Lesson 1: Introduction

Python is a high-level, interpreted, interactive and object-oriented programming language. Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages

The current version of Python (3.x) was released in 2008. This version is not compatible with the prior version (2.x). This course will be based on the syntax of Python version 3.x.

Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python's features include-

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows a student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode, which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features. A few are listed below-

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Installing Python

Discuss in groups and find out the best way / form of python installation

Starting Python

There are three different ways to start Python-

1. **Interactive Interpreter**

You can start Python from a command-line interpreter or shell window. This is achieved by executing the **python** command from a shell prompt. Then you can start coding right away using the interactive interpreter.

2. **Script from the Command-line**

A Python script can be executed at the command line by invoking the interpreter on your application. `Eg  C:>python test.py   #     Windows/DOS`

While on Linux /Unix platforms always ensure the file permission mode allows execution.

3. **Integrated Development Environment**

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python. Examples are **PyCharm**, **Visual Studio Code**, Sublime Text, Vim, Atom, **Jupyter Notebook**, Eclipse + PyDev + LiClipse, GNU Emacs, **Spyder**, Thonny.

Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9). Punctuation characters such as @, $, and % are not allowed within identifiers. Python is a case sensitive programming language.

Here are naming conventions for Python identifiers-

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strong private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language- defined special name.

Reserved Words

and,	as,	assert,	break,	class,	continue,	def,	del
elif,	else,	except,	exec,	finally,	for,	from,	global,
if,	import,	in,	is,	lambda,	not,	or,	pass,
print,	raise,	return,	try,	while,	with,	yield	

Indentation / Blocks

Python does not use braces ({}) to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

```
gender="Male"
if gender=="Male":
      print ("Gender is Male")
      print ("do something …")
else:
      print ("Not Male")
      print ("do a different …")
```

Quotations

Single, double, and triple (''' or """) quotes are used to denote string literals, as long as the same type of quote starts and ends the string. Eg.

```
word = 'word'
```

```
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is made up of multiple lines and
sentences."""
```

Comments

A hash sign (#) that is not inside a string literal is the beginning of a comment. All characters after the #, up to the end of the physical line, are part of the comment and the Python interpreter ignores them.

```
e.g. # My First comment
```

Statements

Statements in Python typically end with a new line. The semicolon (;) allows multiple statements on a single line. The line continuation character (\) allows one statement to break to the next line. However, statements contained within the [], {}, or () brackets do not need to use the line continuation character.