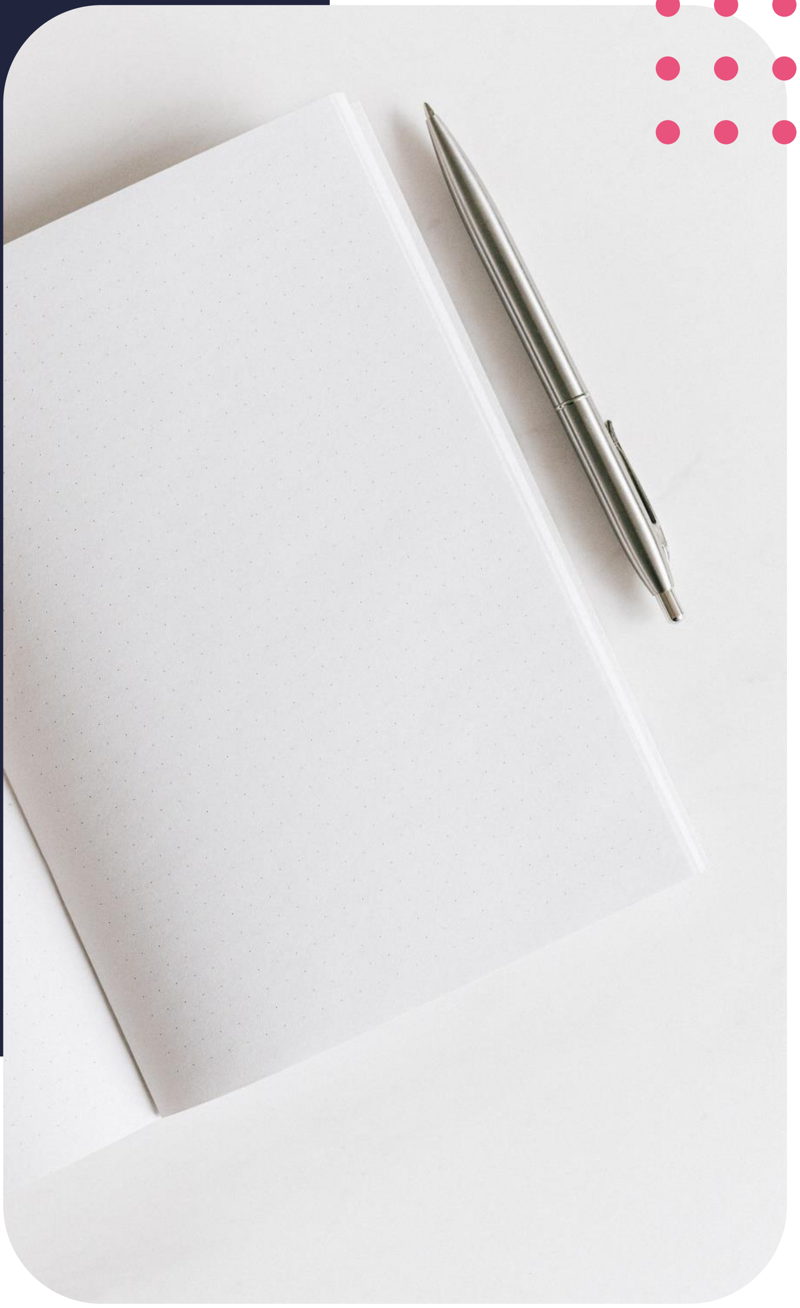
Diploma in Python Programming

Decision Making

Summary Notes

Contents



Lesson outcomes

By the end of this lesson, you should be able to:

- Learn the process of decision making
- Understand python *if* statements
- Apply decision making code

Decision Making

Python *if* statement

The **if** statement is utilised to make a decision in the code block, whether to execute the body of the if statement or not.

When a condition is a True Boolean data type the statement(s) inside the code block will be executed but if the condition is a False data type, the statement(s) inside the code block will not be executed and when this happens the program (code) will skip them entirely.

