Explain the Model view Template Architecture?

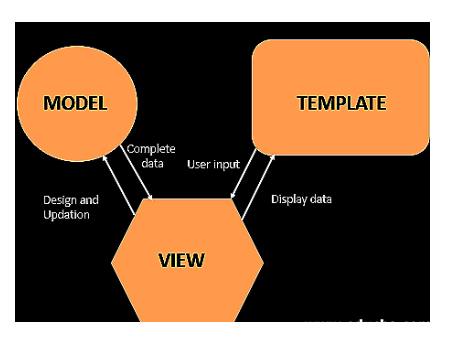
Django is based on **MVT (Model-View-Template)** architecture. MVT is a software design pattern for developing a web application.

**MVT Structure has the following three parts –**

**Model:**The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres).

**View:** The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files.

**Template:**A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.



Explain the Dictionary and Dictionary method with an example?

In Python, a dictionary is one of the core data structures. It is a sequence of key-value pairs separated by commas and surrounded by curly braces.



Python provides more than 10 methods for working with dictionaries.

demo\_dict = {

"key1": "value1",

"key2": "value2",

"key3": "value3"

}

Note that the values can be of any data type and can be duplicated, but the key must not be duplicated. If the keys are duplicated, you will get an invalid syntax error.

**Methods for Working with Python Dictionaries**

first\_dict = {

"name": "freeCodeCamp",

"founder": "Quincy Larson",

"type": "charity",

"age": 8,

"price": "free",

"work-style": "remote",

}

Explain the Tuple in Python with an Example?

A tuple represents a sequence of any objects separated by commas and enclosed in parentheses. A tuple is an **immutable** object, which means it cannot be changed, and we use it to represent fixed collections of items.

Let's take a look at some examples of Python tuples:

* () — an empty tuple
* (1.0, 9.9, 10) — a tuple containing three numeric objects
* ('Casey', 'Darin', 'Bella', 'Mehdi') — a tuple containing four string objects
* ('10', 101, True) — a tuple containing a string, an integer, and a Boolean object

Also, other objects like lists and tuples can comprise a tuple, like this:

a\_tuple = (0, [1, 2, 3], (4, 5, 6), 7.0)

The code above creates a tuple containing an integer, a list, a tuple, and a float number. The following code returns the entire tuple and its data type.

print(a\_tuple)

print(type(a\_tuple))

(0, [1, 2, 3], (4, 5, 6), 7.0)

However, ('A') is not a tuple. Let’s look at its data type:

print(type(('A')))

So, how can we declare a single-value tuple? The answer is easy. We need to add an extra comma just before the closing parenthesis, like this:  
('A',)

What is function? How to declare a function in Python Programming?

**Functions in Python**

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values.

There are three types of functions in Python:

* Built-in functions, such as help() to ask for help, min() to get the minimum value, print() to print an object to the terminal,… You can find an overview with more of these functions [**here**](https://docs.python.org/3/library/functions.html).
* User-Defined Functions (UDFs), which are functions that users create to help them out; And
* Anonymous functions, which are also called lambda functions because they are not declared with the standard def keyword.

**How to Define a Function: User-Defined Functions (UDFs)**

The four steps to defining a function in Python are the following:

1. Use the keyword def to declare the function and follow this up with the function name.
2. Add parameters to the function: they should be within the parentheses of the function. End your line with a colon.
3. Add statements that the functions should execute.
4. End your function with a return statement if the function should output something. Without the return statement, your function will return an object None.
5. def hello():
6. name = str(input("Enter your name: "))
7. if name:
8. print ("Hello " + str(name))
9. else:
10. print("Hello World")
11. return
13. hello()
14. In the above function, you ask the user to give a name. If no name is given, the function will print out “Hello World”. Otherwise, the user will get a personalized “Hello” response.
15. **Remember** also that you can define one or more function parameters for your UDF. You’ll learn more about this when you tackle the Function Arguments section. Additionally, you can or can not return one or multiple values as a result of your function.