

Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Отчет по лабораторной работе №3

Выполнил:
студент группы ИУ5-53Б

Баркалова Ирина

Дата: 26.10.2021

Москва. 2021 г.

Задание:

Задание лабораторной работы состоит из решения нескольких задач.

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

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

№1

Необходимо реализовать генератор `field`. Генератор `field` последовательно выдает значения ключей словаря. Пример:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', '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'}`

- В качестве первого аргумента генератор принимает список словарей, дальше через `*args` генератор принимает неограниченное количество 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.

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/field.py
'Ковер', 'Диван для отдыха'
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
```

№2

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`

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/gen_random.py
2, 1, 3, 1, 1
7, 7, 3
```

№3

- 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.

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/unique.py
Отбор уникальных чисел: 1, 2
Отбор уникальных случайных чисел: 10, 3, 8, 5, 9
Отбор уникальных строк без игнорирования регистра по умолчанию: 'a', 'A', 'b', 'B'
Отбор уникальных строк с игнорированием регистра: 'a', 'b'
Отбор уникальных строк без игнорирования регистра: 'a', 'A', 'b', 'B'
```

№4

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`. Пример:

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

Необходимо решить задачу двумя способами:

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

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/sort.py
Без использования lambda-функции: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
С использованием lambda-функции: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
```

№5

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 выводиться в столбик через знак равенства.

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/print_result.py
```

```
-----
```

```
test_1
```

```
1
```

```
test_2
```

```
iv5
```

```
test_3
```

```
a = 1
```

```
b = 2
```

```
test_4
```

```
1
```

```
2
```

```
-----
```

№6

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

```
with cm_timer_1():  
    sleep(5.5)
```

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`).

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/cm_timer.py  
cm_timer_1 5.502502679824829  
cm_timer_2 5.50265645980835
```

№7

- 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.

```
/usr/bin/python3.9 /home/iris/PyCharmProjects/RIP/lab_3/lab_python_fp/process_data.py
```

```
f1
```

```
1С программист
```

```
2-ой механик
```

```
3-ий механик
```

```
4-ый механик
```

```
4-ый электромеханик
```

```
ASIC специалист
```

```
JavaScript разработчик
```

```
RTL специалист
```

```
Web-программист
```

```
Web-разработчик
```

```
[химик-эксперт
```

```
web-разработчик
```

```
Автожестящик
```

```
Автоинструктор
```

```
Автомаляр
```

```
Автомойщик
```

```
Автор студенческих работ по различным дисциплинам
```

```
Автослесарь
```

```
Автослесарь - моторист
```

```
Автоэлектрик
```

```
Агент
```

```
Агент банка
```

```
Агент нпф
```

```
Агент по гос. закупкам недвижимости
```

```
Агент по недвижимости
```

```
Агент по недвижимости (стажер)
```