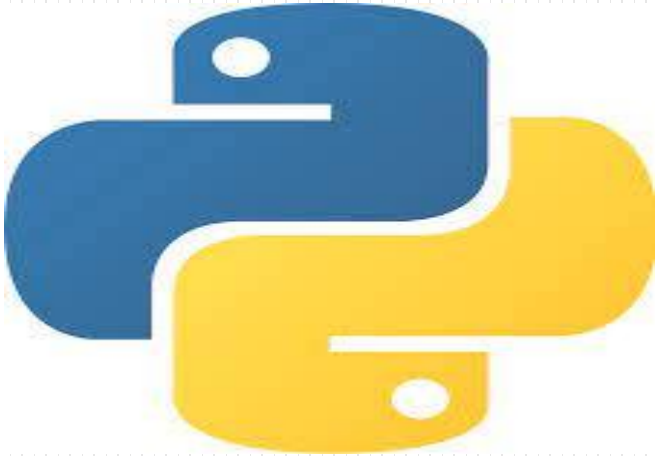
Lecture 0

INT108:: PROGRAMMING IN PYTHON (Program practice)



Course Overview

L T P :

3	0	2

Credit: 4

Text Book:

Fundamentals of Python –First Program Second edition by KENNETH A.LAMBERT,CENGAGE

References Book:

Python Programming: Using Problem solving approach by REEMA THAREJA, OXFORD UNIVERSITY PRESS

Introduction to Programming using Python by Y. DANIEL LIANG, PEARSON

Marks Breakup

- Marks Breakup:

Activity	Marks
Attendance	5
Continuous Assessment (CA)*	65
End-Term Practical (ETP)	30

CA and ETP

CA

Programming Practice using CodeTantra/ or EBox	40
Mini Project	15
Best 1 out of 2 MCQ Based Test	10

ETP

Programming Challenge/Hackathon based

Course Assessment Model

Programming Practice using E-Box/or Codetantra 40

Based on number of questions attempted

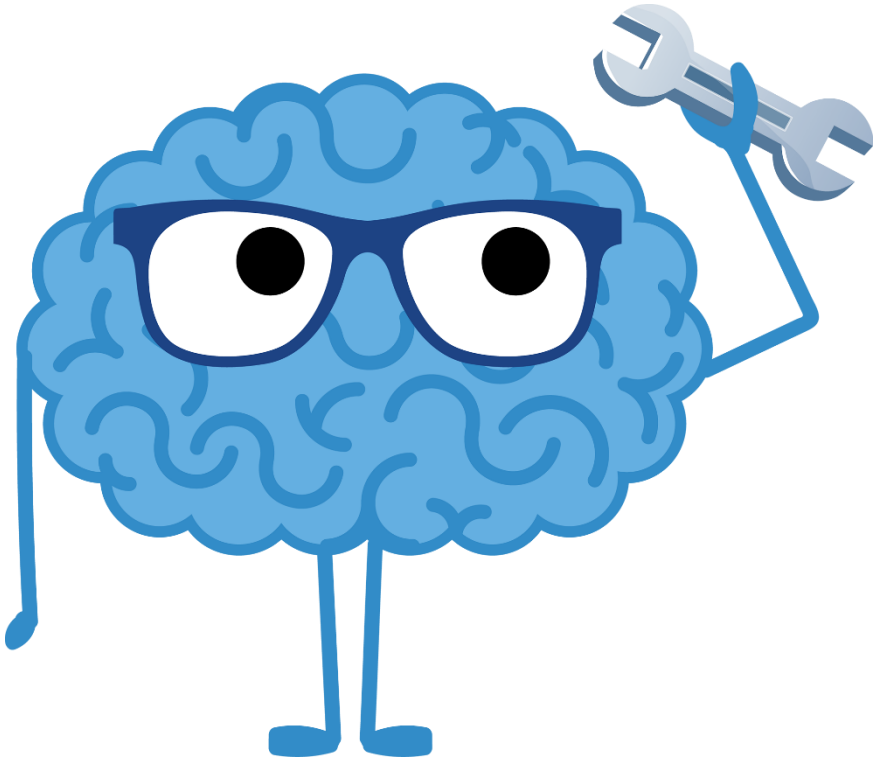
0-50%	->0 Marks
51%	-> 1 Marks
74%	->24 Marks
90-100%	-> 40 Marks

