, 1, 212)

for i in f:
   print(i)
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/gen_random.py
160
208
167
174
90
151
137
67
125
155

Process finished with exit code 0
```

Файл unique.py:

```
# Итератор для удаления дубликатов

from lab_python_fp.gen_random import gen_random
import types

class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых удалится
        # По-умолчанию ignore_case = False
```

```
        self.index = 0
        self.data = items
        self.used_elements = set()
        for key in kwargs:
            if key == 'ignore_case':
                self.ignore_case = kwargs[key]
            else:
                self.ignore_case = False
        pass

    def __next__(self):
        # Нужно реализовать __next__
        while True:
            if(type(self.data) == types.GeneratorType):
                current = next(self.data)
            else:
                if self.index >= len(self.data):
                    raise StopIteration
                current = self.data[self.index]
                if (type(current) == str and self.ignore_case == True):
                    current = current.lower()
                self.index = self.index + 1
            if current not in self.used_elements:
                 # Добавление в множество производится
                 # с помощью метода add
                self.used_elements.add(current)
                return current
        pass

    def __iter__(self):
        return self

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
# data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
# data = gen_random(10, 1, 3)
for i in Unique(data, ignore_case=True):
    print(i)
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/unique.py
a
b

Process finished with exit code 0
```

Файл sort.py:

```
from operator import itemgetter

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

if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda n: abs(n), reverse=True)
    print(result_with_lambda)
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

Файл print_result.py:

```
# Здесь должна быть реализация декоратора

def print_result(func_to_decorate):
    def decorated_func(*args, **kwargs):
        print(func_to_decorate.__name__)
        arg = func_to_decorate(*args, **kwargs)
        if(type(arg) == dict):
            for key in arg:
                print('{} = {}'.format(key, arg[key]))
        elif (type(arg) == list):
            for i in arg:
                print('{}'.format(i))
        else:
            print(arg)
        return arg
    return decorated_func


@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'


@print_result
def test_3():
    return {'a': 1, 'b': 2}


@print_result
def test_4():
    return [1, 2]


if __name__ == '__main__':
    test_1()
    test_2()
    test_3()
    test_4()
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/print_result.py
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0
```

Файл cm_timer.py:

```
import time
from contextlib import contextmanager

class cm_timer_1:

    def __init__(self):
        self.before_time = 0
        self.after_time = 0

    def __enter__(self):
        self.before_time = time.perf_counter()
        # Должен возвращаться значимый объект
        # например, открытый файл
        return time.perf_counter()

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            self.after_time = time.perf_counter()
            print('time: {}'.format(round(self.after_time-self.before_time,
2)))

@contextmanager
def cm_timer_2():
    before_time = time.perf_counter()
    yield time.perf_counter()
    after_time = time.perf_counter()
    print('time: {}'.format(round(after_time - before_time,2)))

with cm_timer_1():
    time.sleep(2.5)

with cm_timer_2():
    time.sleep(2.5)
```

Результаты программы:

```
C:\Users\makst\PycharmProjects\untitled\venv\Scripts\python.exe C:/Users/makst/python/lab3/lab_python_fp/cm_timer.py
time: 2.5
time: 2.51

Process finished with exit code 0
```

Файл process_data.py:

```
import json
import sys
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.unique import Unique
from lab_python_fp.field import field
from lab_python_fp.gen_random import gen_random
# Сделаем другие необходимые импорты

# Необходимо в переменную path сохранить путь к файлу, который был передан
при запуске сценария

path = "data_light.json"
with open(path, encoding='utf-8') as f:
    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise
NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    return list(Unique(sorted(field(arg,'job-name'), key=str),
ignore_case=True))
        #raise NotImplemented


@print_result
def f2(arg):
    return list(filter(lambda x: 'программист' in x, arg))
    #raise NotImplemented


@print_result
def f3(arg):
    return list(map(lambda x: x + " с опытом Python", arg))
    #raise NotImplemented


@print_result
def f4(arg):
    return list(map(lambda x: x + ", зарплата " + str(*gen_random(1, 100000,
200000)) + " руб", arg))
    #raise NotImplemented


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Результаты программы:

```
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
системный программист (c, linux) с опытом Python
старший программист с опытом Python
инженер - программист с опытом Python
педагог программист с опытом Python
f4
1с программист с опытом Python, зарплата 115228 руб
web-программист с опытом Python, зарплата 132404 руб
веб - программист (php, js) / web разработчик с опытом Python, зарплата 171906 руб
веб-программист с опытом Python, зарплата 144235 руб
ведущий инженер-программист с опытом Python, зарплата 102422 руб
ведущий программист с опытом Python, зарплата 194143 руб
инженер - программист асу тп с опытом Python, зарплата 192646 руб
инженер-программист с опытом Python, зарплата 101671 руб
инженер-программист (клинский филиал) с опытом Python, зарплата 188533 руб
инженер-программист (орехово-зуевский филиал) с опытом Python, зарплата 144224 руб
инженер-программист 1 категории с опытом Python, зарплата 132328 руб
инженер-программист ккт с опытом Python, зарплата 124234 руб
инженер-программист плис с опытом Python, зарплата 172736 руб
инженер-программист сапоу (java) с опытом Python, зарплата 107076 руб
инженер-электронщик (программист асу тп) с опытом Python, зарплата 139760 руб
помощник веб-программиста с опытом Python, зарплата 108568 руб
программист с опытом Python, зарплата 159380 руб
программист / senior developer с опытом Python, зарплата 197772 руб
программист 1с с опытом Python, зарплата 152696 руб
программист c# с опытом Python, зарплата 174502 руб
программист c++ с опытом Python, зарплата 132500 руб
программист c++/c#/java с опытом Python, зарплата 163602 руб
программист/ junior developer с опытом Python, зарплата 195272 руб
программист/ технический специалист с опытом Python, зарплата 163784 руб
программистр-разработчик  информационных систем с опытом Python, зарплата 193583 руб
системный программист (c, linux) с опытом Python, зарплата 123833 руб
старший программист с опытом Python, зарплата 172577 руб
инженер - программист с опытом Python, зарплата 107314 руб
педагог программист с опытом Python, зарплата 110356 руб
time: 0.03

Process finished with exit code 0
```