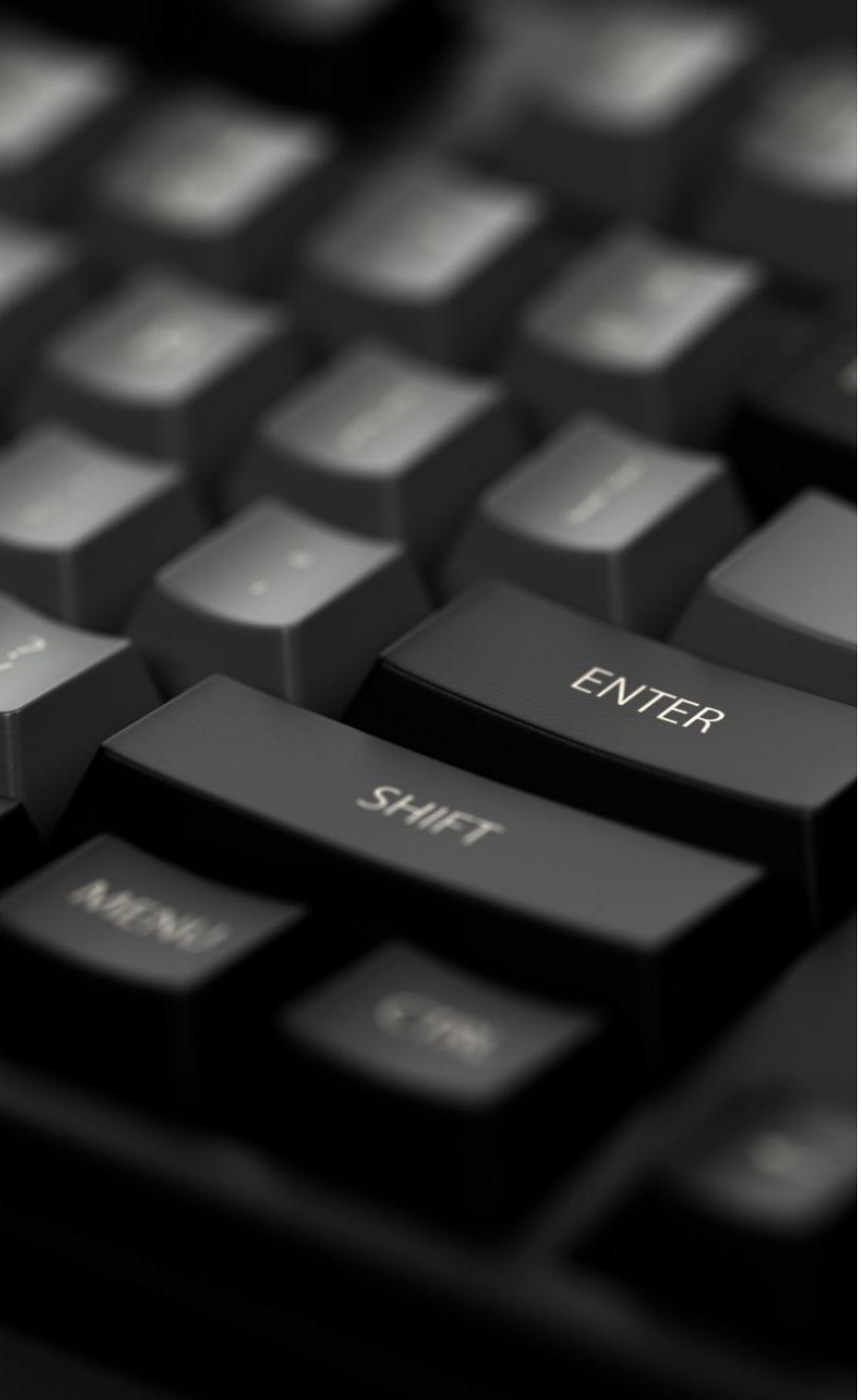
A gamified introduction to Python Programming

Lecture 13
Text files handling and processing

Matthieu DE MARI – Singapore University of Technology and Design

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN



Outline (Chapter 13)