For each % of attempted question after 50%, student will be awarded 1 mark (max up to 45 marks)

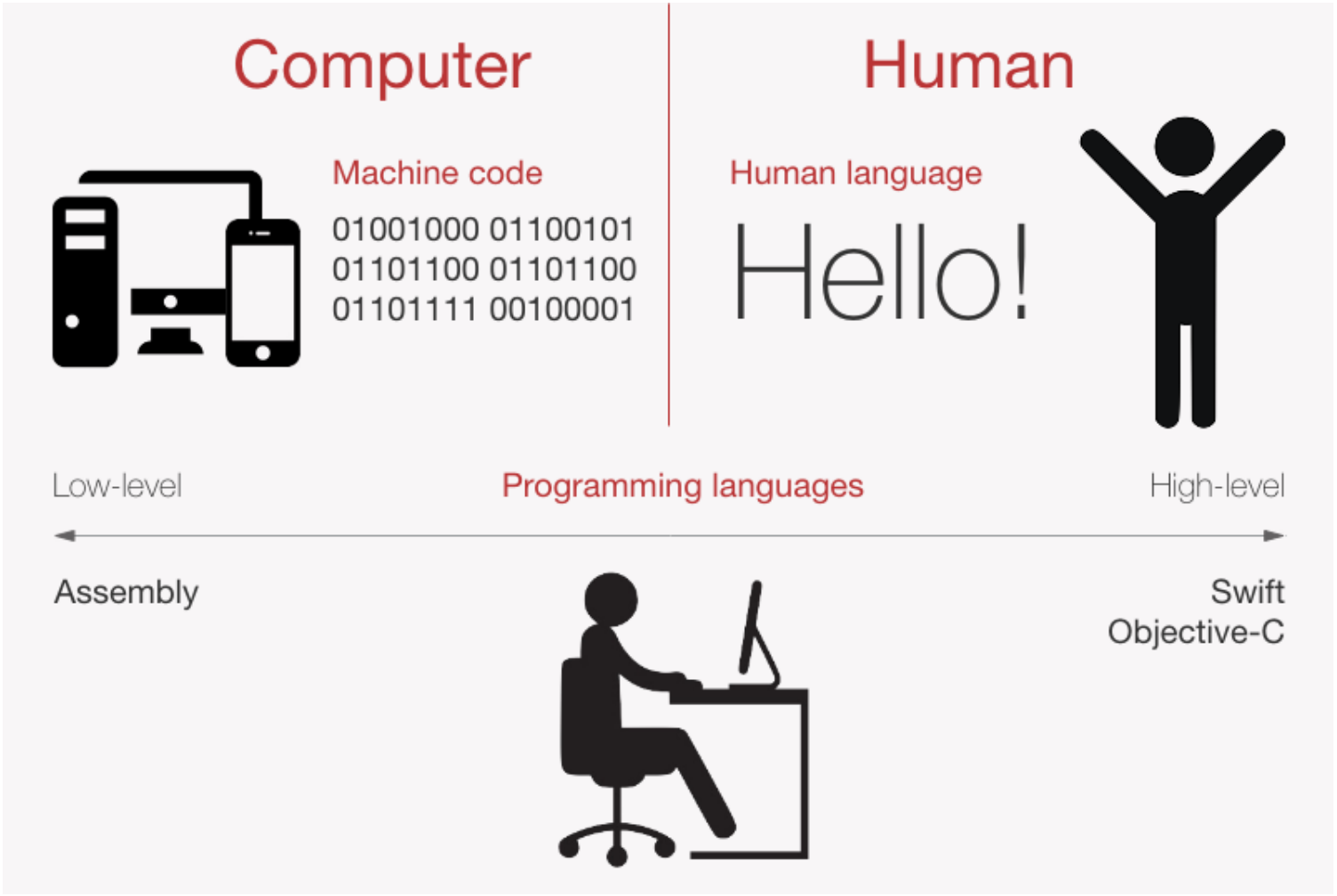
Course Outcomes

- define the installation of python environment and basics of Python language.
- apply the condition and iteration statements for evaluating the appropriate alternates.
- apply to formulate Regular Expressions and use them for Pattern Matching .
