## → Python Try Except (Error Handling)

In programming, error handling refers to the process of anticipating and resolving errors when they arise.
Error handling allow for flexibility in the code as it executes differents blocks of code based on the presence of errors.

- The try block lets us test a block of code for errors.
- The except block lets us handle the error
- The else block lets us execute code when there is no error
- The finally block lets us execute code, regardless of the result of the try - and except blocks

Example:

```
try:
  print(x)
except:
  print("An exception occurred")
# The 'try' block will generate an exception,
  because 'x' is not defined
  => Output: An exception occured
```

Since the try block raises an error, the except block will be executed. Without the try block, the program will crash and raise an error.

## Many Exceptions

We can define as many exception block as we w
(we can execute a special block of code for
a special kind of error)

Example:
Print one message if the try block raises
a NameError and another for other errors

```
try:
  print(x)
except NameError:
  print ("Variable x is not defined")
except:
  print ("Something else went wrong")
# Output: Variable x is not defined
```

## Else

We can use the else keyword to define a block
of code to be executed if no errors were raised

Example:
In this example, the try block does not generate any error:

```
try:
  print ("Hello")
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
# Hello
# Nothing went wrong
```

## Finally

The finally block, if specified, will be executed regardless if the try block raises an error or not

Example:

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished").
```

## Raise an exception

We can choose to throw an exception if a condition occurs. To throw (or raise) an exception, use the raise Keyword

Example:

Raise an error and stop the program if x is lower than 0:

```
x = -1
if x < 0:
  raise Exception("Sorry, no numbers below zero")
#output: ...
        Exception: Sorry, no numbers below zero
```

We can define what kind of error to raise, and the text to print to the user.

Example:

Raise a TypeError if x is not an integer

```
x = "Hello"
if not type(x) is int:
    raise TypeError("Only integers are allowed")
```

→ Python Libraries

1) NumPy

Numpy is a Python library used for working with arrays

It also has functions for working in domain of linear algebra, and matrices.