Control Structure

What is control structures?

A control structure in programming is a set of instructions that govern the flow of execution of a program. It determines the order in which statements are executed and how many times a particular block of code is executed, based on certain conditions or events.

Control structures are used to create complex and sophisticated programs that can make decisions, iterate over data, and perform different actions based on specific conditions. Some examples of control structures in programming include:

1. Conditional statements: These allow you to execute different blocks of code based on certain conditions.
2. Loops: These allow you to execute a block of code repeatedly, either a fixed number of times or until a certain condition is met.
3. Functions: These allow you to group related code together and make it reusable.

By using control structures effectively, you can write programs that are more efficient, flexible, and easier to read and maintain.

Conditional Statements / Decision control statements

In Python, conditional statements are used to execute different blocks of code based on certain conditions. The two main conditional statements in Python are "**if**" and "**else**".

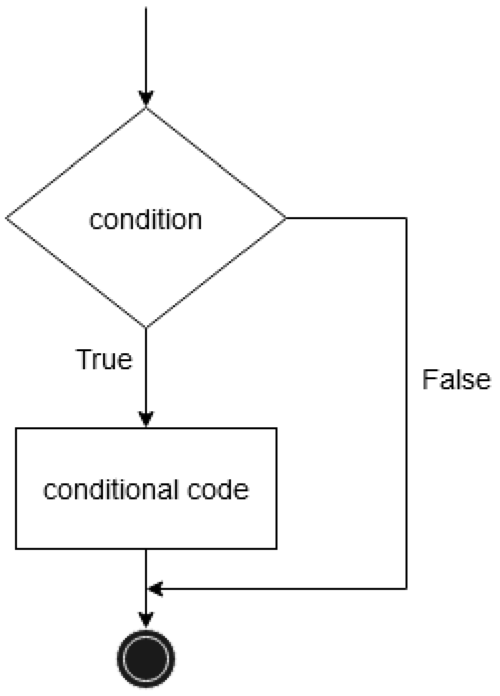
if Statement

In Python, **if** statements are used to execute a block of code conditionally, based on whether a certain condition is `True` or `False`. The syntax for an **if** statement in Python is as follows:

```
if condition:
   # code to execute if the condition is True
```

Here, **condition** is an expression that evaluates to either **True** or **False**. If the condition is **True**, then the code inside the block of the **if** statement will be executed. If the condition is **False**, then the code inside the block will be skipped.

Description of if statement



Example

```
x = 10
if x > 1:
    print("x is greater than 1")
```

In this example, the condition x > 1 is true, so the code block display x is greater than 1 will be executed.

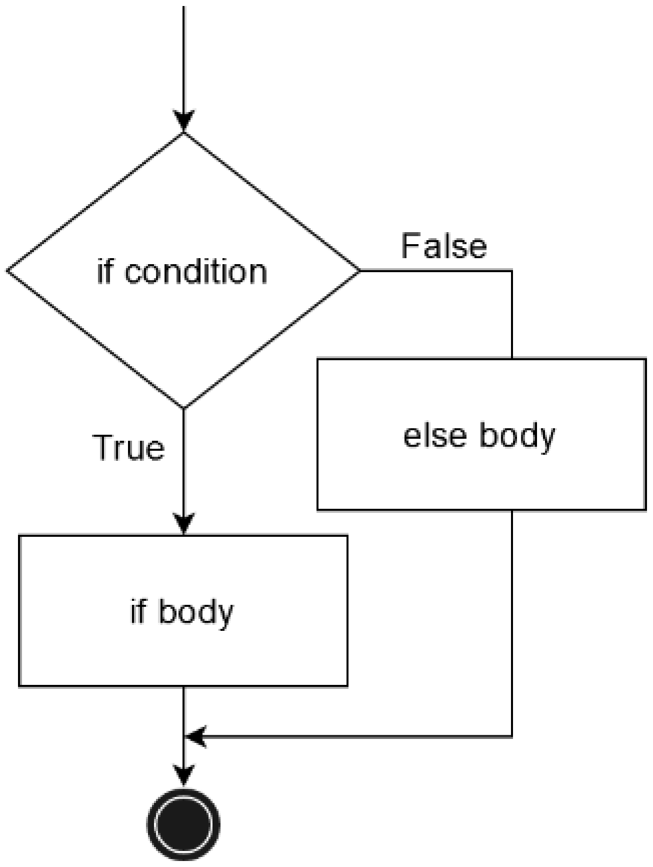
if-else Statement

You can also use **else** statements to execute a different block of code if the condition is **False**. The syntax for an **if-else** statement in Python is as follows:

if condition:
  # code **to execute** if the **condition is True**
else:
  # code **to execute** if the **condition is False**

Here, **if** the condition is **True**, the code inside the first block will be executed. Otherwise, the code inside the second block will be executed.

