Handling exceptions

The try except statement can handle exceptions.

Exceptions may happen when you run a program.

Exceptions are errors that happen during execution of the program. Python won't tell you about errors like syntax errors (grammar faults), instead it will abruptly stop.

An abrupt exit is bad for both the end user and developer.

Instead of an emergency halt, you can use a try except statement to properly deal with the problem. An emergency halt will happen if you do not properly handle exceptions.

Python has built-in exceptions which can output an error. If an error occurs while running the program, it's called an exception.
If an exception occurs, the type of exception is shown. Exceptions needs to be dealt with or the program will crash. To handle exceptions, the try-catch block is used

- **try**: the code with the exception(s) to catch. If an exception is raised, it jumps straight into the except block.
- **except**: this code is only executed *if an exception occured* in the try block. The except block is required with a try block, even if it contains only the pass statement.

It may be combined with the **else** and **finally** keywords.

- **else**: Code in the else block is only executed if no exceptions were raised in the try block.
- **finally**: The code in the finally block is always executed, regardless of if a an exception was raised or not.

```
a=10

print(a/0)
```

In the above example we catch the specific exception ZeroDivisionError. You can