**PYTHON2**

**Python Strings:**

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

'hello' is the same as "hello".

You can display a string literal with the print() function:

Ex:-

Print(“hello”)

Print(‘hello’)

**Strings are arrays:**

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

**Ex:-**

a = "Hello, World!"  
print(a[1])

**Looping through Strings:**

Since strings are arrays, we can loop through the characters in a string, with a for loop.

**Ex:-**

for x in "banana":  
   print(x)

**String Length:**

To get the length of a string, use the len() function.

Ex:-

a = "Hello, World!"  
print(len(a))

# **Slicing Strings:**

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Ex:-

b = "Hello, World!"  
print(b[2:5])

Slice from the start:

b = "Hello, World!"  
print(b[:5])

Slice to the end:

b = "Hello, World!"  
print(b[2:])

# **Modify Strings:**

**Upper Case:**

The upper() method returns the string in upper case:

a = "Hello, World!"  
print(a.upper())

**Lower Case:**

The lower() method returns the string in lower case:

a = "Hello, World!"  
print(a.lower())

**Remove White Spaces:**

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Ex:- The strip() method removes any whitespace from the beginning or the end:

a = " Hello, World! "  
print(a.strip()) # returns "Hello, World!"

**Replace Strings:**

The replace() method replaces a string with another string:

a = "Hello, World!"  
print(a.replace("H", "J"))

**Split Strings:**

The split() method returns a list where the text between the specified separator becomes the list items.

Ex:-

The split() method splits the string into substrings if it finds instances of the separator:

a = "Hello, World!"  
print(a.split(",")) # returns ['Hello', ' World!']

**Concatenation:**

To concatenate, or combine, two strings you can use the + operator.

Merge variable a with variable b into variable c:

a = "Hello"  
b = "World"  
c = a + b  
print(c)

**Python Booleans:**

Booleans represent one of two values: True or False.

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

**Ex:-**

print(10 > 9) #True  
print(10 == 9) #False  
print(10 < 9) #False

**Ex:-**

a = 200  
b = 33  
  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

**Python Operaters:**

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

**Arithmetic Operators:**

Arithmetic operators are used with numeric values to perform common mathematical operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| + | Addition | x + y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_add) |
| - | Subtraction | x - y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_sub) |
| \* | Multiplication | x \* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mult) |
| / | Division | x / y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_div) |
| % | Modulus | x % y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod) |
| \*\* | Exponentiation | x \*\* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_exp) |
| // | Floor division | x // y |  |

**Assignment Operators:**

Assignment operators are used to assign values to variables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Example** | **Same As** |  |  |
| = | x = 5 | x = 5 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass1) |
| += | x += 3 | x = x + 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass2) |
| -= | x -= 3 | x = x - 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass3) |
| \*= | x \*= 3 | x = x \* 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass4) |
| /= | x /= 3 | x = x / 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass5) |
| %= | x %= 3 | x = x % 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass6) |
| //= | x //= 3 | x = x // 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass7) |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass8) |
| &= | x &= 3 | x = x & 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass9) |
| |= | x |= 3 | x = x | 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass10) |
| ^= | x ^= 3 | x = x ^ 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass11) |
| >>= | x >>= 3 | x = x >> 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass12) |
| <<= | x <<= 3 | x = x << 3 |  | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass13) |

**Comparison Operators:**

Comparison operators are used to compare two values:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** |  |
| == | Equal | x == y |  |
| != | Not equal | x != y |  |
| > | Greater than | x > y |  |
| < | Less than | x < y |  |
| >= | Greater than or equal to | x >= y |  |
| <= | Less than or equal to | x <= y |  |

**Logical Operators:**

Logical operators are used to combine conditional statements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** |  |
| and | Returns True if both statements are true | x < 5 and  x < 10 |  |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |  |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |  |

**Identity Operators:**

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** |  |
| is | Returns True if both variables are the same object | x is y |  |
| is not | Returns True if both variables are not the same object | x is not y |  |

**Bitwise Operators:**

Bitwise operators are used to compare (binary) numbers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** |  |
| & | AND | Sets each bit to 1 if both bits are 1 | x & y |  |
| | | OR | Sets each bit to 1 if one of two bits is 1 | x | y |  |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y |  |
| ~ | NOT | Inverts all the bits | ~x |  |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 |  |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 |  |