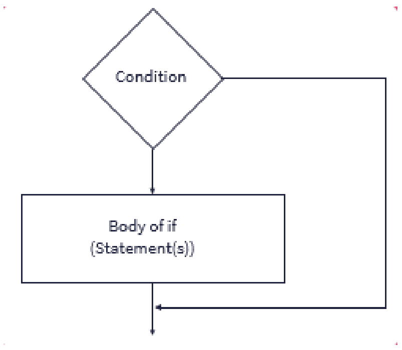
```
if condition:
    statement(s)
```

When the condition expression in the if statement is replaced by a True Boolean data type as indicated in the example, the statement(s) inside the code block will be executed by the Python interpreter.

```
if True:
    statement(s)
```

If the condition is a False data type as shown in the example, the statement(s) inside the code block will not be executed by the Python interpreter.

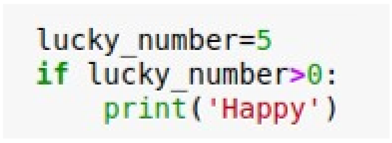
```
if False:
    statement(s)
```



Code displaying *if* statement

We first created a variable named lucky − number and assigned it to the value 5. Then this is followed by an if statement (if lucky − number &gt; 5) and the body statement print(0 Happy 0).

This means that if the lucky − number is bigger than 5, then the print statement will be executed. In this case, since the lucky − number = 5, which is greater than zero, the print statement will be executed since the condition from the if statement is true.

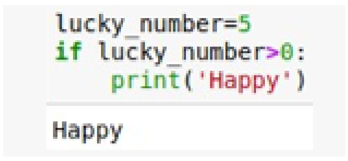


```
lucky_number=5
if lucky_number>0:
    print('Happy')
```

The operator & > in this code is the Python comparison operator, and if the value of the left operand is greater than the value of right operand, then the condition becomes true.

For example 4 > 2

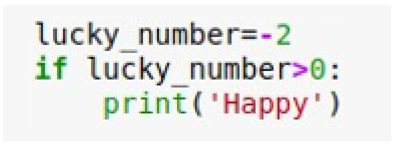
which makes this a True Boolean data type and if 4 < 7 this proves to be a False Boolean data type.



```
lucky_number=5
if lucky_number>0:
    print('Happy')

Happy
```

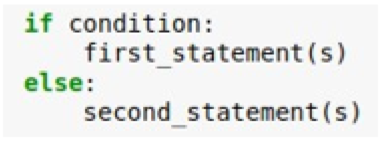
However, if the variable *lucky − number* is assigned a value less than 0, as indicated in the example. The body print statement will not be executed, since the lucky − number is less than zero.



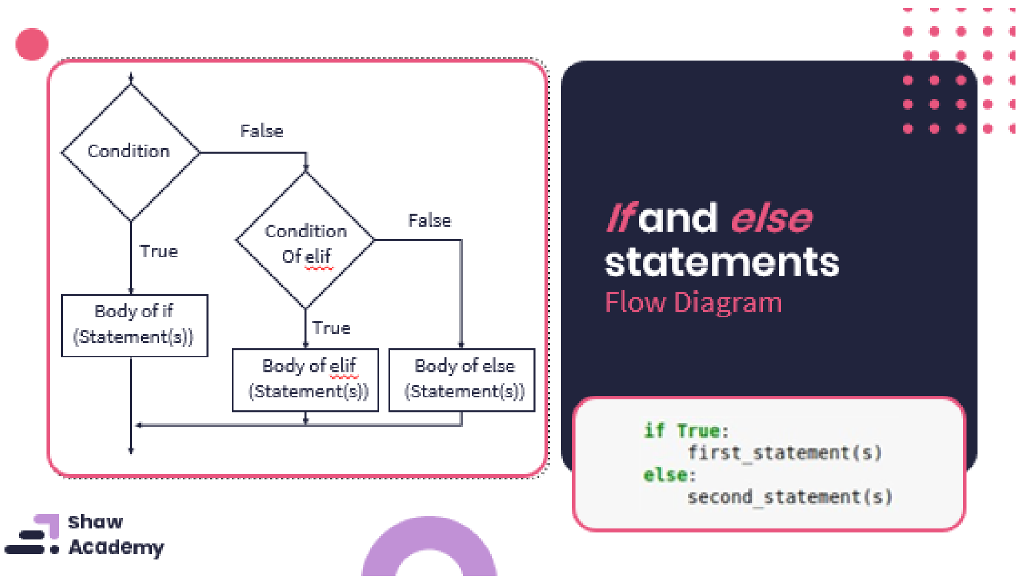
```
lucky_number=-2
if lucky_number>0:
    print('Happy')
```

