**Phong cách viết mã**

Phần mềm sử dụng dựa trên framework Django của Python, là một framework rất mạnh về web. Các tiêu chuẩn viết mã hầu hết dựa trên code style của python và Django, nên ở đây sẽ viết với phong cách PEP8

**Phong cách viết mã PEP8**

**Indentation( thụt lùi):**

Khác với C#,C++,java,các vòng lặp,điều kiện trong python được phân ra bởi các khoảng cách khác với C++,Java là là dấu{})

Mỗi thụt dòng có thể là 1 tab hoặc 4 space(nằm giúp đoạn code có thể phân rõ các phần,đoạn để dễ fix lỗi ,debug code)

Các dòng code phải được căn chỉnh sao cho các phần tử được bao bọc theo chiều dọc bằng cách sử dụng thụt lề của Python nối bên trong dấu ngoặc đơn, dấu ngoặc nhọn hoặc sử dụng thụt lề treo. Khi sử dụng thụt lề cần lưu ý những điều sau; không được có đối số trên dòng đầu tiên và phải sử dụng thụt lề sâu hơn để phân biệt rõ ràng chính nó là dòng tiếp tục.

Ví dụ sau minh họa cách bạn nên sử dụng phương pháp thụt lề thích hợp trong khi viết mã:

**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.**

**Quy định về để các dòng trống**

Tầm quan trọng của hai dòng trống và dòng trống đơn như sau:

**Hai dòng trống** :Hai dòng trống có thể được sử dụng để phân tách các chức năng cấp cao nhất và định nghĩa class, giúp tăng cường khả năng đọc code

**Một dòng trống** Một dòng trống duy nhất có thể được sử dụng trong các trường hợp sử dụng – **ví dụ**: mỗi function bên trong một class có thể được phân tách bằng một dòng duy nhất và các chức năng liên quan có thể được nhóm lại với nhau bằng một dòng duy nhất. Bạn cũng có thể tách phần logic của code bằng một dòng duy nhất.

**Importing a package**

Python là một ngôn ngữ đa tác vụ vì vậy việc import các package là một vấn đề đáng được lưu tâm khi sử dụng Django

Thông thường với các thư viện thì sẽ import 1 dòng tương ứng

**import os**

**import sys**

Với các thư viện đã được import sẵn ta có thể import các function trên cung 1 dòng

**from django.http import Http404, HttpResponse**

Nếu các function bạn import nhiều hơn, bạn có thể sử dụng phương pháp sau để khai báo chúng:

**from django.http import (**

**Http404, HttpResponse, HttpResponsePermanentRedirect**

**)**

**Grouping imported packages**

Package imports can be grouped in the following ways:

**Import các thư viện thông thường** Ví dụ như  *sys*, os, *subprocess*,…

import re  
import simplejson

**Import liên quan đến thư viện bên khác**::

from decimal import \*

**Import Ứng dụng/thư viện đã cài** Điều này bao gồm các mô-đun cục bộ của các dự án của bạn, chẳng hạn như model, view,… v.v.

from models import ModelFoo  
from models import ModelBar

**Quy tắc đặt tên trong Python/Django**

Mọi ngôn ngữ lập trình đều có quy ước đặt tên riêng. Ở Python/Django có nhiều điểm tương đồng,Và thế bạn cần phải tạo tên biến,biến toàn cục,class,module,package… theo chuẩn sau

* **Đặt tên cho các biến là** y: Không bao giờ sử dụng các ký tự đơn lẻ, ví dụ: 'x' hoặc 'X' làm tên biến. Nó có thể ổn đối với các tập lệnh Python thông thường của bạn, nhưng khi bạn đang xây dựng một ứng dụng web hay AI, bạn phải đặt tên biến đúng cách vì nó quyết định khả năng đọc của toàn bộ dự án, giúp cho mọi người có hiểu và sửa chữa những gì bạn viết
* **Đặt tên cho các gói và mô-đun** : Nên sử dụng các tên viết thường và viết tắt cho các mô-đun(vd: Import pandas as pd. Có thể sử dụng dấu gạch dưới nếu việc sử dụng chúng để cải thiện khả năng đọc. Các gói Python g nên có tên ngắn, toàn chữ thường để dễ truy vấn,sử dụng ( thông thường các biến được sử dụng nhiều lần trong chương trình vì thế cần chữ thường để tằng tốc độ viết code), có thể gạch nối dưới
* Vì tên mô-đun được ánh xạ tới tên tệp ( *models.py* , *urls.py* , v.v.), điều quan trọng là tên mô-đun phải được chọn khá ngắn vì một số hệ thống tệp không phân biệt chữ hoa chữ thường và cắt bớt tên dài.
* **Đặt tên mộtclass** : Tên lớp phải tuân theo quy ước đặt tên **CamelCase(**mỗi từ trong cụm được viết hoa chữ đầu) và các class sử dụng nội bộ có thể có dấu gạch dưới ở đầu trong tên của chúng( thường là các lớp khơi tạo,chức năng cơ bản như init).
* **Tên biến toàn cục** : Không nên sử dụng các biến toàn cục, việc ngăn chặn các biến toàn cục được xuất có thể được thực hiện thông qua *\_\_all\_\_* hoặc bằng cách xác định chúng bằng dấu gạch dưới có tiền tố (cách cũ, thông thường ).
* **Tên hàm và đối số phương thức** : Tên hàm phải ở dạng chữ thường và được phân tách bằng dấu gạch dưới và *tự* là đối số đầu tiên để khởi tạo phương thức. Đối với các lớp hoặc phương thức, hãy sử dụng CLS hoặc các đối tượng để khởi tạo.
* **Tên phương thức và biến phiên bản** : Sử dụng quy tắc đặt tên hàm — chữ thường với các từ được phân tách bằng dấu gạch dưới nếu cần để cải thiện khả năng đọc. Chỉ sử dụng một dấu gạch dưới ở đầu cho các phương thức không công khai và các biến phiên bản.
* **Gợi nhớ**: các function sử dụng càng mang tính gợi nhớ, tường minh còn không thì nên viết comment để làm rõ các tính năng

Lưu ý cho từng phần riêng

**Views.py**:Với các views.py của các module: các function sử dụng request cần được đặt ở dưới các Class, nếu một module lớn có nhiều module nhỏ khác thì ta cần phải phân chia code trong view bởi các khoảng trắng để dễ quản lý

**Urls.py**:các tên đường dẫn phải tương tự như tên của function,Class sử dụng, các đường dẫn được cách bằng khoảng cách cho từng mỗi module

**Model.py**: Cần được định nghĩa rõ ràng các model (các function khởi tạo giá trị,save,update)

Các file HTML,CSS của riêng từng module nên được phân chia ứng với từng folder (các file có thể tái sử dụng sẽ được chứa ở đường dẫn cao nhất ngoài cùng của dự án)

**Form.py**: Được tạo khi sử dụng các trang nhập liệu, được dùng để import và sử dụng trong views.py

**Utils.py**: Các function được sử dụng trong không phụ thuộc vào request trong views.py

Các phần installed app ở setting.py cần được chú thích ứng với nhu cầu sử dụng

**Link tham khảo:** https://hub.packtpub.com/code-style-django/