05-if-else-elif, zip

IF Statements:

In Python, an `if` statement is a conditional statement that allows your code to make decisions based on whether a specified condition is true or false.

Basic Syntax:

```
if condition:
    # Code to execute if condition is True
```

How it works:

1. Python evaluates the condition after the `if` keyword
2. If the condition evaluates to `True`, the indented code block is executed
3. If the condition evaluates to `False`, the indented code block is skippe

Real-world Example:

Let's say we want to check if a student has passed an exam (passing score is 60):

```
score = 75

if score >= 60:
    print("Congratulations! You passed the exam.")
    print(f"Your score is: {score}")

print("This line will always execute regardless of the condition.")
```

In this example:

- We set a variable `score` to 75
- The condition `score >= 60` evaluates to `True` because 75 is greater than or equal to 60
- Therefore, the code inside the `if` block executes, displaying the congratulatory message

- The last print statement always executes because it's outside the `if` block

If we changed the score to 45, the condition would be `False`, and the congratulatory message wouldn't appear, but the last line would still execute.

When to Use:

Use `if` statements when you need your program to behave differently based on certain conditions, such as:

- Validating user input
- Checking application state
- Making decisions in game logic
- Handling different cases in data processing

Else:

The `else` statement in Python is used alongside an `if` statement to execute code when the condition in the `if` statement is `False`.

Basic Syntax:

```
if condition:
    # Code to execute if condition is True
else:
    # Code to execute if condition is False
```

How it works:

1. If the condition after `if` evaluates to `True`, the first code block executes
2. If the condition evaluates to `False`, the code block after `else` executes
3. Only one of the blocks will execute - never both

Real-world Example:

Let's extend our student exam example:

```
score = 45

if score >= 60:
    print("Congratulations! You passed the exam.")
    print(f"Your score is: {score}")
else:
    print("Sorry, you did not pass the exam.")
    print(f"Your score is: {score}")
    print("Please consider retaking the course.")

print("Thank you for taking the exam.")
```

In this example:

- The `score` is 45
- The condition `score >= 60` evaluates to `False` because 45 is less than 60
- Therefore, the code inside the `else` block executes, displaying the failure message and advice to retake the course
- The last line always executes regardless of the condition

Elif (Else If):

The `elif` statement allows you to check multiple conditions in sequence. It stands for "else if" and comes after an `if` statement and before an optional `else` statement.

Basic Syntax:

```
if condition1:
    # Code to execute if condition1 is True
elif condition2:
    # Code to execute if condition1 is False and condition2 is True
elif condition3:
    # Code to execute if condition1 and condition2 are False and condition3 i
s True
```

```
else:
    # Code to execute if all conditions are False
```

How it works:

1. Python checks each condition in order, starting with the `if` statement
2. Once a condition evaluates to `True`, its corresponding code block executes
3. All other conditions are skipped, even if they would also be `True`
4. If none of the conditions are `True`, the `else` block executes (if present)

Real-world Example:

Let's grade a student's exam based on their score:

```
score = 85

if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

print(f"Your score is {score}, which gives you a grade of {grade}.")
```

In this example:

- The score is 85
- First, Python checks if `score >= 90`, which is `False`
- Next, it checks if `score >= 80`, which is `True`
- Therefore, `grade = "B"` executes, and all other conditions are skipped
- Finally, the program outputs the grade

The zip() Function:

The `zip()` function in Python is used to combine multiple iterables (like lists, tuples, or strings) into a single iterable of tuples, where each tuple contains elements from the input iterables at the same position.

Basic Syntax:

```
zip(iterable1, iterable2, ...)
```

How it works:

1. Takes any number of iterables as arguments
2. Pairs corresponding elements from each iterable into tuples
3. Stops when the shortest input iterable is exhausted

Real-world Example:

Let's say we have two lists: one with student names and another with their respective scores:

```
students = ["Alice", "Bob", "Charlie", "David"]
scores = [85, 92, 78, 95]

# Combine the two lists using zip()
student_scores = list(zip(students, scores))
print(student_scores)
# Output: [('Alice', 85), ('Bob', 92), ('Charlie', 78), ('David', 95)]

# We can easily iterate through the paired data
for student, score in zip(students, scores):
    if score >= 90:
        print(f"{student} scored an A with {score} points")
    else:
        print(f"{student} scored {score} points")
```

In this example:

- We have two separate lists: `students` and `scores`

- The `zip()` function pairs each student with their corresponding score
- We convert the result to a list to see the tuples clearly
- We then iterate through the pairs to print personalized messages based on scores

Common Uses for zip():

- Parallel iteration over multiple sequences
- Creating dictionaries from two lists (keys and values)
- Transposing matrices (rows to columns and vice versa)
- Combining related data from different sources

Python Conditional Statements and zip() Function Summary

Concept	Syntax	When to Use	Key Points
if statement	`if condition:` `# code`	To execute code only when a condition is true	- Basic decision making - Code inside only runs if condition is True - Can be used alone

else statement	`if condition:`
	`# code`
	`else:`
	`# alternative code`
		To provide an alternative when condition is false	- Must follow an if statement
			- Only executes when if condition is False
			- Only one block (if or else) will run

elif statement	`if condition1:`
	`# code`
	`elif condition2:`
	`# code`
	`else:`
	`# code`
		To check multiple conditions in sequence	- Stands for "else if"
			- Checks conditions in order
			- Stops after first True condition
			- Can have multiple elif blocks

zip() function	`zip(iterable1, iterable2, ...)`	To combine multiple iterables	- Creates pairs/tuples of elements - Stops at shortest iterable - Returns a zip object (iterator) - Common for parallel iteration