# Python Programming

# Course Overview

**Introduction**

- Overview
- Why python
- Versions

**Setup**

- Python installation
- Environment setup
- Git configuration

**Data Structures**

- Basics
- Collections

**Operators**

- Basic operators
- Chained comparison
- Mathematical, logical etc

**Statements**

- Control statements
- Loops

**Functions**

- Named vs Lambda
- Nested functions
- Scopes

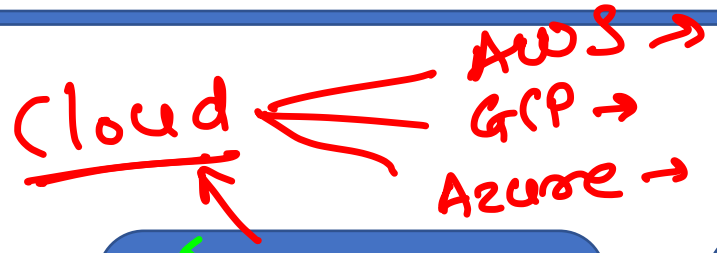
**Advanced features**

- List comprehension
- Generators
- Decorators

**OOP**

- Class vs Object
- Inheritance
- Special methods

# Course Overview




**Modules and packages**

- Module creation
- Custom packages
- Package ecosystem

**Functional Programming**

- Map, filter and reduce
- Environment setup
- Git configuration

**Web Services**

REST DB

- Fundamentals of WS
- WS using Flask

**Numpy**

- Requirements
- Data types
- Data manipulation

**Pandas**

- Series and Data Frames
- Data manipulation

**Data Analysis**

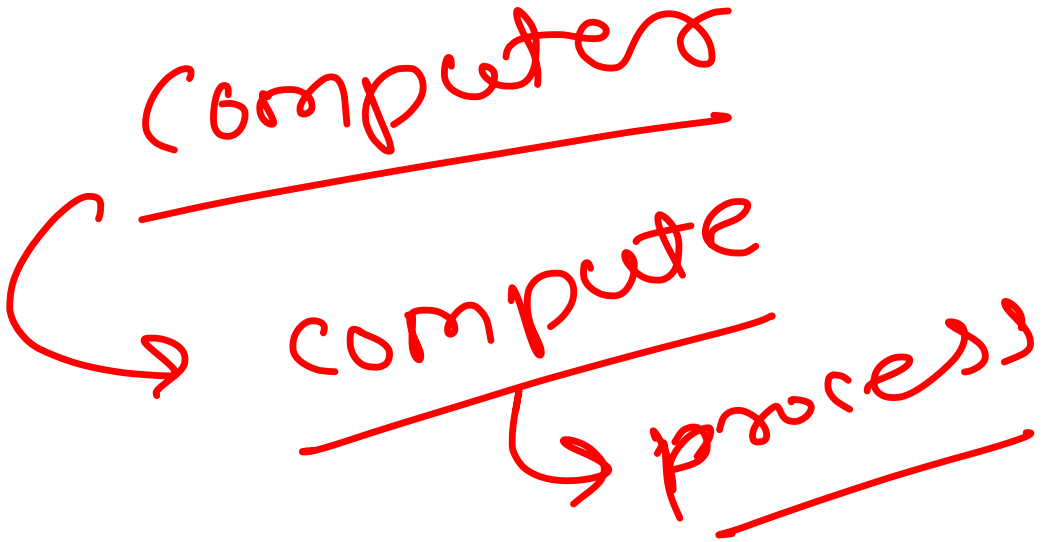
- Fundamentals
- Requirements
- Using numpy and Pandas

**Data visualization**

- Fundamentals
- Using Matplotlib
- Using Seaborn

**Testing**

- Fundamentals
- BeautifulSoup
- Selenium




# Language Fundamentals

# What is a computer language?

- A language is a medium to interact with computer
- A language is used to solve a problem by giving some solution (writing a program)
- Types
  - **Machine languages**: the code written in 0s and 1s and can be directly executed by CPU
  - **Assembly languages**: a language closely related to one or a family of machine languages, and which uses mnemonics to ease writing
  - Programming languages
    - **General purpose languages**: a programming language that is broadly applicable across application domains, and lacks specialized features for a particular domain
    - **Markup languages**: a grammar for annotating a document in a way that is syntactically distinguishable from the text
    - **Stylesheet language**: a computer language that expresses the presentation of structured documents
    - **Configuration language**: a language used to write configuration files
    - **Query language**: a language used to make queries in databases and information systems
    - **Scripting languages**: a language used to write simple to complex scripts

