**What to Say**:

• “In programming, decision-making structures allow the program to make choices based on conditions. Today, we’ll use an **if-else** structure to make decisions.”

• “Think of it like this: if a condition is true, one block of code runs; if it’s false, another block runs. This is how programs control the flow of what happens.”

• “We’ll also introduce loops, which help you repeat tasks efficiently without writing the same code over and over.”

**Example**:

• Show this simple if-else structure:

A black screen with white text

Description automatically generated

**Quick Explanation**:

• “In this example, we check if someone is 18 or older. If yes, we print that they are eligible to vote. Otherwise, we print that they are not eligible.”

**Demonstrating Decision Structures and Loops (10 minutes)**

**Step 1: Demonstrate Decision Structures (3 minutes)**

**What to Say**:

• “Let’s walk through how an **if-else** statement works. It’s like making a choice based on information.”

• “For example, if a student’s grade is 70 or higher, they pass. Otherwise, they fail.”

**Example**:

A computer screen with green text

Description automatically generated

**Hands-On Activity**:

• Ask students to create their own **if-else** example using a different condition, like checking the weather (if temperature > 30, print “It’s hot!”; else, print “It’s cool.”).

**Step 2: Introduce Loops (3 minutes)**

**What to Say**:

• “Loops are a way to repeat tasks. Instead of writing code for every step, you can use a loop to automate the process.”

• “A **for** loop repeats a block of code a set number of times, while a **while** loop continues until a condition becomes false.”

**Example** (For loop):

A black screen with green text

Description automatically generated

**Explanation**:

• “This code prints the same message 5 times, but we only wrote one print statement!”

**Example** (While loop):

A black screen with colorful text

Description automatically generated

**Explanation**:

• “In this case, we loop while count is less than or equal to 5. Each time the loop runs, count increases by 1.”

**Step 3: Combine If-Else and Loops (4 minutes)**

**What to Say**:

• “Now let’s combine decision-making and loops to create a program that checks a condition multiple times.”

• “For example, we can loop through a list of ages and check if each person is eligible to vote.”

**Example**:

A screenshot of a computer program

Description automatically generated

**Explanation**:

• “Here, we loop through the list of ages and use an if-else statement to check if each person is old enough to vote. The loop lets us check multiple people without rewriting the same check for each.”

**Hands-On Activity**:

• Let the students create their own version of this, checking a list of grades to determine if each student passed or failed.

**Q&A and Discussion (3 minutes)**

**What to Say**:

• “Do you have any questions about how decision structures and loops work?”

• “How do you think you could use loops and if-else statements together in real life?”

• “One example could be checking multiple items in an online store to see if they are on sale!”

Encourage students to share their ideas and reinforce how using decision structures and loops can make their code more efficient.

**Closing Remarks**

• “Today, you’ve learned how to make decisions in your code using **if-else** statements and how to repeat tasks with loops. Combining these two concepts is key to making powerful and efficient programs.”

• “Next time you write a Python program, remember that decision structures help your program make choices, and loops help you avoid repetitive work!”

**Timing Summary:**

A screen shot of a computer program

Description automatically generated

**Line-by-Line Explanation**

**Line 1: number = 10**

• **What it does**:

• This line initializes the variable number and assigns it a value of 10.

• We’ll use this variable to check whether the number is even or odd and control how long the loop will run.

**Line 2: while number > 0:**

• **What it does**:

• This is the **while loop**. It checks if the condition (number > 0) is true before every iteration of the loop.

• As long as number is greater than 0, the loop will continue running. Once number is 0 or less, the loop will stop.

• So, initially, since number is 10, the loop begins.

**Line 3: if number % 2 == 0:**

• **What it does**:

• This line is an **if statement** inside the while loop. It checks if the number is even.

• The expression number % 2 is called the **modulus operator**, which returns the remainder of dividing number by 2.

• If the remainder is 0 (number % 2 == 0), that means the number is even, and the code inside this block will run.

**Line 4: print(f"{number} is even")**

• **What it does**:

• If the condition in the if statement (number % 2 == 0) is true, this line will be executed.

• This uses Python’s **f-string** to insert the value of the number variable into the string and print it, along with the message "is even".

• For example, if number is 10, it will print: 10 is even.

**Line 5: else:**

• **What it does**:

• This **else statement** comes into play when the condition in the **if** statement is false.

• If number % 2 == 0 is **not true** (i.e., the number is odd), the code in the else block will run.

**Line 6: print(f"{number} is odd")**

• **What it does**:

• If the number is not divisible by 2 (i.e., the remainder is not 0), this line will be executed.

• It will print a message saying that the number is odd. For example, if number is 9, it will print: 9 is odd.

**Line 7: number -= 1**

• **What it does**:

• This line reduces the value of number by 1 after each loop iteration.

• number -= 1 is shorthand for number = number - 1.

• After each loop, the value of number decreases, which ensures that eventually number will reach 0, and the loop will stop.

• Without this line, the loop would never end because number would always stay the same, causing an **infinite loop**.

**How It Runs:**

• Initially, number is 10.

• The loop checks if number > 0. Since 10 is greater than 0, it enters the loop.

• The if statement checks if 10 % 2 == 0 (which is true), so it prints 10 is even.

• Then, number is reduced by 1, making it 9.

• On the next loop iteration, it checks if number > 0 (9 > 0 is true), so the loop continues.

• This time, 9 % 2 != 0, so it prints 9 is odd, and then reduces number to 8.

• This process repeats until number reaches 0. At that point, the condition number > 0 becomes false, and the loop stops.