- What is a **file handle**?
- What are **opening modes** for files?
- How to **read** and **write strings to a text file**?
- Practice

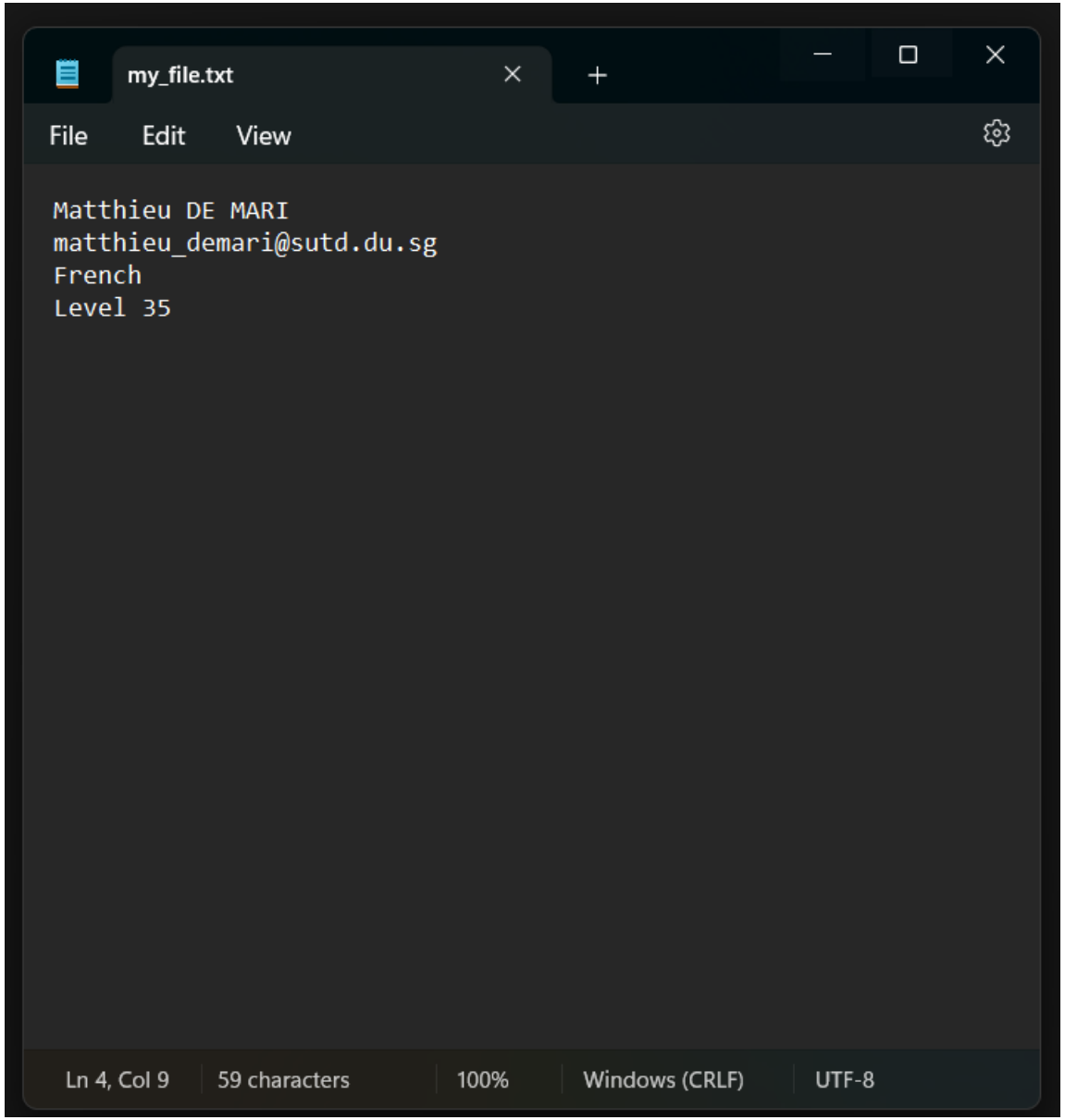
Text files

Definition (text files):

A **text file** is a file that contains **text in plain string format**. It is often saved using the **.txt** file format extension.

