**Python Syntax**

**Execute Python Syntax**

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

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

Hello, World!

**Python Indentations**

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

if 5 > 1:

print("Five is greater than two!")

**Comments**

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

#This is a comment.

print("Hello, World!")

**Docstrings**

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

"""This is a

multiline docstring."""

print("Hello, World!")

**Creating Variables**

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

x = 100

y = "ABHISHEK"

print(x)

print(y)

Variables do not need to be declared with any particular type and can even change type after they have been set.

x = 4 # x is of type int

x = "Abhishek soni" # x is now of type str

print(x)

**Output Variables**

The Python print statement is often used to output variables.

To combine both text and a variable, Python uses the + character:

x = "Abhishek "

print("My name is " + x)

**You can also use the + character to add a variable to another variable:**

x = "Python is "

y = "awesome"

z = x + y

print(z)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x = 50

y = 10

print(x + y)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x = 50

y = "Abhishek"

print(x + y)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Python Numbers**

There are three numeric types in Python:

* int
* float
* complex

x = 1 # int

y = 2.8 # float

z = 1j # complex

**Int**

x = 1

y = 35656222554887711

z = -3255522

print(type(x))

print(type(y))

print(type(z))

**Float**

x = 1.10

y = 1.0

z = -35.59

print(type(x))

print(type(y))

print(type(z))

Float can also be scientific numbers with an "e" to indicate the power of 10.

x = 35e3

**y = 12E4**

z = -87.7e100

print(type(x))

print(type(y))

print(type(z))

**Complex**

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

x = 3+5j

y = 5j

z = -5j

print(type(x))

print(type(y))

print(type(z))

**Python Casting**

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

int() - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), 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

z = int("3") # z will be 3

w = float("4.2") # w will be 4.2

z = str(3.0) # z will be '3.0'

**String Literals**

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

'hello' is the same as "hello".

Strings can be output to screen using the print function. For example: print("hello").

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.

a = "Hello, World!"

print(a[1])

b = "Hello, World!"

print(b[2:5])

a = " Hello, World! "

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

a = "Hello, World!"

print(len(a))

a = "Hello, World!"

print(a.lower())

a = "Hello, World!"

print(a.upper())

a = "Hello, World!"

print(a.replace("H", "J"))

a = "Hello, World!"

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

**Command-line String Input**

Python allows for command line input.

That means we are able to ask the user for input.

The following example asks for the user's name, then, by using the input() method, the program prints the name to the screen:

print("Enter your name:")

x = input()

print("Hello, ", x)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Formatting for string

# Python Program for

# Formatting of Strings

# Default order

String1 = "{} {} {}".format('MSS', 'CYBER', 'SECURITY ')

print("Print String in default order: ")

print(String1)

# Positional Formatting

String1 = "{1} {0} {2}".format('MSS', 'CYBER ', 'SECURITY')

print("\nPrint String in Positional order: ")

print(String1)

# Keyword Formatting

String1 = "{l} {f} {g}".format(g = 'MSS', f = 'CYBER', l = 'SECURITY')

print("\nPrint String in order of Keywords: ")

print(String1)

# Formatting of Integers

String1 = "{0:b}".format(16)

print("\nBinary representation of 16 is ")

print(String1)

# Formatting of Floats

String1 = "{0:e}".format(165.6458)

print("\nExponent representation of 165.6458 is ")

print(String1)

# String alignment

String1 = "|{:<10}|{:^10}|{:>10}|".format('MSS','CYBER','SECURITY')

print("\nLeft, center and right alignment with Formatting: ")

print(String1)