V Skills
Skills for a secure future

Certified Python Developer
Sample Material

V-Skills Certifications

A Government of India
&
Government of NCT Delhi Initiative

V-Skills

Skills for a secure future

1. INTRODUCTION

1.1 Introduction

Python is one of those rare languages which can claim to be both simple and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in.

The official introduction to Python is:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Note - Guido van Rossum, the creator of the Python language, named the language after the BBC show "Monty Python's Flying Circus". He doesn't particularly like snakes that kill animals for food by winding their long bodies around them and crushing them.

1.2 Features of Python

Simple - Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

Easy to Learn - As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

Free and Open Source - Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

High-level Language - When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

Portable - Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.
You can use Python on Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and even PocketPC !

Interpreted - This requires a bit of explanation.
A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

Object Oriented - Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

Extensible - If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

Embeddable - You can embed Python within your C/C++ programs to give 'scripting' capabilities for your program's users.

Extensive Libraries - The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), Tk, and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the 'Batteries Included' philosophy of Python.
Besides the standard library, there are various other high-quality libraries such as wxPython , Twisted, Python Imaging Library and many more.
Python is indeed an exciting and powerful language. It has the right combination of performance and features that make writing programs in Python both fun and easy.

1.3 Why not Perl?

If you didn't know already, Perl is another extremely popular open source interpreted programming language.

If you have ever tried writing a large program in Perl, you would have answered this question yourself! In other words, Perl programs are easy when they are small and it excels at small hacks and scripts to 'get work done'. However, they quickly become unwieldy once you start writing bigger programs.

When compared to Perl, Python programs are definitely simpler, clearer, easier to write and hence more understandable and maintainable. I do admire Perl and I do use it on a daily basis for various things but whenever I write a program, I always start thinking in terms of Python because it has become so natural for me. Perl has undergone so many hacks and changes, that it feels like it is one big (but one hell of a) hack. Sadly, the upcoming Perl 6 does not seem to be making any improvements regarding this.

The only and very significant advantage that I feel Perl has, is its huge CPANlibrary - the Comprehensive Perl Archive Network. As the name suggests, this is a humongous collection of

Perl modules and it is simply mind-boggling because of its sheer size and depth - you can do virtually anything you can do with a computer using these modules. One of the reasons that Perl has more libraries than Python is that it has been around for a much longer time than Python. However this seems to be changing with the growing Python Package Index.

1.4 Why not Ruby?

If you didn't know already, Ruby is another popular open source interpreted programming language.

If you already like and use Ruby, then I would definitely recommend you to continue using it.
For other people who have not used it and are trying to judge whether to learn Python or to learn Ruby, then I would recommend Python, purely from an ease-of-learning perspective. I personally found it hard to grasp the Ruby language, but for people who understand Ruby, they all praise the beauty of the language. Unfortunately, I am not as lucky.

1.5 What Programmers Say

You may find it interesting to read what great hackers like ESR have to say about Python:
Eric S. Raymond is the author of "The Cathedral and the Bazaar" and is also the person who coined the term Open Source. He says that Python has become his favorite programming language. This article was the real inspiration for my first brush with Python.

Bruce Eckel is the author of the famous Thinking in Java and Thinking in C++ books. He says that no language has made him more productive than Python. He says that Python is perhaps the only language that focuses on making things easier for the programmer. Read the complete interview for more details.

Peter Norvig is a well-known Lisp author and Director of Search Quality at Google (thanks to Guido van Rossum for pointing that out). He says that Python has always been an integral part of Google. You can actually verify this statement by looking at the Google Jobs page which lists Python knowledge as a requirement for software engineers.