WELCOME

Python  Series

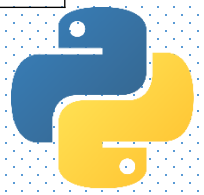
AGENDA

Interpreter Vs Compiler Vs Assembler
Python 2 Vs Python 3
Pypl
Pip
venv
Libraries, Frameworks, Tools

Python  Series

INTERPRETER | COMPILER | ASSEMBLER

Features	INTERPRETER	COMPILER	ASSEMBLER
PURPOSE	Executes source code directly, line by line	Translates source code from a high-level language to machine code	Translates assembly language into machine code
EXECUTION	Executes code directly without producing intermediate machine code	Produces machine code or bytecode before execution begins	Produces machine code for execution later
SPEED	Generally slower, as it translates and executes code line-by-line	Compilation is slow, but execution of compiled code is fast	Fast, as it translates directly to machine code

Python  Series

INTERPRETER | COMPILER | ASSEMBLER

Features	INTERPRETER	COMPILER	ASSEMBLER
MEMORY USAGE	Can be less efficient, as it needs to analyze code on the fly	More efficient at runtime due to optimization during compilation	Efficient, as it generates optimized machine code
ERROR DETECTION	Errors are detected and handled at runtime	Errors are detected during the compilation process, before execution	At the time of assembly
USE CASE	Suitable for scripting, small programs, and rapid development cycles	Ideal for large applications requiring speed and efficiency	Used for system software and critical performance applications
EXAMPLES	Python Interpreter, Ruby Interpreter	GCC for C/C++, Javac for Java	NASM, MASM

PYTHON 2 | PYTHON 3

Features	PYTHON 2	PYTHON 3
Release Date	2000	2008
Print Statement	print "Hello, world!"	print("Hello, world!")
Division of Integers	Dividing two integers performs floor division	Dividing two integers results in a float
Unicode Support	ASCII str type by default, separate unicode type	Unicode (UTF-8) str type by default
Error Handling	except IOError, e:	except IOError as e:
Iterating Items	.iteritems(), .iterkeys(), and .itervalues()	.items(), .keys(), and .values()

Python  Series

PYTHON 2 | PYTHON 3

Features	PYTHON 2	PYTHON 3
Syntax	Less strict syntax rules	Stricter syntax rules (e.g., no mixing of tabs and spaces)
Libraries and Features	Fewer built-in libraries and features	More built-in libraries and modern features, including improved threading, asyncio, etc.
End of Life	January 1, 2020	Still actively developed and maintained
Function Annotations	Not available	Available (provides a standardized way of attaching metadata to function parameters and return values)
2to3 Tool	-	Available to help transition from Python 2 to Python 3

Python  Series

PyPL

- **Full Name**: Python Package Index
- **Purpose**: Official third-party software repository for python. It is used to host and distribute python package to the Python community.
- **Accessibility**: Accessible via package management tools like '**pip**',
- **Contents**: Contains variety of packages (libraries, frameworks, and tools) across different domains (Web Development, Data Analysis, and Machine Learning.

Python  Series

PIP

- **Purpose**: It is a python package installer, a tool for installing and managing python packages from the python package Index (PyPL)

- **Installation**: Comes preinstalled with python version >=3.4.

Commands:

- **pip install package_name –** Installs new package
- **pip install –upgrade package_name –** upgrades the package to latest version.
- **pip uninstall package_name -** removes specific package
- **pip list –** lists down installed packages
- **pip show package_name –** show information about specific package

Python  Series

VENV

- **Full Name**: Stands for Virtual Environment.
- **Purpose**: Create Isolated Virtual Environment, each with its own set of libraries, and package version, separate from system wide python installation.
- **Availability**: Available by default with python version 3.3 or later.
- **Creation**: Can be created in PyCharm while creating the project.
- **Activation Command**:
  - **Windows (cmd) –** 'path\to\venv\Scripts\activate'
  - **MacOS (bash) –** 'source/path/to/venv/bin/activate'
- **Management**: Use 'pip' to install packages locally within the environment.
- **Flexibility**: Allows developers to work on different projects on same machine without risking dependency conflicts.
- **Portability**: Can copied with other machines, ensuring consistency across different setups.

Python Series

LIBRARIES | FRAMEWORKS | TOOLS

LIBRARIES

NumPy

pandas

matplotlib

Requests
http for humans

FRAMEWORKS

python
django

TensorFlow

Flask

PyTorch

TOOLS

jupyter

Streamlit

Python Series

CONGRATULATIONS...

UNIT 2 COMPLETE

Python  Series

UP NEXT...

UNIT 3
PYTHON BASICS

Python  Series