

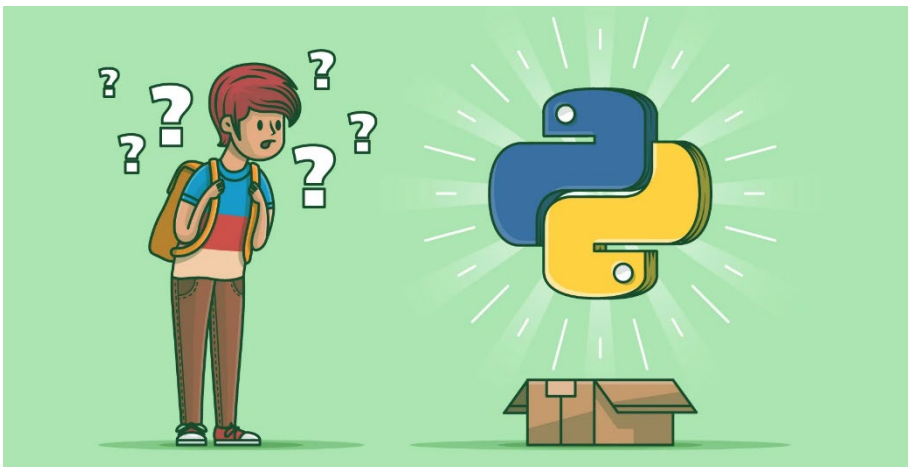
Welcome to the Python Course!

Python is a computer language that may be used to create programs.

Programming languages are just a different way of giving computers instructions to follow. Some of the most common programming languages, such as Java or PHP, are presumably recognisable to you.

Python is getting increasingly popular, and it was just named one of the top ten programming languages to learn in 2018. Indeed, this is one of the reasons why teaching Python programming to children has become so popular.

Python is a programming language that teaches real-world skills. It's utilised in a variety of scenarios to create software and apps. Python is popular among computer programmers because it is simple to read and understand, especially for beginners.



What we will look at to begin learning about python:

Syntax: We have learned in our Fundamentals Course that in computer programming languages, the syntax is essentially the 'spelling and grammar.' A computer cannot understand orders unless they are correctly thought out, much as it is impossible to understand an English phrase without good spelling and grammar. The proper manner to put out commands in programming languages is defined by syntax.

Variables: are a sort of value that may vary in computer programming. In this Python lesson, we'll look at how we may update variables in Python and how it affects the results of our code.

Loops: are a series of instructions that are repeated indefinitely until a specified set of criteria is fulfilled. We'll learn how to tell the difference between a for loop and a while loop in this course.

Some basic concepts to understand before we start:

print() – this writes out whatever is contained in the brackets

input() – this is where the user can write input for whatever is asked

Below is a cheat sheet to help you with the basics:

Python for Beginners – Cheat Sheet

Data types and Collections	Numerical Operators	Comparison Operators
integer 10	+ addition	< less
float 3.14	- subtraction	<= less or equal
boolean True/False	* multiplication	> greater
string 'abcde'	/ division	>= greater or equal
list [1, 2, 3, 4, 5]	** exponent	= equal
tuple (1, 2, 'a', 'b')	% modulus	!= not equal
set {'a', 'b', 'c'}	// floor division	
dictionary {'a':1, 'b':2}		

Operations	Index starts at 0
Strings:	
s[i]	i:th item of s
s[-1]	last item of s
Lists:	
l = []	define empty list
l[i:j]	slice in range i to j
l[i] = x	replace i with x
l[i:j:k]	slice range i to j, step k
Dictionaries:	
d = {}	create empty dictionary
d[i]	retrieve item with key i
d[i] = x	store x to key i
i in d	is key i in dictionary

Logical Operators
and logical AND
or logical OR
not logical NOT

Membership Operators
in value in object
not in value not in object

Conditional Statements
if condition: <code>
elif condition: <code>
else: <code>

Syntax

Indentations

Indentations are simply the spaces you put in all the lines of a block of code following the first line. An example of this could be :

```
if 3 > 2:  
    print("three is greater than two!")
```

Notice how the second line is further in than the first line; this is what we call indentation. All lines in that block of code would need the same indentation.

Comments

Comments are used to describe a code segment and what it is doing. They are not actual codes and are used only for human understanding. To spot comments, they will begin with a ' # ' in Python.

```
#This code will write Hello World.  
print("Hello, World!")
```

As you can see above, the comment explains the code to help readers understand what is going on. This becomes more valuable the more significant the amount of code.

Variables

Variables can be used to assign words values; for example

```
x = 1  
y = "Hello, World!"
```

variables don't need to be assigned a data type but can be assigned any data type such as integer, float, string, etc.

```
x = str(1) # x will be '1'  
y = int(1) # y will be 1  
z = float(1) # z will be 1.0
```

as you can see, specifying the data type will change the formatting of the value.