Python *if, elif* and *else* Statements

Sometimes an if statement can be combined with an else statement, which is executed **when** the if statement Boolean expression (condition) is False or 0. But, **when** the if statement Boolean expression is True, the else statement is not executed.



```
if condition:
    first_statement(s)
else:
    second_statement(s)
```



```
if True:
    first_statement(s)
else:
    second_statement(s)
```

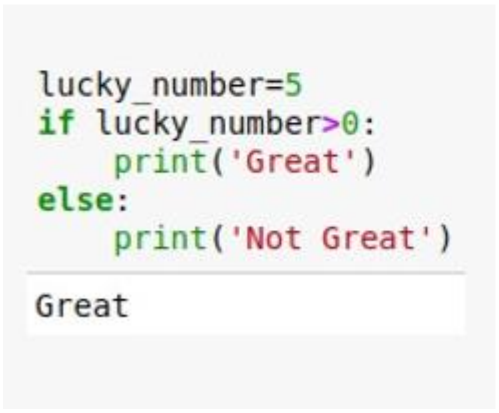
The above is a flow diagram of an If and else statement. Use the condition expression, from the previous code, to equal to a True

Boolean data type. And then the first – statement(s) will be executed and the else (second – statement(s)) will not be executed.

However when the if condition is False the first – statement will not be executed and the else (second – statement(s)) will be executed:

```
if False:
    first_statement(s)
else:
    second_statement(s)
```

Example of an if and else statement combined:



```
lucky_number=5
if lucky_number>0:
    print('Great')
else:
    print('Not Great')
```

```
Great
```

Since the variable lucky_number is equal to 5 and greater than 0, the if statement will yield a True Boolean data type, and the body print(0 Great 0) statement will be executed, the else statement will be ignored, it won’t be executed. Here is another alternative example:

```
lucky_number=-2
if lucky_number>0:
    print('Great')
else:
    print('Not Great')

Not Great
```

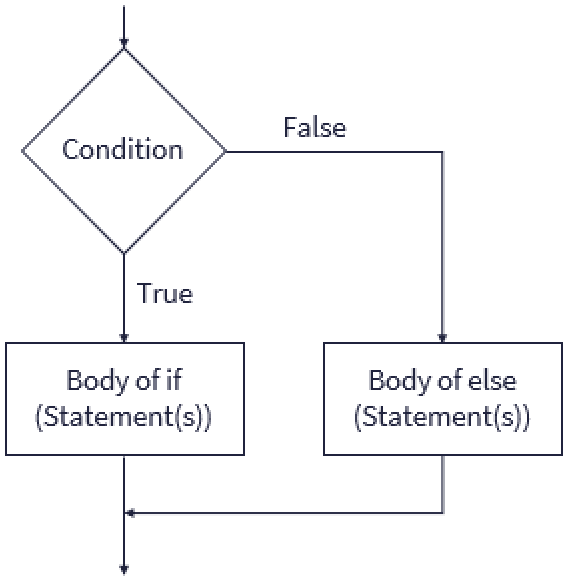
Since the if statement now equals to a False Boolean data type, because -2 is not greater than 0. The body of the else statement (print(0 Not Great 0)) will be executed since the if statement is False.

