HANDLING EXCEPTIONS

We can handle exceptions using the `try..except` statement. We basically put our usual statements within the try-block and put all our error handlers in the except-block.

Example (save as exceptions_handle.py):

```
try:
    text = raw_input('Enter something --> ')
except EOFError:
    print 'Why did you do an EOF on me?'
except KeyboardInterrupt:
    print 'You cancelled the operation.'
else:
    print 'You entered {}'.format(text)
```

Output:

```
# Press ctrl + d
$ python exceptions_handle.py
```

```
Enter something --> Why did you do an EOF on me?


# Press ctrl + c
$ python exceptions_handle.py
Enter something --> ^CYou cancelled the operation.


$ python exceptions_handle.py
Enter something --> No exceptions
You entered No exceptions
```

How It Works

We put all the statements that might raise exceptions/errors inside the `try` block and then put handlers for the appropriate errors/exceptions in the `except` clause/block. The `except` clause can handle a single specified error or exception, or a parenthesized list of errors/exceptions. If no names of errors or exceptions are supplied, it will handle *all* errors and exceptions.

Note that there has to be at least one `except` clause associated with every `try` clause. Otherwise, what's the point of having a try block?

If any error or exception is not handled, then the default Python handler is called which just stops the execution of the program and prints an error message. We have already seen this in action above.

You can also have an `else` clause associated with a `try..except` block. The `else` clause is executed if no exception occurs.

Source: http://www.swaroopch.com/notes/python/