**Module 4. Conditional Structures**

**Learning Objectives**

* Understand and apply the basic syntax of ***if*** statements to control program flow.
* Use ***if-else*** statements to execute different code blocks based on condition evaluation.
* Implement ***if-elif-else*** structures to handle multiple conditions and execute appropriate code blocks.
* Recognize the importance of proper indentation in Python for defining the scope of code blocks.
* Utilize ***match-case*** statements to handle multiple conditions similarly to switch statements in other languages.

Flow control is a fundamental programming concept that enables developers to manage the order of execution of instructions within a program. By using conditionals (if-else statements), loops (for, while), and control statements (break, continue, return), programmers can create dynamic and responsive applications. Learning flow control is crucial for students as it enhances their decision-making capabilities, allows for the automation of repetitive tasks, aids in implementing various algorithms, and improves problem-solving skills. Proper use of flow control also contributes to code readability and maintainability, making it easier to debug and modify. Mastery of these concepts equips students with the tools needed to develop robust, efficient, and flexible programs.

# **The *if* Statement**

In Python, you can use logical conditions to make decisions in your code. These conditions are similar to the ones you’ve seen in mathematics:

* **Equals**: *a == b*
* **Not Equals**: *a != b*
* **Less than**: *a < b*
* **Less than or equal to**: *a <= b*
* **Greater than**: *a > b*
* **Greater than or equal to**: *a >= b*

These conditions are often used in "if statements" and loops to control the flow of your program. An "if statement" allows your program to execute certain code only if a specific condition is true. You can write an if statement using the ***if*** keyword.

|  |
| --- |
| a = 50  b = 100  if b > a:  print("b is greater than a.") |

In this example, we have two variables, ***a*** and ***b***. The if statement checks if ***b*** is greater than ***a***. Since ***a*** is 50 and ***b*** is 100, the condition ***b > a*** is true, so the program prints "b is greater than a".

***Importance of Indentation***

Python uses indentation (spaces or tabs at the beginning of a line) to define the scope of code blocks. This is different from many other programming languages that use curly braces ***{}***.

*Example Without Proper Indentation*:

|  |
| --- |
| a = 50  b = 100  if b > a:  print("b is greater than a.") |

In this example, the lack of indentation for the print statement will raise an error because Python expects the code inside the if statement to be indented. Correct indentation is crucial for your code to work properly.

(Example)

Problem Statement: Write a Python program that takes a single character input from the user and checks if it is a vowel (a, e, i, o, u). If the character is a vowel, print "The character <char> is a vowel".

Instructions:

* Prompt the user to enter a single character.
* Use an if statement to check if the character is in the string 'aeiou'.
* If the condition is true, print "The character <char> is a vowel".

Example Output:

Enter a character: a

The character a is a vowel

**(Solution)**

|  |
| --- |
| # Step 1: Prompt the user to enter a single character  char = input("Enter a character: ")  # Step 2: Check if the character is a vowel  if char in 'aeiou':  # Step 3: Print the result  print(f"The character {char} is a vowel") |

# **The *if* – *else* Statement**

The ***if-else*** statement in Python allows your program to execute one block of code if a condition is true, and another block of code if the condition is false. This helps in creating more complex decision-making processes within your code.

* **If Condition**: The program first evaluates the condition following the if keyword.
* **True Condition**: If the condition is true, the code block under the if statement is executed.
* **False Condition**: If the condition is false, the code block under the else statement is executed.

Here’s the basic syntax of an ***if-else*** statement in Python:

*if condition:*

*# code to execute if the condition is true*

*else:*

*# code to execute if the condition is false*

Let's look at an example to understand how the if-else statement works:

|  |
| --- |
| a = 100  b = 50  if b > a:  print("b is greater than a")  else:  print("b is not greater than a") |

(Explanation)

* **Variables**: We have two variables, ***a*** and ***b***, with values 100 and 50, respectively.
* **Condition**: The condition ***b > a*** is evaluated.
* **True Path**: If ***b*** were greater than ***a***, the program would print "b is greater than a".
* **False Path**: Since ***b*** is not greater than ***a*** (50 is not greater than 100), the program executes the code block under the ***else*** statement and prints "b is not greater than a".

This example shows how you can control the flow of your program based on conditions. The ***if-else*** statement ensures that different blocks of code are executed depending on whether the specified condition is met.

***Importance of Indentation***

Just like with simple ***if*** statements, proper indentation is crucial for ***if-else*** statements. Python relies on indentation to understand the scope of the code blocks. Incorrect indentation will lead to errors.

*Example With Proper Indentation*:

|  |
| --- |
| a = 100  b = 50  if b > a:  print("b is greater than a")  else:  print("b is not greater than a") |

*Example With Improper Indentation*:

|  |
| --- |
| a = 100  b = 50  if b > a:  print("b is greater than a") # This will cause an error  else:  print("b is not greater than a") # This will also cause an error |

In the incorrect example, the lack of indentation for the print statements will cause an error because Python does not recognize them as part of the if-else blocks.

The if-else statement is a fundamental tool in Python programming, allowing you to execute different code paths based on conditions. Mastering this concept is essential for writing dynamic and responsive programs.