It is possible to open and read the content of a text file using a simple notepad/text editor.

Files that cannot be read with a simple text editor, require advanced operations (e.g. docx, pdf, etc.)



Opening a text file in Python

You can open a text file in Python, using the **open() function**.

- It creates a **file handler f**,
- Using the file name specified *(Here "./matrix.txt" means there should be a text file with name matrix.txt in the same location of the Python notebook/file.)*
- Once done manipulating the file, remember to close it using the **f.close()** operation.

```python
# Open file in read ('r') mode
f = open('./matrix.txt', 'r')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix.txt' mode='r' encoding='cp1252'>
```

```python
# Open file in write ('w') mode
# Important: write mode, erases the file!
f = open('./matrix2.txt', 'w')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix2.txt' mode='w' encoding='cp1252'>
```

```python
# Open file in append ('a') mode
# Note: basically, write mode but does not erase the file
f = open('./matrix3.txt', 'a')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix3.txt' mode='a' encoding='cp1252'>
```

Opening a text file in Python

The f.open() method also expects a **file reading/writing mode**:

- **Read ('r'):** Opens the file but only allows Python to read content.
- **Write ('w'):** Opens the file but only allows Python to write content. **Also erases the content of the file!**
- **Append ('a'):** Opens the file but only allows Python to write content. **Does not erase.**

```python
# Open file in read ('r') mode
f = open('./matrix.txt', 'r')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix.txt' mode='r' encoding='cp1252'>
```

```python
# Open file in write ('w') mode
# Important: write mode, erases the file!
f = open('./matrix2.txt', 'w')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix2.txt' mode='w' encoding='cp1252'>
```

```python
# Open file in append ('a') mode
# Note: basically, write mode but does not erase the file
f = open('./matrix3.txt', 'a')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix3.txt' mode='a' encoding='cp1252'>
```

Opening a text file in Python

More advanced modes exist:

- **Read and write ('r+'):** Opens the file and allows all operations.
- **Read and write ('w+'):** Opens the file and allows all operations. **Erases the content of the file!**

There are also additional modes for, say, binary files, but these are simply out-of-scope for CTD.

```python
# Open file in read ('r') mode
f = open('./matrix.txt', 'r')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix.txt' mode='r' encoding='cp1252'>
```

```python
# Open file in write ('w') mode
# Important: write mode, erases the file!
f = open('./matrix2.txt', 'w')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix2.txt' mode='w' encoding='cp1252'>
```

```python
# Open file in append ('a') mode
# Note: basically, write mode but does not erase the file
f = open('./matrix3.txt', 'a')
print(f)
f.close()
```

```
<_io.TextIOWrapper name='./matrix3.txt' mode='a' encoding='cp1252'>
```

Open a text file in Python

Another formatting for opening a text file suggests to use the **with** statement upon opening the file.

- It is used to indent the file input/outputs commands.
- Improves the readability of your code (Preferred).

```
# Using the with statement for read/write commands
with open('./matrix2.txt', 'w') as f:
    # Write the string to the file
    string = "Kueh salat"
    f.write(string)
    f.close()
```

Reading a file with **read()**

Assuming the file has been opened with a valid reading mode, you can read its content with the **read()** function.

- It is applied to the string handle f.
- If no argument given, read everything in the file and store it in a string.

```python
# Reading the entire file with read(), storing it in s
with open("./my_file.txt", "r") as f:
    s = f.read()
    print("s:", s)
    print("Length of s:", len(s))
    f.close()
```

```
s: Matthieu DE MARI
matthieu_demari@sutd.du.sg
French
Level 35
Length of s: 59
```

Reading a file with read()

Character-wise and multiple readings with **read()**.

