Formatted Files

Formatted Files

CSV Files, Serialization, and Sequential vs. Random Access

Review of Writing Files

Review of Writing Files

- `myFile = open("filename.txt", "w")` – “w” = write a new file

Review of Writing Files

- `myFile = open("filename.txt", "w")` – “w” = write a new file
- `myFile.write("Any string you’d like to write")`

Review of Writing Files

- `myFile = open("filename.txt", "w")` – “w” = write a new file
- `myFile.write("Any string you’d like to write")`
- `myFile.write(str(myVar))`

Review of Writing Files

- `myFile = open("filename.txt", "w")` – “w” = write a new file
- `myFile.write("Any string you'd like to write")`
- `myFile.write(str(myVar))`
- `myFile.close()`

Review of Writing Files

- `myFile = open("filename.txt", "w")` – “w” = write a new file
- `myFile.write("Any string you'd like to write")`
- `myFile.write(str(myVar))`
- `myFile.close()`
- `myFile = open("filename.txt", "a")` – “a” = append to existing file

Review of Reading Files

Review of Reading Files

- `myFile = open("filename.txt", "r")` – “r” = read a file

Review of Reading Files

- `myFile = open("filename.txt", "r")` – “r” = read a file
- `myVar1 = myFile.readline()`

Review of Reading Files

- `myFile = open("filename.txt", "r")` – “r” = read a file
- `myVar1 = myFile.readline()`
- `myVar2 = myFile.readline()`

Review of Reading Files

- `myFile = open("filename.txt", "r")` – “r” = read a file
- `myVar1 = myFile.readline()`
- `myVar2 = myFile.readline()`
- `myFile.close()`

Review of Reading in Loop

Review of Reading in Loop

- `myList = []`

Review of Reading in Loop

- `myList = []`
- `myFile = open("filename.txt", "r")`

Review of Reading in Loop

- `myList = []`
- `myFile = open("filename.txt", "r")`

-
  ```
  for line in myFile:
      myList.append(line)
  ```

Review of Reading in Loop

- `myList = []`
- `myFile = open("filename.txt", "r")`

-
  ```
  for line in myFile:
      myList.append(line)
  ```
- `myFile.close()`

Review of Safe Reading

Review of Safe Reading

If an error occurs while reading, then your files won't be closed

Review of Safe Reading

If an error occurs while reading, then your files won't be closed

Use a `with` clause to have them automatically close:

Review of Safe Reading

If an error occurs while reading, then your files won't be closed

Use a `with` clause to have them automatically close:

```
myList = []
with open("filename.txt", "r") as myFile:
    for line in myFile:
        myList.append(line)
```

Review of Safe Reading

If an error occurs while reading, then your files won't be closed

Use a `with` clause to have them automatically close:

```
myList = []
with open("filename.txt", "r") as myFile:
    for line in myFile:
        myList.append(line)
```

https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

Formatted Writing

Formatted Writing

```
myList = [["a", "b", "c"], ["d", "e", "f"]]
```

Formatted Writing

```
myList = [["a", "b", "c"], ["d", "e", "f"]]
myFile = open("filename.txt", "w")
```

Formatted Writing

```
myList = [["a", "b", "c"], ["d", "e", "f"]]
myFile = open("filename.txt", "w")
for row in myList:
    myFile.write(f"row[0],row[1],row[2]\n")
```

Formatted Writing

```
myList = [["a", "b", "c"], ["d", "e", "f"]]
myFile = open("filename.txt", "w")
for row in myList:
    myFile.write(f"row[0],row[1],row[2]\n")
myFile.close()
```

Formatted Reading

Formatted Reading

```
myList = []
```

Formatted Reading

```
myList = []
myFile = open("filename.txt", "r")
```

Formatted Reading

```
myList = []
myFile = open("filename.txt", "r")
for line in myFile:
```

Formatted Reading

```
myList = []
myFile = open("filename.txt", "r")
for line in myFile:
    cols = line.strip().split(",")
```

Formatted Reading

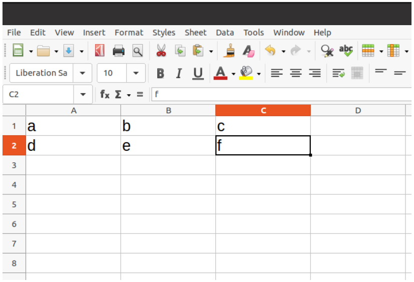
```
myList = []
myFile = open("filename.txt", "r")
for line in myFile:
    cols = line.strip().split(",")
    myList.append([cols[0], cols[1], cols[2]])
```

Formatted Reading

```
myList = []
myFile = open("filename.txt", "r")
for line in myFile:
    cols = line.strip().split(",")
    myList.append([cols[0], cols[1], cols[2]])
myFile.close()
```

