fileio_in_python.md

Reading and writing files in Python Reference Card

- There are no extra modules necessary to read or write to a file in Python (so no need for any `import`).
- To access a disk file in Python you first need to open it with the `open()` function:

  ```
  file_object = open(filename, mode)
  ```

  here
  - `file_object` is the object returned by `open()` that allows the file to be accessed.
  - `filename` is a string giving the name of the file (for example `sequence_abc.txt` or `/homes/fred/.config_abc`)
  - `mode` is a string indicating what you want to do with the file:
    - `'r'` means open as text file for read (this is the default).
    - `'w'` means open as text file for writing to.
    - `'a'` means append (write after the end) if the text file exists.
    - `'rb'` means open as binary file for read.
    - `'wb'` means open as binary file for writing to.
- To read from a file you first have to opened a file object (see previous). Normally it is best to read the complete contents of a file into a string (unless the file is very large), for example to read from `input.txt`:

  ```
  in_file = open('input.txt')
  contents = in_file.read()
  in_file.close()
  ```

  - The complete contents of the file will be in string `contents`. If you want to process a file line by line then use the string function splitlines() that returns a list of lines in string, removing the line breaks.

    ```
    in_file = open('input.txt')
    contents = in_file.read()
    lines = contents.splitlines()
    in_file.close()
    print(lines)
    ```

  - It is important to close a file after you have finished accessing it by calling the file object's `.close()` method
- To write to a file first open it as a file object with a mode specifying write. Then call the file object's `.write()` method to write strings to it. Do not forget to close the file.

  ```
  out_file = open('output_new.txt', 'w')
  out_file.write('top line\n')
  out_file.write('2nd line\n')
  out_file.write('3rd line\nfinal line\n')
  out_file.close()
  ```

  - This will write a text file `output_new.txt` with 4 lines. Note the use of the escape character `\n` for the new lines.
- Files can be automatically closed by using the the `with` construction. For example to read a file into a list `lines`:

  ```
  with open('input.txt') as fin:
      lines = fin.read().splitlines()
  print(lines)
  ```

  - the file `input.txt` is closed as soon as the block of code under the with completes
  - this produces neater code.
- If there is an error opening a file because it does not exist this produces a `FileNotFoundError` Exception. This can be caught in the normal way for instance:

  ```
  import sys
  try:
      with open('input.txt') as fin:
         lines = fin.read().splitlines()
  except FileNotFoundError as error_mess:
      print('ERROR:', error_mess)
      sys.exit(1)
  print('read lines:', lines)
  ```