**Encapsulation in Python:-**

Capsule--->

🡪Grouping/combining data member and member function in single unit is known as Encapsulation.

🡪Restrict access to method and variable

🡪Prevent data from direct modification

Advantage:

1-Security

2-Enhancement will be easy{without affecting outside we can able to perform any type of change internally }

3-Maintainability

4-Modularity

Disadvantage:

1-Time consuming process slows the speed

2-Username/password/OTP

Real example for encapsulation

Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”. Using encapsulation also hides the data. In this example, the data of the sections like sales, finance, or accounts are hidden from any other section.

**\_\_ private variable**

Type Description

private variables 🡪 Accessible only in their own class starts with two underscore

private methods 🡪 Accessible only in their own class or by a method defined with two underscore

class Computer:

def \_\_init\_\_(self):

self.\_\_maxprice = 900

def sell(self):

print("Selling Price: {}".format(self.\_\_maxprice))

def setMaxPrice(self, price):

self.\_\_maxprice = price

c = Computer()

c.sell()

# change the price

c.\_\_maxprice = 1000

c.sell()

# using setter function

c.setMaxPrice(1000)

c.sell()

**Output:**

Selling Price: 900

Selling Price: 900

Selling Price: 1000

9511113102

**HTML**

**Forms:**

Every data goes in backend as string.

For input elements we should define label that means the box will be associated with the label.

<label for="">Enter name</label>

<input id="uname" type="text" name="username" value="enter name">

Mandatory fields:

< input type="...." ..... required>

Placeholder vs value:

<placeholder ="Enter name">

Program for form :-

<html >

<head>

  <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h1 align ="center" >Welcome to informatics</h1>

    <table  align="center" cellspacing="0" cellpading="0">

        <form action="submit\_to.html">

           <tr>

              <div class="block"> <td> <label for="n">Name</label></td><td></td></div>

               <td><input id="n" type="text" name="first\_name"  placeholder="Enter your name></td>

           </tr>

           <tr>

              <div class="block"><td> <label for="em">Email</label></td><td></td></div>

              <td><input id="em" type="email" name="em\_ail"  placeholder="Enter your mail account" required></td>

           </tr>

            <tr>

                <div class="block"><td><label for="f\_name">Father's Name</label></td><td></td></div>

                <td><input id="f\_name" type="text" name="father's\_name"  placeholder="Enter your father's name"></td>

            </tr>

            <tr>

                <td><label for="qual">Qualification</label></td><td></td>

                <td><input id="qual" type="text" name="qualification"  placeholder="Enter your qualification's"></td>

            </tr>

            <tr>

                <td>Gender</td>

                <td></td>

                <td><label for="male">Male</label>

                <input id="male" type="radio" name="gender" >

                <label for="fmale">Female</label>

                <input id="fmale" type="radio" name="gender" >

                </td>

            <tr>

                <td></td>

                <td></td><td><input type="submit"  placeholder="submit"></td>

            </tr>

        </form>

    </table>

</body>

</html>

.h1

{

font-color:red;

}

block.label

{

font:bold;

font-size: 65;

}

Output:

**Welcome to informatics**

|  |  |  |
| --- | --- | --- |
| Name |  |  |
| Email |  |  |
| Father's Name |  |  |
| Qualification |  |  |
| Gender |  | Male  Female |
|  |  |  |

Dropdown :

<select name=’state’>

<option value=”1”>Value</option>

<option value=”1”>Value</option>

<option value=”1”>Value</option>

<option value=”1”>Value</option>

<option value=”1”>Value</option>

</select>

**CSS**

CSS is of 3 types:

**1-Inline CSS**

<h1 style=”color:red”>Welcome</h1>

**2-Internal CSS:**

<head>

<style type=”text/css”>

h1{

color: red ;

}

</style>

</head>

**3-External CSS**

**<head>**

<link rel=”stylesheet” href=”style1.css”>

**</head>**

style1.css

h1{

color: blue;

}

CSS:

IST WAY:-

Color:red;

2nd WAY:-

color: rgb(255,255,255)🡪white

color: rgb(0,0,0)🡪black

3rd WAY:-

Hexadecimal:-

color:#10ff33

rgba a is alpha for transparency

color: rgba(255,22,122,0.1)

Range is ( 0.1-1.0 )

img{

border : blue 20px solid;

}

body{

background: url();

background-repeat: no-repeat;

background-size: cover;

}

**CSS basic selectors:-**

1-Elementary Selectors

2-Id Selectors

Use #

<h1 id="spl" align="center">Welcome to new learnING INdIA</h1>

#spl{

    color:green

}

3-Class Selectors

.class\_name{

}

<h1 class="new" align="center">Welcome to new learnING Usa</h1>

.new {

    color:blue;

}

Advanced CSS Selectors:

1-\*

2-Descendent

3-Adjacent

4-Attribute

5-Nth of type selector

<html>

<head>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>Advanced Css Selectors</h1>

<a href="www.google.com">Click here to go to Google</a>

<ul>

<li>DOG</li>

<li>RAT</li>

<li>MOUSE</li>

</ul>

<h4>List of top movies</h4>

<ul>

<li>Bahuballi</li>

<li>KGF</li>

<li>Parmanu</li>

</ul>

<h4>List of top websites</h4>

<ul>

<li><a href="www.google.com">Flipkart</a></li>

<li><a href="www.snapdeal.com">Snapdeal</a></li>

<li><a href="www.amazon.com">Amazon</li>

<li><a href="www.twitter.com">Twitter</a></li>

</ul>

</body>

</html>

1-\* Selector

\*{

color:blue

}

Output:

