**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**Topic 22 - Nested for Loops**  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**What are Nested for Loops?**

A nested for loop is simply a for loop inside another for loop. With nested loops, you can automate repetitive tasks that involve multiple levels of iteration, such as combining items from two different lists.

**Example Scenario: Generating Rap Star Names**

Imagine Atlantic Records wants a list of potential rap star names. You decide to create separate lists for first names and last names:

plaintext

Copy code

First Names Last Names

----------- ----------

BlueRay Zzz

Upchuck Burp

Lojack Dogbone

Gizmo Droop

Do-Rag

By combining each first name with each last name, you can generate a list of unique names like:

* BlueRay Zzz
* Upchuck Burp
* Lojack Dogbone
* Gizmo Droop

To get 20 unique combinations, you could write each manually. But with a nested for loop, Python can do this for you quickly.

**Using Nested for Loops**

Here’s how you can set up nested for loops to automate this task:

python

Copy code

first\_names = ["BlueRay", "Upchuck", "Lojack", "Gizmo", "Do-Rag"]

last\_names = ["Zzz", "Burp", "Dogbone", "Droop"]

full\_names = []

for a\_first\_name in first\_names:

for a\_last\_name in last\_names:

full\_names.append(a\_first\_name + " " + a\_last\_name)

**How It Works**

1. **Outer loop (first names)**:  
   The outer loop iterates through each item in first\_names.
2. **Inner loop (last names)**:  
   For each first name in the outer loop, the inner loop runs through each item in last\_names.
3. **Combining names**:  
   Each combination is created by concatenating a\_first\_name and a\_last\_name with a space, and then added to full\_names.
4. **Result**:  
   Once all combinations are processed, full\_names contains every possible unique name.

**Detailed Step-by-Step Breakdown**

* **First Pass (Outer Loop)**:  
  The outer loop starts with "BlueRay", and the inner loop pairs "BlueRay" with each last name:
  + BlueRay Zzz
  + BlueRay Burp
  + BlueRay Dogbone
  + BlueRay Droop
* **Second Pass (Outer Loop)**:  
  The outer loop moves to "Upchuck", and the inner loop combines "Upchuck" with each last name:
  + Upchuck Zzz
  + Upchuck Burp
  + Upchuck Dogbone
  + Upchuck Droop

This pattern continues until all names in first\_names have been combined with all names in last\_names, giving 20 unique names.

**Understanding the Nested Loop Syntax**

* **Outer loop**:  
  for a\_first\_name in first\_names:
  + This loop controls which first name is currently being paired with last names.
* **Inner loop**:  
  for a\_last\_name in last\_names:
  + For each first name, this loop iterates through each last name.
* **append() method**:  
  full\_names.append(a\_first\_name + " " + a\_last\_name)
  + Each combination is added to the full\_names list using append().

**Points to Remember**

* **Indentation matters**:  
  Each level of nested loops requires further indentation to show the hierarchical structure.
* **Multiple nesting levels**:  
  You can nest additional loops within a loop, though it may make the code more complex.

**Summary**

* **Nested for loops** are helpful for combining elements from multiple lists or sequences.
* **Outer loop** controls the main iteration, while **inner loops** run through all combinations for each outer loop item.
* Use **meaningful variable names** for clarity in nested loops.

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