# Урок 11.02. Списки. Индексация

Мы довольно много говорили об особенности работы с числами и строками, со списками мы тоже будем работать очень долго. Поэтому мы для работы с ними мы будем изучать как базовые вещи, так и продвинутые, и начнем прямо сейчас.

Задача «Очки».

Возьмём такую простую задачу: мы пишем свою игру, где играет пять игроков и каждый из них в процессе зарабатывает и теряет очки. И все их очки для удобства хранятся в одной переменной список. В игре происходит какое-то событие, условие которого выполняет только второй игрок, тем самым увеличивая свои очки ровно в три раза, а после этого на экран выводится список очков всех игроков. Давай попробуем реализовать такую ситуацию.

Для начала создадим список на пять игроков и выведем их очки:

scores = [8, 5, 10, 7, 4]

print(scores)

Сделали. Теперь нам нужно увеличить количество очков второму игроку в три раза. Первое что должно прийти в голову – это пройтись по списку, найти «5» и увеличить в три раза. Давайте попробуем:

scores = [8, 5, 10, 7, 4]

print(scores)

for score in scores:

 if score == 5:

 ???

Но что же писать в условии? Наверное, надо создать еще один список и в него переписывать новые значения очков. Давайте так и сделаем:

scores = [8, 5, 10, 7, 4]

print(scores)

new\_scores = []

for score in scores:

 if score == 5:

 score \*= 3

 new\_scores.append(score)

print(new\_scores)

Это, конечно, работает, но не правильно. Во-первых, если в нашем списке очков у другого игрока тоже будет 5 очков, то наша программа ему тоже увеличит количество очков, а это не правильно. Во-вторых, и самая важная проблема, если у второго игрока изначально было другое количество очков не 5, то наша программа не будет работать, и нам нужно переписывать программу. Если вернуться назад, к первой программе, то нам нужно просто взять второй элемент в списке и увеличить его в три раза. Или по более правильно, нам нужно обратиться ко второму элементу списка и изменить его. И, конечно, программисты предусмотрели такую очевидную возможность. Такое обращение к элементам списка осуществляется по их индексам, который пишется в квадратных скобках после имени списка. Давайте это сделаем:

scores = [8, 5, 10, 7, 4]

print(scores)

scores[2] \*= 3

print(scores)

Запустим программу, и видим, что в три раза увеличился третий элемент. На самом деле всё просто, и это произошло потому, что номер элемента и индекс считаются немного по-разному, а именно: если номера элементов мы считаем, начиная с 1, то индексы в списках считаются с 0. Тогда, чтобы увеличить второй элемент в три раза нам нужно взять элемент списка с индексом 1, т.е. на один меньше чем номер элемента.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Список* | **8** | **5** | **10** | **7** | **6** |
| *Индекс* | 0 | 1 | 2 | 3 | 4 |

Давайте исправим нашу программу:

scores = [8, 5, 10, 7, 4]

print(scores)

scores[1] \*= 3

print(scores)

Вот теперь у нас всё получилось. Таким образом, мы можем обратиться к любому элементу списка по его индексу, и далее работать как с любой другой переменной.

Давайте рассмотрим еще одну задачу.

Задача «Урон»

У нас есть список из N элементов, в нем хранятся значения урона от всех монстров. Среди них есть такой маг, который добавляет свой урон к значению урона всех остальных монстров, чей урон меньше 100, кроме себя. Давайте посмотрим, как можно реализовать такую программу.

Первое, нам нужно спросить у пользователя, сколько монстров будет в списке.

Второе, нам нужно спросить под каким номером будет наш маг.

Третье, нам нужно спросить урон каждого монстра.

И четвертое, нам нужно посчитать урон мага.

monsters\_count = int(input('Количество монстров: '))

mage\_index = int(input('Номер мага в списке: '))

for monster in range(monsters\_count):

 print('Введите урон', monster + 1, 'монстра:', end=' ')

 damage = int(input())

 monsters\_damage.append(damage)

for i\_monster in range(monsters\_count):

 if monsters\_damage[i\_monster] < 100 and i\_monster != mage\_index - 1:

 monsters\_damage[i\_monster] += monsters\_damage[mage\_index - 1]

print('Итоговый урон монстров:', monsters\_damage)

Запустим программу.

Количество монстров: 5

Номер мага в списке: 3

Введите урон 1 монстра: 10

Введите урон 2 монстра: 100

Введите урон 3 монстра: 30

Введите урон 4 монстра: 60

Введите урон 5 монстра: 150

[40, 100, 30, 90, 150]

Получим вот такой результат. Всё получилось.

И ещё тут мы познакомимся с новой ошибкой, если введем номер мага больше, чем количество элементов в нашем списке.

IndexError: list index out of range

Данный тип ошибки переводится как «Ошибка индекса: индекс списка за границами». И она вам будет встречаться очень часто, так что её нужно запомнить.

Продолжим изучение наших списков. И как всегда на примерах. Рассмотрим нашу первую задачу, в которой случилось другая игровая ситуация: первому и третьему игроку начисляется столько очков, сколько игроков играет.

Напишем такую программу

scores = [8, 5, 10, 7, 4]

print(scores)

scores[0] += 5

scores[2] += 5

print(scores)

Отлично, но не совсем. А что если количество игроков увеличится, и вообще заранее не известно сколько человек будет играть. Для этого мы можем просто посчитать количество игроков или длину списка, и данную задачу (подсчет длины списка) вынесем в отдельную функцию.

def length(my\_list):

 count = 0

 for \_ in my\_list:

 count += 1

 return count

scores = [8, 5, 10, 7, 4]

print(scores)

scores[0] += length(scores)

scores[2] += length(scores)

print(scores)

Вот, уже хорошо. Однако подсчет количества элементов в списках очень распространенная задача поэтому данная функция имеется в python по умолчанию. И называется аналогично нашей функции за исключением того, что название сокращено до трех букв – len.

scores = [8, 5, 10, 7, 4]

print(scores)

scores[0] += len(scores)

scores[2] += len(scores)

print(scores)

Рассмотрим еще одну стандартную ситуацию. На нужно посмотреть последний элемент списка. На примере игры программа будет выглядеть так:

scores = [8, 5, 10, 7, 4]

print(scores)

print(scores[len(scores) - 1])

Справились, отлично. А предпоследний:

scores = [8, 5, 10, 7, 4]

print(scores)

print(scores[len(scores) - 2])

Хорошо. Однако в python такой вид записей ещё сократили так:

scores = [8, 5, 10, 7, 4]

print(scores)

print(scores[-1])

print(scores[-2])

Таким образом, индексы можно «считать» с конца, если поставить перед ними знак минус, но счет начинается с "-1".