What is a Conditional Statement?

- A conditional statement in Python is a way to make decisions in your program.
- It checks if something is True or False, and then runs different code depending on the result.
- The most common conditional statement is if.

In [ ]:
```
age = 18 # Example age variable

if age >= 18:     # condition check
    print("You are an adult.")
else:
    print("You are a minor.")
```

```
You are an adult.
```

Explation

- if age >= 18: → Python checks if the condition (age >= 18) is True.
- If it's True, it runs the code under if.
- If it's False, it runs the code under else.

1. if only

In [ ]:
```
if 5 > 3: # condition check
    print("Yes, 5 is greater than 3")
```

```
Yes, 5 is greater than 3
```

1. if else

In [4]:
```
x = 10
if x % 2 == 0:
    print("Even number")
else:
    print("Odd number")
```

```
Even number
```

1. if elif else

In [6]:
```
marks = 75
if marks >= 90:
    print("Grade A")
elif marks >= 60:
    print("Grade B")
else:
    print("Grade C")
```

```
Grade B
```

Difference between if, elif, and else

- `if` checks a condition and runs the code block if it's True.
- `elif` (else if) checks another condition if the previous `if` was False.
- `else` runs a code block if all previous conditions were False.

In [ ]:
```
# if
# Used to check the first condition.
# Runs only if the condition is True.

x = 5
if x > 0:
    print("Positive number")
```

```
Positive number
```

In [ ]:
```
# elif (short for "else if")
# Used when you want to check more than one condition.
# Runs only if the previous if (or other elif) was False.

x = 0
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
```

```
Zero
```

In [11]:
```
# else
# Runs when all previous conditions are False.
# It’s like a “default option”.

x = -5
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

```
Negative
```

In [12]:
```
if True:
    print('Data Science')
```

```
Data Science
```

In [13]:
```
if True:
print('ds')
```

```
  Cell In[13], line 2
    print('ds')
    ^
IndentationError: expected an indented block after 'if' statement on line 1
```

In [14]:
```
if False:
    print('Data Science')
print('bye for now')
```

```
bye for now
```

In [15]:
```
if True:
    print('Data Science')
print('bye for now')
```

```
Data Science
bye for now
```

In [16]:
```
if True:
    print('Data Science')
else:
    print('bye for now')
```

```
Data Science
```

In [17]:
```
if False:
    print('Data Science')
else:
    print('bye for now')
```

```
bye for now
```

In [18]:
```
x = 4
r = x % 2

if r == 0:
    print('Even number')
```

```
Even number
```

In [19]:
```
x = 5
r = x % 2

if r == 0:
    print('Even number')
```

In [20]:
```
x = 5
r = x % 2

if r == 0:
    print('Even number')
else:
    print('odd number')
```

```
odd number
```

In [21]:
```
x = 4
r = x % 2

if r == 0:
    print('Even number')

if r == 1:
```

```
    print('odd number')

if r == 2:
    print('even number')
```

```
Even number
```

In [22]:
```
x = 1

if x == 1:
    print('one')
if x == 2:
    print('Two')
if x == 3:
    print('Three')
if x == 4:
    print('four')
```

```
one
```

In [23]:
```
x = 5
r = x % 2

if r == 0:
    print('Even number')
else:
    print('Odd Number')
```

```
Odd Number
```

In [24]:
```
x = 3
r = x % 2

if r == 0:
    print('Even number')
    if x>5:
        print('greater number')

else:
    print('Odd Number')
```

```
Odd Number
```

In [25]:
```
x = 4
r = x % 2

if r == 0:
    print('Even number')
    if x>5:
        print('greater number')

else:
    print('Odd Number')
```

```
Even number
```

In [26]:
```
x = 4
r = x % 2
```

```
if r == 0:
    print('Even number')

    if x>5:
        print('greater number')


    else:
        print('number is lesser number')

else:
    print('Odd Number')
```

```
Even number
number is lesser number
```

In [27]:
```
# elif it wont check till the block once you find the output it wont go to next line
# you can try with multiple parameter 1, 2 & 3 value in x

x = 4

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')
```

```
four
```

In [28]:
```
# elif it wont check till the block once you find the output it wont go to next line
# you can try with multiple parameter 1, 2 & 3 value in x

x = 7

if(x == 1):
    print('one')
elif(x == 2):
    print('Two')
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')
```

In [29]:
```
# elif it wont check till the block once you find the output it wont go to next line
# you can try with multiple parameter 1, 2 & 3 value in x

x = 7

if(x == 1):
    print('one')

elif(x == 2):
    print('Two')
```

```
elif(x == 3):
    print('Three')
elif(x == 4):
    print('four')

else:
    print('number not found')
```

```
number not found
```

Loops

- A loop in Python is a way to repeat a block of code multiple times without writing it again and again.
- Think of it like telling the computer:
- "Keep doing this task until I say stop" or "Do this task for each item in a list."

There are two main types of loops in Python:

1. **For Loop**: Used for iterating over a sequence (like a list, tuple, or string) or a range of numbers.
2. **While Loop**: Repeats a block of code as long as a specified condition is true.

1. `for loop` – used when you want to repeat something a certain number of times or go through items in a list, tuple, string, etc.

In [32]:
```
for i in range(5):
    print("Hello")
```

```
Hello
Hello
Hello
Hello
Hello
```

2. `while loop` – used when you want to keep repeating as long as a condition is true.

In [ ]:
```
count = 1
while count <= 5:
    print("Hello")
    count += 1
```

```
Hello
Hello
Hello
Hello
Hello
```

1 - `for loop`

- Used when you **already know** how many times you want to repeat.
- Often used to go through a list, string, or a fixed range of numbers.

```
Example:
for i in range(5):   # repeat 5 times
    print("Hello")
```

2 - `while loop`

- Used when you **don't know** in advance how many times to repeat.
- It keeps running as long as a condition is true.

```
Example:
count = 1
while count <= 5:   # repeat until condition is false
    print("Hello")
    count += 1
```

Main Difference in Simple Words

- `for loop` → repeat a fixed number of times (or through each item in something).
- `while loop` → repeat until a condition becomes false.

Analogy:

- `for loop` → Like setting an alarm for 5 times – you know exactly how many rings.
- `while loop` → Like charging your phone until battery is full – you don't know the exact time, it depends on the condition.

In [34]:
```
i = 1

while i<=5:    # condition
    print('data science')
    i = i + 1  # increment
```

```
data science
data science
data science
data science
data science
```

In [35]:
```
i = 1

while i<=5:
    print('data science') # when we mention end then new line will not create
    j = 1
    while j<=4:
        print('technology')
        j = j + 1

    i = i + 1
    print()

    # the output which we got is very lengthy but how to make them one line lets refer to below code
```

```
data science
technology
technology
technology
technology

data science
technology
technology
technology
technology

data science
technology
technology
technology
technology

data science
technology
technology
technology
technology

data science
technology
technology
technology
technology
```

In [36]:
```
i = 1
while i<=5:
    print(' datascience', end = "") # when we mention end then new line will not create
    j = 1
    while j<=4:
        print(' technology', end="")
        j = j + 1

    i = i + 1
    print()
```

```
datascience technology technology technology technology
datascience technology technology technology technology
datascience technology technology technology technology
datascience technology technology technology technology
datascience technology technology technology technology
```

In [ ]:
```

```