Elif statement

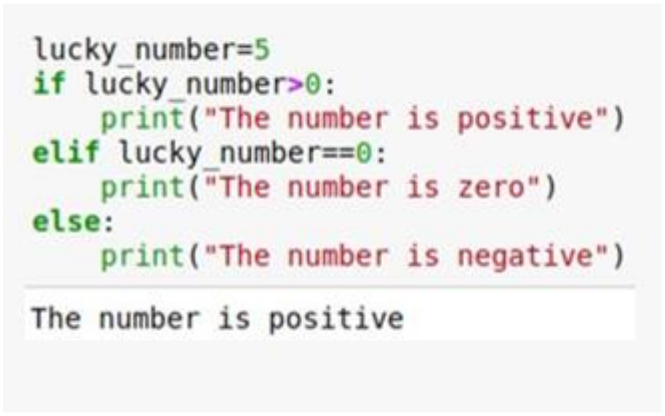
When we have multiple conditions to check, we can use the elif statement which is the combination of the if and else statements. Heres an example of the if, elif and else statements:

```
if condition_1:
    first_statement(s)
elif condition_2:
    second_statement(s)
else:
    last_statement(s)
```

If the first condition (condition − 1), the if statement is False, then the next condition (condition − 2) which is the elif statement will be checked. If the elif condition is also False, then the next condition will be checked, and so on. If all the conditions are False, then the last statement, which is the else statement will be executed.



When the variable lucky − number is positive, the string, The number is positive, is printed from the if body statement. If the variable lucky − number is equal to 0, the string, The number is zero, will be printed from the elif body statement. And when the variable lucky − number is negative, the string, The number is negative from the else body statement will be printed.



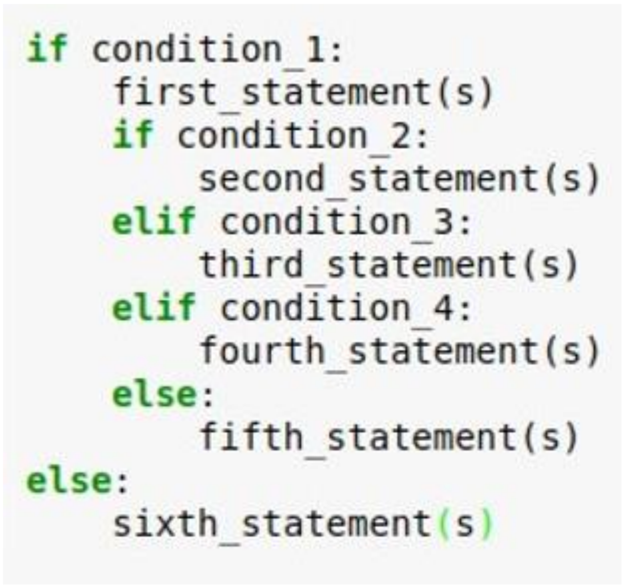
```
lucky_number=5
if lucky_number>0:
    print("The number is positive")
elif lucky_number==0:
    print("The number is zero")
else:
    print("The number is negative")
```

```
The number is positive
```

