Решение задач на цикл с предусловием. Алгоритм Евклида

Аннотация

В этом уроке мы повторим, как проверить, не пора ли выходить из цикла, научимся искать минимум и максимум и узнаем один из классических алгоритмов — алгоритм Евклида по поиску НОД двух целых чисел.

1.Сигнал остановки

Рассмотрим такую задачу: пользователь вводит числа. Пусть это будут цены на купленные в магазине товары, а наша программа — часть программного обеспечения кассового аппарата. Ввод -1 — сигнал остановки. Нужно сосчитать сумму всех введенных чисел (сумму чека).

Поскольку требуется повторить нечто (ввод очередной цены) неизвестное количество раз, потребуется цикл `while`. Нам понадобится как минимум две переменные: `price` для цены очередного товара и `total` — для общей суммы.

Если бы мы знали точно, что пользователю надо купить ровно три товара, цикл (и ввод -1 как условие его прерывания) был бы не нужен. Тогда программа могла бы выглядеть так:

```
total = 0
price = float(input())
total = total + price
price = float(input())
total = total + price
price = float(input())
total = total + price
print('Сумма введённых чисел равна', total)
```

Обратите внимание: мы назвали переменные осмысленно. Это очень облегчит жизнь программисту, который будет читать наш код позже, даже если это будете вы сами неделю спустя. Однако интерпретатор Python к этому факту совершенно равнодушен. Чтобы значения переменных соответствовали названиям и тому смыслу, который мы

в них закладываем, нужно поддерживать переменные в актуальном состоянии. И только вы, программист, можете это сделать.

С переменной `price` все относительно понятно: ее значение обновляется при считывании с клавиатуры на каждой итерации цикла, как это делалось во многих других задачах. `total` сначала равно нулю: до начала ввода цен их сумма, конечно, ноль. Однако значение переменной `total` устаревает каждый раз, когда пользователь вводит цену очередного товара. Поэтому нам нужно прибавить к значению `total` только что введенную цену, чтобы эта переменная по-прежнему обозначала сумму цен всех купленных товаров.

Если бы мы хотели сократить запись, можно было бы организовать цикл, который выполнился бы ровно три раза. Для этого нам потребуется переменная-счетчик, которая внутри цикла будет считать каждую итерацию цикла. А условием выхода обозначим выполнение нужного количества итераций:

```
count = 0
total = 0
while count < 3:
    price = float(input())
    total = total + price
    count = count + 1
print('Сумма введённых чисел равна', total)
```

Обратите внимание: `total` и `count` должны обнуляться до цикла.

Однако у нас в задаче количество товаров неизвестно, поэтому понадобится цикл до ввода сигнала остановки (−1). С учетом сказанного выше программа будет выглядеть так:

```
total = 0
print('Вводите цены; для остановки введите
-1.')
price = float(input())
while price > 0:
    total = total + price  # можно заменить на
 аналогичную запись
    # total += price
    price = float(input())
print('Общая стоимость равна', total)
```

2. Поиск максимума и минимума

Очень часто в задачах приходится использовать различные статистические алгоритмы: поиск максимума, минимума, среднего значения, медианы и моды чисел, главный из которых — определение

максимального и минимального значений на множестве данных.

Рассмотрим алгоритм в общем виде.

1)Заведем отдельную переменную для хранения максимума и минимума. В качестве начального значения можно задать:

-Заведомо малое для анализируемых данных значения, для максимума это будет совсем маленькое число: например, если мы вычисляем максимальный балл за экзамен, можно взять `maximum = 0`, тогда гарантированно произойдет замена максимума. Минимуму же, наоборот, присваивается заведомо большое значение

-Первый элемент данных

2)В теле цикла каждый подходящий элемент данных обрабатывается операторами по принципу:

-Если текущий элемент больше максимума, меняем максимум

-Если текущий элемент меньше минимума, заменяем минимум

Рассмотрим пример. Витя анализировал список литературы и решил, что хочет начать с самой большой по объему книги. Напишем программу, которая поможет мальчику определить, сколько страниц ему предстоит прочитать. Витя последовательно вводит количество страниц каждой книги из списка, а окончанием ввода служит ввод любого отрицательного числа.

```
biggest_book = 0
n = int(input())
while n > 0:
    if n > biggest_book:
        biggest_book = n
    n = int(input())
```

```
print(biggest_book)
```

Так как книга не может содержать в себе 0 страниц, для значения максимума мы можем взять 0.

После этого Витя начинает вводить количество страниц: например, он вводит 148. `148 > 0` — условие цикла выполняется, и мы переходим к операции сравнения. На данном шаге `148 > 0`, значит, `biggest_book = 148`. Снова считываем число.

Предположим теперь введено 120. `120 > 0` — продолжаем работать в цикле. `120 > 148` — условие не выполняется, переходим к вводу новых данных, `biggest_book` все еще равен 148.

В этот раз мальчик ввел 486, мы заходим в цикл `486 > 148`, производим замену `biggest_book = 486`. Продолжаем ввод. И так далее до тех пор пока не будет введено отрицательное число.

3.Алгоритм Евклида. Наибольший общий делитель

Наибольший общий делитель

Наибольший общий делитель (НОД) двух целых чисел — это наибольшее число, на которое можно разделить оба числа.

Алгоритм нахождения НОД назван по имени древнегреческого математика Евклида Александрийского, предложившего данный метод. Существует две реализации — вычитанием и делением. Рассмотрим обе.

Алгоритм Евклида (вычитанием)

1)Если числа равны, то любое из них является НОДом.

2)Если нет, то выберем большее и вычтем из большего числа меньшее.

3)Вернемся к пункту 1, только теперь будем работать с меньшим числом и результатом вычитания.

Вот так выглядит реализация этого алгоритма:

```
a = int(input())
b = int(input())
while a != b:
    if a > b:
        a -= b
    else:
        b -= a
print(b)
```

Алгоритм нахождения НОД (делением)

1)Большее число делим на меньшее.

2)Если делится без остатка — меньшее число и есть НОД, выходим из цикла.

3)Если остаток не 0, большее число заменяем на этот остаток.

4)Возвращаемся к пункту 1.

Воспользуйтесь любым вариантом поиска НОД для решения задачи «Сократить дробь».