**SESSION - 1**

**WHY USE PYTHON**

1. Software Quality

* Python’s focus on readability, coherence, and software quality.
  + Python code is designed to be readable, and hence reusable and maintainable much more so than traditional scripting languages.
* Python has deep support for more advanced software reuse mechanisms such as object-oriented (OO) and function programming.

1. Developer productivity

* Python boosts developer productivity many times beyond compiled or statically typed languages such as C, C++, and Java.
* Python code is typically one-third to one-fifth the size of equivalent C++ or Java code. This means there is less to type, less to debug, and less to maintain after the fact.
* Python programs also run immediately, without the lengthy compile and link steps required by some other tools, further boosting programmer speed.

 It is deliberately optimized for speed of development—its simple syntax, dynamic typing, lack of compile steps, and built-in toolset allow programmers to develop programs in a fraction of the time needed when using some other tools.

1. Program portability

* Most Python programs run unchanged on all major computer platforms.
* Porting Python code between Linux and Windows, for example, is usually just a matter of

copying a script’s code between machines.

1. Support libraries

* Python comes with a large collection of prebuilt and portable functionality, known as the standard library. This library supports an array of application-level programming tasks, from
  + text pattern matching to network scripting.

In addition, Python can be extended with both homegrown libraries and a vast collection of

third-party application support software.

1. Component integration - Product customization and extension tool

* Python code can:
  + @ invoke C and C++ libraries,
  + @ can be called from C and C++ programs,
  + @ can integrate with Java and .NET components,
  + @ can communicate over frameworks such as COM and Silverlight,
  + @ can interface with devices over serial ports, and
  + @ can interact over networks with interfaces like SOAP, XML-RPC, and CORBA

**WHAT CAN YOU DO WITH PYTHON**

1. System Programming
   * Python’s built-in interfaces to operating-system services make it ideal for writing portable, maintainable system-administration tools and utilities.
   * Python programs can search files and directory trees, launch other programs, do parallel processing with processes and threads, and so on.

Support for:

* + environment variables,
  + files,
  + sockets,
  + pipes,
  + processes,
  + multiple threads,
  + regular expression
  + pattern matching,
  + command-line arguments,
  + standard stream interfaces,
  + shell-command launchers,
  + filename expansion,
  + zip file utilities, XML and JSON parsers, CSV file handlers, and more.

1. GUI's

* Python comes with a standard object-oriented interface to the Tk GUI API called tkinter (Tkinter in 2.X) that allows Python programs to implement portable GUIs with a native look and feel.

1. Internet Scripting

* Python comes with standard Internet modules that allow Python programs to perform a wide variety of networking tasks, in client and server modes.

Scripts can communicate over sockets:

* extract form information sent to server-side CGI scripts;
* transfer files by FTP;
* parse and generate XML and JSON documents;
* send, receive, compose, and parse email;
* fetch web pages by URLs;
* parse the HTML of fetched web pages;
* communicate over XML-RPC, SOAP, and Telnet; and more.

Python’s libraries make these tasks remarkably simple.

1. Component Integration
2. Database Programming

* There are Python interfaces to all commonly used relational database systems—Sybase, Oracle, Informix, ODBC, MySQL, PostgreSQL, SQLite, and more.

1. Rapid Prototyping
2. Numeric and Scientific Programming

**DISADVANTAGES**

Only significant universal downside to Python is that,

* its execution speed may not always be as fast as that of fully compiled and lower-level languages such as C and C++.