- construct the core data structures like lists, dictionaries, tuples and sets in Python to store, process and sort the data.
- Apply the concepts of Object-oriented programming as used in Python using encapsulation, polymorphism, and inheritance .
- Apply the external modules for creating and writing data to excel files and inspect the file operations to navigate the file systems.

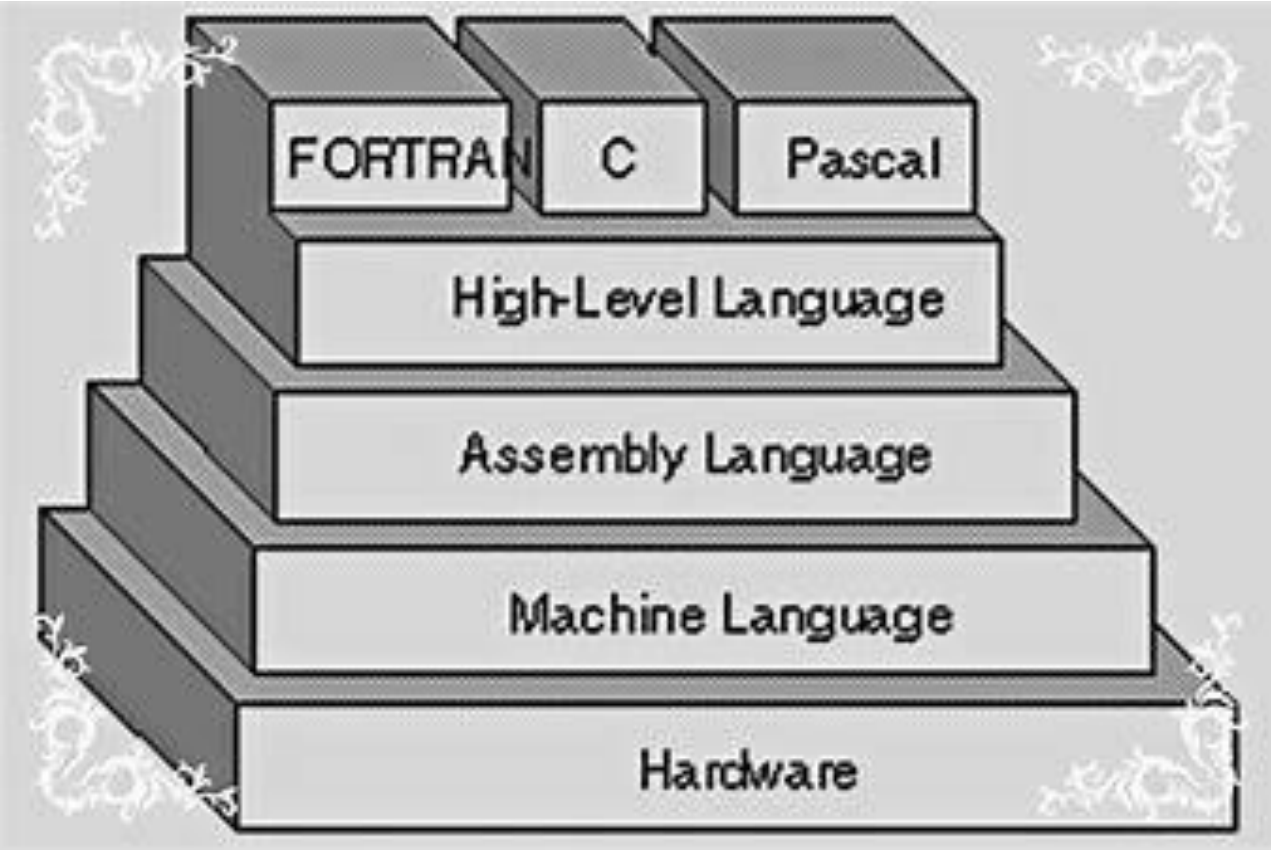
Why languages?



