JANUARY 24TH, 2025



CE 311K - L06



Review: Multiple Conditions

else is a catch-all case

```
if humidity > 0.95:
    print("It is probably raining")
else:
    print("It is likely not raining")
```

Provide more conditions to check with elif

```
if temperature > 75:
    print("It is warm!")
elif temperature > 65:
    print("It is pleasant!")
else:
    print("It is cold")
```



What are arrays?

An **array** is a data structure used to store multiple **elements** in a single, **ordered** collection

Arrays are memory efficient, have indexed elements, and **typically** contain homogenous values

In Python, we use a data type called a *list* which is very similar

```
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "cherry"]
mixed = [1, "apple", True]
```

Python lists are more flexible than traditional arrays but less memory efficient



Accessing List Elements

Access elements by referencing their Index

- Python uses zero-based indexing
- Negative indices access elements from the end of the list
- Accessing an element with an invalid index returns an IndexError

```
fruits = ["apple", "banana", "cherry", "orange"]
print(fruits[0])  # Output: apple
print(fruits[2])  # Output: cherry
print(fruits[-1])  # Output: orange
print(fruits[4])  # Output: IndexError - max index is 3
```

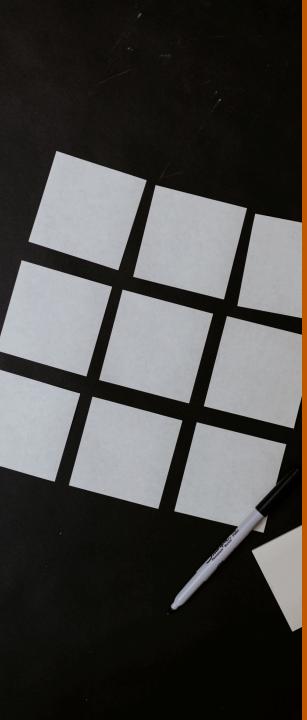


List Slicing

You can access a subset of the list by **slicing**

- Start index is **inclusive**, end index is **exclusive**
- Excluding the start index will slice from the **beginning**
- Excluding end index will slice to the end

```
fruits = ["apple", "banana", "cherry", "orange"]
print(fruits[1:3]) # Output: ['banana', 'cherry']
print(fruits[:2]) # Output: ['apple', 'banana']
print(fruits[2:]) # Output: ['cherry', 'orange']
```



Lists of Lists

Nested Lists are a list where each element is also a list

```
matrix = [[1,2,3], [4,5,6], [7,8,9]]
```

Access elements by specifying indices in two dimensions

```
print(matrix[1][2]) # Output: 6
```

Slicing only works on the outer level

```
print(matrix[:2]) # Output: [[1, 2, 3], [4, 5, 6]]
```

To slice inner level, you will need to first access the list element from the outer row and then slice



Simple List Operations

Join two lists together with the addition operator

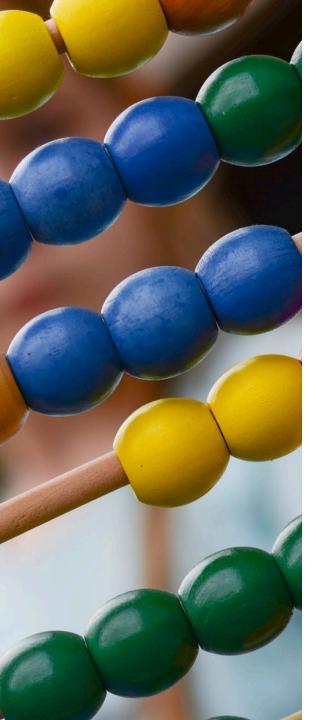
```
combined_list = [1, 2, 3] + [4, 5, 6] # Result: [1, 2, 3, 4, 5, 6]
```

Repeat lists with the multiplication operator to create larger lists

```
numbers = [1, 2, 3] * 3 # Result: [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Update elements by reassignment

```
combined_list[5] = 7 # Result: [1, 2, 3, 4, 5, 7]
```



Simple List Operations

Compare lists with the equality operator

Use the *in* keyword to check for membership

```
fruits = ['orange', 'cherry', 'banana', 'apple']
print("apple" in fruits) # Output: True
print("grape" in fruits) # Output: False
```

Summary

Arrays store multiple **elements** in a **single**, **ordered** collection

In Python, we use the more flexible *list*

You can access elements and sublists

Python lists are zero-indexed

Use brackets to get single elements and slices

Python provides simple ways to modify lists

Combine lists with addition operator

Update elements by overriding the value in that element using the index

Check for membership using in