**Advanced Css Selectors**

[Click here to go to Google](file:///D:\web-techy\www.google.com)

* DOG
* RAT
* MOUSE

**List of top movies**

* Bahuballi
* KGF
* Parmanu

**List of top websites**

* [Flipkart](file:///D:\web-techy\www.google.com)
* [Snapdeal](file:///D:\web-techy\www.snapdeal.com)
* [Amazon](file:///D:\web-techy\www.amazon.com)
* [Twitter](file:///D:\web-techy\www.twitter.com)

2-Descendent Selectors:

li  a{

background: blue;

color: red;

}

Output:

**Advanced Css Selectors**

[Click here to go to Google](file:///D:\web-techy\www.google.com)

* DOG
* RAT
* MOUSE

**List of top movies**

* Bahuballi
* KGF
* Parmanu

**List of top websites**

* [Flipkart](file:///D:\web-techy\www.google.com)
* [Snapdeal](file:///D:\web-techy\www.snapdeal.com)
* [Amazon](file:///D:\web-techy\www.amazon.com)
* [Twitter](file:///D:\web-techy\www.twitter.com)

**Attribute Selector:-**

a[href]{

    color:red;

}

**Output:**

**Advanced Css Selectors**

[Click here to go to Google](file:///D:\web-techy\www.google.com)

* DOG
* RAT
* MOUSE

**List of top movies**

* Bahuballi
* KGF
* Parmanu

**List of top websites**

* [Flipkart](file:///D:\web-techy\www.google.com)
* [Snapdeal](file:///D:\web-techy\www.snapdeal.com)
* [Amazon](file:///D:\web-techy\www.amazon.com)
* [Twitter](file:///D:\web-techy\www.twitter.com)

**Font and text in CSS:-**

1-font-family 🡪 cssfontstack.com

2- font-size: x px

3- font-weight 🡪 font-weight:100-900;

4-font-height:1.5 🡪the space between two line

5-text-align 🡪 text-align: center

6-text-decoration:line-through;

**Box Model:**

4 Edges

1-content

2-padding

3-border

4-margin

The space between content and border padding.

Hello

Padding

**Points to revise**

Functions types

Lambda,map,filters

Decorators,Generator,Iterator

Logging

**Oops**

Encapsulation -->Done/22/12/2020

Variables & Methods--> Done/22/12/2020

Super

Inheritance

Polymorphism

Abstraction(Abstract class & Interface)

Exception Handling

File Handling

Mongo Db:-

Humonguos 🡪 Extremely large

To handle extremely large amount of data

MongoDB is the most popular and trending database.

Vendor—MongoDB

MDB internally uses Mozilla’s spider monkey java script engine

Databases:

1-Relational database🡪Tables (Row & Column) Relation between data of the tables. Fixed schema/structure

2-No SQL/document database🡪

🡪Data will be stored in documents.

🡪Each document is independent from each other

🡪Documents are like row/record if compared in context of relational database.

**MongoDB structure:-**

MongoDB database contains several logical databases.

Each database contains several collections.

Collections contain several documents. Collection are like table in context of RDBMS

**How data represented in MongoDB?**

In Java script object notation [JSON].

**Features of MongoDB:-**

**1-Performance is good because retrieval is fast.**

**2-Schema less**

**3-**

**MongoDB shell/client vs MongoDB Server:-**

Once we install MongoDB we will get MongoDB shell and MongoDB server.

These are Javascript based applications

**Django:**

To create a project command:

**django-admin startproject project\_name**

**django-admin startproject firstproject**

Inside the project a subfolder will be created and a file named manage.py will be created once the above code is run as shown below:

**firstproject**

**manage.py**

**firstproject**

**\_\_init\_\_.py –** It is used as a python package file.Initially it will be empty

**asgi.py –** Asynchronous Server Gateway Interface

**urls.py –** All urls is managed by this file

**wsgi.py--** webserver gateway interface /used when application is moved to production.

**settings.py** --All project level setting is managed by this file.(databases and all)

**How to create an app :**

Go inside the project directory and then we need to run the command:

**python manage.py startapp app\_name**

**python manage.py startapp testapp**

Once the app is created default files will be automatically generated inside the app as shown below

**manage.py**

**firstproject**

**\_\_init\_\_.py –** It is used as a python package file.Initially it will be empty

**asgi.py –** Asynchronous Server Gateway Interface

**urls.py –** All urls is managed by this file

**wsgi.py--** webserver gateway interface /used when application is moved to production.

**settings.py** --All project level setting is managed by this file.(databases and all)

**testapp**

**migrations**

**\_\_init\_\_.py --**

**admin.py --**

**apps.py --**

**models.py --**

**tests.py --**

**views.py --**

Go to the directory where the file manage.py is there in command prompt then run the below command:

**python manage.py runserver**

**View**

**http request**

**http response**

**Class based**

**Function based**

**Views.py**

We need to import the HttpResponse from Django.http

from django.http import HttpResponse

Instead of request we can use any argument like r, req based on your mood

def hello\_world\_view(request):

    return HttpResponse('<h1>This is response from django application</h1>')