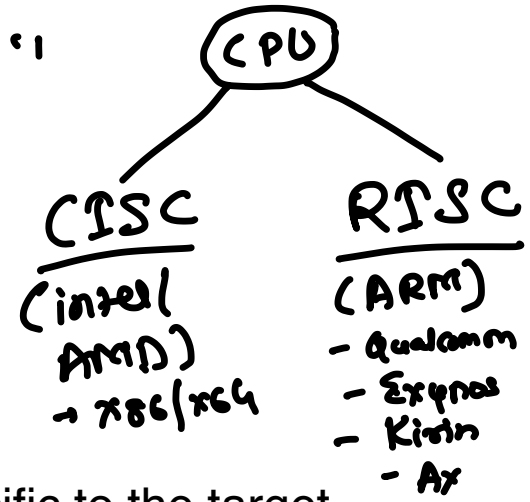
# Low Level vs High Level Languages

| Low Level Language | High Level Language |
|---|---|
| It is programmer friendly language | It is a machine friendly language |
| High level language is less memory efficient | Low level language is high memory efficient |
| It is easy to understand | It is tough to understand |
| It is simple to debug | It is complex to debug comparatively |
| It is simple to maintain | It is complex to maintain comparatively |
| It is portable | It is non-portable |
| It can run on any platform | It is machine-dependent |
| It needs compiler or interpreter for translation | It needs assembler for translation |
| E.g. Python | E.g. Assembly |

# Compiled language → code executes using one executable

- A programming language which involves an executable to execute the logic instead of executing the source code directly
- E.g. C, C++, Pascal, Objective-C, Swift etc.
- Stages
  - Pre-processing
    - Used to preprocess the code
    - Comments removal →
    - Code expansion → macro #define
    - Conditional compilation

