

# Assignment 7 – File I/O

## File I/O

Python supports two general file types, text files and binary files.

### Text Files

#### *Open a file for Reading*

1. Code the following:
  - a. Note: This will throw a file not found error if the file does not exist

#### Screen Capture #1 (1 point)

```
1  #!/usr/bin/env python3
2
3  # Open a file - read mode
4  file = open("test.txt", "r")
5
```

#### *Open a file for Writing - Overwrite*

2. Code the following:
  - a. Note: If the file exists, it will overwrite all existing data

#### Screen Capture #2 (1 point)

```
1  #!/usr/bin/env python3
2
3  # Open a file - write mode (overwrite)
4  file = open("test.txt", "w")
5
```

#### *Open a file for Writing - Append*

3. Code the following:
  - a. Note: If the file exists, it will append the data to the existing data

```
1  #!/usr/bin/env python3
2
3  # Open a file - write mode (append)
4  file = open("test.txt", "a")
5
```

*Writing to a file*

4. Add a line to write to the file.

```
1  #!/usr/bin/env python3
2
3  # Open a file - write mode (append)
4  file = open("test.txt", "a")
5
6  # Writing to the file
7  file.write("Hello World")
8
```

*Closing a file*

5. Update the code to close the file.

Screen Capture #3 (1 point)

```
1  #!/usr/bin/env python3
2
3  # Open a file - write mode (append)
4  file = open("test.txt", "a")
5
6  # Writing to the file
7  file.write("Hello World")
8
9  # Close the file
10 file.close()
11
```

## Working with Text Files

Now that we've covered the basics, let's put it to use.

### Writing to a Text File

6. Create a new python file.
7. Code the following:
  - a. Note: You want to save the file to a location on the computer to be able to easily find it, here we will save to the user's desktop. If you have any trouble with this, just remove everything but the "hello.txt" from the file location and the file will be in the same location as the python code file.

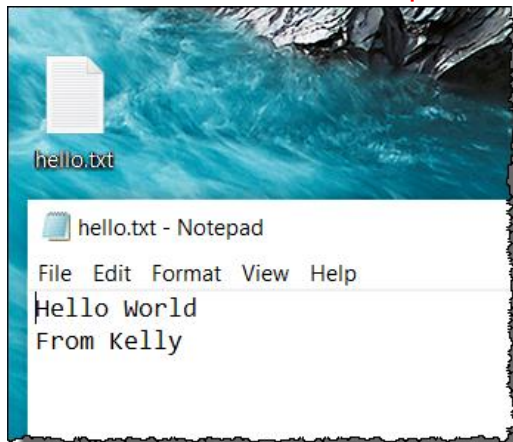
Example: `file = open("hello.txt", "w")`

### Windows Users - Screen Capture #4 (1 point)

```
1  #! /usr/bin/env python3
2  import os
3
4  # open a file in write mode (overwrite)
5  file = open(os.path.expandvars("c:/Users/$USERNAME/Desktop/hello.txt"), "w")
6
7  # write to the file
8  file.write("Hello World\n")
9  file.write("From Kelly") # use your name here
10
11 # close the file
12 file.close()
13
```

8. Run the program...

### Windows Users - Screen Capture #5 (1 point)

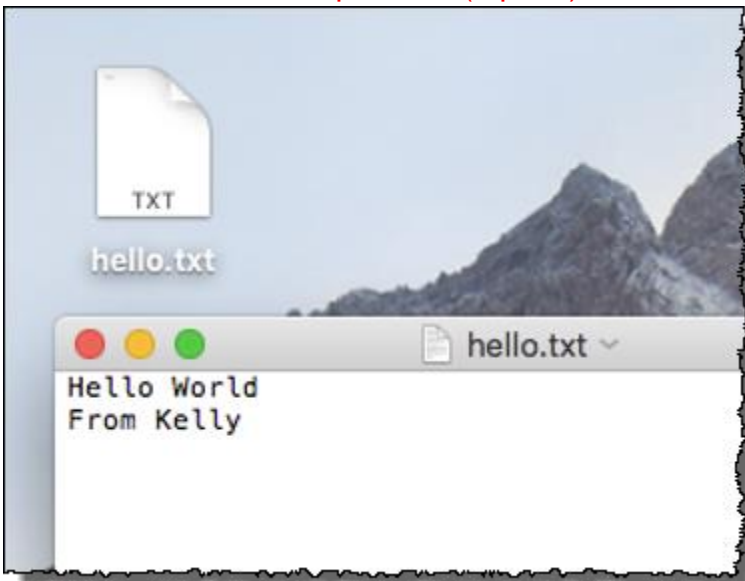


Mac Users - Screen Capture #4 (1 point)

```
1  #!/use/bin/env python3
2
3  import os
4
5  # open a file in write mode (overwrite)
6  file = open(os.path.expanduser("~/Desktop/hello.txt"), "w")
7
8  # write to the file
9  file.write("Hello World\n")
10 file.write("From Kelly") # use your name here
11
12 # close the file
13 file.close()
14
```

