COL1000: Introduction to Programming

File I/O

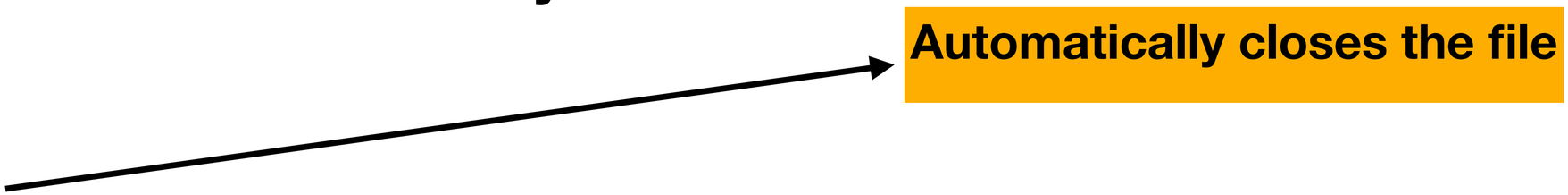
Subodh Sharma | Lec 30 | Oct 28

Announcement

- **(Usually) Monday's — 5 pm to 7 pm; Doubt learning sessions in Bharti 419**
  - **This week, it's today from 5:30 pm onwards!**
-

File Open — with exception handling

- **Opening the file in a safer way:**




```
try:
    with open("data.csv", "w") as f:
        content = f.read()
        print content
    #automatic closing of the file
except FileNotFoundError:
    print("The file was not found!")
```

File Open — Modes

Mode	Symbol	Description
Read	`'r'`	**Reads** from a file. This is the default. Raises an error if the file does not exist.
Write	`'w'`	**Writes** to a file. **Overwrites** the entire file if it exists, or creates a new one if not.
Append	`'a'`	**Appends** to the end of a file. Creates a new file if it does not exist.
Create	`'x'`	**Exclusively creates** a new file. Fails with an error if the file already exists.
Text	`'t'`	Opens in **text mode**. This is the default.
Binary	`'b'`	Opens in **binary mode** (for non-text files like images or executables).
Update	`'+'`	Opens for **updating** (reading and writing). Can be combined with other modes (e.g., `r+`, `w+`).

File Reading — Other options

- **Reading one line:** `f.readline()`
- **Reading all lines as a list:** `f.readlines()`
- **Reading the content iteratively:**

```
try:
    with open("data.csv", "r") as f:
        for line in f:
            print(line.strip())
except FileNotFoundError:
    print("The file was not found!")
```

File Reading— All at once

- **Reading all at one:**

```
try:
    with open("data.csv", "w") as f:
        content = f.read()
        print content
    #automatic closing of the file
except FileNotFoundError:
    print("The file was not found!")
```

File Reading— Other Variations

- **Reading all at one:**

```
try:
    with open("data.csv", "r") as f:
        print(f.readline())
        f.seek(0)
        print(f.readlines())
except FileNotFoundError:
    print("The file was not found!")
```

Reading one line at a time

Resets the file cursor/ptr to the start

Reading in a list all at once from the wherever the file cursor is!

- **Note about `seek(offset)`: sets to 0 when file opened in r/w mode, and to EoF in a mode**

File Reading— Other Variations

- **Reading the lines iteratively, one by one.**

```
try:
    with open("data.csv", "r") as f:
        for line in f:
            print(line.strip())
except FileNotFoundError:
    print("The file was not found!")
```

File Writing - Write mode

- **Write mode — overwrites the contents, if the file existed before!**

```
try:
    with open("data.csv", "w") as f:
        f.write("Replacing the CSV entries to string data\n")
        f.write("3,5,7\n")
except FileNotFoundError:
    print("The file was not found!")
```

File Writing - Append mode

- **Append mode — Preserves previous content, the new content is appended toward the end**

```
try:
    with open("data.csv", "w") as f:
        f.write("Replacing the CSV entries to string data\n")
        f.write("3,5,7\n")
except FileNotFoundError:
    print("The file was not found!")
```

Common File I/O Exceptions

Exception Name	Typical Scenario
`FileNotFoundError`	Opening non-existent file
`PermissionError`	Opening file without required access or permissions
`IsADirectoryError`	Opening a directory as a file
`IOError` / `OSError`	General file I/O (read/write) failures
`EOFError`	Reading past the end of file

- **Special kind of IOError**
- **Disk is full, file is corrupted, hardware error**

FileException - Example

- **Shell Commands:**

```
touch protected_file.txt

chmod u-w protected_file.txt
```

```
file_path = "./protected_file.txt"
try:
    with open(file_path, "w") as f:
        f.write("Attempting to overwrite!")
except PermissionError:
    print(f"Error: Permission denied. Cannot write to '{file_path}'.")
```

Some File IO Exercises

- Write a program that performs the service of the command `wc`
  - **Lines, words, chars and bytes counter**
- Unique word counter
- Read a CSV file and
  - compute a column average
  - Search a target string and replace it with some pattern

Support in PathLib for FileIO

- Lot of convenient methods

File Content Operations:

- `.read_text()`: Reads the content of a file in text mode.
- `.read_bytes()`: Reads the content of a file in binary mode.
- `.write_text()`: Writes string data to a file in text mode.
- `.write_bytes()`: Writes binary data to a file in binary mode.
- `.open()`: Opens the file associated with the path and returns a file object.

Support in PathLib for FileIO

- Lot of convenient methods

File and Directory Creation/Deletion:

- `.mkdir()`: Creates a new directory. `exist_ok=True` prevents an error if the directory already exists.
- `.touch()`: Creates a new empty file.
- `.unlink()`: Deletes a file.
- `.rmdir()`: Deletes an empty directory.
- `.rename()`: Renames a file or directory.
- `.replace()`: Renames a file or directory, overwriting the destination if it exists.

Support in PathLib for FileIO

- Lot of convenient methods

Permissions and Ownership:

- `.chmod()`: Changes the file mode and permissions.
- `.owner()`: Returns the name of the file owner.
- `.group()`: Returns the name of the file's group owner.

Path Construction and Manipulation:

- `Path()`: Creates a `Path` object representing a file or directory path.
- `/` (Operator Overload): Concatenates path components, providing a clean way to build paths.

Python

```python
from pathlib import Path
p = Path("my_directory") / "my_file.txt"
```

- `.parent`: Accesses the parent directory of a path.
- `.name`: Gets the final path component (filename or directory name).
- `.stem`: Gets the final path component without its suffix.
- `.suffix`: Gets the file extension.
- `.suffixes`: Gets a list of all file extensions.
- `.joinpath()`: Joins the path with another path or string.
- `.absolute()`: Returns the absolute path.
- `.resolve()`: Returns the absolute path, resolving any symbolic links and `..` references.
- `.relative_to()`: Returns the relative path to another path.

ToDo — Next class

- Command line arguments
- A case study