Exceptions

$\downarrow$

unexpected error that occurs during the execution of the program

Python will normally stop & generate the below error message

SyntaxError: Occurs when the Python interpreter encounters an error in the syntax of the code, preventing it from executing. This can include issues like incorrect indentation, missing parentheses, or invalid keywords.
IndentationError: A specific type of SyntaxError that occurs when the indentation of code blocks is not consistent. Python uses indentation to define blocks of code, so incorrect or inconsistent indentation can lead to errors.
TypeError: Occurs when an operation or function is applied to an object of inappropriate type. For example, performing arithmetic on non-numeric data types or passing arguments of the wrong type to a function.
NameError: Occurs when a variable or function name used in the code is not defined or cannot be found in the current scope.
ValueError: Occurs when a function receives an argument of the right type but inappropriate value. For example, passing an invalid argument to a function that expects a certain range of values.
IndexError: Occurs when trying to access an element from a list, tuple, or string using an index that is out of its range. It indicates that the index provided does not exist in the sequence.
KeyError: Occurs when trying to access a dictionary with a key that does not exist in the dictionary. It indicates that the specified key is not found in the dictionary.
AttributeError: Occurs when trying to access an attribute or method that does not exist for a given object. It indicates that the object does not have the specified attribute or method.
ZeroDivisionError: Occurs when attempting to divide a number by zero, which is mathematically undefined. It indicates that the denominator in a division operation is zero.

Exception Handling

↳ so that python program should not stop ideally

- ↳ try ✓ $\longrightarrow$ critical code
- ↳ except $\longrightarrow$ backup plan/strategies
- ↳ else $\longrightarrow$ code to execute if no exceptions were raised
- ↳ finally

raise $\longrightarrow$ used to generate an exception intentionally

Modules → .py Code file

↳ functions that are already defined inside modules

- built-in modules
  - ↳ Numpy, Pandas
- user-defined modules

The main purpose of a module is to organize and reuse code across multiple programs or parts of a program. By modularizing code, you can break down complex programs into smaller, more manageable parts that can be shared and maintained more easily.