9. Run the program...

Mac Users - Screen Capture #5 (1 point)



*Reading from a Text File*

10. Code these statements which read the entire file and print out in various ways:

Note: When using the “with”, we will not need to explicitly close the file, it will automatically be closed at the end of the with block.

```

10  # Close the file
11  file.close()
12
13  # Read and print the entire file
14  print("This will read and the print the entire file")
15  with open("hello.txt") as file:
16      text = file.read()
17      print(text)
18
19  print()
20
21  # Read the entire file and print 1 line at a time.
22  print("This will read entire file "
23        "and then print 1 line at a time")
24  with open("hello.txt") as file:
25      for line in file:
26          print(line, end="")
27

```

11. Code these statements which read the file as a list and then one line at a time.

```

26      print(line, end="")
27
28  print("\n")
29
30  # Read the entire file as a list
31  print("This will read file as a list "
32        "and then print 1 list item at a time")
33  with open("hello.txt") as file:
34      listItems = file.readlines()
35      for item in listItems:
36          print(item, end="")
37
38  print("\n")
39
40  # Read and print the file 1 line at a time
41  print("This will read and print the file "
42        "1 line at a time")
43  with open("hello.txt") as file:
44      line = file.readline()
45      print(line, end="")
46      line = file.readline()
47      print(line)
48

```

12. Run the program...

Screen Capture #6 (2 points)

```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe
This will read and the print the entire file
Hello World
From Kelly

This will read entire file and then print 1 line at a time
Hello World
From Kelly

This will read file as a list and then print 1 list item at a time
Hello World
From Kelly

This will read and print the file 1 line at a time
Hello World
From Kelly

Process finished with exit code 0
```

## Working with Lists in a Text File

Because it's common to store data as a list...

*Writing a List to a File and then Reading the data back into a list*

13. Code the following and run the program:

### Screen Capture #7 (1 point)



```
1  #!/usr/bin/env python3
2
3  OS = ["Mac OS", "Windows", "Android", "Linux"]
4
5  # open the file in overwrite mode
6  with open("os.txt", "w") as file:
7      # write 1 element at a time
8      for item in OS:
9          file.write(item + "\n")
10
11  new_OS = []
12  # open the file in read (default) mode
13  with open("os.txt") as file:
14      # read 1 element at a time
15      for line in file:
16          line = line.replace("\n", "")
17          new_OS.append(line)
18
19  print(new_OS)
```

Run: lists x

C:\Users\Saddleback\AppData\Local\Programs\Python\Python38-32\python.exe

['Mac OS', 'Windows', 'Android', 'Linux']

Process finished with exit code 0

## The Movie List Program – Using a Text File

For this version of the application, we'll store the list of movie titles only into a text file.

The functions we'll need to add/modify will include:

- Read – We'll need to add a function to read the movie list from the movie text file.
- Write – We'll need to add a function to write the movie list to the movie text file.
- Main – We'll need get the movie list from the read function instead of the list we created.
- List – We're going to update the code to display the movie list to be a bit cleaner.
- Add – We'll need to update the movie list when a movie is added to the list
- Delete – We'll need to update the movie list when a movie is deleted from the list.

Make a copy of **movie-1.py** file and rename to **movie-3.py**.

Download the **movies.txt** from canvas.

### Code Validation (+5 Points)

#### Create a global for the file name

This will make it easier to track the file name and if the file name changes, you will only need to update it in one place.

```
1  #!/usr/bin/env python3
2  FILENAME = "movies.txt"
3
```

#### Create a read function

This function will be used to read the text file and return a list of movies. Notice that we strip off the carriage return character at the end of each line.

```
def read_movies():
    movies = []
    with open(FILENAME) as file:
        for line in file:
            line = line.replace("\n", "")
            movies.append(line)
    return movies
```



*Create a write function*


This function will take in a list of movies in the form of a parameter and write that list to file. Since we want each movie entry to be on its own line in the text file, we need to append a carriage return character to the end of each movie entry.

```
def write_movies(movies):
    with open(FILENAME, "w") as file:
        for movie in movies:
            file.write(movie + "\n")
```