(Example)

Problem Statement: Write a Python program that takes an input number from the user and checks if it is even or odd. If the number is even, print "The number is even". If the number is odd, print "The number is odd".

Instructions:

* Prompt the user to enter a number.
* Convert the input to an integer.
* Use an ***if-else*** statement to check if the number is divisible by 2 (i.e., even).
* If the condition is true, print "The number is even".
* If the condition is false, print "The number is odd".

Example Output:

Enter a number: 4

The number is even

Enter a number: 7

The number is odd

**(Solution)**

|  |
| --- |
| # Step 1: Prompt the user to enter a number  number = int(input("Enter a number: "))  # Step 2: Check if the number is even or odd  if number % 2 == 0:  # Step 3: Print the result if the number is even  print("The number is even")  else:  # Step 4: Print the result if the number is odd  print("The number is odd") |

# **The *if* – *elif* – *else* Statement**

An ***if-elif-else*** statement checks multiple conditions one by one. The program executes the block of code under the first condition that is true. If none of the conditions are true, the code block under the else statement is executed.

*How It Works*

* **If Condition**: The program first evaluates the condition following the ***if*** keyword.
* **Elif Condition**: If the ***if*** condition is false, the program evaluates the condition following the ***elif*** keyword. You can have multiple ***elif*** conditions.
* **Else Condition**: If none of the ***if*** or ***elif*** conditions are true, the program executes the code block under the ***else*** statement.

Here’s the basic syntax of an ***if-elif-else*** statement in Python:

*if condition1:*

*# code to execute if condition1 is true*

*elif condition2:*

*# code to execute if condition1 is false and condition2 is true*

*else:*

*# code to execute if none of the above conditions are true*

Let's look at an example to understand how the if-elif-else statement works:

|  |
| --- |
| a = 20  b = 20  if b > a:  print("b is greater than a")  elif a == b:  print("a and b are equal")  else:  print("b is less than a") |

* **Variables**: We have two variables, ***a*** and ***b***, both set to 20.
* **If Condition**: The condition ***b > a*** is evaluated. Since ***b*** is not greater than ***a*** (both are equal), this condition is false, and the program moves to the next condition.
* **Elif Condition**: The condition ***a == b*** is evaluated. Since ***a*** equals ***b***, this condition is true, and the program prints "a and b are equal".
* **Else Condition**: If neither of the above conditions were true, the program would execute the code block under else. However, in this case, the ***elif*** condition was true, so the else block is not executed.

The ***if-elif-else*** structure allows the program to handle multiple conditions gracefully, executing only the first true condition's code block.

***Importance of Indentation***

As with other Python control structures, proper indentation is crucial for ***if-elif-else*** statements. Python uses indentation to define the scope of code blocks.

(Example)

Problem Statement: Write a Python program that takes a numerical grade input from the user and categorizes it into a letter grade based on the following criteria:

* A: 90-100
* B: 80-89
* C: 70-79
* D: 60-69
* F: 0-59

Instructions:

* Prompt the user to enter a numerical grade.
* Convert the input to an integer.
* Use an ***if-elif-else*** statement to categorize the grade into A, B, C, D, or F.
* Print the corresponding letter grade.

Example Output:

Enter a numerical grade: 85

The letter grade is B

Enter a numerical grade: 72

The letter grade is C

**(Solution)**

|  |
| --- |
| # Step 1: Prompt the user to enter a numerical grade  grade = int(input("Enter a numerical grade: "))  # Step 2: Categorize the grade using if-elif-else statements  if grade >= 90 and grade <= 100:  letter\_grade = 'A'  elif grade >= 80 and grade <= 89:  letter\_grade = 'B'  elif grade >= 70 and grade <= 79:  letter\_grade = 'C'  elif grade >= 60 and grade <= 69:  letter\_grade = 'D'  else:  letter\_grade = 'F'  # Step 3: Print the corresponding letter grade  print(f"The letter grade is {letter\_grade}") |

Explanation:

* User Input: The program asks the user to input a numerical grade.
* Conversion: The input is then converted to an integer using ***int()***.
* Condition Check:
  + The if statement checks if the grade is between 90 and 100 (inclusive) and assigns the letter grade 'A'.
  + The elif statements check for the ranges 80-89, 70-79, and 60-69, assigning the letter grades 'B', 'C', and 'D' respectively.
  + The else statement covers all grades below 60, assigning the letter grade 'F'.
* Print Statement: The program prints the corresponding letter grade based on the evaluated conditions.

# **The *match* Statement**

Python does not have a built-in ***switch*** statement like some other languages (e.g., C, C++, Java, JavaScript). However, Python offers several ways to handle multiple conditions that can serve similar purposes. You can use the ***match-case*** statement introduced in Python 3.10 to achieve the functionality of a switch statement.

