ERRORS AND EXCEPTION HANDLING

- Mistakes in the code that Python doesn't like and will result in the abrupt termination of the program.
- Two main types of errors:

  1.Syntax errors

  2.Exceptions

SYNTAX ERRORS

- Happen due to incorrect syntax of our code.
- Syntax errors occur at compile time
- They also are known as parsing errors
- The most common kind of complaint you get
- Easy to fix

EXCEPTIONS

- Represents an error
- If the normal flow of the program gets disrupted in spite of being syntactically correct, Python raises an exception
- If not handled properly, the program will terminate.
- Always occurs at runtime
- Various in-built exceptions (ZeroDivisionError, NameError and TypeError)
- Self-defined exceptions can be created easily

RAISING EXCEPTIONS

- An exception instance can be raised with the raise statement
- When an exception is raised, no further statements in the current block of code are executed unless the exception is handled.

```
# custom input
num = int(input())
# raise exception if input is negative
if num < 0:
   raise Exception('{} is negative, please enter a positive number'.format(num))
# print input number if it is not negative
```

```
print('Your number is accepted!')
```

HOW DOES PYTHON HANDLE EXCEPTIONS

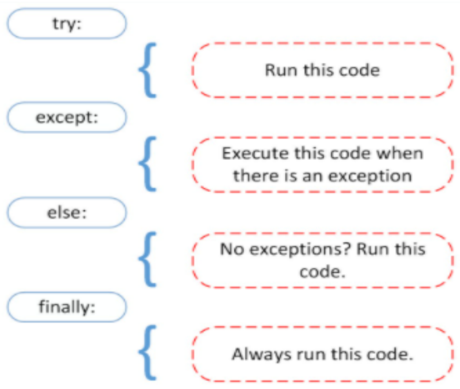
- Try and except blocks
- Any code that we think might throw an error is placed inside the try block
- If an exception is raised, control flow leaves this block immediately and goes to the except block which handles the corresponding exception.

```
try:
        Operational/Suspicious Code
except SomeException:
        Code to handle the exception
```

```
try:
        div = num1/num2
        message = "Quotient is" + " " + str(div)
        print(message)
except ZeroDivisionError:
        message = "Cannot divide by zero"
        print(message)
```

ELSE AND FINALLY CLAUSES

- The code inside **else clause** runs only when the exception doesn't occur
- The code inside **finally clause** always executes after the other blocks, even if there was an uncaught exception or a return statement in one of the other blocks. This block is optional.


try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.
finally:
Always run this code.

Built-in python exceptions:

- **Exception:** The base class for all kinds of exceptions. All kind of exceptions are derived from this class
- **ArithmeticError:** Base class for the exception raised for any arithmetic errors.
- **EOFError:** Raised when `input()` function read End-of-File without reading any data.
- **ZeroDivisionError:** Raise when the second argument of a division or modulo operation is zero
- **AssertionError:** Raised when an `assert` statement fails
- **FloatingPointError:** Raised when a floating-point operation fails
- **KeyError:** Raised when a mapping (dictionary) key is not found in the set of existing keys
- **KeyboardInterrupt:** Raised when the user hits the interrupt key (normally Control-C or Delete). During execution, a check for interrupts is made regularly.