



Python Conditional Statements & String

If-else statements



If statement

Python uses boolean logic to evaluate conditions

Example

Test if `a` is greater than `b`, AND if `c` is greater than `a`:

```
a = 200
b = 33
c = 500
if a > b and c > a:
    print("Both conditions are True")
```



If - else example

Example

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```



Nested if


If Inside If

You can have `if` statements inside `if` statements, this is called *nested if* statements.

Example

```
x = 41

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```



Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello" .

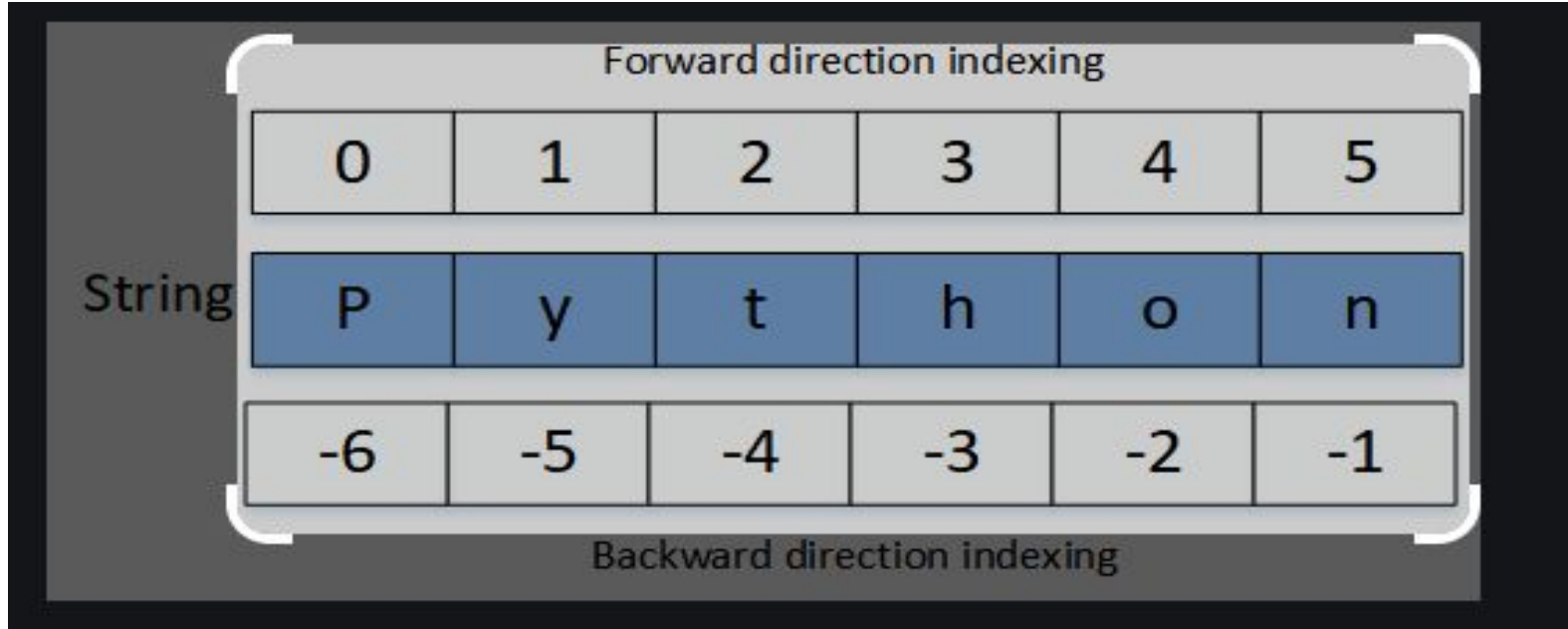
```
# defining strings in Python
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
                the world of Python"""
print(my_string)
```

String index





Python substring

Python has no **substring methods** like `substring()` or `substr()`. Instead, we use slice syntax to get parts of existing strings. **Python slicing** is a computationally fast way to methodically access parts of your data. The colons (:) in subscript notation make slice notation - which has the arguments, **start, stop and step** . It follows this template:

- Parameters are enclosed in the square brackets.
- Parameters are separated by colon.

```
string[start: end: step]
```



How to access characters in String

```
#Accessing string characters in Python
str = 'programiz'
print('str = ', str)

#first character
print('str[0] = ', str[0])

#last character
print('str[-1] = ', str[-1])

#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])

#slicing 6th to 2nd last character
print('str[5:-2] = ', str[5:-2])
```




Concatenation of two or more strings

```
# Python String Operations
str1 = 'Hello'
str2 = 'World!'

# using +
print('str1 + str2 = ', str1 + str2)

# using *
print('str1 * 3 =', str1 * 3)
```