# Cordas de fatiamento em Python

Em Python, uma string é uma sequência ordenada de caracteres Unicode . Cada caractere da string possui um índice exclusivo na sequência. O índice começa com 0. O primeiro caractere da string tem seu índice posicional 0. O índice continua aumentando no final da string.

Se uma variável de string for declarada como var="HELLO PYTHON", o índice de cada caractere na string é o seguinte -

| H | E | L | L | O | | P | Y | T | H | O | N |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

## Indexação de strings Python

Python permite acessar qualquer caractere individual da string por meio de seu índice. Neste caso, 0 é o limite inferior e 11 é o limite superior da string. Portanto, var[0] retorna H, var[6] retorna P. Se o índice entre colchetes exceder o limite superior, Python gera IndexError.

## Exemplo

```
>>> var="HELLO PYTHON"
>>> var[0]
'H'
>>> var[7]
'Y'
>>> var[11]
'N'
>>> var[12]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

## Indexação negativa e positiva de strings Python

Um dos recursos exclusivos dos tipos de sequência do Python (e, portanto, de um objeto string), ele também possui um esquema de indexação negativo. No exemplo acima, um esquema de indexação positivo é usado onde o índice aumenta da esquerda para a direita. No caso de indexação negativa, o caractere no final possui índice -1 e o índice diminui da direita para a esquerda, como resultado o primeiro caractere H possui índice -12.

| H | E | L | L | O | | P | Y | T | H | O | N |
|---|---|---|---|---|---|---|---|---|---|---|---|
| -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

Vamos usar indexação negativa para buscar caracteres N, Y e H.

## Exemplo

```
>>> var[-1]
'N'
>>> var[-5]
'Y'
>>> var[-12]
'H'
>>> var[-13]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Mais uma vez, se o índice ultrapassar o intervalo, IndexError será encontrado.

Podemos, portanto, usar índice positivo ou negativo para recuperar um caractere da string.

## Exemplo

```
>>> var[0], var[-12]
('H', 'H')
>>> var[7], var[-5]
('Y', 'Y')
>>> var[11], var[-1]
('N', 'N')
```

Em Python, string é um objeto imutável. O objeto é imutável se não puder ser modificado no local, uma vez armazenado em um determinado local da memória. Você pode recuperar qualquer caractere da string com a ajuda de seu índice, mas não pode substituí-lo por outro caractere. Em nosso exemplo, o caractere Y está no índice 7 em HELLO PYTHON. Tente substituir Y por y e veja o que acontece.

## Exemplo

```
var="HELLO PYTHON"
var[7]="y"
print (var)
```

Ele produzirá a seguinte **saída** -

```
Traceback (most recent call last):
 File "C:\Users\users\example.py", line 2, in <module>
  var[7]="y"
  ~~~^^^
TypeError: 'str' object does not support item assignment
```

O TypeError ocorre porque a string é imutável.

Python define ":" como operador de fatiamento de string. Ele retorna uma substring da string original. Seu uso geral é -

```
substr=var[x:y]
```

## Fatiamento de strings em Python

O operador ":" precisa de dois operandos inteiros (ambos podem ser omitidos, como veremos nos exemplos subsequentes). O primeiro operando x é o índice do primeiro caractere da fatia desejada. O segundo operando y é o índice do caractere próximo ao último na string desejada. Então var(x:y] separa caracteres da x-ésima posição para (y-1)-ésima posição da string original.

### Exemplo

```
var="HELLO PYTHON"

print ("var:",var)
print ("var[3:8]:", var[3:8])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[3:8]: LO PY
```

## Fatiamento de strings em Python com indexação negativa

Índices negativos também podem ser usados para fatiamento.

### Exemplo

```
var="HELLO PYTHON"
print ("var:",var)
```

```
print ("var[3:8]:", var[3:8])
print ("var[-9:-4]:", var[-9:-4])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[3:8]: LO PY
var[-9:-4]: LO PY
```



## Valores padrão de índices com divisão de strings

Ambos os operandos do operador Slice do Python são opcionais. O primeiro operando é padronizado como zero, o que significa que se não fornecermos o primeiro operando, a fatia inicia o caractere no índice 0, ou seja, o primeiro caractere. Ele corta a substring mais à esquerda até os caracteres "y-1".

### Exemplo

```
var="HELLO PYTHON"
print ("var:",var)
print ("var[0:5]:", var[0:5])
print ("var[:5]:", var[:5])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[0:5]: HELLO
var[:5]: HELLO
```

### Exemplo

Da mesma forma, o operando y também é opcional. Por padrão, é "-1", o que significa que a string será dividida da x-ésima posição até o final da string.

```
var="HELLO PYTHON"
print ("var:",var)
print ("var[6:12]:", var[6:12])
print ("var[6:]:", var[6:])
```

Ele produzirá a seguinte saída -

```
var: HELLO PYTHON
var[6:12]: PYTHON
var[6:]: PYTHON
```

## Exemplo

Naturalmente, se ambos os operandos não forem utilizados, a fatia será igual à string original. Isso ocorre porque "x" é 0 e "y" é o último índice+1 (ou -1) por padrão.

```
var="HELLO PYTHON"
print ("var:",var)
print ("var[0:12]:", var[0:12])
print ("var[:]:", var[:])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[0:12]: HELLO PYTHON
var[:]: HELLO PYTHON
```

O operando esquerdo deve ser menor que o operando direito, para obter uma substring da string original. Python não gera nenhum erro, se o operando esquerdo for maior, bu retorna uma string nula.

## Exemplo

```
var="HELLO PYTHON"
print ("var:",var)
print ("var[-1:7]:", var[-1:7])
print ("var[7:0]:", var[7:0])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[-1:7]:
var[7:0]:
```

## Tipo de retorno de fatiamento de string

O fatiamento retorna uma nova string. Você pode muito bem executar operações de string, como concatenação ou fatiar a string fatiada.

## Exemplo

```
var="HELLO PYTHON"

print ("var:",var)
print ("var[:6][:2]:", var[:6][:2])

var1=var[:6]
print ("slice:", var1)
print ("var1[:2]:", var1[:2])
```

Ele produzirá a seguinte **saída** -

```
var: HELLO PYTHON
var[:6][:2]: HE
slice: HELLO
var1[:2]: HE
```