IF Statement

Syntax

```
if condition is true:

               Statement A
```

If the IF condition is true, then execute statement A.

You can have more than one statement inside an if statement. A group of statements is known as a block of code. We group statements together in a block because they are related. The statements need to be run together.

```
if condition is true:

               Statement A

               Statement B

               Statement C

               Statement D

               Some more code
```

The above code would execute statements A, B, C and D and any other code following it only if the condition in the if statement is true.

Notice that every if statement needs to be followed with a **colon**. Any statements within the IF statement must be indented by exactly **four** spaces or a tab space. Most Python code editors will automatically indent code after IF statements by four spaces.

```
if condition is true:

               Statement A

               Statement B

               Statement C

            Statement D
```

In the above example, Statement D is only indented by three spaces and is therefore not part of the if statement code block that includes statements A, B and C. Statement D will execute whether or not the if condition is true.

Conversely, in the next example, Statement D is indented by more than four spaces and Python will display a **Syntax error** as it expects you to use the same number of spaces for all the statements in the block.

```
age= 25

   if  age > 20:

       print("You're too old! )

         print("Why are you here?")

    SyntaxError: unexpected indent
```

Conditional Operators

Symbol	Definition
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!	Not

Examples

1. The program below assigns the value 13 to the variable, age. It then checks to see if the value in age is greater than 19. If true, it outputs a message stating that you are no longer a teenager.

```
age = 13

  if age > 19:

      print("You are no longer a teenager")
```

2. The following program assigns a value to a variable, age, and checks to see if the value equals to 10. If so, it outputs a message stating that you are 10 years old.

```
age = 10

  if age == 10:

      print("You are 10 years old")
```

3. The program below accepts a number and displays if the number entered is even.

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

  if n % 2 == 0:

      print("The number is even")
```

4. The following program accepts a number and displays if the number is odd.

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

  if n % 2 != 0:

      print("The number is odd")
```

IF...ELSE Statement

Syntax

```
if condition is true:

              Statement A

      else:

              Statement B
```

If the if condition is true, then execute statement A, else execute Statement B.

Notice that the else also needs to be followed with a **colon**. Any statements within the else statement must be indented by exactly **four** spaces or a tab space, just like the if statement. Most Python code editors will automatically indent code after else statements by four spaces. An if statement can only have one and only one else statement.

Examples

1. The following program accepts a response from the user. If the user enters, “Yes”, the program will output the message, “Bye Bye!” If the user enters any other response, the program will output the message, “Thanks for staying around!”

```
answer = input("Are you sure you want to exit the program?")

if answer == "Yes":

         print("Bye Bye!")

else:

         print("Thanks for staying around!")
```

2. The program below outputs the message, “You have passed” if the score entered is 60 or above. If not, it outputs the message, “You have failed!”

```
score = float(input("Enter percent the score: "))

if score >= 60:

         print(“You have passed”)

else:

          print(“You have failed”)
```

3. The program below outputs whether the person is old enough to vote or not based on the value in the variable, age.

```
age = 17

if age >= 18:

         print(“You are old enough to vote!”)

         print(“Have you registered to vote yet?”)

else:

         print(“Sorry, you are too young to vote.”)

         print(“Please register to vote as soon as you turn 18!”)
```

IF…ELIF…ELSE Statement

Syntax

```
if condition_1:
        statement_block_1
elif condition_2:
        statement_block_2
elif condition_3:
        statement_block_3
...
else:
        statement_block_3
```

We can extend an if statement even further with elif, which is short for else if. For example, we can check if a person's age is 10, 11, 12 and so on and have the program do something different based on the answer.

The IF…ELIF code will test each condition in turn until it finds one that evaluates to true. It will then execute the code inside that elif statement and exit the if statement completely – in other words, it will not evaluate anymore of the elif statements. If none of the elif conditions evaluate to true, then the code inside the else statement will execute.

Examples

1. The following program outputs the age of a person based on the value assigned to the variable age.

```
age=12
if age == 10:
        print("You are 10 years old.")
elif age == 11:
        print("You are 11 years old.")
elif age == 12:
```

```
        print("You are 12 years old.")
elif age == 13:
        print("You are 13 years old.")
else:
        print("You are either less than 10 years or older than 13 years.")
```

2. In the following example the user inputs an alphabet from the user. If the user enters a, e, i, o or u then the program should display a message indicating that the entered letter is a vowel. Otherwise the program will display a message indicating that the letter is a consonant.

```
letter = input("Enter a letter in the alphabet in lowercase: ")
if letter == "a":
        print("The letter you entered is a vowel")
elif letter == "e":
          print("The letter you entered is a vowel")
elif letter == "i":
          print("The letter you entered is a vowel")
elif letter == "o":
          print("The letter you entered is a vowel")
elif letter == "u":
          print("The letter you entered is a vowel")
else:
         print("The letter you entered is a consonant")
```

3. This program accepts a value of the number of sides of an object between 2 and 10 and outputs the name of the object depending on the number of sides entered, using the table below as a guide.

Number of Sides	Shape
2	Line
3	Triangle

Number of Sides	Shape
4	Rectangle or Square
5	Pentagon
6	Hexagon
7	Septagon
8	Octagon
9	Nanogon
10	Decagon

```
sides = int(input("Enter the number of sides: "))

if sides == 2:

         print(“The object is a line”)

elif sides == 3:

           print(“The object is a triangle”)

elif sides == 4:

           print(“The object is a rectangle or square”)

elif sides == 5:

           print(“The object is a pentagon”)

elif sides == 6:

           print(“The object is a hexagon”)

elif sides == 7:

           print(“The object is a septagon”)

elif sides == 8:

           print(“The object is a octagon”)

elif sides == 9:

           print(“The object is a nanogon”)
```

```
elif sides == 10:
            print("The object is a decagon")
else:
          print("Invalid number entered")
```

Compound IF Statement

Syntax

If condition_1 and condition_2 and condition_3 and…condition_n:
    statement_block_1

If condition_1 or condition_2 or condition_2 or…condition_n:
    statement_block_1

We can combine conditions in an IF statement by using the keywords **and** and **or**, which produces shorter and simpler code.

