Conditional Statements

Conditional statements in Python allow you to execute code only if certain conditions are met. They are useful for controlling the flow of a program based on specific conditions.

There are three types of conditional statements in Python: `if`, `elif`, and `else`.

`if` Statements

An `if` statement is used to execute code if a certain condition is met. The syntax for an `if` statement is:

```
if condition:
    # do something
```

The `condition` is a boolean expression that evaluates to `True` or `False`. If the condition is `True`, the code indented below the `if` statement is executed. If the condition is `False`, the code is skipped. For example:

```
x = 10
if x > 5:
    print("x is greater than 5")
```

The indentation is important here, as it tells Python which code to execute if the condition is `True`. The indentation is four spaces by default, but you can use any number of spaces or tabs. The important thing is that you are consistent. You can also use the `tab` key to indent.

Also, note that the colon (`:`) at the end of the `if` statement is required.

`if-else` Statements

An `if-else` statement is used to execute one block of code if a certain condition is `True`, and another block of code if the condition is `False`. The basic syntax for an `if-else` statement is as follows:

```
if condition:
    # do something if condition is True
else:
    # do something else if condition is False
```

Again, note the usage of indentation and colons. This applies to all conditional statements in Python.

Here's an example to demonstrate the usage of an if-else statement:

```
x = 10

if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

`if-elif-else` Statements

You can also use multiple conditions in an `if-else` statement by using the `elif` keyword.

Here's the syntax for an `if-elif-else` statement in Python:

```
if condition1:
    # do something if condition1 is True
elif condition2:
    # do something different if condition1 is False and condition2 is True
else:
    # do something else if both condition1 and condition2 are False
```

Here's an example to demonstrate the usage of an `if-elif-else` statement:

```
x = 10

if x < 5:
    print("x is less than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is greater than 5")
```

Note that only one block of code among the `if`, `elif`, and `else` blocks will be executed, depending on the conditions. The order of the conditions matters here. If the first condition is `True`, the code in the `if` block will be executed, and the rest of the conditions will be skipped. If the first condition is `False`, the next condition will be checked, and so on. If none of the conditions are `True`, the code in the `else` block will be executed. If there is no `else` block, nothing will be executed if none of the conditions are `True`.

Nested `if` Statements

You can use nested conditional statements to create complex decision-making logic. A nested conditional statement is a conditional statement that is nested inside another conditional statement. In other words, the nested statement is executed only when the condition of the outer statement is met. For example:

```
x = 5
y = -10

if x > 0:
    if y > 0:
        print("x and y are both positive")
    else:
        print("x is positive, but y is negative")
else:
    print("x is negative")
```

Note that the indentation is important here. The code in the `if` block is indented by four spaces, and the code in the nested `if` block is indented by eight spaces. This is because the nested `if` statement is nested inside the outer `if` statement. The outer `if` statement is executed only if the condition is `True`, and the nested `if` statement is executed only if the outer `if` statement is executed.

Multiple Conditions

You can use multiple conditions in a conditional statement by using the `and`, `or` or `not` keywords. The `and` keyword is used to check if all the conditions are `True`. The `or` keyword is used to check if any of the conditions is `True`. The `not` keyword is used to check if a condition is `False`. For example:

```
x = 5
y = -10

if x > 0 and y > 0:
    print("x and y are both positive")
```

Remember the order of precedence of the `and`, `or` and `not` keywords. The `not` keyword has the highest precedence, followed by `and`, and then `or`. This means that `not` is evaluated first, then `and`, and then `or`. You can use parentheses to override the order of precedence.

Exercises

Note: Exercise 10 requires a function not covered so far in this course. Try to think how you would solve it by hand, then research the internet to try and find a way to codify your solution.

1. Write a program that takes a temperature as input and prints "too hot" if the temperature is greater than 90, "too cold" if the temperature is less than 60, and "just right" otherwise.
2. Write a program that takes a number as input and prints "positive" if the number is positive, "negative" if the number is negative, and "zero" if the number is zero.
3. Write a program that takes a character as input and prints "uppercase" if the character is an uppercase letter, "lowercase" if the character is a lowercase letter, and "not a letter" otherwise.
4. Write a program that takes a year as input and prints "leap year" if the year is a leap year, and "not a leap year" otherwise. (A leap year is a year that is divisible by 4, except for years that are divisible by 100 but not by 400.)
5. Write a program that takes two numbers as input and prints "both positive" if both numbers are positive, "both negative" if both numbers are negative, and "mixed" otherwise.
6. Write a program that takes an integer as input and prints "even" if the number is even, and "odd" otherwise.
7. Write a program that takes a number as input and prints "multiple of 3" if the number is divisible by 3, "multiple of 5" if the number is divisible by 5, "multiple of both 3 and 5" if the number is divisible by both 3 and 5, and "not a multiple of 3 or 5" otherwise.
8. Write a program that takes a string as input and prints "palindrome" if the string is a palindrome (reads the same forwards and backwards), and "not a palindrome" otherwise.
9. Write a program that takes a grade as input and prints "A" if the grade is greater than or equal to 90, "B" if the grade is greater than or equal to 80, "C" if the grade is greater than or equal to 70, "D" if the grade is greater than or equal to 60, and "F" otherwise.
10. Write a program that takes two strings as input and prints "anagram" if the strings are anagrams (contain the same letters in a different order), and "not anagram" otherwise.

Project: Simple Calculator

Project Description

Create a Python program that acts as a simple calculator. The program should ask the user to enter two numbers, and then ask which operation they want to perform: addition, subtraction, multiplication, or division. The program should then perform the selected operation and display the result to the user.

Project Requirements

1. The program should ask the user to enter two numbers.
2. The program should ask the user which operation they want to perform: addition, subtraction, multiplication, or division.
    - Feel free to expand this list of operations if you want to!
3. The program should perform the selected operation and display the result to the user.
4. The program should handle errors appropriately, for example, division by zero.

Example Output

```
Welcome to Simple Calculator!
Please enter the first number: 10
Please enter the second number: 5
Please select the operation to perform:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 3

The result is: 50
```

Further Reading

Check out the following resources for more information on conditional statements in Python:

- W3Schools
    - Python If...Else
- Real Python

  - Conditional Statements in Python

- Python Documentation
  - The `if` Statement
  - The `if-else` Statement
  - The `if-elif-else` Statement