<https://www.w3schools.com/python/python_variables.asp>

**Python basic:**

Python is object oriented program (OOP) or Graphical user interface .

* [Integers](https://www.digitalocean.com/community/tutorials/understanding-data-types-in-python-3#integers) are whole numbers that can be positive, negative, or 0 (…, -1, 0, 1, …).
* [Floats](https://www.digitalocean.com/community/tutorials/understanding-data-types-in-python-3#floating-point-numbers) are real numbers, they contain a decimal point (as in 9.0 or -2.25).
* Python Variables :
* In Python variables are created the moment you assign a value to it

### Example : Variables in Python:

x = 5  
 y = "Hello, World”

## Python Indentations

1. Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.
2. Python uses indentation to indicate a block of code.

### **Example**

if 5 > 2:  
   print("Five is greater than two!")

## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* You can also use the + character to add a variable to another variable

**Comment:**

* 1. Comments can be used to explain Python code.
  2. Comments can be used to make the code more readable.

## Comments can be used to prevent execution when testing code.

## Creating a Comment :

1. Comments starts with a #, and Python will ignore them.
2. Comments can be placed at the end of a line, and Python will ignore the rest of the line.

## Python Numbers :

There are three numeric types in Python:

* int
* float
* complex

Variables of numeric types are created when you assign a value to them:

### **Example**

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex (j=i)

## Type Conversion:

You can convert from one type to another with the int(), float(), and complex() methods:

### **Example**

Convert from one type to another:

x = 1 # int  
y = 2.8 # float  
z = 1j # complex  
  
#convert from int to float:  
a = float(x)  
  
#convert from float to int:  
b = int(y)  
  
#convert from int to complex:  
c = complex(x)  
  
print(a)  
print(b)  
print(c)  
  
print(type(a))  
print(type(b))  
print(type(c))

## String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

### **Example**

age = 36  
txt = "My name is John, I am " + age  
print(txt)

out : TypeError: must be str, not int

But we can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

age = 36  
txt = "My name is John, and I am {}"  
print(txt.format(age))

### **Example**

quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want {} pieces of item {} for {} dollars."  
print(myorder.format(quantity, itemno, price).

Output: I want 3 pieces of item 567 for 49.95 dollars.

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

### **Example**

quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."  
print(myorder.format(quantity, itemno, price))

output: I want to pay 49.95 dollars for 3 pieces of item 567

Python Operators

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

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

## Python Assignment Operators

Assignment operators are used to assign values to variables:

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

## Python Comparison Operators

Comparison operators are used to compare two values:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| == | Equal | x == y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare1) |
| != | Not equal | x != y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare2) |
| > | Greater than | x > y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare4) |
| < | Less than | x < y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare5) |
| >= | Greater than or equal to | x >= y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare6) |
| <= | Less than or equal to | x <= y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_compare7) |

## Python Logical Operators

Logical operators are used to combine conditional statements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| and | Returns True if both statements are true | x < 5 and  x < 10 | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical1) |
| or | Returns True if one of the statements is true | x < 5 or x < 4 | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical2) |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical3) |

## Python 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** | **Try it** |
| is | Returns true if both variables are the same object | x is y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_identity1) |
| is not | Returns true if both variables are not the same object | x is not y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_identity2) |

## Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:s

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| in | Returns True if a sequence with the specified value is present in the object | x in y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_membership1) |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_membership2) |

## Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

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

Top of Form

## Test Yourself With Exercises

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members. Rep by []
* **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members. Rep by ()
* **Set** is a collection which is unordered and unindexed. No duplicate members. Rep by {}
* **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.In Python dictionaries are written with curly brackets, and they have keys and values.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.