If you do much work on computers, eventually you find that there’s some task you’d like to automate. For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you’d like to write a small custom database, or a specialized GUI application, or a simple game.

If you’re a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you’re writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you’ve written a program that could use an extension language, and you don’t want to design and implement a whole new language for your application.

Python is just the language for you.

You could write a Unix shell script or Windows batch files for some of these tasks, but shell scripts are best at moving around files and changing text data, not well-suited for GUI applications or games. You could write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program. Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you get the job done more quickly.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

* the high-level data types allow you to express complex operations in a single statement;
* statement grouping is done by indentation instead of beginning and ending brackets;
* no variable or argument declarations are necessary.

Python is extensible: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

By the way, the language is named after the BBC show “Monty Python’s Flying Circus” and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!

Now that you are all excited about Python, you’ll want to examine it in some more detail. Since the best way to learn a language is to use it, the tutorial invites you to play with the Python interpreter as you read.

In the next chapter, the mechanics of using the interpreter are explained. This is rather mundane information, but essential for trying out the examples shown later.

The rest of the tutorial introduces various features of the Python language and system through examples, beginning with simple expressions, statements and data types, through functions and modules, and finally touching upon advanced concepts like exceptions and user-defined classes.

Python is a general purpose high level programming language.

General purpose means:

We can use Python to implement any kind of applications like

* Web applications
* Data Science applications
* Machine learning applications
* Desktop applications
* Database applications
* Networking applications
* Games
* Data Analysis
* AI
* IOT applications

Python is Dynamically Typed Language

Python was developed by integrating lot of features from other languages

* Functional programming features from C
* OOPs from C++
* Scripting language features from Perl and Shell
* Modular programming from Modula-3

Features of Python

1. Simple and easy to learn
2. Freeware and Open source
3. High level programming language
4. Platform independent
5. Portability
6. Dynamically typed
7. Both Procedure oriented and Object oriented
8. Interpreted
9. Extensible (We can integrate code written in other languages with Python)
10. Embedded (Python programs can be embedded in any language or anywhere)
11. Extensive library

Limitations of Python:

1. Performance
2. Mobile applications

Flavors of Python:

1. **CPython** (Standard flavor of Python, It is used to work with C language applications)
2. **Jython** or **JPython** (This is for JAVA applications)
3. **IronPython** (This is developed to work with C# applications)
4. **Pypy** (If high performance is needed we can go for this flavor: Inside Python Virtual Machine there is JIT compiler)
5. **RubyPython** (Can be used for Ruby platform)
6. **AnacondaPython** (To handle large volume of data processing)
7. **Stackless** (Python for concurrency (multithreading) )

Python Versions:

* Python 1.0 @1994
* Python 2.0 @2000
* Python 3.0 @2008
  + Python 3.6 @2016

Python new versions will not provide a backward compatibility to previous versions of Python.

Identifiers:

Any thing which has a name is called an identifier

Ex: Variable names, function names class names etc.

Rules to define identifiers in Python:

1. Alphabet symbols (both uppercase and lowercase) and underscores also
2. Identifier should not start with digits
3. Identifiers are case sensitive
4. Keywords are not allowed to use to define variables
5. If identifier starts with underscore then it indicates it is Private
6. If identifier starts with two underscore then it indicates it is strongly Private
7. If identifier starts with two underscores and ends with two underscores then it indicates it is language specific identifier (special identifier)

Reserved Words:

33 Reserved words are there in Python

* True, False, None
* and, or, not, is
* if, else, elif
* while, for, break, continue, return, in, yield
* try, except, finally, raise, assert
* import, from, as, class, def, pass, global, nonlocal, lambda, del, with

All Reserved words contain only alphabets

Except first 3 remaining reserved words are in lowercase.

Following syntax can print all the keywords:

1. Import keyword
2. keyword.kwlist

Data Types:

1. int
2. float
3. complex
4. bool
5. str
6. bytes
7. bytearray
8. range
9. list
10. tuple
11. set
12. frozenset
13. dict
14. None

Out of these the first 5 data types are inbuilt data types or Fundamental Data types.

Python considers any data type as an object

Following are the generally used inbuilt functions in Python:

print(), type(). Id()

Strings:

In Python for stings there will be positive as well as negative indexes will exist

For example:

Str = “sachin”

Str[0] will print s

Str[-1] will print n