Computer Language vs. Human Language



Different Computer Languages



Language Translators

- Computers are electronic devices that can only understand machine-level binary code (0/1 or on/off), and it is extremely difficult to understand and write a program in machine language, so developers use human-readable high level and assembly instructions.
- To bridge that gap, a translator is used, which converts high-level instructions to machine-level instructions (0 and 1).
- The translator is a programming language processor that converts a high-level or assembly language program to machine-understandable low-level machine language without sacrificing the code's functionality.
- There are 3 types:

  1) Compiler

  2) Interpreter

  3) Assembler

Compiler

- The compiler is a language translator program that converts code written in a human-readable language, such as high-level language, to a low-level computer language, such as assembly language, machine code, or object code, and then produces an executable program.

Characteristics of Compiler:

- Source code is converted to machine code before runtime. So, code execution at runtime is faster.
- Takes a lot of time to analyze and process the program. The compiling process is complicated.
- But Program execution is Fast
- Cannot create an executable program when there is a compile type error in the program.

Interpreter

- The Interpreter's source code is transformed into machine code at run time. The compiler, however, converts the code to machine code, i.e. an executable file, before the program starts.
- The interpreter program executes directly line by line by running the source code. So, it takes the source code, one line at a time, and translates it and runs it by the processor, then moves to the next line, translates it and runs it, and repeats until the program is finished.

Characteristics of Interpreter:

- Spends less time converting to machine code.
- No compilation stage is present in the interpreter while generating machine instructions.
- Program execution is slower because it gets converted to machine code at runtime.
- Easy for debugging and finding errors.

Compiler vs Interpreter

Compiler	Interpreter
Translate High-level language program into machine code before runtime	Translate High-level language program into machine code at runtime
The compiler needs a lot of time for the whole source code to be analyzed	It takes less time for the source code to analyze
The overall program execution time is relatively faster.	Overall program execution time is relatively slower.
The compiler only generates an error message only after scanning the whole program. And all the errors are shown at the same time.	Interpreter only shows one error at a time and if solved and again after interpreting the code then shows the next error if exists.
Debugging is relatively more difficult since there can be an error anywhere in the code.	It is easier to debug since it continues to translate the program until the error is fixed. Show only one error at a time, and if solved then shows the next error if exists.
The compiler compiles the code before execution.	Compilation and execution take place simultaneously.
Difficult error detection and removal	Easy for error detection and removal
The programming language that uses Compiler: C, C++, C#, Scala	The programming language that uses Interpreters: Python, Perl, Ruby, PHP

Assembler

Assembler converts code written in assembly language into machine-level code. Assembly language contains machine opcode mnemonics so that assemblers translate from mnemonics to machine code

Assembly Example

```
          .MODEL SMALL              ;select small model
          .DATA                     ;start data segment
DATA1     DW     2000H              ;define DATA1
DATA2     DW     3000H              ;define DATA2
          .CODE                     ;start code segment
          .STARTUP                  ;start program
          LEA SI,DATA1              ;address DATA1 with SI
          MOV DI,OFFSET DATA2       ;address DATA2 with DI
          MOV BX,[SI]               ;exchange DAT1 with DATA2
          MOV CX,[DI]
          MOV [SI],CX
          MOV [DI],BX
          .EXIT
          END
```