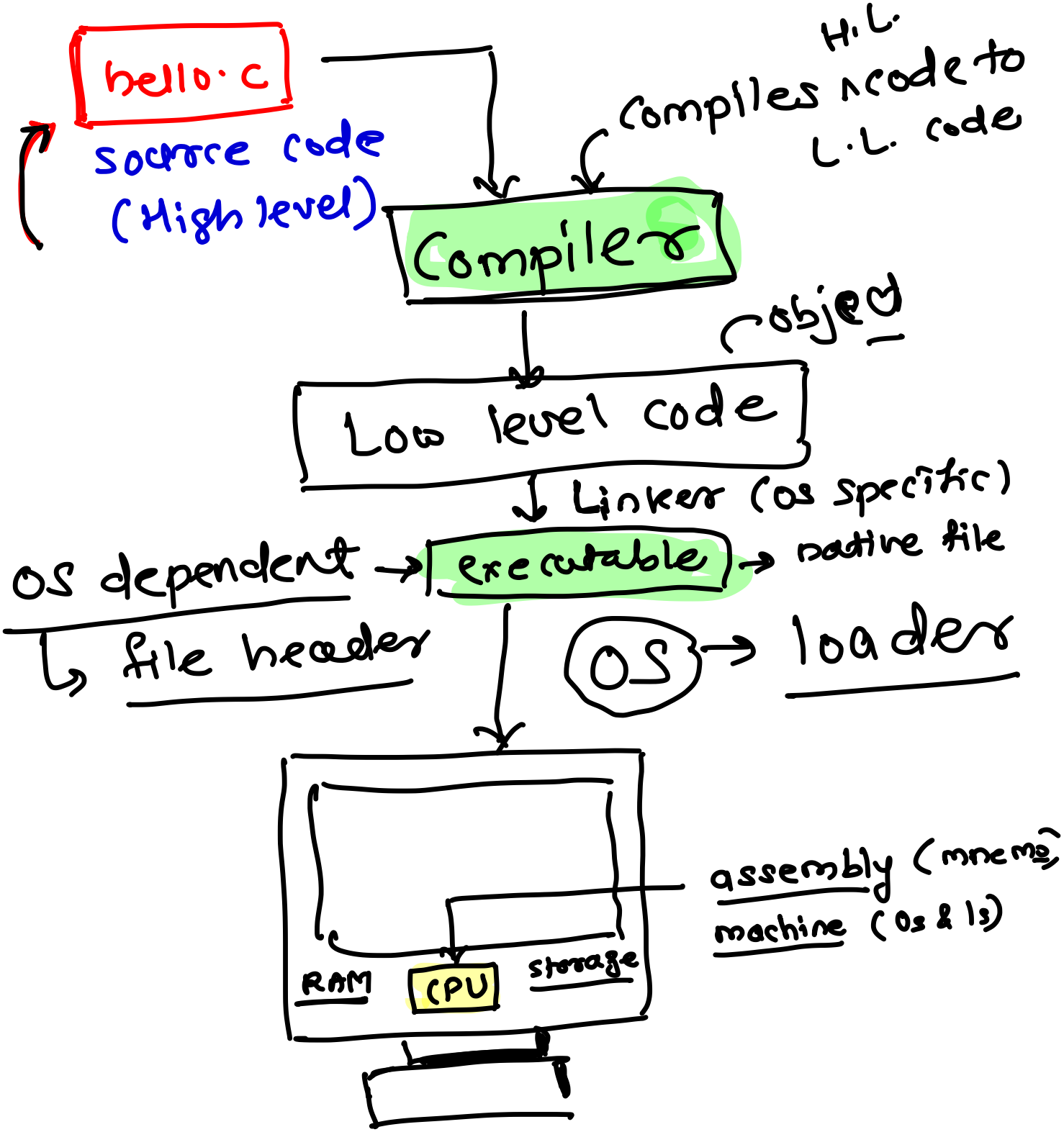
#include " "
headers
modules
package




  - Compiling
    - Compiler is used to translate pre-processed code into assembly instructions specific to the target processor architecture

- object code -asm
- eg ls

  - Assembling
    - Assembler is used to translate the assembly instructions to object code
  - Linking
    - Linker used in this stage re-arranges the code and insert missing files to emit an executable



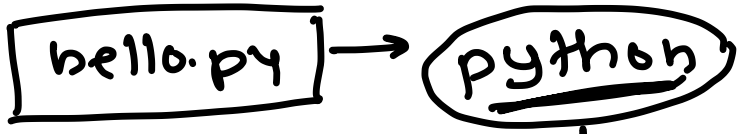

Executable file format

1. macOS : Mach-O ←
2. Linux : ELF
3. Windows : PE / COFF ←

magic number
↳ os specific

# Interpreted Language

- A language mostly executes source code directly and freely, without previously compiling a programm into an executable
- E.g. Python, JavaScript, Perl, BASIC etc.
- Interpretation
  - Interpreter used in the language executes the high level source code directly




Interpreter

↳ interpretes the code

↳ reads the code line by line from top to bottom

executes

→ slower than compiled languages

→ OS independent (write once & run anywhere)

→ source code gets executed

# Python

# Introduction

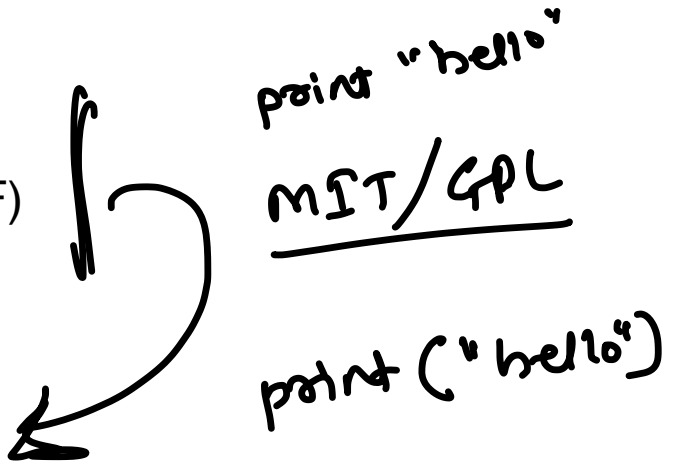
- Python is a general purpose, high level and interpreted language
- It is dynamically typed and garbage collected language
- It supports programming paradigms like
  - Procedural programming
  - Object oriented programming
  - Functional programming
  - Aspect oriented programming (metaprogramming and magic metaobjects)
- Heavily dependent on indentation to create blocks which makes it very easy to read the code
- It has more than 130,000 packages included with wide range of functionality
  - Graphical user interfaces
  - Web frameworks
  - Networking
  - Automation
  - Web scraping

# History

- Python was conceived in the late 1980s by Guido Rossum in Netherlands
- It is considered as a successor to the ABC language (itself inspired by SETL)
- Its implementation began in December 1989 ← 1.0 → 1991
- Van Rossum continued as Python's lead developer until July 12, 2018
- When he announced his permanent vacation from his responsibilities, a team of five members was developed in Jan 2019 to lead the project

# Versions

- Version 0.9.0 [Feb 1991]
  - Having features like classes with inheritance, exception handling, functions etc.
  - One of the major versions of python
- Version 1 [Jan 1994]
  - The major new features included in this release were the functional programming tools like labda, map, filter, reduce
  - The last version released was 1.6 in 2000
- Version 2 [Oct 2000]
  - Introduced features like list comprehension, garbage collection, generators etc.
  - Introduced its own license known as Python Software Foundation License (PSF)
  - The last version released was 2.7.16 in Mar 2019
- Version 3 [Dec 2008]
  - Python 3.0 is also called "Python 3000" or "Py3K"
  - It was designed to rectify fundamental design flaws in the language
  - Python 3.0 had an emphasis on removing duplicative constructs and modules
  - The last version released was 3.7.4 in Jul 2019

