FILE I/O

Objectives

- Read text files in Python
- Write text files in Python
- Use "with" blocks when reading / writing files
- Describe the different ways to open a file
- Read CSV files in Python
- Write CSV files in Python

Reading Files

- You can read a file with the `open` function
- `open` returns a file object to you
- You can read a file object with the `read` method

File Read Example

story.txt

```
This short story is really short.
```

reading_files.py

```
file = open("story.txt")
file.read()
```

Cursor Movement

- Python reads files by using a cursor
- This is like the cursor you see when you're typing
- After a file is read, the cursor is at the end
- To move the cursor, use the `seek` method
- To read only part of a file, pass a number of characters into `read`, or use `readline`
- To get a list of all lines, use `readlines`

Closing a File

- You can close a file with the close method
- You can check if a file is closed with the closed attribute
- Once closed, a file can't be read again
- Always close files - it frees up system resources!

`with` Blocks

Option 1

```
file = open("story.txt")
file.read()
file.close()

file.closed # True
```

Option 2

```
with open("story.txt") as file:
    file.read()

file.closed # True
```

Writing to Text Files

Writing Files

- You can also use `open` to write to a file
- Need to specify the "w" flag as the second argument

Writing Files Example

```
with open("haiku.txt", "w") as file:
    file.write("Writing files is great\n")
    file.write("Here's another line of text\n")
    file.write("Closing now, goodbye!")
```

IMPORTANT: If you open the file again to write, the original contents will be deleted!

Modes for Opening Files

- r - Read a file (no writing) - this is the default
- w - Write to a file (previous contents removed)
- a - Append to a file (previous contents not removed)
- r+ - Read and write to a file (writing based on cursor)

Truncating Files

file.truncate - removes all text starting from the current cursor position

YOUR TURN

Reading CSV Files

Reading CSV Files

- CSV files are a common file format for tabular data
- We can read CSV files just like other text files
- Python has a built-in CSV module to read/write CSVs more easily

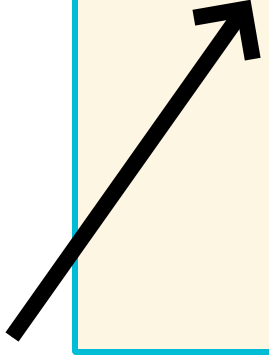
Reading CSV Example

fighters.csv

```
Name,Country,Height (in cm)
Ryu,Japan,175
Ken,USA,175
Chun-Li,China,165
Guile,USA,182
E. Honda,Japan,185
Dhalsim,India,176
Blanka,Brazil,192
Zangief,Russia,214
```

reading_csv.py

```
with open("fighters.csv") as file
    file.read()
```



Don't do this!

CSV Module

- `reader` - lets you iterate over rows of the CSV as lists
- `DictReader` - lets you iterate over rows of the CSV as OrderedDicts
- Keys are determined by the header row
- An OrderedDict is like a dictionary, but it remembers the order in which keys were inserted

CSV Module Examples

reader

```
from csv import reader
with open("fighters.csv") as file:
    csv_reader = reader(file)
    for row in csv_reader:
        # each row is a list!
        print(row)
```

DictReader

```
from csv import DictReader
with open("fighters.csv") as file:
    csv_reader = DictReader(file)
    for row in csv_reader:
        # each row is an OrderedDict
        print(row)
```

Other Delimiters

Readers accept a delimiter kwarg in case your data isn't separated by commas.

```
from csv import reader
with open("example.csv") as file:
    csv_reader = reader(file, delimiter="|")
    for row in csv_reader:
        # each row is a list!
        print(row)
```

Writing to CSV Files

Writing CSV Files

Using lists

- `writer` - creates a writer object for writing to CSV
- `writerow` - method on a writer to write a row to the CSV

Writing CSV Files Example

Using lists

```
from csv import writer
with open("fighters.csv", "w") as file:
    csv_writer = writer(file)
    csv_writer.writerow(["Character", "Move"])
    csv_writer.writerow(["Ryu", "Hadouken"])
```

Writing CSV Files

Using Dictionaries

- `DictWriter` - creates a writer object for writing using dictionaries
- `fieldnames` - kwarg for the DictWriter specifying headers
- `writeheader` - method on a writer to write header row
- `writerow` - method on a writer to write a row based on a dictionary

Writing CSV Files Example

Using Dictionaries

```
from csv import DictWriter
with open("example.csv", "w") as file:
    headers = ["Character", "Move"]
    csv_writer = DictWriter(file, fieldnames=headers)
    csv_writer.writeheader()
    csv_writer.writerow({
        "Character": "Ryu",
        "Move": "Hadouken"
    })
```

Dictionaries or Lists?

- Using lists often means less code
- Using dictionaries is often more explicit

Recap

- You can read and write text files using *open* and *with*
- Files are automatically closed after use when using *with*
- You can set the mode for open to read, write, or append
- The CSV module lets you read / write CSV files
- You can read / write CSVs using lists or dictionaries

YOUR
TURN