**Flask Introduction**

**Python:**

Before learning Flask, you must have the basic knowledge of Python concepts

**Python Introduction**

* Python Basics
  + Python Introduction and History
  + Python Installation and Environment setup
  + How to run Hello World program in Shell and as well as Terminal/Command Prompt
  + Python Comments

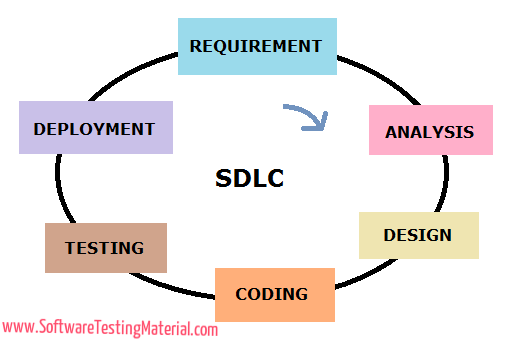
* Python Classes and Objects
  + Classes
    - Definition of Class
    - Create a Class
    - \_\_init\_\_() Function
  + Objects
    - Object Definition
    - self Parameter
    - Modify Object
    - Delete Object

* Python Functions
  + Declaration of Function
    - Required Arguments
    - Keyword Arguments
    - Defaults Arguments
    - Variable length Arguments

* Python Libraries, Packages and Modules
  + Pre-Define Packages
  + User-Define Packages

**Sample Project Demo:**

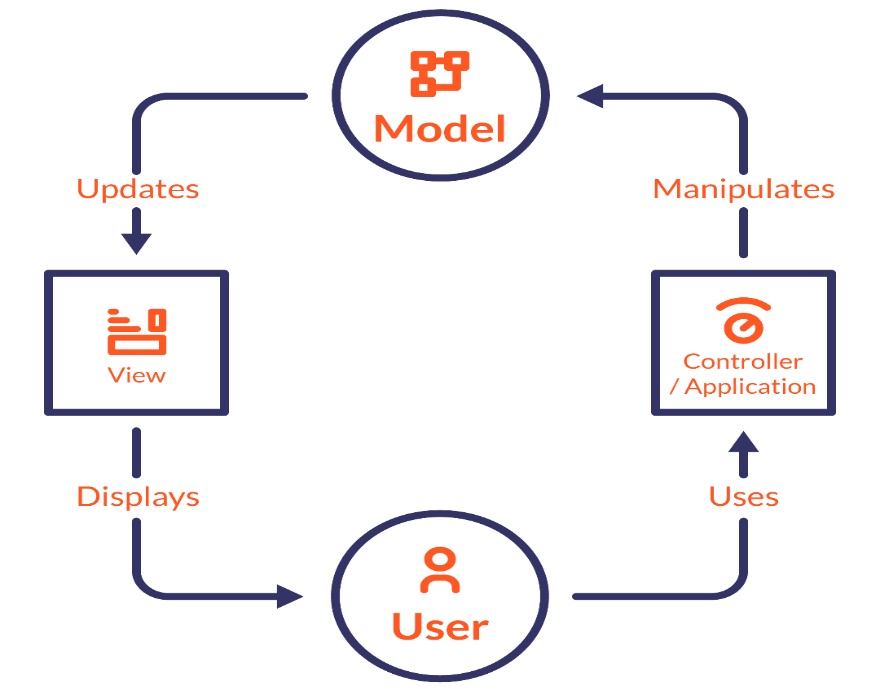
* Sample Project Explanation with real time example
* Explain About **SDLC** (Software Development Life Cycle)
  + Architecture of SDLC



* + Requirements
    - Hardware Requirements
    - Software Requirements
  + Analysis
  + Design
  + Coding
  + Testing
  + Deployment
* What is the purpose of MVC (**Model- View-Controller**)

**MVC:** MVC Stands for “Model-View Controller” is an application design model comprised of three interconnected parts. They include the Model (data), the View (User Interface),and Controller(Processes that handle input).

The MVC model is commonly used for developing modern user interfaces. It is provides that fundamental pieces for designing a program for desktop or mobile, as well as web applications. It works well with object-oriented programming, since the different models, views, and controllers can be treated as objects and reused with in an application.



**Below is a description of each aspect of MVC:**

**Model – Database (Data Layer in OSI Model):**

A model is data used by a program. This may be a database file, file or simple object, such as an icon or a character in a video game.

**View - Frontend (Presentation Layer in OSI Model):**

A view is the means of displaying objects with in an application. Examples include displaying a window or buttons or text within a window. It includes anything that the user can see

**Controller – Programming languages (Application Layer in OSI Model):**

A controller updates both models and views. It accepts [input](https://techterms.com/definition/input) and performs the corresponding update. For example, a controller can update a model by changing the attributes of a character in a video game. It may modify the view by displaying the updated character in the game.

The three parts of MVC are interconnected (see diagram). The view displays the model for the user. The controller accepts user input and updates the model and view accordingly. While MVC is not required in application design, many [programming languages](https://techterms.com/definition/programming_language) and [IDEs](https://techterms.com/definition/ide) support the MVC architecture, making it an common choice for developers.