And Operator

Condition_1	Condition_2	Condition_1 and Condition_2
True	True	True
True	False	False
False	False	False
False	False	False

Or Operator

Condition_1	Condition_2	Condition_1 or Condition_2
True	True	True
True	False	True
False	True	True

Condition_1	Condition_2	Condition_1 or Condition_2
False	False	False

Not Operator

Condition_1	NOT Condition_1
True	False
False	True

Examples

1. The following program outputs the age of a person based on the value assigned to the variable age. If the value in the variable age is equals 10 or 11 or 12 or 13, the program will output, “You are between 10 and 13 years old.” If not, it will output, “You are younger than 10 or older than 13.”

```
age = 12

 if age == 10 or age == 11 or age == 12 or age ==13:

        print(“You are between 10 and 13 years old.”)

else:

        print(“You are younger than 10 or older than 13.”)
```

We could also rewrite the above example as follows and obtain the same result. If the value in the variable age is less than 10 or great than 13, the program will output, “You are younger than 10 or older than 13.” Otherwise, it will output, “You are between 10 and 13 years old.”

```
age = 12

if age < 10 or age > 13

       print(“You are younger than 10 or older than 13.”)

else:

       print(“You are between 10 and 13 years old.”)
```

2. The next program outputs the age of a person based on the value assigned to the variable age, this time using the and operator. If the value in the variable age is between 10 and 13, the program will output, “You are between 10 and 13 years old.” If not, it will output, “You are younger than 10 or older than 13.”

```
age = 12

 if age >= 10 and age <=13:

        print(“You are between 10 and 13 years old.”)

else:

        print(“You are younger than 10 or older than 13.”)
```

3. The following example makes use of the and operator. The first condition checks if the allowance (allowance) is greater than or equal to the price of the toy (toy_price). This condition ensures that you have enough money to buy the toy.

The second condition checks if the toy price is less than or equal to 10. This condition ensures that the toy is within an acceptable price range.

If both conditions are true, this means you have enough money and the toy is within an acceptable price range. So, it will print “You have enough money to buy the toy!” indicating that you can buy the toy. Otherwise, if any of the conditions is false, it prints “Sorry, you don’t have enough money to buy the toy.” indicating that the you cannot afford the toy.

```
allowance = 10

 toy_price = 8


if allowance >= toy_price and toy_price <= 10:

    print("You have enough money to buy the toy!")

else:

    print("Sorry, you don't have enough money to buy the toy.")
```

4. In this example, the code uses the or and not operators to check the conditions if you want to go outside and play.

The first condition checks if it’s raining (is_raining). If it’s raining, this condition evaluates to True. The second condition uses the not operator to check if it’s not the weekend (is_weekend). If it’s not the weekend, this condition evaluates to True.

The or operator combines these two conditions. It checks if either condition is true. If either condition is true, it means you cannot go outside to play. In that case, it prints “Sorry, you cannot go outside to play.”

On the other hand, if both conditions are false, meaning it’s not raining and it’s the weekend, the code inside the if statement will not execute, and it will move to the else block. In this case, it prints “You can go outside and play!”

```
is_raining = True

is_weekend = False


if is_raining or not is_weekend:

         print("Sorry, you cannot go outside to play.")

else:

         print("You can go outside and play!")
```

Nested IF Statements

Syntax

```
if condition_1:

          print( "true")

          if condtion_2:

                 print("yes")

           else:

                 print("no")

   else:

          print("false")
```

If condition_1 evaluates to true, the program will then evaluate whether the condition_2 also evaluates to true. If both cases are true, the output will be:

true
yes

If, however, condition_1 evaluates to true, but condition_2 evaluates to false, then the output will be:

true
no

And if condition_1 evaluates to false, the nested if-else statement will not run, so the else statement will run alone, and the output will be:

false

Nested if statements are like solving puzzles one after another. You have a main puzzle (outer if statement), and if you solve it, you can move on to a smaller puzzle (inner if statement) inside it. The smaller puzzle helps you make a more specific decision based on the outcome of the main puzzle.

Examples

1. In this example, the nested if statements check multiple conditions to determine if you can go to the park. Each if statement represents a different condition to be met. First, it checks if it's sunny outside. If it is, it moves on to the next condition: do you have permission to go?

If both conditions are true, it prints "You can go to the park!" indicating that you can go. However, if any of the conditions is false, it prints an appropriate message explaining why you can't go to the park.

```
is_sunny = True

has_permission = True


if is_sunny:

         if has_permission:

                        print("You can go to the park!")


            else:

                         print("Sorry, you don't have permission to go.")

else:

         print("It's not sunny outside. Wait for a sunny day to go to the
park.")
```

2. In the following program, you need to have both the key and the map to find the treasure. If you have both the key and the map, it prints “You have both the key and the map. You can find the treasure!”

If you have the key but not the map, it prints “You don’t have the map. Keep exploring to find it!”

If you don’t have the key, it skips the nested if statement and prints “You don’t have the key. Keep searching to find it!”

```
has_key = False

has_map = True

if has_key:

        print("You have the key!")

        if has_map:

                print("You have both the key and the map. You can find the
treasure!")

        else:

                print("You don't have the map. Keep exploring to find it!")

else:

        print("You don't have the key. Keep searching to find it!")
```