Application Development Lifecycle Phases

1. Requirement Gathering
   1. User Requirements
   2. Business Requirements
   3. Technical Requirements
   4. Determining any constraints and viability
2. Analysis (Verify and Revise)
3. Design (Verify and Revise, Document)
4. Code and Test (unit, performance, integration testing)
5. User Testing and System Testing
6. Production (final tested program -> promote to production -> production repo)
7. Maintenance (upgrades)

Note : Code each functionality in separate files for easy n efficient code maintenance

Web Apps

* Stored on a remote server and delivered over the internet
* A web app requires three components to process a client request:
  + a web server to manage raised requests
  + an app server to execute the requested task
  + a database to store the information needed to complete the task
* API’s are the software components that enables apps to communicate
* All web apps are APIs, but not all APIs are web apps. Both share data between apps, but not all APIs require networks like web apps do.

Static Code Analysis

* Static code analysis, or static analysis, is an application code efficiency verification activity that analyses source code for quality, reliability, and security without executing the code. Static code analysis is an essential part of any application development cycle and is available as a part of multiple frameworks with Python.
* One of the most popular frameworks is PyLint. PyLint basically evaluates the code against compliance with the PEP8 coding style guide and generates comments wherever it finds an issue.
* The PEP8 guidelines for code readability include the following:
  + Four spaces for indentation
  + Blank lines to separate functions and classes
  + Spaces around operators and after commas
* The PEP8 coding conventions for consistency and manageability include:
  + Add larger blocks of code inside functions
  + Name functions and files using lowercase with underscores
  + Name classes using CamelCase
  + Name constants in capital letters with underscores separating words

Unit Testing

* During code development, you will test each unit. The test is performed in two phases.
* In the first phase, you will test the unit on your local system. If the test fails, you will determine the reason for the failure and fix the issue. Then, you will again test the unit.
* After the unit test passes, you will need to test the unit in a server environment, such as the Continuous Integration Continuous Delivery, or CICD, test server. If the unit fails the server test, you will receive the failure details. You will need to determine and fix the issue. Once the unit passes the server test, the unit is integrated into the final code base.

Packaging

* A Python module is a .py file containing Python definitions, statements, functions, and classes.
* A package is a collection of Python modules into a directory with an init.py file.
* A library is a collection of packages, or it can be a single package.
* To create a package, create a folder with the package name, create an empty init.py file, create the required modules.
* In the init.py file, add code to reference the modules needed in the package.
* You can verify the package via the bash terminal.
* After creating the package, you can use it on other scripts if the package folder is in the same directory.
* To create a package, create a folder with the package name, create an empty init.py file, create the required modules. In the init.py file, add code to reference the modules needed in the package. You can verify the package via the bash terminal.