CSV Files

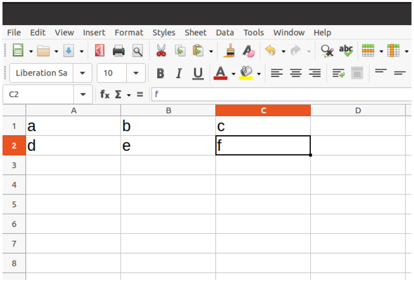
CSV Files


File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sa
10
C2
f

	A	B	C	D
1	a	b	c	
2	d	e	f	
3				
4				
5				
6				
7				
8				

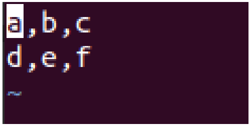
A Spreadsheet

CSV Files


File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sa
10
C2
f

	A	B	C	D
1	a	b	c	
2	d	e	f	
3				
4				
5				
6				
7				
8				

A Spreadsheet



```
a,b,c
d,e,f
~
```

A CSV file

Writing CSV Files

Writing CSV Files

```
import csv
```

Writing CSV Files

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
```

Writing CSV Files

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "w") as myFile:
```

Writing CSV Files

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "w") as myFile:
    csvwriter = csv.writer(myFile)
```

Writing CSV Files

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "w") as myFile:
    csvwriter = csv.writer(myFile)
    for row in myList:
        csvwriter.writerow(row)
```

CSV Reading

CSV Reading

```
import csv
```

CSV Reading

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
```

CSV Reading

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "r") as myFile:
```

CSV Reading

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "r") as myFile:
    csvreader = csv.reader(myFile)
```

CSV Reading

```
import csv
myList = [["a", "b", "c"], ["d", "e", "f"]]
with open("filename.txt", "r") as myFile:
    csvreader = csv.reader(myFile)
    for row in csvreader:
        myList.append(row)
```

https://docs.python.org/3/library/csv.html

Object Serialization

Object Serialization

To save an object, just save all of its attributes in a list

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing*

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing* `import csv`

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing* `import csv`

```
from person import Person
```

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing* `import csv`

```
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
```

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing* `import csv`

```
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.csv", "w") as fp:
```

Object Serialization

To save an object, just save all of its attributes in a list

This is called *serializing* `import csv`

```
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.csv", "w") as fp:
    daveWriter = csv.writer(fp)
    daveWriter.write([dave.firstName, dave.lastName,
                      dave.birthDay, dave.birthMonth,
                      dave.birthYear])
```

The pickle module

The pickle module

Python provides a std lib for serializing objects

The pickle module

Python provides a std lib for serializing objects

Called `pickle`

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.pickle", "wb") as fp:
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.pickle", "wb") as fp:
    pickle.dump(dave, fp, pickle.HIGHEST_PROTOCOL)
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.pickle", "wb") as fp:
    pickle.dump(dave, fp, pickle.HIGHEST_PROTOCOL)

with open("dave.pickle", "rb") as fp:
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.pickle", "wb") as fp:
    pickle.dump(dave, fp, pickle.HIGHEST_PROTOCOL)

with open("dave.pickle", "rb") as fp:
    dave2 = pickle.load(fp)
```

The pickle module

Python provides a std lib for serializing objects

Called `pickle` – a Python object serialized in this way is said to be *pickled*

```
import pickle
from person import Person
dave = Person(name="Dave Davison", DOB="1/1/2001")
with open("dave.pickle", "wb") as fp:
    pickle.dump(dave, fp, pickle.HIGHEST_PROTOCOL)

with open("dave.pickle", "rb") as fp:
    dave2 = pickle.load(fp)
```

https://docs.python.org/3/library/pickle.html

Sequential Access

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
```

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
line1 = myFile.readline()
```

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
line1 = myFile.readline()
line2 = myFile.readline()
```

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
line1 = myFile.readline()
line2 = myFile.readline()
line3 = myFile.readline()
```

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
line1 = myFile.readline()
line2 = myFile.readline()
line3 = myFile.readline()
line4 = myFile.readline()
```

Sequential Access

```
myFile = open("sequentialFile.txt", "r")
line1 = myFile.readline()
line2 = myFile.readline()
line3 = myFile.readline()
line4 = myFile.readline()
myFile.close()
```

Random Access

Random Access

Directly access lines in a file by their line number (out-of-order)

Random Access

Directly access lines in a file by their line number (out-of-order)

In Python, use `linecache` std lib:

Random Access

Directly access lines in a file by their line number (out-of-order)

In Python, use `linecache` std lib:

```
import linecache
```

Random Access

Directly access lines in a file by their line number (out-of-order)

In Python, use `linecache` std lib:

```
import linecache
line4 = linecache.getline("myfile.txt", 4)
```

Random Access

Directly access lines in a file by their line number (out-of-order)

In Python, use `linecache` std lib:

```
import linecache
line4 = linecache.getline("myfile.txt", 4)
line2 = linecache.getline("myfile.txt", 2)
```

Random Access

Directly access lines in a file by their line number (out-of-order)

In Python, use `linecache` std lib:

```
import linecache
line4 = linecache.getline("myfile.txt", 4)
line2 = linecache.getline("myfile.txt", 2)
```

https://docs.python.org/3/library/linecache.html