*Get the movie list using the read function*


We need to update from the list we created in the previous app to now use the read function to pull the list from the text file.

```
def main():
    display_menu()
    movies = read_movies()
    while True:
        command = input("Command: ")
        if command == "list":
            list_movies(movies)
```


*Get the list*

We need to update from the list we created in the previous app to now use the read function to pull the list from the text file.

```
def add(movie_list):
    movie = input("Movie: ")
    movie_list.append(movie)
    write_movies(movie_list)
    print(movie + " was added.\n")
```



### *Display the list*

While this will produce no visible change to the output, this code will be a bit cleaner.

```
def list_movies(movies):  
    for i in range(len(movies)):  
        movie = movies[i]  
        print(str(i+1) + ". " + movie)  
    print()
```

### *Add movie*

When a movie is added to the list, we want to update the text file to reflect the change.

```
def add_movie(movies):  
    movie = input("Movie: ")  
    movies.append(movie)  
    write_movies(movies)  
    print(movie + " was added.\n")
```

### *Delete movie*

When a movie is delete from the list, we want to update the text file to reflect the change

```
def delete_movie(movies):  
    number = int(input("Number: "))  
    if number < 1 or number > len(movies):  
        print("Invalid movie number.\n")  
    else:  
        movie = movies.pop(number - 1)  
        write_movies(movies)  
        print(movie + " was deleted.\n")
```

## CSV Files

CSV (comma-separated values) files are text files that have rows of information, and each row is typically made up of fields that are separated by commas. To make it easier to work with CSV files in python, a csv module is provided that can be imported into your projects.

### Writing to a CSV File

14. Code the following:

```

1  #! /usr/bin/env python3
2
3  import csv
4
5  OS = [
6      ["Mac OS", "10.6"],
7      ["Windows", "10"],
8      ["Android", "7"]]
9
10 with open("os.txt", "w", newline="") as file:
11     writer = csv.writer(file)
12     writer.writerows(OS)

```

### Reading from a CSV File

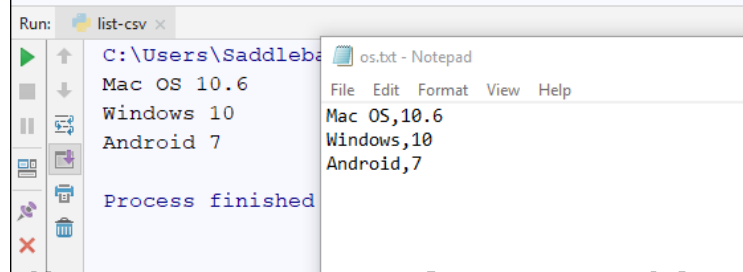
15. Code the following to add the reading of the csv file:

#### Screen Capture #8 (1 points)

```

1  #! /usr/bin/env python3
2
3  import csv
4
5  OS = [
6      ["Mac OS", "10.6"],
7      ["Windows", "10"],
8      ["Android", "7"]]
9
10 with open("os.txt", "w", newline="") as file:
11     writer = csv.writer(file)
12     writer.writerows(OS)
13
14 with open("os.txt", newline="") as file:
15     reader = csv.reader(file)
16     for row in reader:
17         print(row[0], row[1])

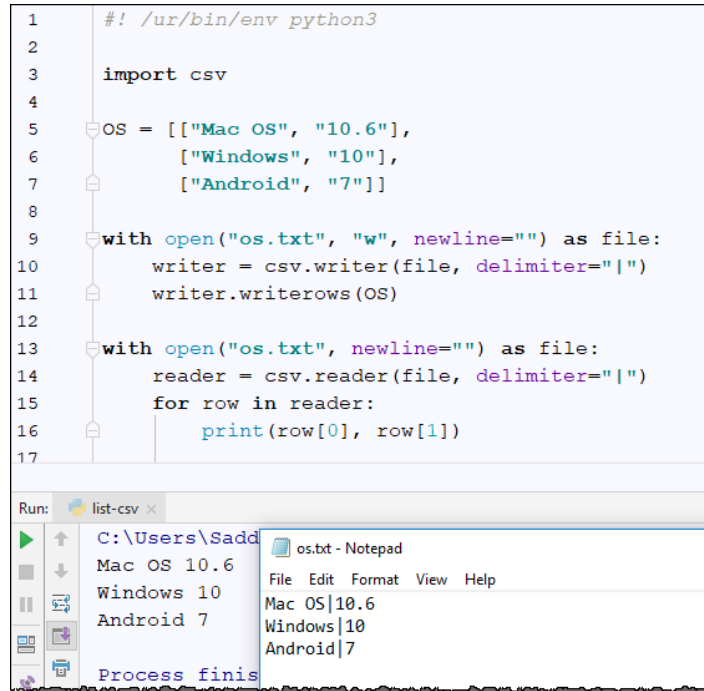
```



*Changing the Delimiter*

By default, the comma is the delimiter (note image #8) but we can change it.

