Существует несколько типов коллекций для хранения данных в Python. Позиционно упорядоченные коллекции элементов обычно называются **последовательностями**, и им принадлежат как списки, так и строки. Каждый элемент в списке, так же как и каждый символ в строке, имеет **индекс**, соответствующий его позиции. Индексы используются для доступа к элементам внутри последовательности. Индексирование *основано на нулевом значении*, поэтому, если вы видите человека, который считает с нуля, вы, должно быть, встречали программиста.

**Индексы элементов**

Чтобы получить доступ к элементу списка по его индексу, вам нужно использовать **квадратные скобки.**Вы добавляете скобки после списка и между ними записываете индекс элемента, который хотите получить.

Не забывайте,**что**индексы начинаются с 0, поэтому индекс первого элемента равен 0. Индекс последнего элемента равен len(list) - 1.

Давайте посмотрим на пример ниже:

colors = ['red', 'green', 'blue']

first\_elem = colors[0] # 'red'

second\_elem = colors[1] # 'green'

third\_elem = colors[2] # 'blue'

Строки работают таким же образом:

pet = "cat"

first\_char = pet[0] # 'c'

second\_char = pet[1] # 'a'

third\_char = pet[2] # 't'

**Потенциальные подводные камни**

При использовании индексов важно оставаться в пределах диапазона вашей последовательности: вы получите ошибку (вызываемую IndexError), если попытаетесь получить доступ к элементу с несуществующим индексом!

colors = ['red', 'green', 'blue']

pet = "cat"

print(colors[3]) # IndexError: list index out of range

print(pet[3]) # IndexError: string index out of range

На вашем пути есть еще одно препятствие. Представьте, что вы хотите изменить один из элементов в списке. Это может быть легко сделано:

colors = ['red', 'green', 'blue']

colors[1] = 'white'

print(colors) # ['red', 'white', 'blue']

Однако, когда дело доходит до строк, такое переназначение невозможно. Строки, в отличие от списков, неизменяемы, поэтому вы не можете изменять их содержимое с помощью индексов:

pet = "cat"

pet[0] = "b"

# TypeError: 'str' object does not support item assignment

Не волнуйтесь, после некоторой практики вы не столкнетесь с этими ошибками.

**Отрицательные индексы**

Самый простой способ получить доступ к элементам в конце списка или строки - использовать **отрицательные индексы**: минус перед числом каким-то образом меняет вашу перспективу, и вы смотрите на последовательность с конца. Итак, последний элемент списка, в данном случае, имеет индекс, равный **-1**, а первый элемент списка имеет индекс -len(list) (длина списка).

Например:

colors = ['red', 'green', 'blue']

last\_elem = colors[-1] # 'blue'

second\_elem = colors[-2] # 'green'

first\_elem = colors[-3] # 'red'

pet = "cat"

last\_char = pet[-1] # 't'

second\_char = pet[-2] # 'a'

first\_char = pet[-3] # 'c'

Как вы можете видеть, это работает одинаково для списков и строк.

Если вы напишете несуществующий отрицательный индекс, вы также получите IndexError. Будьте осторожны с индексами, чтобы избежать отдельных ошибок в вашем коде.

На следующем рисунке показана общая концепция индексов в списке:

**Краткие сведения**

В этом разделе мы узнали, какие положительные и отрицательные индексы есть в Python, как находить элементы в списке по их индексам и какие ошибки могут возникнуть при их использовании. Поскольку вы изучили концепцию индексов, мы надеемся, что с этого момента вы не столкнетесь с какими-либо трудностями при их использовании!

**Практика**