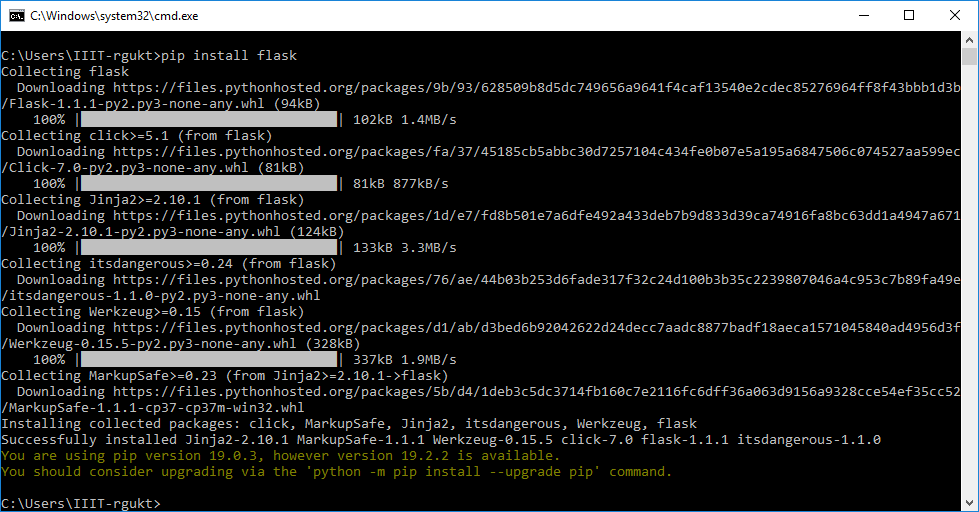
**Flask:**

* Flask is web framework in python
* Flask is a to build lightweight web application in python
* Developed by **Armin Ronacher**
* It is based on **WSGI toolkit** and **jinja2**
  + **WSGI -** Web Server Gateway Interface. It is the interface between webserver and Web application
  + **Jinja2 -** Jinja is a web template engine which is related to dynamic web pages

**Flask environment:**

Open Command prompt or terminal and type below command for flask installation

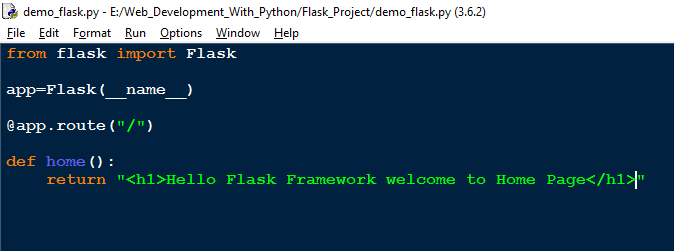
* **pip install flask -**  In Windows
* **pip3 install flask –** In Linux



**Directory Creation:**

* Open any directory and create a new folder (**Flask\_Project**)
* The folder is a project name
* Inside project folder create one python file which is main file to developing flask web application
* Now try to execute **Sample** **Message** in browser

**Program:**

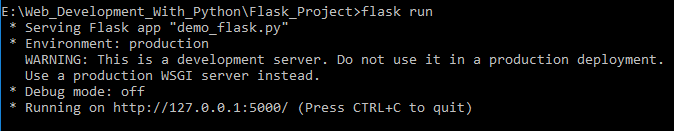


Now open command prompt and run the python file

* Before running the program we need to set the flask application
* **Set FLASK\_APP=demo\_file.py (in Windows)**
* **export FLASK\_APP=demo\_file.py (in Linux)**

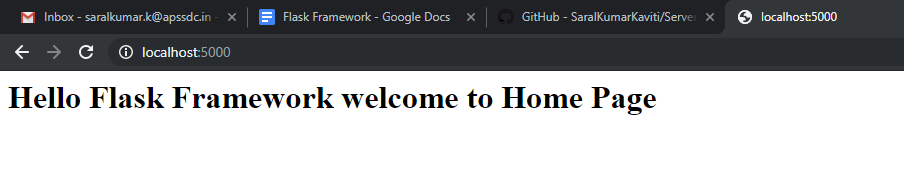
Now Open command prompt to run server

**flask run**

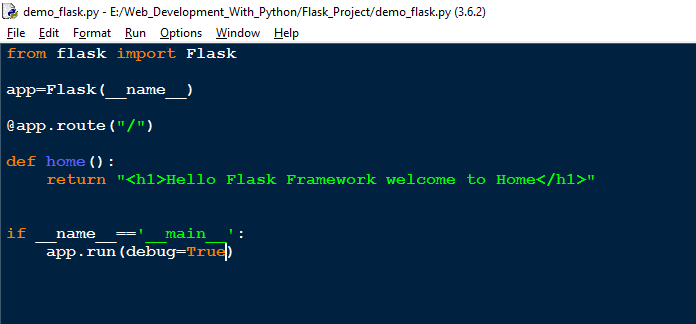
****

Now Open browser and type local IP (127.0.0.1) or (localhost)

