Introduction to python

Introduction to python.

- **Python** is a general purpose, dynamic, high-level, and interpreted programming language.
- It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.
- Python is *easy to learn* yet powerful and versatile scripting language, which makes it attractive for Application Development.
- Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.
- Python supports *multiple programming pattern*, including object-oriented, imperative, and functional or procedural programming styles.
- Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.
- We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to assign an integer value in an integer variable.
- Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

history

- Python was invented by **Guido van Rossum** in 1991 at CWI in Netherland. The idea of Python programming language has taken from the ABC programming language or we can say that ABC is a predecessor of Python language.
- There is also a fact behind the choosing name Python. Guido van Rossum was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name **Python** for his newly created programming language.
- Python has the vast community across the world and releases its version within the short period.

Features of python

- Easy to use and Learn
- Expressive Language
- Interpreted Language
- Object-Oriented Language
- Open Source Language
- Extensible
- Learn Standard Library
- GUI Programming Support
- Integrated
- Embeddable
- Dynamic Memory Allocation
- Wide Range of Libraries and Frameworks

Where?

- Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.
- Data Science
- Date Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

Python Popular Frameworks and Libraries

- Python has wide range of libraries and frameworks widely used in various fields such as machine learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.
- The ***Dynamic Websites*** – ***Server-side programming*** topic is a series of modules that show how to create dynamic websites; websites that deliver customised information in response to HTTP requests.
- **Web development (Server-side) -** Django, Flask, Pyramid, CherryPy
- **GUIs based applications -** Tk, PyGTK, PyQt, PyJs, etc.
- **Machine Learning -** TensorFlow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
- **Mathematics -** Numpy, Pandas, etc.

Index	Object-oriented Programming	Procedural Programming
1.	Object-oriented programming is the problem-solving approach and used where computation is done by using objects.	Procedural programming uses a list of instructions to do computation step by step.
2.	It makes the development and maintenance easier.	In procedural programming, It is not easy to maintain the codes when the project becomes lengthy.
3.	It simulates the real world entity. So real-world problems can be easily solved through oops.	It doesn't simulate the real world. It works on step by step instructions divided into small parts called functions.
4.	It provides data hiding. So it is more secure than procedural languages. You cannot access private data from anywhere.	Procedural language doesn't provide any proper way for data binding, so it is less secure.
5.	Example of object-oriented programming languages is C++, Java, .Net, Python, C#, etc.	Example of procedural languages are: C, Fortran, Pascal, VB etc.

- To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\name>python –version
```

- Execute in command line:

```
C:\Users\name\Desktop\python>python
>>> print(“web programming")
>>> exit()
```

- Exexute by creating a file with .py extension:

```
C:\Users\name\Desktop\python>python first.py
```

Python Indentation

- Indentation refers to the spaces at the beginning of a code line.
- Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
- Python uses indentation to indicate a block of code.

```
if 7 > 3:
    print("Five is greater than two!")
```

- Python comments: explain Python code , make the code more readable , prevent execution when testing code.

```
#This is a comment
print("Hello, World!")
```

- To add a multiline comment you could insert a # for each line:
- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

```
""“ This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Creating Variables

- Python has no command for declaring a variable.A variable is created the moment you first assign a value to it.Variable names are case-sensitive.

```
x = 500
y = “programming"
print(x)
print(y)
```

- If you want to specify the data type of a variable, this can be done with casting.

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

- You can get the data type of a variable with the type() function.

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

- **Variable Names**
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

- **Many Values to Multiple Variables**
- Python allows you to assign values to multiple variables in one line:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

- **One Value to Multiple Variables**
- And you can assign the *same* value to multiple variables in one line:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

- **Unpack a Collection**
- If you have a collection of values in a list, tuple etc. Python allows you extract the values into variables. This is called *unpacking*.

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Output Variables

- The Python print statement is often used to output variables.
- To combine both text and a variable, Python uses the + character:

```
x = “good subject"
print("Python is " + x)
```

- You can also use the + character to add a variable to another variable:

```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

- For numbers, the + character works as a mathematical operator:

```
x = 5
y = 10
print(x + y)
```