Python Language Cheat Sheet

Operators

Arithmetic Operators

Arithmetic operators are used 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

Assignment Operators

Assignment operators 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

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)

Membership Operators

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

Operator	Description	Example

in	Returns True if a sequence with the specified value is present in the object	x in y
is not	Returns True if both variables are not the same object	x is not y

Identity Operators

Identity operators are used to compare the objects to see 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
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
&	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

Variables

Variables are used to temporarily store data in the computer's memory.

Example:

```
price = 100

rating = 4.7

course_name = "Python for Beginners"

published = True
```

Functions

Functions are used to break up the code into small chunks. These chunks are easier to read and maintain. It is also easier to find bugs in a small chunk rather than going through the entire program. These chunks can also be reused.

Example:

```
def greet_user(name):

   print(f"Hi {name}")

greet_user("John")
```

Parameters are the placeholders for data to pass through functions. Arguments are the values passed.

There are two types of values - positional arguments (their position matters) and keyword arguments (their position doesn't matter).

Positional Argument Example:

```
greet_user("John", "Smith")
```

Keyword Argument Example:

```
calculate_total(order=30, shipping=8, tax=0.1)
```

Functions can return values. If in case you don't use the return statement, None is returned by default. None represents the absence of a value.

Example:

```
def square(number):
     return number * number
result = square(2)
print(result)
```

The above function will print 4 as a result.

If-Else Statements

The if-else statement is used to execute both the true and false parts of a given condition. If the condition is true, the "if" block code is executed. If the condition is false, the "else" block code is executed. The elif keyword prompts your program to try another condition if the previous one is not true.

```
if age < 4:
   price = 0
elif age < 18:
   price = 10
else:
   price = 20
```

Loops

For Loop

The for loop is used to iterate over a sequence such as a list, string, tuple, etc.

Example:

```
for i in range(10):
   print(i)
```

While Loop

The while loop enables you to execute a set of statements as long as the condition is true.

Example:

```
i = 1
while i < 8:
   print(i)
   i += 1
```