Coding Standards

We have used Django as a framework and HTML CSS for the front end development. Most of Django's important practices are based on Python.

A. DJANGO/PYTHON

1. Indentation

While writing a Python code, indentation plays a very important role. Using 4 spaces for indentation is better than eight, and if there are a few nested blocks, using eight spaces for each indentation may take up more characters that can be shown in a single line. Example of good indentation is illustrated below.

```
bar = some_function_name(var_first, var_second,
                                                var_third, var_fourth)
# Here indentation of arguments makes them grouped, and stand clear
from others.
def some_function_name(
     var_first, var_second, var_third,
     var_fourth):
   print(var_first)

# This example shows the hanging intent.
```

2. Importing a package

Importing a package is a direct implication of code-reusability. Therefore, always import statements are placed at the top of the source file. It is advisable to import different packages in different lines rather than importing it in the same line. The best way to import packages is illustrated as follows:

```
import os

import sys
```

3. **Naming conventions in Python/Django**

   The common naming convention that we should follow:

   - **Variable Names**- Avoid using single characters or symbols for denoting variables. Using proper variable names can increase the readability of the code and can be useful for debugging.
   - **Package Names**- Lowercase and short names along with the usage of underscore is recommended to improve readability.
   - **Class Names**- Naming a class should follow CamelCase naming convention,and internal classes can have an underscore leading in their name.
   - **Global Variable Names**-Avoid using global variables but if required prevention of global variables from getting exported can be done via _all_.
   - **Function names,method arguments and instance variables**- naming should be in lowercase and separated by an underscore and self as the first argument to instantiate methods. For classes or methods,use CLS or the objects for initialisation.

4. **Template style**

   In Django template code, put only one space between the braces and tag the contents.

   Do this: `{{ hello }}`
   Don't do this: `{{hello}}`

5. **View style**

   The first parameter in a view function should be called request.

   Do this: `def my_view(request,hello):`
   Don't do this: `def my_view(req,hello):`

6. **Model Style**

   Field names should be all lowercase using underscores. CamelCase should not be used.

   Do this:

```
class Person(models.Model):
    first_name = models.CharField(max_length=20)
    last_name = models.CharField(max_length=40)
```

Don't do this:

```
class Person(models.Model):
    FirstName = models.CharField(max_length=20)
    Last_Name = models.CharField(max_length=40)
```

7. **Model-View-Controller Architecture**

   Organising files as models, views and control will help in better task management. Django is structured with a number of modules like urls, models and views,etc. We can easily maintain our application by this separation. The model consists of all the classes and methods concerned. The view has all the user interfaces of the application. The Controller contains required processes which synchronizes model and view to work whole as an application. Another separation of Database Connection is created in order to deal with data management from mysql database.

B. HTML/CSS

1. **Spacing**- A comma followed by an argument should be further followed by blank space. A keyword followed by braces should be separated by a space.

   Example:

   ```
   p.intro {font-family: Verdana; font-size: 16em;}
   ```

2. **Wrapping lines**- An expression should have a break after an operator or a comma. Alignment of a new line should match the previous line in order to increase the readability.

   Example of Alignment:

   ```
   <table>
     <tr>
       <th>Name</th>
       <th>Description</th>
     </tr>
     <tr>
   ```

```
    <td>A</td>
    <td>Description of A</td>
  </tr>
 </table>
```

3. **Comments**- Inline comments should be used in order to explain the function of any particular code snippet. This improves the readability subsequently.

   One-line comment
   ```
   <!-- This is a comment →
   ```

   Multi-line comment
   ```
   <!--
     This is a long comment example. This is a long comment
   example.
     This is a long comment example. This is a long comment
   example.
   →
   ```

4. **Program Statements**- Statements should be limited per line. Nested and multiple statements should be avoided in order to ease the debugging.
5. **Use of Braces**- Parentheses/Braces improve the readability
6. **Indentation**- Proper indentation along with the emphasis of certain conditional statements or a loop is important for maintainability and readability. At Most 3 spaces can be used for indentation.

   ```
   <body>

   <h1>Famous Cities</h1>

   <h2>Tokyo</h2>
   <p>Tokyo is the capital of Japan, the center of the Greater
   Tokyo Area,
   and the most populous metropolitan area in the world.
   It is the seat of the Japanese government and the Imperial
   Palace,
   and the home of the Japanese Imperial Family.</p>

   </body>
   ```

7. **Structured Programming**- Modular programming enhances productivity and makes the task easy.
8. **Source File Names**- Naming the source files according to their function helps the programmer to manage files easily.
9. **MetaData-** The <title> element is necessary in HTML. Therefore,making it as meaningful as possible is important.Also, to ensure correct indentation and indexing, language and character encoding should be defined.

   Example:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML Syntax and Coding Style</title>
</head>
```

10. **Image attributes-**Always add the 'alt' attribute to images. This attribute is important when the image for some reason cannot be displayed. Also, always define image width and height. It reduces flickering because the browser can reserve space for the image before loading.

    Bad Example: `<img src="html5.gif">`
    Good Example:

```
<img src="html5.gif" alt="HTML5"
style="width:128px;height:128px">
```

11. **Using Lowercase attribute names-** Even though HTML allows mixing of uppercase and lowercase letters in attribute names. Lowercase is easier to write and looks cleaner, thus recommended.

    Example: `<div class="menu">`

12. **Closing all HTML elements-** It is recommended to close all HTML tags in order to make the debugging task easier.

    Example:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```