Строки

Операции

Строка - последовательность символов.

Также как и с другими типами данных строку можно записывать в переменную:

In [2]:
```
s = "Привет!"
```

Выводить на экран:

In [3]:
```
print(s)
```

```
Привет!
```

Из чуть более неочевидного, складывать:

In [5]:
```
s1 = "Привет"
s2 = "Вася"
s = s1 + ", " + s2 + "!"
print(s)
```

```
Привет, Вася!
```

И умножать:

In [6]:
```
s="Ку"
s5=s*5
print(s5)
```

```
КуКуКуКуКу
```

Делить и вычитать строки нельзя.

Обращение

Строку можно определить не просто как последовательность, а как неизменяемый массив. Также как в массиве можно обращаться по индексу к конкретному элементу:

In [13]:
```
s="Привет!"
print(s[1])
```

```
р
```

В отличии от массивов нельзя изменить конкретный символ в строке:

In [14]:
```
s[2]="е"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 s[2]="e"

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

Строка - это неизменяемый объект

Но, используя операции можно составить новую строку из уже существующей

In [17]:
```
s = "Привет!"
print(s)
s1 = s + " Всем!"
print(s1)
s2 = s[0]+s[4]+s[5]+s[1]
print("Составлено из частей предыдущей строки:",s2)
```

```
Привет!
Привет! Всем!
Составлено из частей предыдущей строки: Петр
```

Продвинутые способы обращения

На примере строк познакомимся с более продвинутыми способами обращения к элементам, но все эти способы также применимы к обычным массивам.

Отрицательные индексы

Если в качестве индекса передать отрицательное число, то будет возвращен элемент из конца строки:

In [11]:
```
print(s[-1])
```

```
!
```

Чтобы получить последний элемент можно обратиться к нему по индексу -1 или определить длину и вычесть единицу (`len(s)-1`). Для определения длины строки используется та же функция, что и для массивов.

In [12]:
```
print(s[-1])
print(s[len(s)-1])
```

```
!
!
```

Последний элемент имеет индекс -1, предпоследний -2 и так далее.

Срезы

Указав в квадратных скобках (оператор образения) диапазон через двоеточие можно получить соответсвующую часть строки. Также как и в `range()` первый

индекс включительно, а последний не включительно.

In [18]:
```
s = "0123456789"
print(s[3:8])
```

```
34567
```

При использования среза можно не указывать, что вы начинаете с начала строки (первый индекс - 0):

In [19]:
```
print(s[:3])
print(s[0:3])
```

```
012
012
```

И можно не указывать, что вы идете до конца строки (до последнего индекса):

In [22]:
```
print(s[7:])
print(s[7:len(s)])
```

```
789
789
```

Если вы идете сначала и до конца, то индексы можно не указывать, но эта запись идентичная просто обращению к переменной:

In [31]:
```
print(s[:])
print(s)
```

```
0123456789
0123456789
```

Помимо начального и конечного индекса в срезе можно указать шаг, также как и в `range()`. Это будет третье число через ещё одно двоеточие:

In [33]:
```
print(s[1:5:1])
print(s[1:5:2])
```

```
1234
13
```

Также же последовательность можно развернуть указав в шаге -1:

In [26]:
```
print(s)
print(s[::-1])
```

```
0123456789
9876543210
```

Либо указав любой другой нужный вам шаг:

In [27]:
```
print(s[::2])
```

```
02468
```

В случае отрицательного шага нужно также развернуть индексы и учитывать, что первый включительно, а последний не включительно:

In [29]:
```
print(s[3:8:1])
print(s[7:2:-1])
```

```
34567
76543
```

В срезах также можно использовать отрицательные индексы, вот несколько способов записи одного и того же среза, начиная с самого короткого и заканчивая самым полным.

In [35]:
```
print(s[:-2])
print(s[:len(s)-2])
print(s[0:len(s)-2:1])
```

```
01234567
01234567
01234567
```

Операции со срезами

Удаление:

In [37]:
```
s = "0123456789"
s = s[:3] + s[9:]
print(s)
```

```
0129
```

Для удаления определенного промежутка в строке нужно взять срез промежутка до и промежутка после и сложить их.

Вставка:

In [40]:
```
s = "0123456789"
s = s[:3] +"___"+ s[3:]
print(s)
```

```
012___3456789
```