16. Code the following:

**Screen Capture #9 (1 points)**

```
1  #! /usr/bin/env python3
2
3  import csv
4
5  OS = [
6      ["Mac OS", "10.6"],
7      ["Windows", "10"],
8      ["Android", "7"]]
9
10 with open("os.txt", "w", newline="") as file:
11     writer = csv.writer(file, delimiter="|")
12     writer.writerows(OS)
13
14 with open("os.txt", newline="") as file:
15     reader = csv.reader(file, delimiter="|")
16     for row in reader:
17         print(row[0], row[1])
```

Run: list-csv x

C:\Users\Sadd

Mac OS 10.6

Windows 10

Android 7

Process finis

os.txt - Notepad

File Edit Format View Help

Mac OS|10.6

Windows|10

Android|7

## The Movie List Program - CSV

For this version of the application, we'll store the list of the movies and year in a CSV file. Nothing from a UI perspective should change, we just need to add the ability to retrieve and write the list to a file.

Make a copy the **movie-2.py** file and rename to **movie-4.py**.

Hints:

- Create the movies.csv file
  - Copy and rename the movies.txt to movies.csv.
    - Update as follows:
 


1	Star Wars, 1977
---	-----------------
    - Or download **movies.csv** from canvas.
  - Or download **movies.csv** from canvas.

### Code Validation (+4 Points)

#### Import the csv module

This will allow you to use functions for reading and writing to csv files.

```
#!/usr/bin/env python3
import csv
FILENAME = "movies.csv"
```



#### Create a read function

This function will be used to read the csv file and return a list of movies.

```
def read_movies():
    movie_list = []
    with open(FILENAME, newline="") as file:
        reader = csv.reader(file)
        for row in reader:
            movie_list.append(row)
    return movie_list
```

*Create a write function*

This function will take in a list of movies in the form of a parameter and write that list to file.

```
def write_movies(movie_list):
    with open(FILENAME, "w", newline="") as file:
        writer = csv.writer(file)
        writer.writerows(movie_list)
```

*Get the movie list using the read function*

We need to update from the list we created in the previous app to now use the read function to pull the list from the csv file.

```
def main():
    # Create and Initialize the movie list
    movie_list = read_movies() ←
    display_menu()

    while True:
```

*Update the movie list in both the add and delete functions*

```
def add(movie_list):
    name = input("Name: ")
    year = input("Year: ")
    movie = (name, year)
    movie_list.append(movie)
    write_movies(movie_list) ←
    print(movie[0] + " was added.\n")
```

```
def delete(movie_list):
    number = int(input("Number: "))
    if number < 1 or number > len(movie_list):
        print("Invalid movie number.\n")
    else:
        movie = movie_list.pop(number - 1)
        write_movies(movie_list) ←
        print(movie[0] + " was deleted.\n")
```

