JANUARY 31ST, 2025

Looping Continued

CE 311K - L09

CE 311K Touchpoint Survey

Review: Loops

A *while* loop repeats as long as a condition is true

```
count = 0
while count < 5:
    print("Count is:", count)
    count += 1
```

A *for* loop iterates over a collection of elements

```
loads = [100, 200, 150, 300]
total_load = 0
for load in loads:
    total_load += load
```

Control Flow in Loops: *break*

The *break* keyword immediately **exits** the loop, **skipping** any remaining iterations

Commonly used when a specific condition is met

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        print("Found 3, stopping the loop.")
        break
    print(num)
```

```
1
2
Found 3, stopping the loop.
```

Control Flow in Loops: *continue*

The *continue* keyword skips the remaining code in the current iteration and moves on to the next one

Used to bypass certain conditions without completing exiting the loop

```
for i in range(5):
    if i % 2 == 1:
        continue  # Skip odd numbers
    print(i) # if the program reaches continue, this statement won't execute
```

```
0
2
4
```

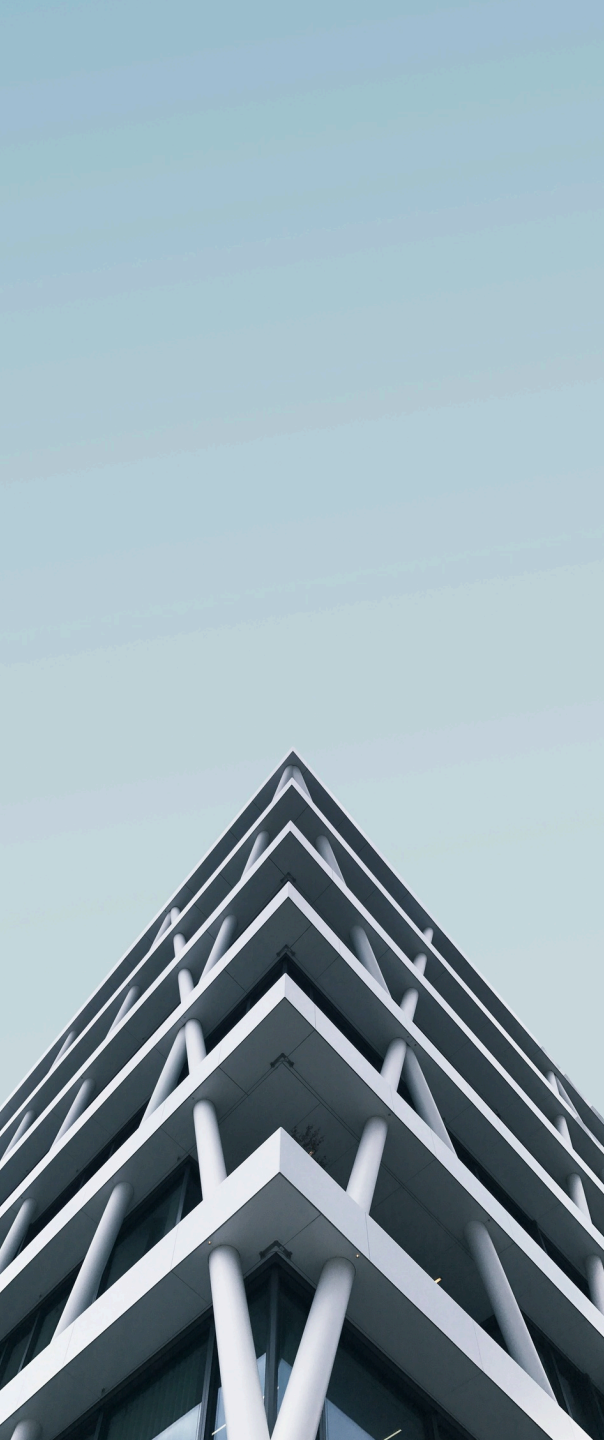
Nested Loops

Loops inside loops

Useful for tasks like matrix operations, grid calculations, or combining elements from multiple lists

```
for i in range(1, 6):
    for j in range(1, 4):
        print("*", end="")  # Print stars in the same row
    print()  # Move to the next line
```

