# Learn Python Programming

## Course outline for module 1:

| |
|---|
| An Introduction to Python, A Brief History of Python Python Versions, Installing Python |
| Environment Variables, Executing Python from the Command Line, IDLE, Editing Python Files |
| Python Documentation, Getting Help, Dynamic Types, Python Reserved Words, Naming Conventions |
| Basic Python Syntax, Basic Syntax Comments, String Values, String Methods The format Method, String Operators |
| Numeric Data Types, Conversion Functions Simple Input and Output The % Method, The print Function |

Python is a powerful high-level, object-oriented programming language created by Guido van Rossum.

It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

This is a comprehensive guide on how to get started in Python, why you should learn it and how you can learn it.

## What is Python (Programming)? - The Basics

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

## History of Python

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

## Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

## Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

## Release Dates of Different Versions

| Version | Release Data |
|---|---|
| **Python 1.0 (first standard release)** | January 1994 |
| **Python 1.6 (Last minor version)** | September 5, 2000 |
| **Python 2.0 (Introduced list comprehensions)** | October 16, 2000 |
| **Python 2.7 (Last minor version)** | July 3, 2010 |
| **Python 3.0 (Emphasis on removing duplicative constructs and module)** | December 3, 2008 |
| **Python 3.5 (Last updated version)** | September 13, 2015 |

**Features of Python Programming**

1. **A simple language which is easier to learn**
   Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.
   If you are a newbie, it's a great choice to start your journey with Python.
2. **Free and open-source**
   You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.
   Python has a large community constantly improving it in each iteration.
3. **Portability**
   You can move Python programs from one platform to another, and run it without any changes.
   It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
4. **Extensible and Embeddable**
   Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
   This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
5. **A high-level, interpreted language**
   Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
   Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.
6. **Large standard libraries to solve common tasks**
   Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using `import MySQLdb` .
   Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
7. **Object-oriented**
   Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
   With OOP, you are able to divide these complex problems into smaller sets by creating objects.

## Applications of Python

### Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS.

Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

## Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on.

Also, the language is heavily used in machine learning, data mining and deep learning.

## Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must.

However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

## Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies.

It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use syntax.

## 4 Reasons to Choose Python as First Language

1. **Simple Elegant Syntax**

   Programming in Python is fun. It's easier to understand and write Python code. **Why?** The syntax feels natural. Take this source code for an example:

```
a = 2
b = 3
sum = a + b
print(sum)
```

   Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

2. **Not overly strict**

   You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

   Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. **Expressiveness of the language**

   Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

## Run Python on Your Operating System

### Install and Run Python in Windows

1. Go to Download Python page on the official site and click **Download Python 3.6.0** (You may see different version name).
2. When the download is completed, double-click the file and follow the instructions to install it. When Python is installed, a program called IDLE is also installed along with it. It provides graphical user interface to work with Python.
3. Open IDLE, copy the following code below and press enter.
4. 
```
print("Hello, World!")
```
5. To create a file in IDLE, go to **File > New Window** (Shortcut: **Ctrl+N**).
6. Write Python code (you can copy the code below for now) and save (Shortcut: **Ctrl+S**) with **.py** file extension like: hello.py or your-first-program.py
```
print("Hello, World!")
```
7. Go to **Run > Run module** (Shortcut: **F5**) and you can see the output. Congratulations, you've successfully run your first Python program.

### Your First Python Program

Often, a program called "Hello, World!" is used to introduce a new programming language to beginners. A "Hello, World!" is a simple program that outputs "Hello, World!".

However, Python is one of the easiest language to learn, and creating "Hello, World!" program is as simple as writing `print("Hello, World!")`. So, we are going to write a different program.

### Program to Add Two Numbers

```
# Add two numbers
num1 = 3
num2 = 5
sum = num1+num2
print(sum)
```

### How this program works?

**Line 1:** # Add two numbers

Any line starting with # in Python programming is a comment.

Comments are used in programming to describe the purpose of the code. This helps you as well as other programmers to understand the intent of the code. Comments are completely ignored by compilers and interpreters.

**Line 2:** num1 = 3

Here, `num1` is a variable. You can store a value in a variable. Here, 3 is stored in this variable.

**Line 3:** num2 = 5

Similarly, 5 is stored in `num2` variable.

**Line 4:** sum = num1+num2

The variables `num1` and `num2` are added using `+` operator. The result of addition is then stored in another variable `sum`.

**Line 5:** print(sum)

The `print()` function prints the output to the screen. In our case, it prints 8 on the screen.

## Few Important Things to Remember

To represent a statement in Python, newline (enter) is used. The use of semicolon at the end of the statement is optional (unlike languages like C/C++, JavaScript, PHP). In fact, it's recommended to omit semicolon at the end of the statement in Python.

Instead of curly braces { }, indentations are used to represent a block.

```
im_a_parent:
 im_a_child:
     im_a_grand_child
 im_another_child:
     im_another_grand_child
```

Python is a terrific language. The syntax is simple and code length is short which makes is easy to understand and write.

If you are getting started in programming, Python is an awesome choice. You will be amazed how much you can do in Python once you know the basics.

