Error Handling

- If a function encounters an error (error reading a file, network error, given values are inappropriate, etc.) how does it communicate to its caller that something went wrong?
  - Return a special value indicating an error
    - Returning None is common in python
    - Doesn't allow for details about the error to be returned

- Raise an exception
  - Information about the error can be communicated through the type of the exception, and an error message and other data can be contained with the exception object

Exceptions in Python

- Many built-in exceptions
  - ValueError - a function argument is the right type, but an inappropriate value

    int('42') returns 42

    int('hello') raises a ValueError
  - OSError - base class for all exceptions related to system errors

    FileNotFoundError, PermissionError

- All exceptions inherit from Exception

- try / except blocks
  - If an exception is raised within a try block and there is an except block that can handle the exception type, the except block is immediately executed
  - If an exception is not handled by an except block, the executing function immediately exits and the exception is propagated to the caller

- If an exception propagates through the entire call stack, the program crashes and a stack trace is printed

- An else block can be provided which will be executed if no except block was executed

- A finally block is executed regardless of whether any except block was executed

- Custom exceptions

  - Extend an existing exception class to create a custom exception

  - Libraries typically define their own exception types

    configparser.ParsingError

  - Larger programs can benefit from custom exceptions too