CS 49 Section

Week 10 Bonus Slides

Surajit A Bose

Accessing data from files

- Python can read and write plain text files
- Usually these have the filename extension **`.txt`**
- **`.py`**, **`.html`**, **`.cpp`**, or other source code files are also typically plain text
- Sometimes plain text files lack an extension altogether
- Notepad, TextEdit, BBEdit, and other text editors can create plain text files
- Microsoft Word and other word processors can export files as plain text
- Opening a text file in Python can be a security risk
- Open only files from a trusted and verified source (anyone can send a file purporting to be from your boss, your bank, your family member ...)

Accessing data from files

- To open a file for reading or writing, use the **`with open`** statement:

```
with open ('path/to/file.txt', 'mode') as fh:
    # indented block to read or write the file
```

- **`fh`** is whatever variable name you want to use for the file
  - fh is typical, as it's short for **`file handle`**
  - **`infile`** is also common for files to be read
  - **`outfile`** is also common for files to be written
- **`path/to/file`** is a **`str`** specifying the location of the file on the system
  - The path can be relative or absolute
  - It is often a constant or received as input from the user, in which cases you might not need to supply the quotation marks

Accessing data from files

- To open a file for reading or writing, use the **`with open`** statement:

```
with open ('path/to/file.txt', 'mode') as fh:
    # indented block to read or write the file
```

- **`mode`** is a **`str`** specifiying one of three options:
  - **`'r'`** for read. This is the default and can be omitted
  - **`'w'`** for write
    - if the file does not exist, it will be created
    - if the file already exists, it will be overwritten!
  - **`'a'`** for append
    - if the file exists, the new data will be added to the end
    - if the file does not exist, it will be created

Reading files

- Python reads the file one line at a time, each line as a `str`
- It's usually efficient to loop over all the lines and add them to a list
- Each element of the list is one line of the file as a `str`
- Sample code:

```
lines = []
with open ('file_to_read.txt', 'r') as infile:
    for line in infile:
        lines.append(line.strip())
```

- `strip()` removes whitespace from the beginning and end of the lines, such as tabs `'\t'` or newlines `'\n'`

Writing to a file

- Python can write strings out to a file
- Remember that the `'w'` option will overwrite the file if it exists!
- Say you have a list of names: `['Jane', 'John', 'June', 'Joe']`
- The following code will write the names out to the file, one name per line:

```
with open ('file_to_write.txt', 'w') as outfile:
    for name in names:
        outfile.write(f'{name}\n')
        # or outfile.write(name + '\n')
```

- Note that unlike `print()`, you must explicitly supply `'\n'` to `write()`

That's all, folks!