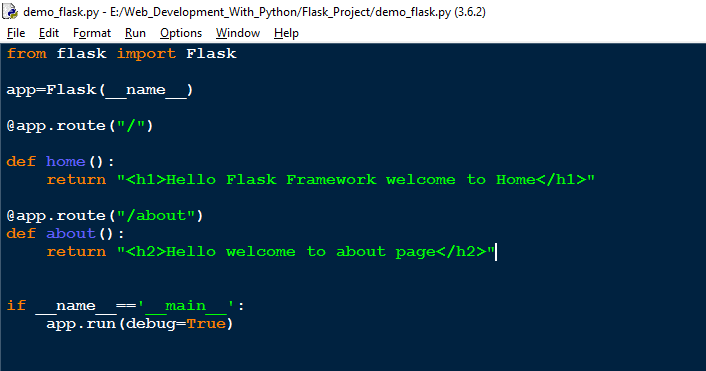
**http://localhost:5000**

****

By Default Debug mode is off so that we need to on the debug mode then open py file add the if condition debug mode in the below.

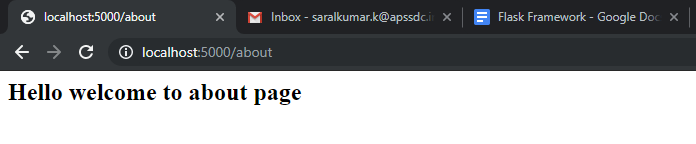


**Number of URL’s:**



Output file

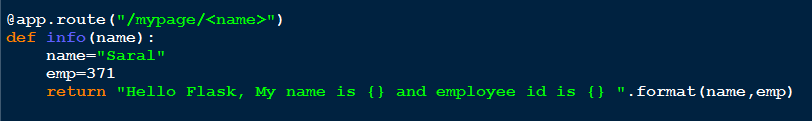
[**http://localhost:5000/about**](http://localhost:5000/about)



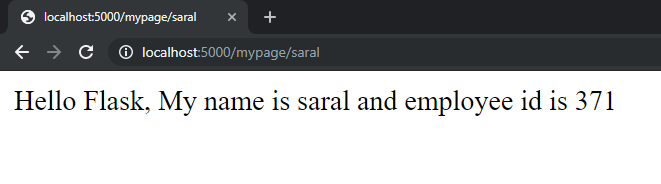
**Variables:**

Variable rules

* int **@app.route(“mypage/<int:emp>”)**
* str **@app.route(“/mypage/<name>”)**
* float **@app.route(“/mypage/<float:income>”)**



**Output:**

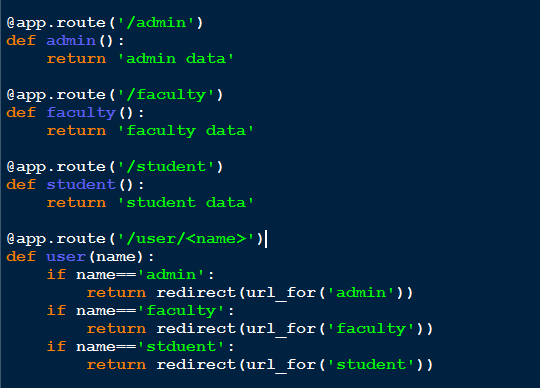


**Url\_for and redirect:**

Before using above concepts we need to import modules like **url\_for** and **redirect** in the controller file.

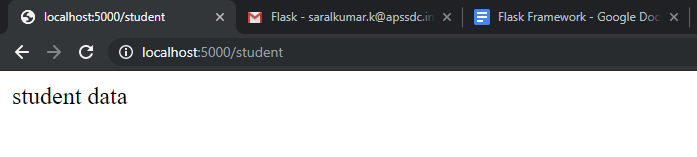
The url\_for() function is used to build a URL to the specific function dynamically. The first argument is the name of the specified function, and then we can pass any number of keyword argument corresponding to the variable part of the URL.

This function is useful in the sense that we can avoid hard-coding the URLs into the templates by dynamically building them using this function.



**http://localhost:5000/user/admin**

**Output:**



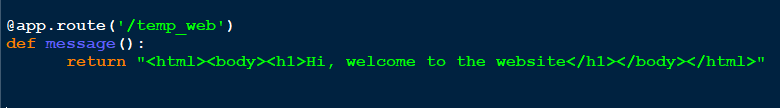
**Templates and Static Files:**

In the previous examples, we have returned the simple string as the response from the view function. Although, flask facilitates us to return the response in the form of HTML templates. In this section of the tutorial, we will go through the ways using which we can return the HTML response from the web applications.

**Example:**

The following flask script contains a view function, i.e., the message() which is associated with the URL '/'. Instead of returning a simple plain string as a message, it returns a message with <h1> tag attached to it using HTML.

[**http://localhost:5000/temp\_web**](http://localhost:5000/temp_web)



**Output:**

