DAY 54

FLASK: BACKEND PYTHON FRAMEWORK

BASIC CODE:

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route('/')  
def hello\_world():  
 return 'Hello world!'

TO RUN AND VIEW THE OUTPUT ON THE BROWSER:

* Pip install Flask
* flask --app hello run (command)

In your Flask code, \_\_name\_\_ is a special variable in Python. It represents the name of the module (file) where the code is written.

* If the code is run directly (e.g., python app.py), the value of \_\_name\_\_ will be "\_\_main\_\_".
* If the code is imported into another file as a module, \_\_name\_\_ will be the name of the file (e.g., "app")

PYTHON DECORATOR:@app.route('/')

A decorator function is simply a function that wraps another function and gives it some additional functionality or modifies it.

EXAMPLE:

import time  
  
def decorator\_function(function):  
 def wrapped\_function():  
 time.sleep(2)  
 function()  
 return wrapped\_function  
  
@decorator\_function  
def hello():  
 print("hello")  
  
def whatsup():  
 print("what's up")  
  
decorated\_function = decorator\_function(whatsup)  
decorated\_function()

two methods of using decorator function one is to use the @symbol followed by the decorator function name or to assign the decorator function to a new variable and call that variable.

USEFUL COMMANDS:

pwd -> print working directory -> current working path

ls -> list -> lists all the files and folders in the current working directory.

cd -> change directory

mkdir folder\_name -> create a new folder in the directory where you are.

touch filename.extension -> touch main.py -> creates a file in the directory.

rm main.py -> delete a file

cd .. -> takes one step before in the path , that is the parent folder .

rm -rf foldername -> recursively delete the folders in the location.