It's easy to overlook the fact that Python is a powerful language. Not only is it good for learning programming, it's also a good language to have in your arsenal. Change your idea into a prototype or create games or get started with data Science, Python can help you in everything to get started.

## How to Get Started With Python?

**Learn how to install Python on your operating system and different ways to run it. Also, you will learn to write "Hello, World!" program in Python**

Python is a cross-platform programming language, meaning, it runs on multiple platforms like Windows, Mac OS X, Linux, Unix and has even been ported to the Java and .NET virtual machines. It is free and open source.

Now there are various ways to start Python.

**1. Immediate mode**

---

Typing `python` in the command line will invoke the interpreter in immediate mode. We can directly type in Python expressions and press enter to get the output.

```
>>>
```

is the Python prompt. It tells us that the interpreter is ready for our input. Try typing in `1 + 1` and press enter. We get `2` as the output. This prompt can be used as a calculator. To exit this mode type `exit()` or `quit()` and press enter.

**2. Script mode**

---

This mode is used to execute Python program written in a file. Such a file is called a **script**. Scripts can be saved to disk for future use. Python scripts have the extension `.py`, meaning that the filename ends with `.py`.

For example: `helloWorld.py`

To execute this file in script mode we simply write `python helloWorld.py` at the command prompt.

**Integrated Development Environment (IDE)**

We can use any text editing software to write a Python script file.

We just need to save it with the `.py` extension. But using an IDE can make our life a lot easier. IDE is a piece of software that provides useful features like code hinting, syntax highlighting and checking, file explorers etc. to the programmer for application development.

Using an IDE can get rid of redundant tasks and significantly decrease the time required for application development.

IDEL is a graphical user interface (GUI) that can be installed along with the Python programming language and is available from the official website.

Hello World Example

Now that we have Python up and running, we can continue on to write our first Python program.

Type the following code in any text editor or an IDE and save it as `helloWorld.py`

```
print("Hello world!")
```

Now at the command window, go to the loaction of this file. You can use the `cd` command to change directory.

To run the script, type `python helloWorld.py` in the command window. We should be able to see the output as follows:

```
Hello world!
```

If you are using PyScripter, there is a green arrow button on top. Press that button or press `Ctrl+F9` on your keyboard to run the program.

In this program we have used the built-in function `print()`, to print out a string to the screen. String is the value inside the quotation marks, i.e. `Hello world!` . Now try printing out your name by modifying this program.

Congratulations! You just wrote your first program in Python.

As we can see, it was pretty easy. This is the beauty of Python programming language.

**Python Keywords and Identifier**

**You will learn about keywords (reserved words in Python) and identifiers (name given to variables, functions etc).**

## Python Keywords

**Keywords are the reserved words in Python.**

We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.

In Python, keywords are case sensitive.

There are 33 keywords in Python 3.3. This number can vary slightly in course of time.

All the keywords except `True`, `False` and `None` are in lowercase and they must be written as it is. The list of all the keywords are given below.

| **Keywords in Python programming language** | | | | |
|---|---|---|---|---|
| **False** | class | finally | is | return |
| **None** | continue | for | lambda | try |
| **True** | def | from | nonlocal | while |
| **and** | del | global | not | with |

|  |  |  |  |  |
|---|---|---|---|---|
| **as** | elif | if | or | yield |
| **assert** | else | import | pass |  |
| **break** | except | in | raise |  |

**Rules for writing identifiers**

1. Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_). Names like `myClass`, `var_1` and `print_this_to_screen`, all are valid example.
2. An identifier cannot start with a digit. `1variable` is invalid, but `variable1` is perfectly fine.
3. Keywords cannot be used as identifiers.

```
global = 1
```

4. Identifier can be of any length.

**Things to care about**

Python is a case-sensitive language. This means, `Variable` and `variable` are not the same. Always name identifiers that make sense.

