1- Conditional Statements if/else

Conditional statements are statements that execute different blocks of code depending on whether a condition is true or false. They are often used to control the flow of a program or to make decisions based on some criteria. In Python, the most common conditional statements are if, elif and else1

1.1- The if statement

In [3]:
```
if 2 > 1:
    print("This is a True statement!")
```

```
This is a True statement!
```

In [5]:
```
var_1 = 3
var_2 = 2
if var_1 > var_2:
    print("This is True")
```

```
This is True
```

1.2- else statement

In [7]:
```
var1 = 5
var2 = 3
if var1 > var2:
    print("This is True")
else:
    print("That was False!")
```

```
This is True
```

In [9]:
```
var1 = 2
var2 = 3
if var1 > var2:
    print("This is True")
else:
    print("That was False!")
```

```
That was False!
```

In [11]:
```
value = 8
if value < 10:
    print("That's a good amount!")
elif 10 <= value <= 20:
    print("I will pay that")
else:
    print("Wow! This is huge amount!")
```

```
That's a good amount!
```

In [12]:
```
value = 12
if value < 10:
    print("That's a good amount!")
elif 10 <= value <= 20:
    print("I will pay that")
else:
    print("Wow! This is huge amount!")
```

```
I will pay that
```

In [13]:
```
value = 21
if value < 10:
    print("That's a good amount!")
elif 10 <= value <= 20:
    print("I will pay that")
else:
    print("Wow! This is huge amount!")
```

```
Wow! This is huge amount!
```

2- Boolean Operations in Python

Boolean operations in Python are used to compare values and return True or False. There are three main types of Boolean operations: and. or and not

In [14]:
```
# Or operator
x = 10
y = 20
if x < 10 or y > 15:
    print("This statement was True!")
```

```
This statement was True!
```

In [15]:
```
x = 10
y = 12
if x < 10 or y > 15:
    print("This statement was True!")
```

In [16]:
```
# and operator
x = 10
y = 10
if x == 10 and y == 15:
    print("This statement was True")
else:
    print("The statement was False!")
```

```
The statement was False!
```

In [17]:
```
x = 10
y = 15
if x == 10 and y == 15:
    print("This statement was True")
else:
    print("The statement was False!")
```

```
This statement was True
```

In [18]:
```
# Not example
my_list = [1, 2, 3, 4]
x = 10
if x not in my_list:
    print("'x' is not in the list, so this is True!")
```

```
'x' is not in the list, so this is True!
```

In [19]:
```
my_list = [1, 2, 3, 4]
x = 2
if x not in my_list:
    print("'x' is not in the list, so this is True!")
```

In [20]:
```
x = 10
if x != 11:
    print("x is not equal to 11!")
```

```
x is not equal to 11!
```

In [21]:
```
x = 10
if x != 10:
    print("x is not equal to 11!")
```

In [23]:
```
my_list = [1, 2, 3, 4]
x = 8
z = 10
if x not in my_list and z != 8:
    print("This is True!")
```

```
This is True!
```

3- None

None is a keyword in Python that represents a null value or no value at all. It is a data type of its own (NoneType) and only None can be None

In [25]:
```
# None examples
empty_list = []
empty_tuple = ()
empty_string = ""
nothing = None
```

In [27]:
```
# None examples
if empty_list == []:
    print("It's an empty list!")
```

In [29]:
```
if empty_tuple:
    print("It's not an empty tuple!")
```

In [33]:
```
if not empty_string:
    print("This is an empty string!")
```

```
This is an empty string!
```

In [34]:
```
if not nothing:
    print("Then it's nothing!")
```

```
Then it's nothing!
```

In [35]:
```
if empty_string == "":
    print("This is an empty string!")
```

```
This is an empty string!
```

In [36]:
```
empty_list == empty_string
```

Out[36]:
```
False
```

In [37]:
```
empty_string == nothing
```

Out[37]:
```
False
```

In [38]:
```
empty_string != nothing
```

Out[38]:
```
True
```

4- Special Characters in Strings

In [40]:
```
# 4.1- Special characters n
print("I have a \n new line in the middle")
```

```
I have a
 new line in the middle
```

In [41]:
```
# 4.2- Special character t
print("This sentence is \ttabbed!")
```

```
This sentence is        tabbed!
```

In [42]:
```
#4.3 Special character \
print("This is a backslash \\")
```

```
This is a backslash \
```

In [ ]:
```

```