Description of if-else statement



Example

```
n=2

if n % 2 == 0:
   print("n is even")
else:
   print("n is odd")
```

In this example, the condition n%2 == 0 is true, so the code block print("n is even"), the condition n%2 == 0 is false, so the code block print("n is odd") will be executed.

if- elif-else Statement

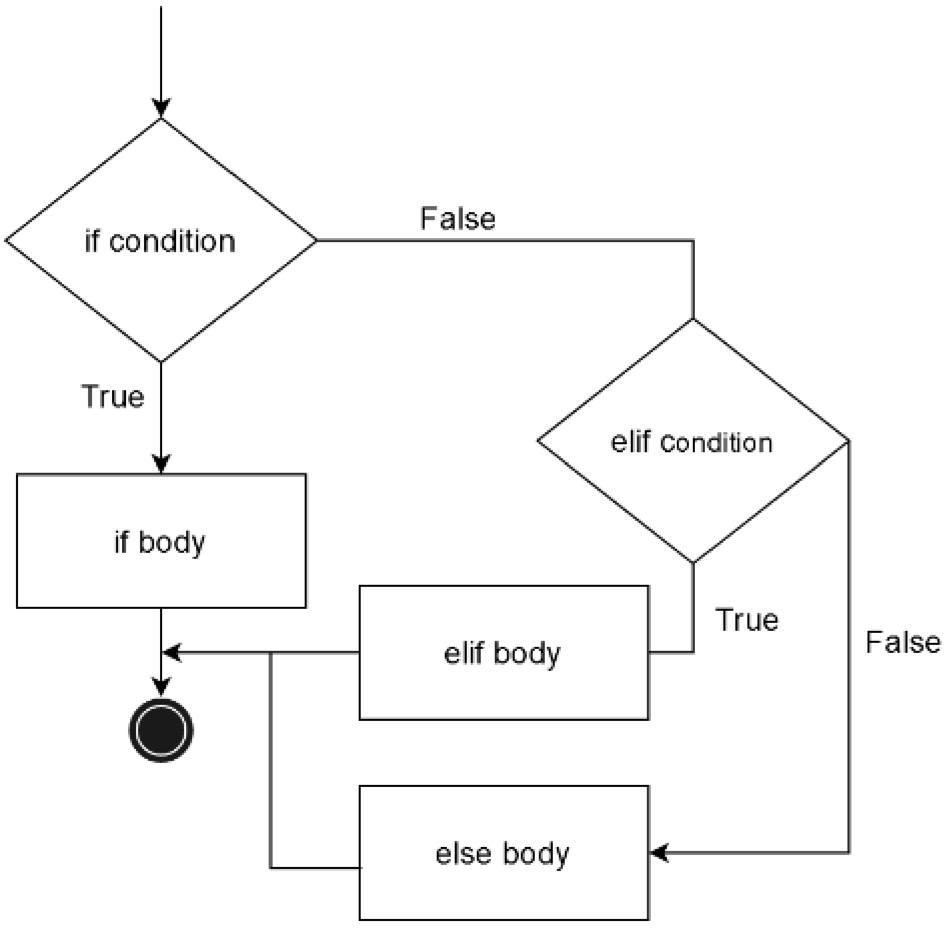
You can also use **elif** statements to test multiple conditions. The syntax for an **if-elif-else** statement in Python is as follows:

```
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 both condition1 and condition2 are False

Here, if **condition1** is **True**, the code inside the first block will be executed. If **condition1** is **False** and **condition2** is **True**, the code inside the second block will be executed. If both **condition1** and **condition2** are **False**, the code inside the third block will be executed.

Description of if- elif-else Statement



Example