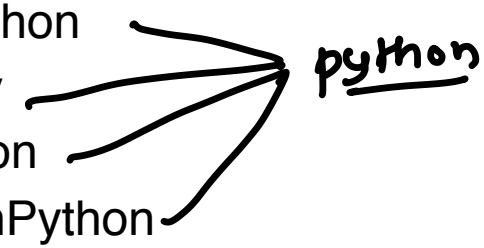


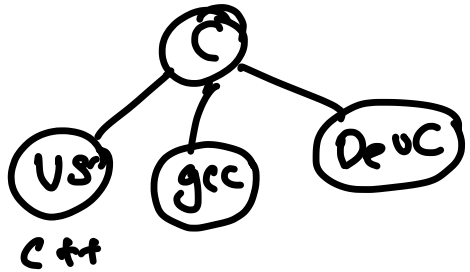
# Environment Setup

- Windows:
  - Download python installer
  - Download pycharm community edition
- Ubuntu
  - Install python3 on using apt installer
  - Download pycharm community edition
- macOS
  - Download python 64 bit installer
  - Download pycharm community edition

# What have you downloaded

- Python has many versions like
  - CPython
  - Pypy
  - Jython
  - InronPython




- The standard version is called as CPython which is what most of us get when we download from the official site
- Henceforth when we say we are using python, it will be CPython

# Hello world program explained

- Python is not a compiled language: misconception
- CPython has a compile base
- Python compiles into bytecodes
- CPU can not understand bytecodes
- We need an interpreter to understand and execute the bytecodes
- This interpreter is nothing but the python virtual machine

# Python Virtual Machine (PVM)

- Written in C
- Compiles the bytecode into machine language
- It emulates the machine or CPU
- Executes bytecodes similar to the way a CPU executes the machine instructions

hello.py → python3

① Lexical analyzer -

- Syntax checking
- tokens (identifiers)

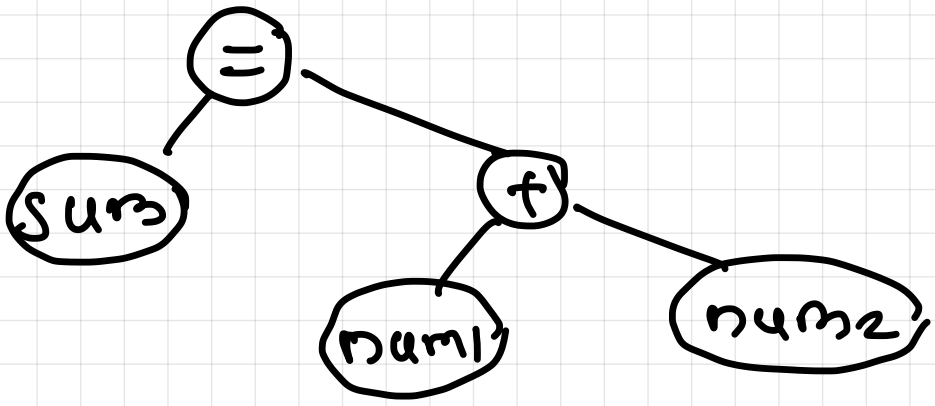

- symbol table →

| symbol | D.T. | Add |
|---|---|---|
| | | |

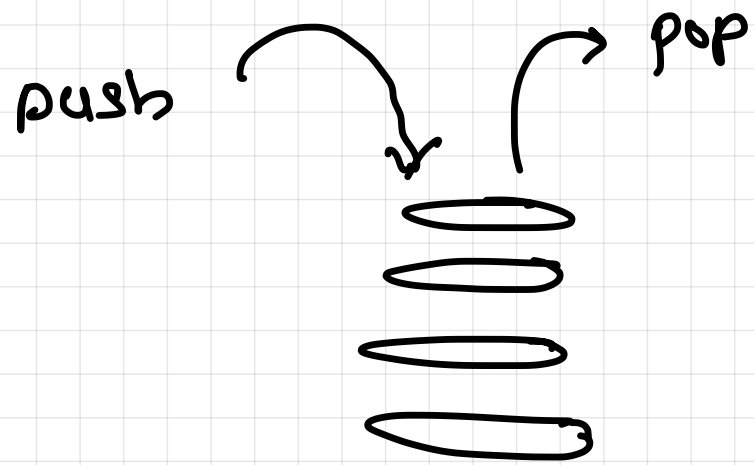
✓

② AST - Abastract Syntax Tree

sum = num1 + num2



③ Stack




④ byte codes → similar to machine code

① CPU does not understand byte code
② It is not meant for CPU
③ It is meant for PVM