Python 3.10 introduced the ***match-case*** statement, which offers functionality similar to a ***switch*** statement in other languages. This new syntax allows for pattern matching and can be more expressive than using other sequence data types, which you will learn in later modules.

|  |
| --- |
| value = 2  match value:  case 1:  print("One")  case 2:  print("Two")  case 3:  print("Three")  case \_:  print("Invalid value") |

In this example,

* **Case 1**: If ***value*** is 1, it prints "One".
* **Case 2**: If ***value*** is 2, it prints "Two".
* **Case 3**: If ***value*** is 3, it prints "Three".
* **Case \_**: The underscore ***\_*** acts as a wildcard or default case, executing if none of the previous cases match.

Explanation of ***match-case*** Syntax:

* ***match*** Keyword: Starts the pattern matching block, similar to *switch*.
* ***case*** Keyword: Each ***case*** keyword introduces a new pattern to match against the value.
* Wildcard ***\_*** Case: Acts as a default case, catching any values that don't match the specified patterns.

This structure ensures that only one case is executed, and there's no need for explicit ***break*** statements, making the code cleaner and less error-prone.

**Summary**

1. Understand and apply the basic syntax of ***if*** statements to control program flow.
2. Use ***if-else*** statements to execute different code blocks based on condition evaluation.
3. Implement ***if-elif-else*** structures to handle multiple conditions and execute appropriate code blocks.
4. Recognize the importance of proper indentation in Python for defining the scope of code blocks.
5. Utilize ***match***-***case*** statements to handle multiple conditions similarly to switch statements in other languages.
6. Flow control enables developers to manage the order of execution of instructions within a program.
7. Learning flow control enhances decision-making capabilities and automates repetitive tasks.
8. Proper use of flow control contributes to code readability and maintainability.
9. The if statement allows execution of code only if a specific condition is true.
10. Conditions like equals, not equals, less than, and greater than are used in if statements.
11. Proper indentation is crucial for Python code to work correctly.
12. The ***if-else*** statement executes one block of code if a condition is true, another if false.
13. The ***if-elif-else*** statement checks multiple conditions one by one, executing the first true condition's code block.
14. Python 3.10 introduced ***match***-***case*** statements for pattern matching, similar to switch statements in other languages.
15. Match-case statements use the ***match*** keyword to start the block and ***case*** keywords for patterns to match against.

**Programming Exercises**

**Exercise 1: Check Negative Number**

Write a program that takes an integer input from the user and checks if the number is negative. If the number is negative, print "The number is negative".

**Exercise 2: Voting Eligibility**

Write a program that asks the user to enter their age. If the age is greater than or equal to 18, print "You are eligible to vote".

**Exercise 3: Temperature Check**

Write a program that asks the user to enter a temperature in Fahrenheit. If the temperature is below 32, print "It's freezing". Otherwise, print "It's not freezing".

**Exercise 4: Divisibility by 5**

Write a program that takes a number as input from the user and checks if the number is divisible by 5. If it is, print "The number is divisible by 5". If it is not, print "The number is not divisible by 5".

**Exercise 5: Number Classification**

Write a program that takes an integer input from the user and prints "Positive", "Negative", or "Zero" based on whether the number is positive, negative, or zero.

**Exercise 6: Day Type Identifier**

Write a program that takes a day of the week as input (e.g., "Monday", "Tuesday") and prints whether it is a weekday or weekend using the match-case statement.

**Exercise 7: Month Name Finder**

Write a program that takes an integer input from the user and uses a match-case statement to print the corresponding month name (1 for January, 2 for February, etc.). Print "Invalid month" if the input is not between 1 and 12.

**Exercise 8: Character Type Identifier**

Write a program that takes an input from the user and checks if it is an uppercase letter, a lowercase letter, a digit, or a special character. Use an ***if-elif-else*** statement to print the appropriate category.

**Exercise 9: Day Type Identifier**

Write a program that takes a day of the week as input (e.g., "Monday", "Tuesday") and prints whether it is a weekday or weekend using the match-case statement.

**Exercise 10: Month Name Finder**

Write a program that takes an integer input from the user and uses a match-case statement to print the corresponding month name (1 for January, 2 for February, etc.). Print "Invalid month" if the input is not between 1 and 12.

**Exercise 11: Discount Calculator**

Write a program that takes the total purchase amount as input and applies a discount based on the following criteria:

* If the amount is greater than $100, apply a 10% discount.
* If the amount is between $50 and $100, apply a 5% discount.
* If the amount is less than $50, no discount.

Use an ***if-elif-else*** statement to calculate and print the final amount after discount.

**Exercise 12: BMI Calculator**

Write a program that calculates the Body Mass Index (BMI) based on user input for weight (in kilograms) and height (in meters). Use ***if-elif-else*** statements to categorize the BMI into:

* Underweight (BMI < 18.5)
* Normal weight (18.5 <= BMI < 25)
* Overweight (25 <= BMI < 30)
* Obesity (BMI >= 30)

**Exercise 13: Traffic Light Simulator**

Write a program that takes a traffic light color (red, yellow, green) as input from the user and prints the corresponding action using a match-case statement. For example, "red" should print "Stop", "yellow" should print "Caution", and "green" should print "Go".

**Exercise 14: Season Identifier**

Write a program that takes a month number (1-12) as input and prints the corresponding season using a match-case statement. Assume:

* December, January, February: Winter
* March, April, May: Spring
* June, July, August: Summer
* September, October, November: Autumn