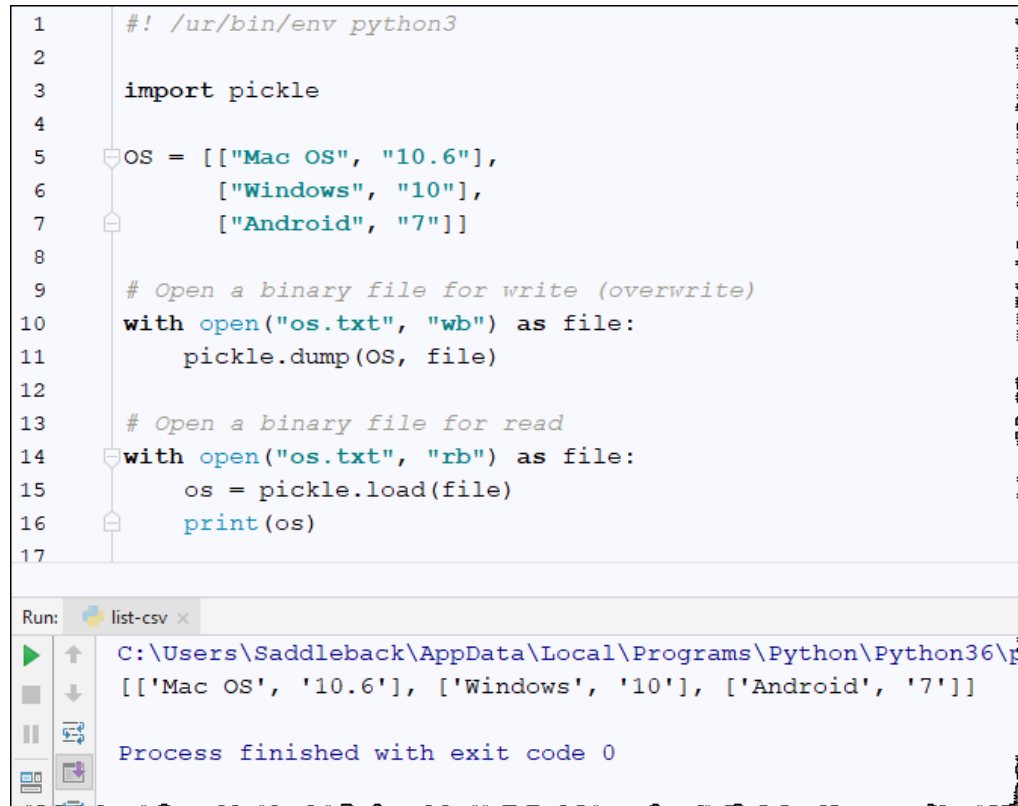
## Binary Files

Like the python csv module, there is a module to assist with working with binary files called pickle.

### *Writing to and Reading from a Binary File*

17. Code the following:

#### Screen Capture #10 (1 points)



```
1  #! /usr/bin/env python3
2
3  import pickle
4
5  OS = [{"Mac OS", "10.6"},
6        ["Windows", "10"],
7        ["Android", "7"]]
8
9  # Open a binary file for write (overwrite)
10 with open("os.txt", "wb") as file:
11     pickle.dump(OS, file)
12
13 # Open a binary file for read
14 with open("os.txt", "rb") as file:
15     os = pickle.load(file)
16     print(os)
17
```

Run: list-csv x

C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe

[['Mac OS', '10.6'], ['Windows', '10'], ['Android', '7']]

Process finished with exit code 0

For more information on pickle and what it does:

<https://pythontips.com/2013/08/02/what-is-pickle-in-python/>

## Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

### Extra Credit #1 – Wizard Inventory (+1 Extra Credit)

Create a program that keeps track of the item that a wizard can carry.

```
The Wizard Inventory program

COMMAND MENU
walk - Walk down the path
show - Show all items
drop - Drop an item
exit - Exit program

Command: walk
While walking down a path, you see a scroll of uncursing.
Do you want to grab it? (y/n): y
You picked up a scroll of uncursing.

Command: walk
While walking down a path, you see an unknown potion.
Do you want to grab it? (y/n): y
You can't carry any more items. Drop something first.

Command: show
1. a wooden staff
2. a scroll of invisibility
3. a crossbow
4. a scroll of uncursing

Command: drop
Number: 3
You dropped a crossbow.

Command: exit
Bye!
```

#### Specifications:

- You can download the wizard\_all\_items.txt that contains a list of all the items that a wizard can carry.
- When the user selects the walk command, the program should read the items from the file, randomly pick one, and give the user the option to grab it.
- Your program should create another file that stores the items that a wizard is carrying. Make sure to update this file every time the user grabs or drops an item.
- The wizard can only carry four items at a time.
- For the drop command, display an error message if the user enters an invalid number for the item.



**Extra Credit #2 – Monthly Sales (+1 Extra Credit)**

Create a program that reads the sales for 12 months from a file and calculates the total yearly sales as well as the average monthly sales. In addition, this program should let the user edit the sales for any month.

```
Monthly Sales program

COMMAND MENU
monthly - View monthly sales
yearly  - View yearly summary
edit    - Edit sales for a month
exit    - Exit program

Command: monthly
Jan - 14317
Feb - 3903
Mar - 1073
Apr - 3463
May - 2429
Jun - 4324
Jul - 9762
Aug - 15578
Sep - 2437
Oct - 6735
Nov - 88
Dec - 2497

Command: yearly
Yearly total: 66606
Monthly average: 5550.5

Command: edit
Three-letter Month: Nov
Sales Amount: 8854
Sales amount for Nov was modified.

Command: exit
Bye!
```

*Specifications:*

- You can download the CSV file named `monthly_sales.csv` from Canvas that contains the month and sales data shown above.
- For the edit command, display an error message if the user doesn't enter a valid three-letter abbreviation for the month.
- When the user edits the sales amount, the data should be saved to the CSV file immediately. That way, no data is lost, even if the program crashes later.
- Round the results of the monthly averages to a maximum of 2 decimal digits.