How to Interface with Python in HTML

Jinja2 allows us to **embed programming logic** in an HTML file, thereby providing a clear distinction between logic and design. While this document serves as an introduction to Python and Jinja2 tidbits among the HTML code, it is useful to think of situations on your own in which you can interface the HTML with the Python backend. **The following key points will best serve you after completing the corresponding lesson and before attempting its problem set.**

Key Ideas for Unit 3
- 3.1 Control Statements
  - Question 3.1
- 3.2 Template Inheritance

Key Ideas for Unit 3

3.1 Control Statements

Embedding control structures into your HTML allows you to situationally apply widgets and other pieces of code when the conditions call for them. Suppose we are dealing with an HTML file that is called by one of the classes that handles a webpage in your app, and given an object initialized inside of that class, you want to render some code given that the object actually exists. Your statement might look something along the lines of:

```
<body>

    <div class=...>
        {% if object %}
        …
        {% endif %}
    </div>
    …

</body>
```

There are two very important things to note here. First, as a best practice, **you always want to make sure that your code is semantically clear**. If I'm refactoring your work, it needs to be obvious that the div section is part of the body of your HTML, and that the conditional statement is associated with that specific division.

Secondly, you can impose external logic or expressions by enclosing the relevant statements in ***{% … %}*** brackets.

Question 3.1
Given an entity named *entity* in a list of datastore entities called *entities*, implement a for loop in HTML such that each entity is rendered sequentially with strings not automatically escaped. To get you started, you should loop over this piece of code:

```
{{ entity.property | safe }}
<br><br>
```

Assume that *property* is a property of the entity previously defined. Hint: How would you end the iteration to ensure that you only loop over the previous piece of code? Think about how we escaped the if statement in the canonical example. Can you write a parallel expression that relates to for loops?

3.2 Template Inheritance

The most powerful aspect of Jinja2 is that it allows you to inherit templates. **Template inheritance** allows you build a base HTML skeleton in which **you can define blocks that other templates can override**. More information can be found ***here***.

What this means is that **within the body of your main or base HTML document, you can indicate at which point you would like to extend to another template**:

```
{% block content %}
{% endblock %}
```

To connect to that section of code in the base template **from the child**

template, you need to specify a point where the base is inherited and a section specifying what code is to be substituted into the base. When rendering the child template, the effect is as if you had written the base and child as if they were in the same document. Inheritance naturally enforces object-oriented structure into your frontend design, and that is especially useful because you will find yourself frequently reusing templates to write content on top of.

```
{% extends “base.html” %}
{% block content %}
…
{% endblock %}
```