Московский государственный технический университет имени Н.Э.Баумана

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

Отчет по лабораторной работе №3-4 по курсу Базовые компоненты «Функциональные возможности языка Python»

Исполнитель
Студент группы РТ5-31Б _____ Татаев С.А.
«__»_____ 2022 г.

Проверил
Доцент кафедры ИУ5 _____ Гапанюк Ю.Е.
«__»_____ 2022 г.

г. Москва – 2022

Задание

Общее:

- 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. Пример:

gen_random (5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1

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:

- С использованием lambda-функции.
- 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.

После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).

cm_timer_1 и cm_timer_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

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.

Листинг файла field.py

```
def field(items, *args):
    assert len(args) > 0
    for elem in items:
        if len(args) == 1:
            if elem.get(args[0]) != None:
                yield """ + elem[args[0]] + """
        else:
            line = {}
            for one in args:
                line[one] = elem[one]
            yield line


goods = [{'title': 'Ковёр', 'price': 2000, 'color': 'green'},
         {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
print((", ").join(field(goods, 'title')))
result = [elem for elem in field(goods, 'title', 'price')]
print(*result, sep=", ")
```

Листинг файла gen_random.py

```
from random import randint

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield randint(begin, end)

print(*list(gen_random(5, 1, 3)), sep=", ")
```

Листинг файла unique.py

```
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore = kwargs.get("ignore_case", False)
        self.items = list(items)
        self.were = []
        self.n = 0

    def __next__(self):
        while (self.n < len(self.items)):
            one = self.items[self.n]
            if (self.ignore):
                if str(one).lower() not in self.were:
                    self.were.append(str(one).lower())
                    self.n += 1
                    return one
                else:
                    self.n += 1
            else:
                if one not in self.were:
                    self.were.append(one)
                    self.n += 1
                    return one
                else:
                    self.n += 1
        raise StopIteration

    def __iter__(self):
        return self

data = list(gen_random(10, 1, 3))
print(data)
i = iter(Unique(list(data)))
for elem in i:
    print(elem)
data1 = ['a', 'A', 'B', 'b', 'a', 'A', 'B', 'b']
print(data1)
j = iter(Unique(data1))
for elem in j:
    print(elem)
print(data1)
k = iter(Unique(data1, ignore_case=True))
for elem in k:
    print(elem)
```

Листинг файла sort.py

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

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

```
    print(sorted(data, key=abs, reverse=True))
```

Листинг файла print_result.py

```
# 5
def print_result(func):
    def result(*arg):
        a = func(*arg)
        if isinstance(a, list):
            print(func.__name__, *a, sep="\n")
        elif isinstance(a, dict):
            print(func.__name__, *[f"{key} = {val}" for key, val in a.items()], sep="\n")
        else:
            print(func.__name__, a, sep="\n")
        return a

    return result

@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__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

Листинг файла cm_timer.py

```
# 6
import time
from contextlib import contextmanager

class cm_timer_1:
    def __init__(self):
        self.start = 0

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

    def __exit__(self, exp_type, exp_value, traceback):
```

```
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            b = time.time() - self.start
            print(f"time: {b}")

@contextmanager
def cm_timer_2():
    start = time.time()
    yield 1
    b = time.time() - start
    print(f"time: {b}")

with cm_timer_1():
    for i in range(100000):
        pass
with cm_timer_2():
    for i in range(1000000):
        pass
```

Листинг файла process_data.py

```
import json
import sys
from cm_timer import cm_timer_1
from print_result import print_result
from unique import Unique
from random import randint

path = r"D:\documents\Basic components\lab3-4\data_light.json"

with open(path, 'r', 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(data):
    return list(sorted(list(iter(Unique([i.get("job-name") for i in data], ignore_case=True))), key=str.lower))

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

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

@print_result
def f4(arg):
    a = [str(randint(100_000, 200_000)) for i in range(len(arg))]
```

```
    return [(", зарплата ").join(i) for i in zip(arg, a)]

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

Результаты работы программы field.py

```
>>> [анализируем field.py]
    'Ковёр', 'Диван для отдыха'
    {'title': 'Ковёр', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
>>>
```

Результаты работы программы gen_random.py

```
>>> [анализируем gen_random.py]
    3, 1, 1, 2, 1
>>>
```

Результаты работы программы unique.py

```
>>> [анализируем unique.py]
    [2, 1, 3, 2, 2, 3, 2, 1, 3, 1]
    2
    1
    3
    ['a', 'A', 'B', 'b', 'a', 'A', 'B', 'b']
    a
    A
    B
    b
    ['a', 'A', 'B', 'b', 'a', 'A', 'B', 'b']
    a
    B
>>>
```

Результаты работы программы sort.py

```
>>> [анализируем sort.py]
    [123, 100, -100, -30, 4, -4, 1, -1, 0]
    [123, 100, -100, -30, 4, -4, 1, -1, 0]
>>>
```

Результаты работы программы print_result.py

```
>>> [анализируем print_result.py]
    !!!!!!!!
    test_1
    1
    test_2
    iu5
    test_3
    a = 1
    b = 2
    test_4
    1
    2
>>>
```

Результаты работы программы cm_timer.py

```
>>> [анализируем cm_timer.py]
    time: 0.009997129440307617
    time: 0.030004501342773438
>>>
```

Результаты работы программы process_data.py

```
>>> [анализируем process_data.py]
    f11C программист
    2-ой механик
    3-ий механик
    4-ый механик
    4-ый электромеханик
    [химик-эксперт
    ASIC специалист
    JavaScript разработчик
    RTL специалист
    Web-программист
    web-разработчик
    Автожестянщик
    Автоинструктор
    Автомаляр
    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 130544
Программист / Senior Developer с опытом Python, зарплата 181937
Программист 1С с опытом Python, зарплата 179468
Программист C# с опытом Python, зарплата 155895
Программист C++ с опытом Python, зарплата 187906
Программист C++/C#/Java с опытом Python, зарплата 142593
Программист/ Junior Developer с опытом Python, зарплата 185976
Программист/ технический специалист с опытом Python, зарплата 161654
Программистр-разработчик  информационных систем с опытом Python, зарплата 184325
time: 0.04999351501464844
```