- If an argument **n** is given to the read() function, read **n** characters from the file.
- If successive reads, keep on reading the file, going from the position where the previous read ended.
- First read, begins at the first character in the file.

```python
# Reading the a few characters at a time in file with read(n),
# storing it in s1, s2, etc.
with open("./my_file.txt", "r") as f:
    # Read the first 8 characters
    s1 = f.read(8)
    print("s1:", s1)
    print("Length of s1:", len(s1))
    # Read three more characters
    s2 = f.read(3)
    print("s2:", s2)
    print("Length of s2:", len(s2))
    # Read the rest of the characters
    s3 = f.read()
    print("s3:", s3)
    print("Length of s3:", len(s3))
    f.close()
```

```
s1: Matthieu
Length of s1: 8
s2:  DE
Length of s2: 3
s3:  MARI
matthieu_demari@sutd.du.sg
French
Level 35
Length of s3: 48
```

Reading a file with readlines()

It is also possible to read all lines at once with **readlines()**.

- It simply produces a list of strings, containing each line in the file.
- Note that each line ends with the end-of-line character \n, except for the last line. If needed, slicing or strip() string method to remove it.

```
# It is also possible to use readlines()
# Stores all lines as strings, in a list.
# Note that each line will end with the end-of-line
# character \n, except for the last line.
with open("./my_file.txt", "r") as f:
    l = f.readlines()
    print("l:", l)
    print("Length of s:", len(l))
    f.close()
```

```
l: ['Matthieu DE MARI\n', 'matthieu_demari@sutd.du.sg\n', 'French\n', 'Level 35']
Length of s: 4
```

For loop with a file handle

It is possible to use a file handle as a generator in a for loop.

- **The iteration variable will read each line, one at a time.**
- This would be equivalent to replacing **f** with **f.readlines()** in the for loop definition.

```python
# A file handle can be used as a generator
# in a for loop.
# It will give the lines, one at a time.
with open("./my_file.txt", "r") as f:
    for line in f:
        print("------")
        print("line:", line)
        print("Length of line:", len(line))
    f.close()
```

```
------
line: Matthieu DE MARI

Length of line: 17
------
line: matthieu_demari@sutd.du.sg

Length of line: 27
------
line: French

Length of line: 7
------
line: Level 35
Length of line: 8
```

Reminder on split() and strip()

The string method offered two methods that can help:

- The **strip()** method removes characters at the beginning and the end of a string. When no argument is passed, it removes extra spaces and end-of-line markers **\n**.
- The **split()** method separates a string into a list of strings using a specified separator. When no argument is passed, it uses spaces as separators.

Refer to Lesson 4 materials if you need more details!

Writing to a file with write()

Similarly, assuming we have opened a file in valid write mode

- We can write strings to text files using the **write()** method.
- Successive write will start writing where the previous write stopped. Same idea as in read().

```
# Writing several lines to a file with write()
# Can open the file manually, erase its content,
# and run the code to confirm it works!
s1 = "NPC: Hello there adventurer!\n"
s2 = "Hero: What is going on good sir?\n"
s3 = "NPC: I lost my daugther! Please help me find her.\n"
s4 = "NPC: I saw her going towards the volcano area.\n"
s5 = "Hero: You are the worst parent I have ever met."
with open("./dialog.txt", "w") as f:
    f.write(s1)
    f.write(s2)
    f.write(s3)
    f.write(s4)
    f.write(s5)
    f.close()
```

Writing to a file with writelines()

Similarly, assuming we have opened a file in valid write mode

- We can write strings to text files using the **write()** method.
- Successive write will start writing where the previous write stopped. Same idea as in read().
- You can also write multiple lines stored in a list, at once, with the **writelines()** function.

```
# Writing several lines to a file with writelines()
# Can open the file manually, erase its content,
# and run the code to confirm it works!
s1 = "NPC: Hello there adventurer!\n"
s2 = "Hero: What is going on good sir?\n"
s3 = "NPC: I lost my daugther! Please help me find her.\n"
s4 = "NPC: I saw her going towards the volcano area.\n"
s5 = "Hero: You are the worst parent I have ever met."
l = [s1, s2, s3, s4, s5]
with open("./dialog.txt", "w") as f:
    f.writelines(l)
    f.close()
```