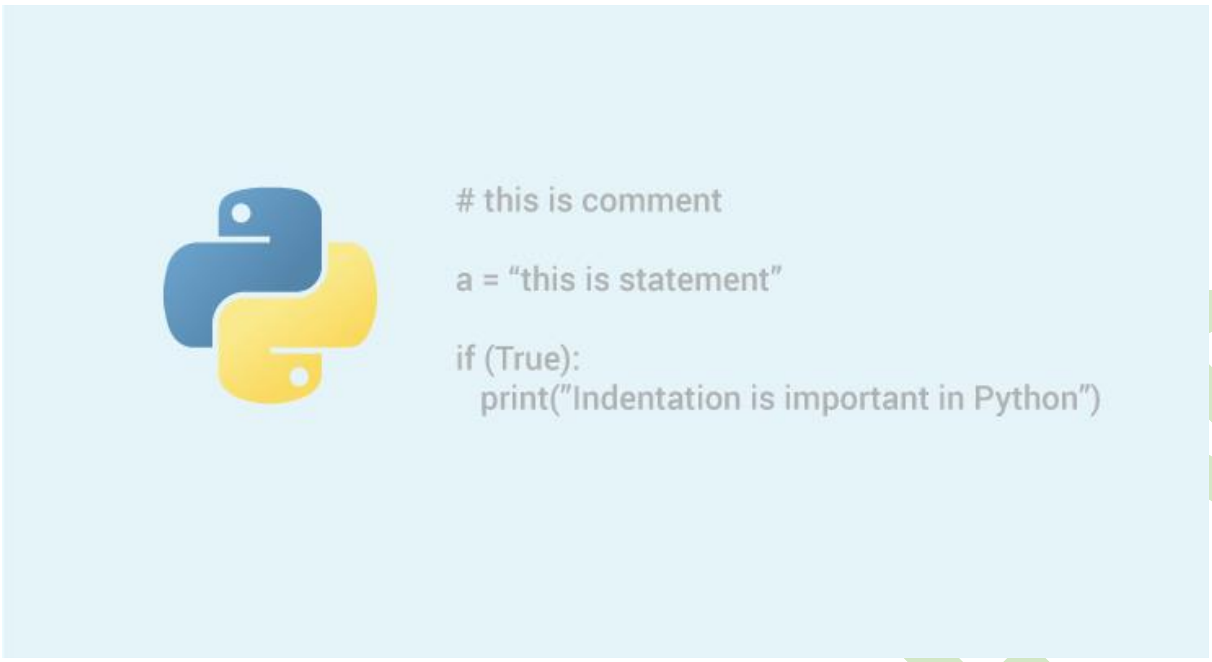
While, `c = 10` is valid. Writing `count = 10` would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.

Multiple words can be separated using an underscore, `this_is_a_long_variable`.

We can also use camel-case style of writing, i.e., capitalize every first letter of the word except the initial word without any spaces. For example: `camelCaseExample`

**Python Statement, Indentation and Comments**

**In this article, you will learn about Python statements, why indentation is important and use of comments in programming.**



```
# this is comment

a = “this is statement”

if (True):
  print(”Indentation is important in Python”)
```

## Python Statement

Instructions that a Python interpreter can execute are called statements. For example, `a = 1` is an assignment statement. `if` statement, `for` statement, `while` statement etc. are other kinds of statements which will be discussed later.

### **Multi-line statement**

In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\). For example:

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

This is explicit line continuation. In Python, line continuation is implied inside parentheses ( ), brackets [ ] and braces { }. For instance, we can implement the above multi-line statement as

```
a = (1 + 2 + 3 +
    4 + 5 + 6 +
    7 + 8 + 9)
```

Here, the surrounding parentheses ( ) do the line continuation implicitly. Same is the case with [ ] and { }. For example:

```
colors = ['red',
          'blue',
          'green']
```

We could also put multiple statements in a single line using semicolons, as follows

```
a = 1; b = 2; c = 3
```

## Python Indentation

Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.

```
for i in range(1,11):
    print(i)
    if i == 5:
        break
```

The enforcement of indentation in Python makes the code look neat and clean. This results into Python programs that look similar and consistent.

Indentation can be ignored in line continuation. But it's a good idea to always indent. It makes the code more readable. For example:

```
if True:
    print('Hello')
    a = 5
```

and

```
if True: print('Hello'); a = 5
```

both are valid and do the same thing. But the former style is clearer.

Incorrect indentation will result into `IndentationError`.

## Python Comments

Comments are very important while writing a program. It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out. You might forget the key details of the program you just wrote in a month's time. So taking time to explain these concepts in form of comments is always fruitful.

In Python, we use the hash (#) symbol to start writing a comment.

It extends up to the newline character. Comments are for programmers for better understanding of a program. Python Interpreter ignores comment.

```
#This is a comment
#print out Hello
print('Hello')
```

## **Multi-line comments**

If we have comments that extend multiple lines, one way of doing it is to use hash (#) in the beginning of each line. For example:

```
#This is a long comment
#and it extends
#to multiple lines
```

Another way of doing this is to use triple quotes, either `'''` or `"""`.

These triple quotes are generally used for multi-line strings. But they can be used as multi-line comment as well. Unless they are not docstrings, they do not generate any extra code.

```
"""This is also a
perfect example of
multi-line comments"""
```

**Python Strings**

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, `'''` or `"""`.

```
s = "This is a string"
s = '''a multiline
```

Like list and tuple, slicing operator [ ] can be used with string. Strings are immutable.

s = 'Hello world!'

# s[4] = 'o'

print("s[4] = ", s[4])

# s[6:11] = 'world'

print("s[6:11] = ", s[6:11])

# Generates error

# Strings are immutable in Python

s[5] ='d'

**Conversion between data types**

We can convert between different data types by using different type conversion functions like int(), float(), str() etc.

```
float(5)
5.0
```

Conversion from float to int will truncate the value (make it closer to zero).

```
int(10.6)
10
int(-10.6)
-10
```

Conversion to and from string must contain compatible values.

```
float('2.5')
2.5
str(25)
'25'
int('1p')
Traceback (most recent call last):
  File "<string>", line 301, in runcode
  File "<interactive input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '1p'
```