History of Python

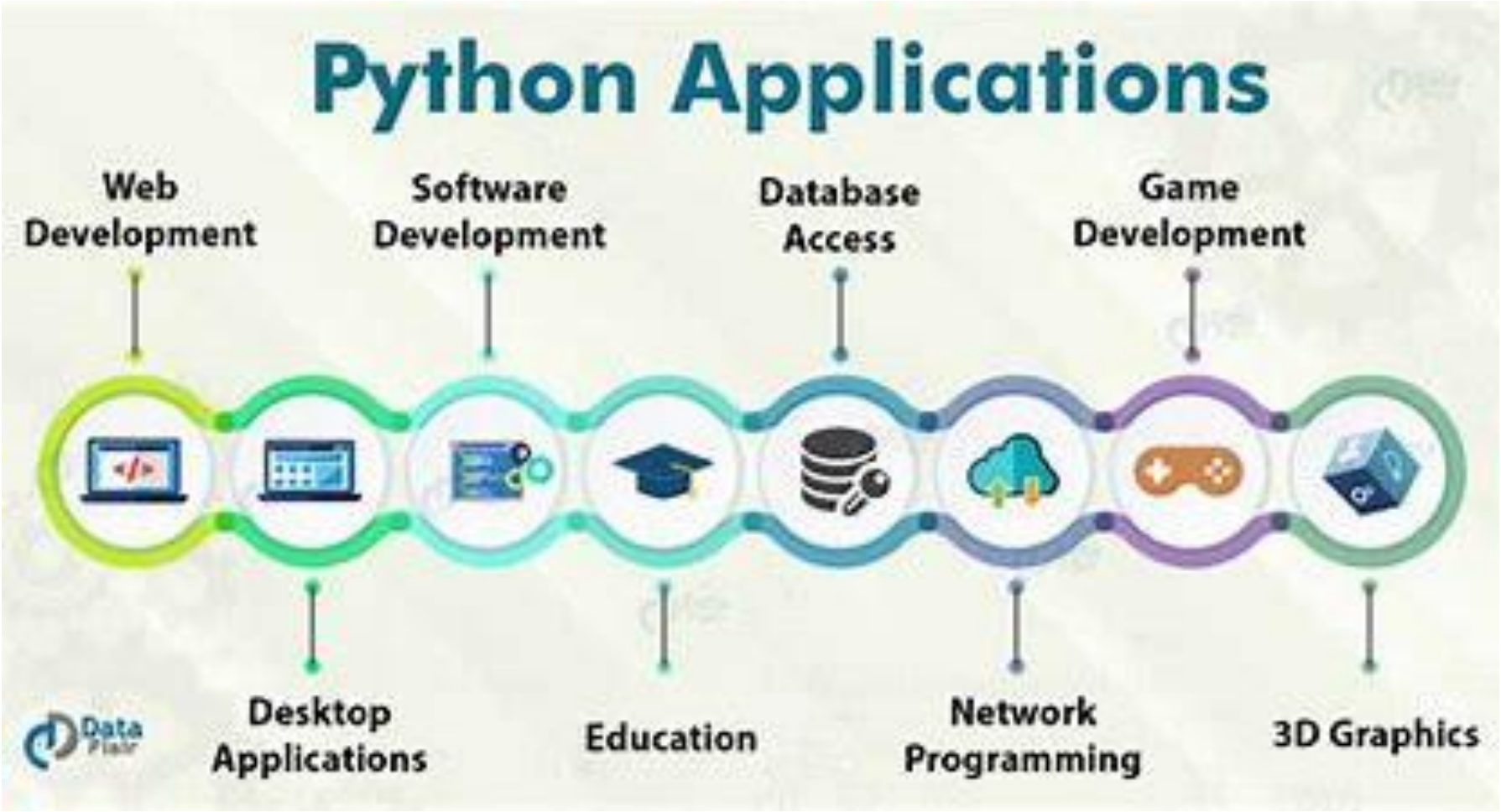
- Python is a widely-used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991(first release Python 0.9.0) and developed by Python Software Foundation.
- It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
- Work on Python was started in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL(Set language), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.
- Latest version: 3.10.6
- Download link:

  https://www.python.org/downloads/

Why Python

- **Python** is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming.
- Following are some of the features:

- **Free and Open Source**
- **Easy to code**
- **Easy to read**
- **Object Oriented language**
- **GUI Programming Support**
- **Easy to debug**
- **It is a portable language**
- **It is an integrated language**
- **Large standard library**
- **Dynamically typed language**
- **Frontend and Backend development**


Python Applications
Web Development
Software Development
Database Access
Game Development
Desktop Applications
Education
Network Programming
3D Graphics
Data Flair

16 Famous Companies that uses PYTHON

Overview of Unit 1

Unit 1

- **Setting up your Programming Environment:** Python versions, Python on windows, running a 'Hello World' program

- **Variables, Expression and Statements:** Naming and using variables, Avoiding Name Error when using variables, Values and types, variables, variables name and keywords, statements, operators and operand, order of operations, operations on string, composition and comments

Overview of Unit 2

If

Elif

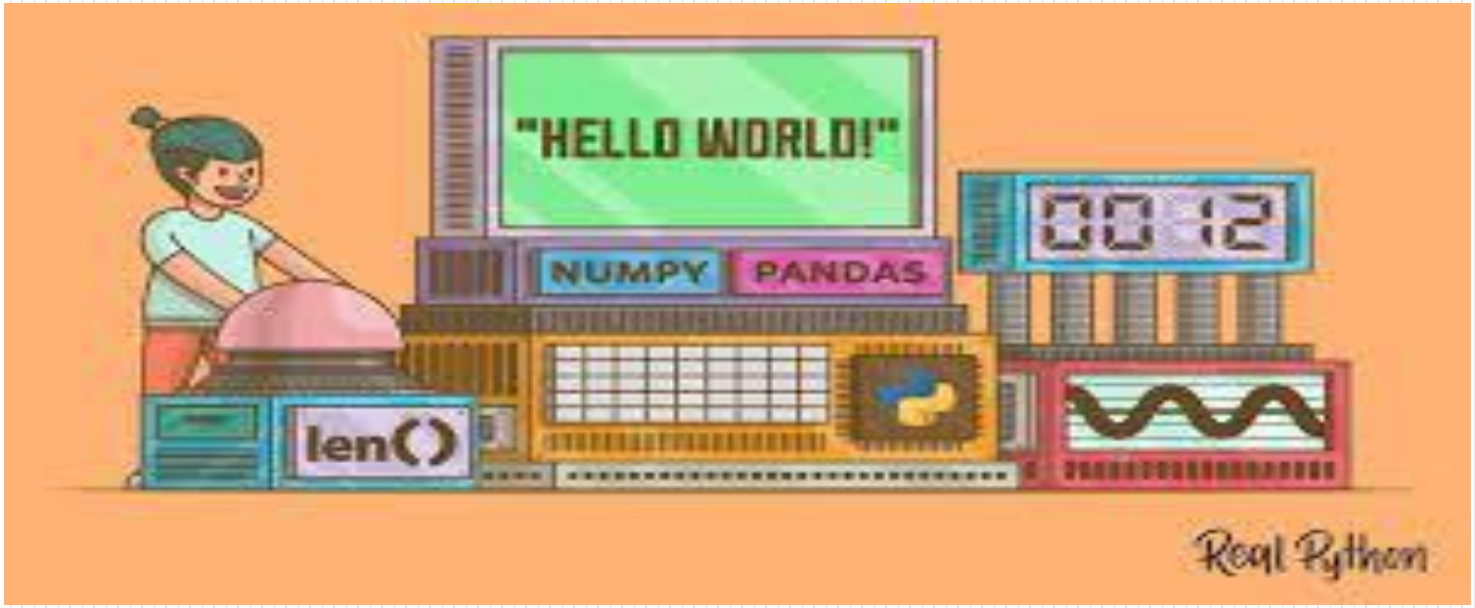
Else Ladder

Unit 2

- **Conditional statements:** modulus operator, Boolean expressions, logic operators, conditional, alternative execution, nested conditionals

- **Iterative statements:** while statements, for loop statement, Nested for, Nested while, Random numbers in loops, encapsulation and generalization