```
***
***
***
***
***
```

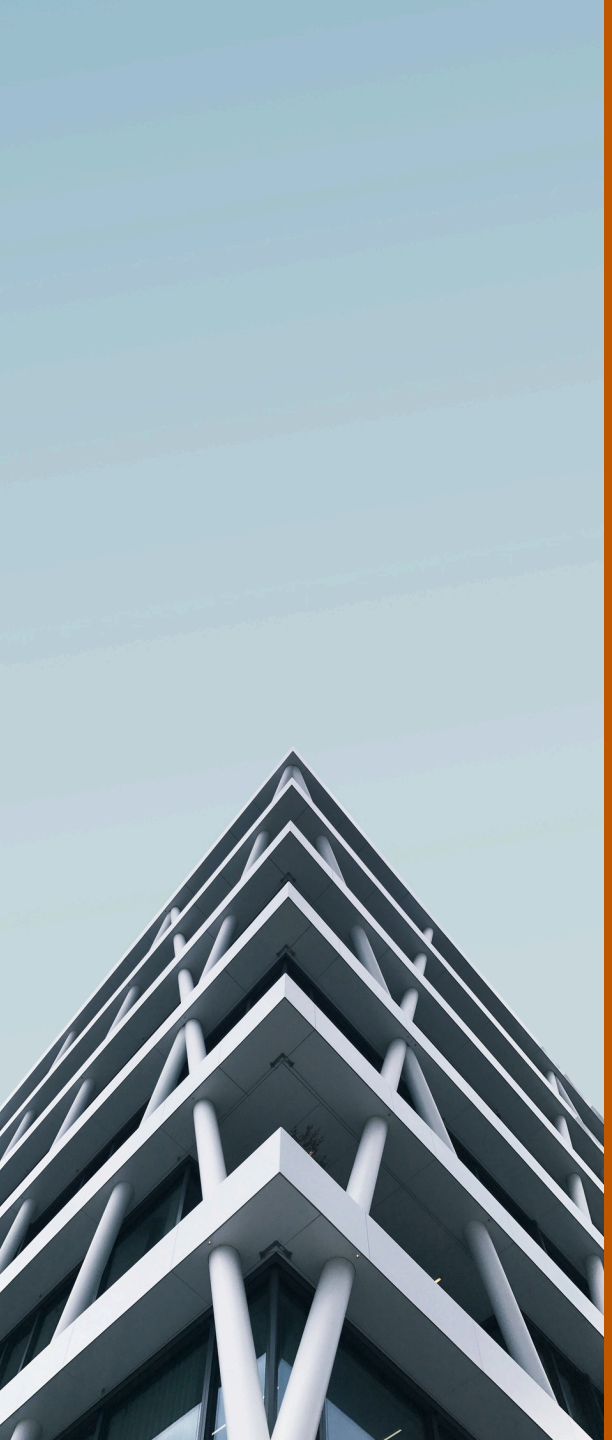


Iterating Over Structured Data

You can loop through **multidimensional** collections, unpacking multiple variables per iteration

```
coords = [(0,1), (0,3), (1,5)]
for x, y in coords: # unpacks each element into components
  output = "Coordinate: (" + str(x) + ", " + str(y) + ")"
  print(output)
```

```
Coordinate: (0, 1)
Coordinate: (0, 3)
Coordinate: (1, 5)
```



Iterating Over Structured Data

You can use the built-in *enumerate()* function to get indices and elements

```
materials = ["Concrete", "Steel", "Timber"]
for index, material in enumerate(materials): # pulls the index and element
    output = str(index) + ": " + material
    print(output)
```

```
0: Concrete
1: Steel
2: Timber
```

Iterating Over Structured Data

You can use the built-in *zip()* function to combine **same-length** lists and iterate through their elements together

```
names = ["Alice", "Bob", "Charlie"]
scores = [85, 92, 78]
for name, score in zip(names, scores): # pulls elements from each list at index
    output = name + ": " + str(score)
    print(f"{name}: {score}")
```

```
Alice: 85
Bob: 92
Charlie: 78
```

Summary

Impose more control in loops

break exits the loop

continue ignores remaining code and moves to next iteration

Nest loops to perform multiple operations per outer loop iteration

Iterate over structured data, **unpacking** multiple elements

Useful for multidimensional collections

enumerate() gets index and element

zip() combines collections to get elements from multiple collections at the same index