Python nested *if* Statements

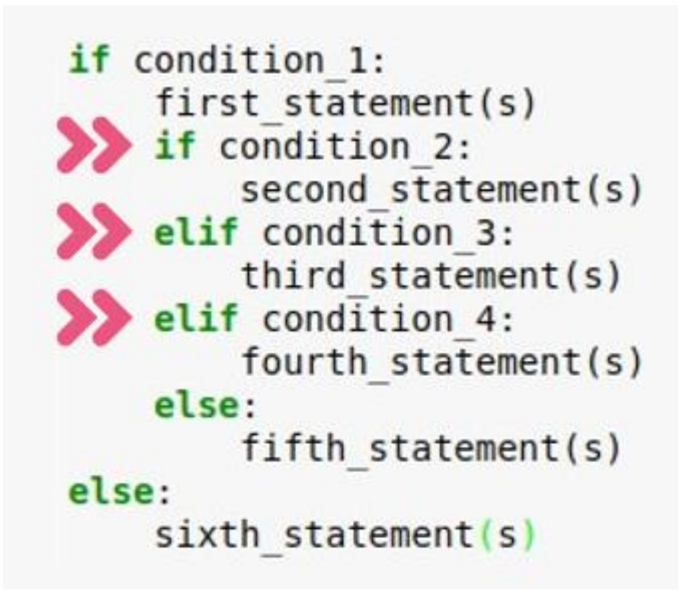
Multiple cascading conditions

We will start of with Multiple cascading conditions. We place another condition statement inside the code block - not below - very important, to further check whether the contained condition if it is true or not. In such instances we use the nested if statements, where we have if, elif and else statements inside another if, elif and else statements. Let me show you what I mean. Too the right-hand side of the screen you'll find a Python nested if example, In this example condition − 2, condition − 3 and condition − 4 are the condition statements inside which means contained under condition − 1 statement, including the first else statement fifth_statement(s). The last else statement is executed when the first condition − 1 statement is False.



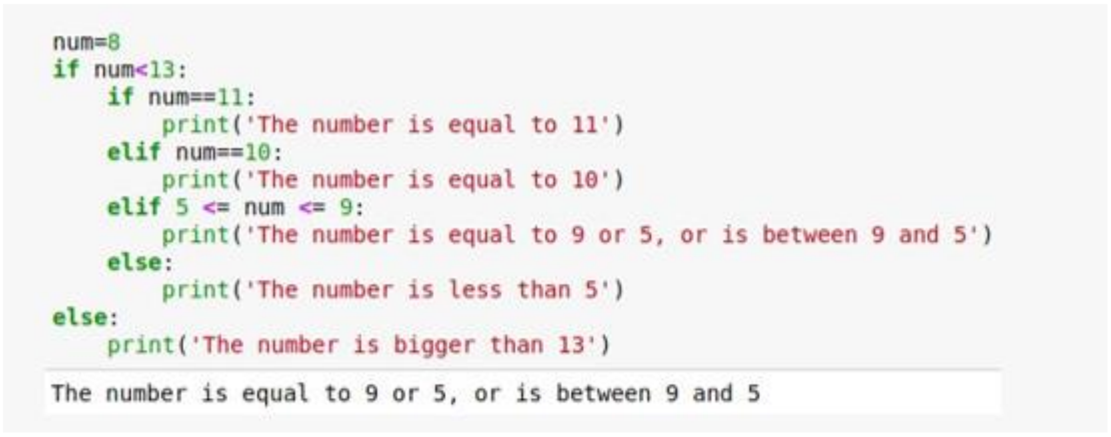
```
if condition_1:
    first_statement(s)
    if condition_2:
        second_statement(s)
    elif condition_3:
        third_statement(s)
    elif condition_4:
        fourth_statement(s)
    else:
        fifth_statement(s)
else:
    sixth_statement(s)
```

In the below example condition − 2, condition − 3 and condition – 4 are the condition statements inside, which means contained under condition − first statement, including the first else statement, fifth_statement(s). The last else statement is executed when the first condition condition − first statement is False.



```
if condition_1:
    first_statement(s)
    if condition_2:
        second_statement(s)
    elif condition_3:
        third_statement(s)
    elif condition_4:
        fourth_statement(s)
    else:
        fifth_statement(s)
else:
    sixth_statement(s)
```

Here, the first condition is true since the variable num = 8 is less than 13. The code checks the condition statements contained inside the first condition and finds the second elif condition statement to be True, since the variable num = 8 is between 5 and 9. Then the print statement, “The number is equal to 5 or 9, or is between 5 and 9”, is printed.



```
num=8
if num<13:
    if num==11:
        print('The number is equal to 11')
    elif num==10:
        print('The number is equal to 10')
    elif 5 <= num <= 9:
        print('The number is equal to 9 or 5, or is between 9 and 5')
    else:
        print('The number is less than 5')
else:
    print('The number is bigger than 13')
```

```
The number is equal to 9 or 5, or is between 9 and 5
```