What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Features in Python:

1. Easy to code:
Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

2. Free and Open Source:
Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword.

3. Object-Oriented Language:
One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

5. High-Level Language:
Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. Extensible feature:
Python is a **Extensible** language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language.

7. Python is Portable language:
Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language:
Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

What is a framework?

A framework, or software framework, is a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform. For example, a framework may include predefined classes and functions that can be used to process input, manage hardware devices, and interact with system software. This streamlines the development process since programmers don't need to reinvent the wheel each time they develop a new application

A framework is similar to an application programming interface (API), though technically a framework includes an API. As the name suggests, a framework serves as a foundation for programming, while an API provides access to the elements supported by the framework. A framework may also include code libraries, a compiler, and other programs used in the software development process.
Several different types of software frameworks exist. Popular examples include ActiveX and .NET for Windows development, Cocoa for Mac OS X, Cocoa Touch for iOS, and the Android Application Framework for Android. Software development kits (SDKs) are available for each of these frameworks and include programming tools designed specifically for the corresponding framework. For example, Apple's Xcode development software includes a Mac OS X SDK designed for writing and compiling applications for the Cocoa framework.

Django

Django is a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization established as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

Advantages of Django:-

Advantages of Django

Here are the most important reasons why to use Django for custom websites. We think these reasons apply equally well to the startup projects and for business gurus.

#1 – It's a Python Language

As we mentioned above, Django is written in Python. The Python language is truly simple to learn and seems as it was created for newbies.

#2 – Django and Python are Core Solutions in:

- Internet of Things
- Blue chip companies
- IT giants (NASA, Google and more)
- FinTech companies in Silicon Valley

> Learning the language and its popular framework should guarantee you a full employment or, for example, give a possibility to create your own product as an outsourcing company

#3 – Batteries included

Django aims to follow Python's "batteries included" philosophy. It means Django provides a wide range of features and functionalities including:

- Magical ORM;
- Multi-site and multi-language support;
- MVC (Model/View/Control) layout;
- RSS and Atom feeds;
- AJAX support;
- Free API;
- URL routing;
- Easy Database Migrations;
- Session handling;
- HTTP libraries and templating libraries;
- Code Layout (you can plug new capabilities by using applications);
- Default Admin section and more

#4 – Stellar Documentation and Tutorials

You will not have any problems as Django includes benefits such as:

- Requirements and quick start details
- Detailed release notes
- Backwards-incompatible changes
- Online topics and discussion on development and scalability

#5 – Administration Interface

- The administration interface provided by Django is one of the coolest things. It's truly simple to create and it's really one of the key advantages when using the framework. You get a fully featured admin interface from writing only a few lines of code:

#6 – Community

As an open source and available for free online, Django is supported by active volunteers who constantly provide updates and resources on djangoproject.com and on Github (in the last, there are 11,300 Django stars). As well, people in the community are cool and they support each other via:

- Mailing list
- IRC channel
- Blog posts
- Stackoverflow

#7 – Django is Immensely Scalable

One of the nicest advantages of Django is that it can handle traffic and mobile app API usage of more than 400 million+ users helping maximize scalability and minimize web hosting costs. And when talking about hosting, we need to mention that the number of hosts is high and hosting price is relatively cheap and even free.

#8 – Customizable Framework

The developers must have put a lot of work into the architectural layout of using Django for a project. Are you in need of serializer classes? Here it goes, serializing data is a very simple operation: from django.core import serializers.

#9 – Robust Built-In Template System

One of the benefits of Django framework is a built-in template language that facilitates the process of building applications. +1 for this!

#10 – Best Security

For one thing, Django hides your website's source code. The framework has protection against XSS and CSRF attacks, SQL injections, clickjacking, etc. Django notifies of a number of common security mistakes better than PHP (you can count it as one of the main advantages of Django over PHP).

#11 – Future

Django literally grew up largely over the last years. What does that mean? Companies who choose Django over other frameworks can aim attention at what makes their projects exclusive, and pay less attention to general supporting issues in web development or framework upgrade performance. One can be guaranteed that the Django functionalities selected now will continue to operate well together in the future.

Disadvantages of Django

Is there any reason not to use Django framework while developing? Probably not. But it would be unfair not to include some disadvantages as well.

#1 – Regex to specify its URL:

You can create simple and easy-to-read URLs. However, Django uses regex to specify its URL routing patterns and that makes the code larger and creates complicated syntaxes. This is an example straight from the documentation:

#2 – While maintaining backwards compatibility, it's moving far too slowly

The framework has dedicated itself to backwards compatibility. What does that mean? It tends to get bigger and heavier over time. Speed is a valid issue even in Python, but Django has explicitly chosen to designate other stuff. Django worries more about dev productivity and backwards compatibility than its speed.

#3 – Is Django too monolithic?

For sure, and this is the reason why Django maintains large, tightly-coupled product. The framework is monolithic and it pushes you into given patterns, but it is more fun when you

can program yourself — choosing architecture, structure and design patterns. Besides this, components get deployed together.

#4 – It feels like it has too much software for minor projects

Vast functionality and heavy structure are always with you, even if it is not used for simple stuff. A framework is a complex tool for simple usages, so if you ready to face with low flexibility, put your heart and soul into it.

#5 – Template errors fail silently by default

It seems like the framework developers didn't pay attention to mistakes when they stepped to their own class-based views. They are extended via inheritance, that means it will make everyone’s life entangled in the subsequent versions.

#6 – A process only handles a single request at a time

Unluckily, WSGI-based servers cannot be utilized to develop real-time apps, as WSGI protocol is synchronous. WSGI server can handle only one request at a time.