Overview of Functions and Recursion



Unit 3

- **Functions and recursion:** function calls, type conversion and coercion, math functions, adding new function, parameters and argument, recursion and its use

Lists and Tuples

LIST 1→

LIST2→

LIST3→

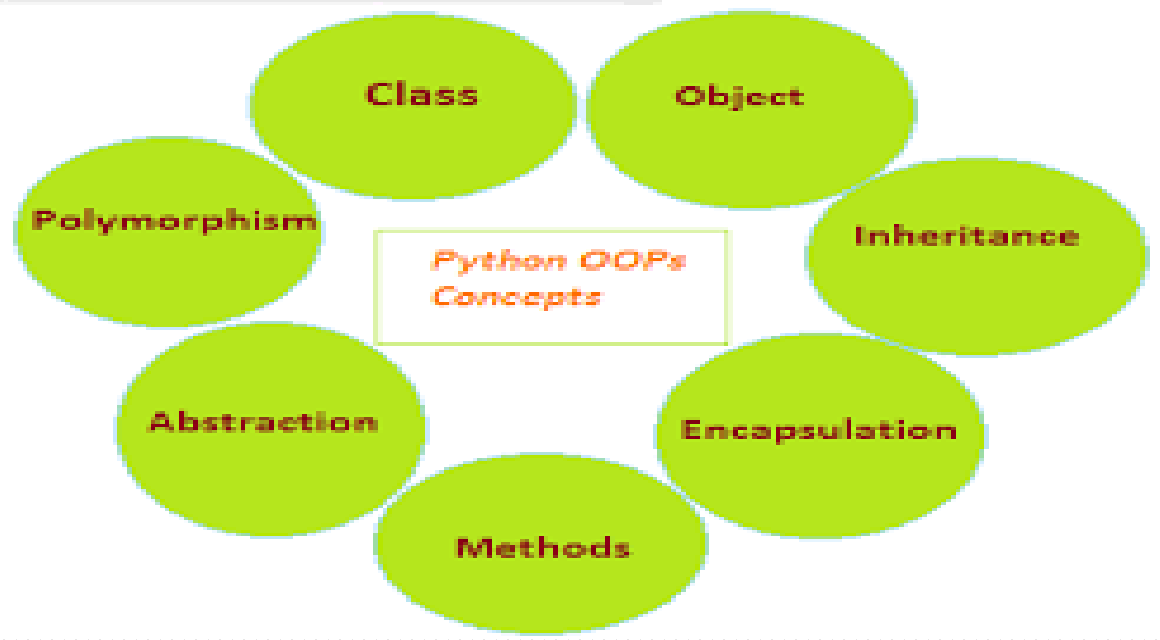
Unit 4

- **String:** string a compound data type, length, string traversal, string slices, comparision, find function, looping and counting

- **Lists:** list values, length, membership, operations, slices, deletion, accessing elements, list and for loops, list parameters and nested list.

- **Tuples and Dictionaries:** mutability and tuples, tuple assignment, tuple as return values, dictionaries operations and methods, sparse matrices, aliasing and coping

OOP


Class
Object
Polymorphism
Inheritance
Python OOPs Concepts
Abstraction
Encapsulation
Methods

Unit 5

- **Classes and objects:** Creating classes, creating instance objects, accessing attributes

- **Object oriented programming terminology:** Class Inheritance, Overriding Methods, Data Hiding, Function Overloading

Unit 6

- **Files and Exceptions:** text files, writing variables, Reading from a file, writing to a file, directories, pickling, handling the zero Division error exception, using try-except blocks, The else block, Handling the File Not found error exception

- **Regular Expressions** – Concept of regular expression, various types of regular expressions, using match function, Web Scraping by using Regular Expressions

Different Python IDEs and Code Editors

IDE	Size in MB	Developed in
PyCharm	BIG	JAVA, PYTHON
Spyder	BIG	PYTHON
PyDev	MEDIUM	JAVA, PYTHON
Idle	MEDIUM	PYTHON
Wing	BIG	C, C++, PYTHON

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing


Thank
You