Python: Агрегация данных (Числа)

Отдельный класс задач, который не обходится без циклов, называется **агрегированием данных**. К таким задачам относятся: поиск максимального или минимального значения, суммы, среднего арифметического. В их случае результат зависит от всего набора данных.

Чтобы рассчитать сумму, нужно сложить все числа, а чтобы вычислить максимальное, нужно их сравнить. С такими задачами хорошо знакомы бухгалтеры или маркетологи. Они работают в таблицах Microsoft Excel или Google Tables.

В этом уроке разберем, как агрегация применяется к числам и строкам.

Допустим, нам нужно найти суммы набора чисел. Реализуем функцию, которая складывает числа в указанном диапазоне, включая границы. **Диапазон** — ряд чисел от конкретного начала до определенного конца. Например, диапазон [1, 10] включает целые числа от одного до десяти.

Пример:

```
sum_numbers_from_range(5, 7)  # 5 + 6 + 7 = 18
```

```
sum_numbers_from_range(1, 2)  # 1 + 2 = 3

# [1, 1] диапазон с одинаковым началом и концом — тоже диап
# Он включает одно число — саму границу диапазона
sum_numbers_from_range(1, 1)      # 1
sum_numbers_from_range(100, 100)  # 100
```

Чтобы реализовать такой код, понадобится цикл, так как сложение чисел — это итеративный процесс, то есть повторяется для каждого числа. Количество итераций зависит от размера диапазона.

Посмотрите код ниже:

```
def sum_numbers_from_range(start, finish):
    # Технически можно менять start
    # Но входные аргументы нужно оставлять в исходном значе
    # Это сделает код проще для анализа
    i = start
    sum = 0  # Инициализация суммы
    while i <= finish:  # Двигаемся до конца диапазона
        sum = sum + i   # Считаем сумму для каждого числа
        i = i + 1       # Переходим к следующему числу в ди
    # Возвращаем получившийся результат
    return sum
```

https://replit.com/@hexlet/python-basics-loops-aggregation-numbers

Структура цикла здесь стандартная: есть счетчик, который инициализируется начальным значением

диапазона, цикл с условием остановки при достижении конца диапазона и изменение счетчика в конце тела цикла. Количество итераций в таком цикле равно `finish - start + 1`. Для диапазона [5, 7] — это 7 - 5 + 1, то есть три итерации.

Главные отличия от обычной обработки — логика вычислений результата. В задачах на агрегацию всегда есть переменная, которая хранит внутри себя результат работы цикла. В коде выше это `sum`. Она изменяется на каждой итерации цикла — прибавляется следующее число в диапазоне: `sum = sum + i`.

Этот процесс выглядит так:

```
# Для вызова sum_numbers_from_range(2, 5)
sum = 0
sum = sum + 2  # 2
sum = sum + 3  # 5
sum = sum + 4  # 9
sum = sum + 5  # 14
# 14 — результат сложения чисел в диапазоне [2, 5]
```

У переменной `sum` есть начальное значение — с него начинается любая повторяющаяся операция. В примере выше — это `0`.

В математике есть понятие **нейтральный элемент**, и у каждой операции он свой. Операция с этим

элементом не изменяет то значение, над которым работает. Например, в сложении любое число плюс ноль дает само число. При вычитании — то же самое. У конкатенации тоже есть нейтральный элемент — это пустая строка: `'' + 'one'` будет 'one'.

Задание

Реализуйте функцию `multiply_numbers_from_range()`, которая перемножает числа в указанном диапазоне включая границы диапазона. Пример вызова:

```
multiply_numbers_from_range(1, 5)  # 1 * 2 * 3 * 4 * 5 = 12
multiply_numbers_from_range(2, 3)  # 2 * 3 = 6
multiply_numbers_from_range(6, 6)  # 6
```

- ▶ Упражнение не проходит проверку — что делать? 😶
- ▶ В моей среде код работает, а здесь нет 😐
- ▶ Мой код отличается от решения учителя 🤔
- ▶ Прочитал урок — ничего не понятно 🙄

Полезное

- Итерация

Определения

- Агрегация — Накопление результата во время итераций и работа с ним после цикла.