Link cheat sheet:

<https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf>

<https://www.datacamp.com/community/data-science-cheatsheets?tag=python>

<https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf>

<https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf>

<https://www.utc.fr/~jlaforet/Suppl/python-cheatsheets.pdf>

<https://web.itu.edu.tr/iguzel/files/Python_Cheat_Sheets.pdf>

Tutorial webs:

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

<https://www.tutorialspoint.com/python3/index.htm>

## Python Arithmetic Operators

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

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| + | Addition | x + y |
| - | Subtraction | x - y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Modulus | x % y |
| \*\* | Exponentiation | x \*\* y |
| // | Floor division | x // y |

Python Assignment Operators

Assignment operators are used to assign values to variables:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| \*= | x \*= 3 | x = x \* 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| \*\*= | x \*\*= 3 | x = x \*\* 3 |
| &= | x &= 3 | x = x & 3 |
| |= | x |= 3 | x = x | 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

Python 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 |

Python 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) |

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:

### Example

print("Hello")  
print('Hello')

## Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

### Example

a = "Hello"  
print(a)

## Multiline Strings

You can assign a multiline string to a variable by using three quotes:

### Example

You can use three double quotes:

a = """UEH,

UII,

Lap trinh can ban."""  
print(a)

Or three single quotes:

### Example

a = ''' UEH,

UII,

Lap trinh can ban.'''  
print(a)

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

### Example

Get the character at position 1 (remember that the first character has the position 0):

a = "Hello, World!"  
print(a[1]) #index

## String Length

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

### Example

The len() function returns the length of a string:

a = "Hello, World!"  
print(len(a))

## Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

### Example

Check if "free" is present in the following text:

txt = "The best things in life are free!"  
print("free" in txt)

## Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

### Example

Check if "expensive" is NOT present in the following text:

txt = "The best things in life are free!"  
print("expensive" not in txt)

## Slicing

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.

### Example

Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"  
print(b[2:5])

## Slice From the Start

By leaving out the start index, the range will start at the first character:

### Example

Get the characters from the start to position 5 (not included):

b = "Hello, World!"  
print(b[:5])

## Slice To the End

By leaving out the end index, the range will go to the end:

### Example

Get the characters from position 2, and all the way to the end:

b = "Hello, World!"  
print(b[2:])

## Negative Indexing

Use negative indexes to start the slice from the end of the string:

### Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

b = "Hello, World!"  
print(b[-5:-2])

## String Concatenation

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

### Example

Merge variable a with variable b into variable c:

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

### Example

To add a space between them, add a " ":

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

Text, letter

Description automatically generated

**Some useful String method:**

[count()](https://www.w3schools.com/python/ref_string_count.asp)

[find()](https://www.w3schools.com/python/ref_string_find.asp)

[isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp)

[isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp)

[isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp)

[islower()](https://www.w3schools.com/python/ref_string_islower.asp)

[isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp)

[isupper()](https://www.w3schools.com/python/ref_string_isupper.asp)

[lower()](https://www.w3schools.com/python/ref_string_lower.asp)

[replace()](https://www.w3schools.com/python/ref_string_replace.asp)

[split()](https://www.w3schools.com/python/ref_string_split.asp)

[upper()](https://www.w3schools.com/python/ref_string_upper.asp)

**List**

**Some useful List method:**

[append()](https://www.w3schools.com/python/ref_list_append.asp)

[count()](https://www.w3schools.com/python/ref_list_count.asp)

[remove()](https://www.w3schools.com/python/ref_list_remove.asp)

[sort()](https://www.w3schools.com/python/ref_list_sort.asp)

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing table

Description automatically generated

Table

Description automatically generated with medium confidence

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

A picture containing table

Description automatically generated

Text, letter

Description automatically generated

Chart

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated with medium confidence

Text, letter

Description automatically generated





Text

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with medium confidence

**FUNCTION**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with low confidence

**BRANCHING STATEMENT**

Text

Description automatically generated

Graphical user interface, text, application, Teams

Description automatically generated

**FOR LOOP**

Text

Description automatically generated with medium confidence Text

Description automatically generated with low confidence

for x in range(6):  
  print(x)

for x in range(2, 30, 3):  
  print(x)

**WHILE LOOP**

i = 1  
while i < 6:  
  print(i)  
  i += 1

Graphical user interface, text, application

Description automatically generated

A picture containing diagram

Description automatically generated