If you are unsure of what data type a variable is,, simply type

```
print(type(x)) # would write class int for integer
```

variable names are case sensitive so the code below will be two different variables

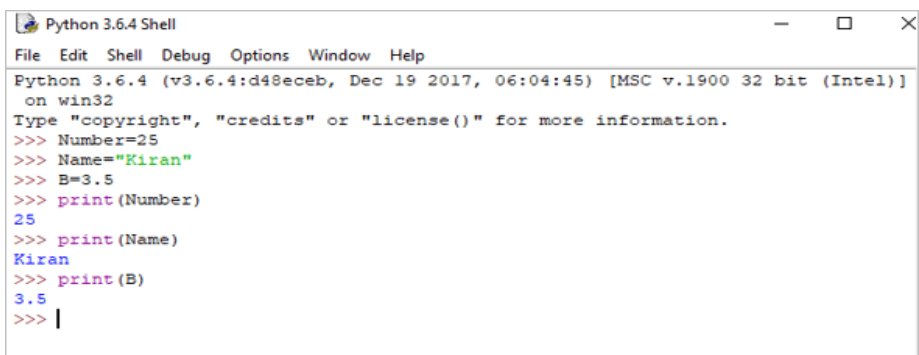
```
a = 10
```

```
A = "Hello"
```

A variable can have a short name (such as x and y) or a longer name (such as age, carname, or totalvolume).

Variables in Python have the following rules:

- The name of a variable must begin with a letter or the underscore character.
- A number cannot be the first character in a variable name.
- Only alpha-numeric characters and underscores (A-z, 0-9, and _) are allowed in variable names.
- Case matters when it comes to variable names (age, Age and AGE are three different variables)



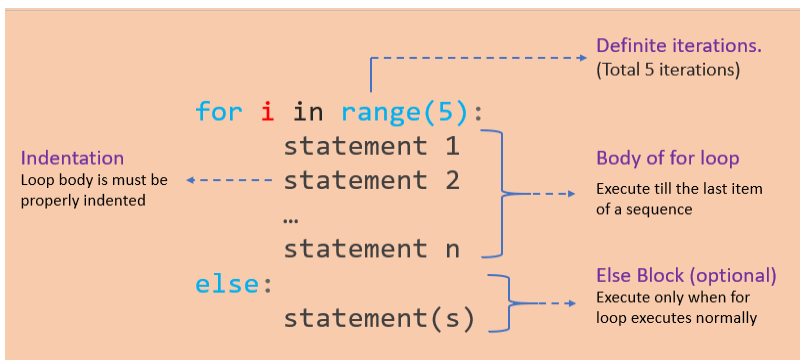
```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> Number=25
>>> Name="Kiran"
>>> B=3.5
>>> print(Number)
25
>>> print(Name)
Kiran
>>> print(B)
3.5
>>> |
```

Loops

1. **for** loops:

A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

With the **for** loop, we can execute a set of statements, once for each item in a list, tuple, set etc.

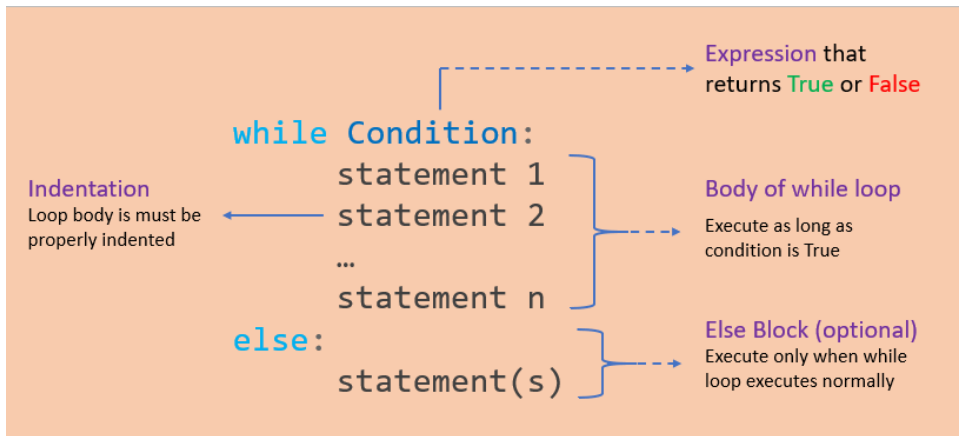


Example:

```
In [7]: #iterate through a list  
colors = ["red", "green", "yellow", "black"]  
for x in colors:  
    print(x)  
  
#iterate through a string  
s = "red"  
for x in s:  
    print(x)  
  
red  
green  
yellow  
black  
r  
e  
d
```

2. `while` loops:

With the `while` loop, we can execute a set of statements as long as a condition is true. This is similar to telling the computer to do something or a set of instructions ‘`while`’ the condition that you’ve provided is true.



Example:

The screenshot shows a code editor with the following Python code:

```
1 i = 5
2 while i >= 1:
3     print(i)
4     i = i - 1
5
```

A dashed green arrow points from the loop body to a list of numbers 5, 4, 3, 2, 1, which are also enclosed in an orange box. A callout box with an orange border contains the text:

This is a “**while** loop”. It repeats code until the expression evaluates **False**.