**PYTHON**

**Python Syntax**

 Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")  
Hello, World!

**Python identation**

Indentation refers to the spaces at the beginning of a code line.Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.Python uses indentation to indicate a block of code.

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

**python variables**

In Python, variables are created when you assign a value to it:

x = 5  
y = "Hello, World!"

**comments**

Comments can be used to explain Python code.Comments can be used to make the code more readable.Comments can be used to prevent execution when testing code.

#This is a comment.  
print("Hello, World!")

**Variables**

variables are containers for storing data value

**casting**

If you want to specify the data type of a variable, this can be done with casting.

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

You can get the data type of a variable with the type() function.

x = 5  
y = "John"  
print(type(x))  
print(type(y))

String variables can be declared either by using single or double quotes

Variable names are case-sensitive.

**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)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

Python allows you to assign values to multiple variables in one line:

x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

**global variables**

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

x = "awesome"  
  
def myfunc():  
  print("Python is " + x)  
  
myfunc()

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

**The global keyword**

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

If you use the global keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

**python data types**

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

You can get the data type of any object by using the type() function:

x = 5  
print(type(x))

**python numbers**

There are three numeric types in Python:

* int
* float
* complex

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Complex numbers are written with a "j" as the imaginary part:

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

**Python casting**

Casting in python is therefore done using constructor functions:

* int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
* float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
* str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

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

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

## **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.

b = "Hello, World!"  
print(b[2:5])

The first character has index 0.

## **Slice From the Start**

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

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

b = "Hello, World!"  
print(b[:5])

## **String Concatenation**

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

**Python format**

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))

## **Escape Character**

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

Other escape characters used in Python:

\’ single quote

\\ backslash

\n new line

\t tab

\b backspace

**Boolean**

booleans represent one of two values: True or False

The bool() function allows you to evaluate any value, and give you True or False in return,

## **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:

+ addition

\_