**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**Topic 21 - for Loops**  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What is a for Loop?**

A for loop in Python is a powerful tool that lets you iterate over a sequence, such as a list, so that you can perform actions on each element in that sequence. Instead of repeating code for each element, you write it once, and the for loop takes care of the repetition.

**Example Scenario: Checking for a Clean City**

Suppose you want to check if a particular city is among the five environmentally cleanest cities in the U.S. You can start by assigning the city in question to a variable:

python

Copy code

*city\_to\_check = "Tucson"*

Next, you assign the cleanest cities to a list:

python

Copy code

*cleanest\_cities = ["Cheyenne", "Santa Fe", "Tucson", "Great Falls", "Honolulu"]*

Without a for loop, you would need to write separate checks for each element in the list, like this:

python

Copy code

*if city\_to\_check == cleanest\_cities[0]:*

*print("It's one of the cleanest cities")*

*elif city\_to\_check == cleanest\_cities[1]:*

*print("It's one of the cleanest cities")*

*elif city\_to\_check == cleanest\_cities[2]:*

*print("It's one of the cleanest cities")*

*elif city\_to\_check == cleanest\_cities[3]:*

*print("It's one of the cleanest cities")*

*elif city\_to\_check == cleanest\_cities[4]:*

*print("It's one of the cleanest cities")*

This approach is lengthy and not efficient. Instead, Python provides a simpler solution with the for loop.

**Using a for Loop**

The for loop is concise and efficient. Here’s how it works:

python

Copy code

*for a\_clean\_city in cleanest\_cities:*

*if city\_to\_check == a\_clean\_city:*

*print("It's one of the cleanest cities")*

**How It Works**

* **Looping through each element**:  
  The loop pulls each city name from cleanest\_cities, one at a time.
* **Temporary variable**:  
  Each city name is temporarily assigned to a\_clean\_city.
* **Comparison check**:  
  For each a\_clean\_city, Python checks if it matches city\_to\_check. If it does, it prints "It's one of the cleanest cities."

**Step-by-Step Breakdown**

1. **First pass**:  
   The loop checks if the first city, "Cheyenne," is equal to "Tucson." It isn’t, so it moves to the next city.
2. **Second pass**:  
   The loop checks if "Santa Fe" is equal to "Tucson." It isn’t, so it moves to the third city.
3. **Third pass**:  
   The loop finds "Tucson," which matches city\_to\_check, and prints "It's one of the cleanest cities."

Once a match is found, the message is displayed.

**Optimizing with break**

To stop the loop once a match is found, add a break statement:

python

Copy code

*for a\_clean\_city in cleanest\_cities:*

*if city\_to\_check == a\_clean\_city:*

*print("It's one of the cleanest cities")*

*break*

The break statement immediately exits the loop, saving time by preventing further checks once a match is found.

**Syntax Overview**

Here’s a quick guide to the for loop syntax:

* **for keyword**:  
  Begins the loop and is followed by a temporary variable name (a\_clean\_city in this case).
* **Loop variable**:  
  A temporary variable to hold each element in the sequence, one at a time.
* **in keyword**:  
  Signals that you’re working within a specific list or sequence.
* **List or sequence**:  
  The sequence you’re looping through, like cleanest\_cities.
* **Indented code block**:  
  Each line of code inside the loop is indented, and each inner block is indented further to show hierarchy.

**Readable Variable Names**

Readable code makes it easier to understand. Avoid cryptic names like x, y, or z in your loops. Although they work, they can make code harder to follow:

python

Copy code

*for x in y:*

*if x == z:*

*print("It's one of the cleanest cities")*

Instead, using meaningful names, like a\_clean\_city, makes the code clearer.

**Summary**

* **Purpose of for loop**:  
  It allows you to efficiently process each element in a list, avoiding repetitive code.
* **Syntax essentials**:  
  for <variable> in <list>: followed by an indented block for code inside the loop.
* **break for efficiency**:  
  Stops the loop once a match is found, making the code faster and more efficient.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**