```
a = 1
c = 1
if a == c:
  print("Both are Equal")
elif a > c:
   print("a is greater than c")
else:
   print("a is smaller than c")
```

In this example, initially if statement checks condition whether 1 is equal to 1. If not, then elif condition will verify that 1 is greater than 1. If this condition is true, the code block inside elif will be executed. Otherwise, the code in the else block will be executed.

nested if else statements

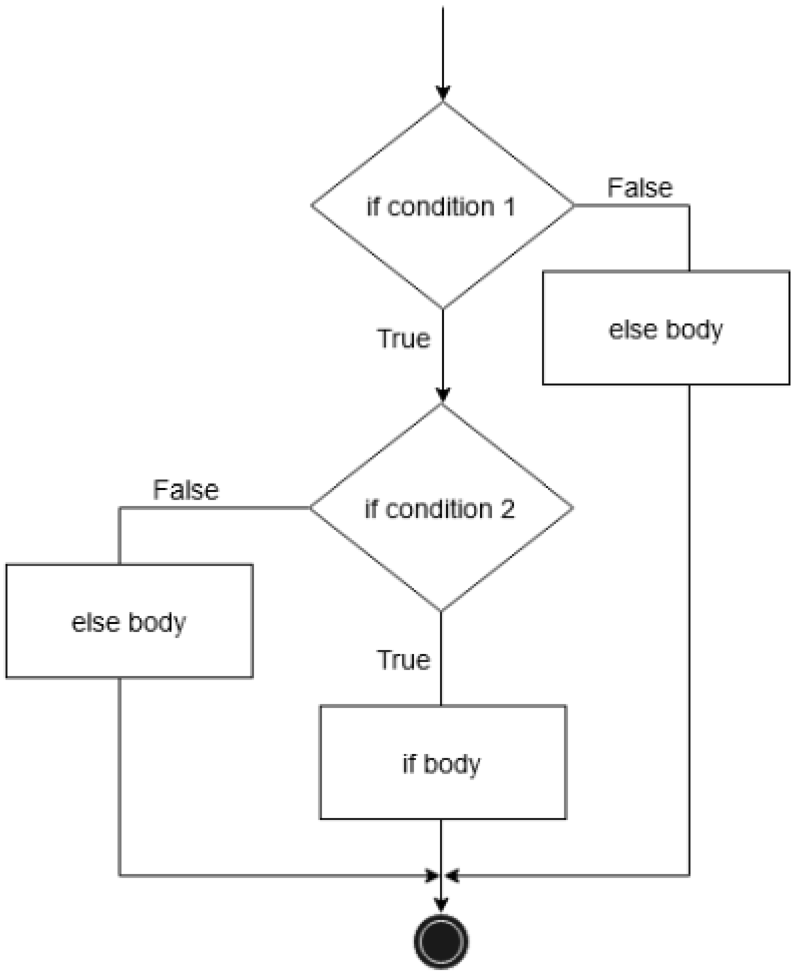
In Python, you can nest conditional statements inside other conditional statements. This allows you to create more complex logic and handle multiple conditions. Here is the syntax for a nested **if-else** statement:

```
if condition1:
   if condition2:
      # code to execute if both condition1 and condition2 are True
   else:
      # code to execute if condition1 is True and condition2 is False
else:
   # code to execute if condition1 is False
```

Here, **if** **condition1** is **True**, the inner **if-else** statement will be executed. If **condition1** is **False**, the code inside the **else** block will be executed.

If **condition1** is **True**, then **condition2** is checked. If **condition2** is also **True**, then the code inside the first block will be executed. If **condition2** is **False**, then the code inside the second block will be executed.

Description of nested if else Statement



Example

```
x = 0
y = 10
z = 5
if x < y:
   if x < z:
      print("x value is small")
   else:
       print("z value is small")
elif y < z:
    print("y value is small")
else:

     print("z is small")
```

In example, First if condition checks x is lesser than y. If yes, then another if condition that checks x is also lesser than z. If yes, then if body will be executed.

Examples

Example 1 of a simple `if` statement program in Python:

Write a program to check the given number is even

```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("The number is even.")
```

In this program, the user is prompted to enter a number. Then, an **if** statement is used to check whether the number is **even**.

If the number is divisible by **2** (i.e., the remainder when divided by **2** is **0**), the program will print "The number is even." If the number is not divisible by **2** (i.e., the remainder when divided by **2** is **1**), the program will not print anything.

Note that there is **no else** statement in this program. This is because we only want to print a message if the number is even. If the number is odd, we don't need to do anything.

Example 2 of an if-else statement program in Python:

Write a program to check if a number is positive or zero, negative

```
num = float(input("Enter a number: "))

if num >= 0:
    print("The number is positive or zero.")

else:

    print("The number is negative.")
```

In this program, the user is prompted to enter a number. Then, an **if-else** statement is used to check whether the number is positive or negative.

If the number is greater than or equal to **0**, the program will print "The number is positive or zero." If the number is less than **0**, the program will print "The number is negative."

Note that the **else** statement is used to handle the case where the number is negative. If the number is not negative, then the **if** statement will be True and the program will execute the code inside the first block.

Example 3 of an if and elif and else statement program in Python:

Write a Program to check if a number is positive, zero or negative

```
num = float(input("Enter a number: "))

if num > 0:
    print("The number is positive.")
elif num == 0:
    print("The number is zero.")
else:
    print("The number is negative.")
```

In this program, the user is prompted to enter a number. Then, an **if** statement is used to check whether the number is positive, negative, or zero.

If the number is greater than **0**, the program will print "The number is positive." If the number is equal to **0**, the program will print "The number is zero." If the number is less than **0**, the program will print "The number is negative."

Note that the **elif** statement is used to check for the case where the number is equal to **0**. Without this statement, the program would incorrectly classify **0** as a negative number.

Example 3 of an nested if else statement program in Python:

Write a Program to check if a number is positive, negative or zero

```
num = float(input("Enter a number: "))
```

```
if num >= 0:
   if num == 0:
      print("The number is zero.")
   else:
      print("The number is positive.")
else:
   print("The number is negative.")
```

In this program, the user is prompted to enter a number. Then, a nested **if-else** statement is used to check whether the number is positive, negative, or zero.

If the number is greater than or equal to **0**, the program will execute the first block of code. Then, the program will check whether the number is equal to **0**. If the number is equal to **0**, the program will print "The number is zero." If the number is not equal to **0**, the program will print "The number is positive."

If the number is less than **0**, the program will execute the code inside the else block and print "The number is negative."

Note that the second **if** statement is nested inside the first **if** statement. This allows the program to handle the case where the number is equal to **0** separately from the case where the number is positive.

Example 4

write a program to check the serious of earthquake
if ritcher value is greater than 5.5 the serious damage
if ritcher value is less than 5.5 and greater than 5 the some damage
if ritcher value is less than 5 the little or no damage
condition : ritcher value always greater than 0

```
ritcher = float(input())
if ritcher>0:
            if ritcher >5.5:
                        print ("Serious damage")
            elif ritcher >=5 and ritcher<=5.5:
                        print ("Some damage")
            elif ritcher <5:
                        print ("Little or no damage")
else:
            print ("Invalid input")
```

In this program, takes the Richter scale value as input and returns the corresponding seriousness level based on the conditions you provided.

Exercise :

1. Write a Python program that prompts the user to enter two numbers and an operator (+, -, *, /). The program should then perform the corresponding operation on the two numbers and print the result.
2. Write a Python program that prompts the user to enter a integer number and then check the number is leap year or not.
3. Write a program that prompts the user to enter a letter and then prints whether the letter is a vowel or a consonant.
4. Write a program that prompts the user to enter three numbers and then prints the largest of the three numbers.
5. Write a program that prompts the user to enter their exam score and then prints their letter grade based on the following scale: 90-100 (A), 80-89 (B), 70-79 (C), 60-69 (D), below 60 (F).
6. Write a Python program to prompt the user to enter their age, and then print a message indicating whether they are a child, a teenager, or an adult. Use the following criteria to determine the age group:

   - Children: ages 0-12
   - Teenagers: ages 13-19
   - Adults: ages 20 and above

   Your program should output one of the following messages, depending on the user's age:

   - If the user is a child: "You are a child."
   - If the user is a teenager: "You are a teenager."
   - If the user is an adult: "You are an adult."

   Hint: Use a series of `if-elif-else` statements to check the age group.

7. Write a Python program that prompts the user to enter a number between 1 and 7, and then prints the corresponding day of the week. For example, if the user enters "1", the program should print "Monday", and if the user enters "7", the program should print "Sunday". If the user enters a number outside of the range 1-7, the program should print an error message.

   Hint: Use a series of `if-elif-else` statements to check the user's input and print the corresponding day of the week. You can use a dictionary to store the day of the week names and their corresponding numbers.

8. Write a program that prompts the user to enter their weight (in kilograms) and height (in meters) and then calculates and prints their body mass index (BMI). Then, based on their BMI, print whether they are underweight (BMI < 18.5), normal weight (18.5 <= BMI < 25), overweight (25 <= BMI < 30), or obese (BMI >= 30).

9. Write a program that prompts the user to enter their hourly wage and the number of hours they worked in a week. If they worked more than 40 hours, they should be paid time and a half for the overtime hours. Print their gross pay for the week.
10. Write a program that prompts the user to enter a month (as an integer between 1 and 12) and a year (as a four-digit integer). The program should then print the number of days in that month. Be sure to handle leap years correctly (i.e., February should have 29 days in leap years).
11. Write a program that prompts the user to enter a number between 1 and 100. The program should then print "Fizz" if the number is divisible by 3, "Buzz" if the number is divisible by 5, and "FizzBuzz" if the number is divisible by both 3 and 5. If the number is not divisible by 3 or 5, just print the number itself.