Writing to a file with **writelines()**

Similarly, assuming we have opened a file in valid write mode

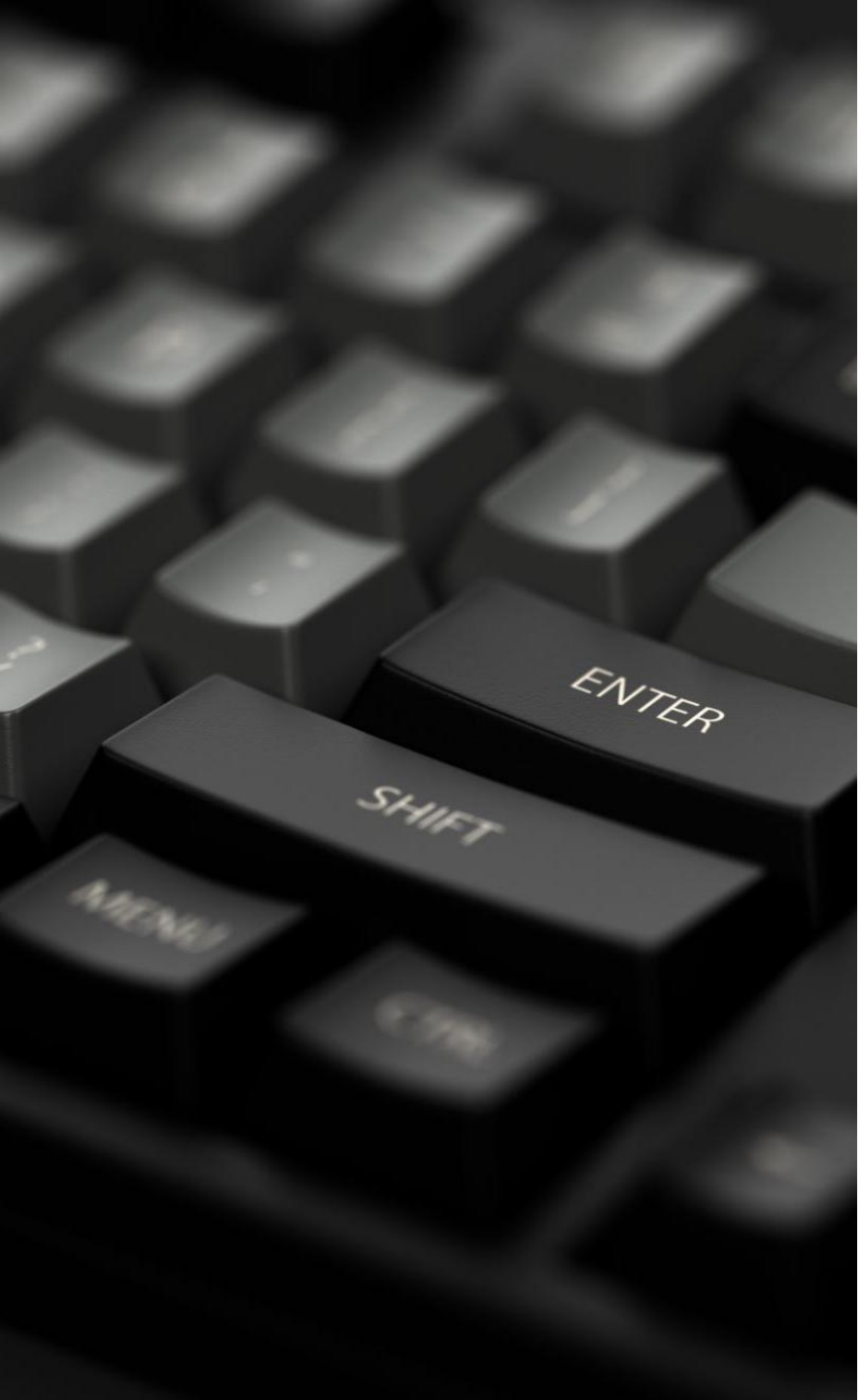
- We can write strings to text files using the **write()** method.
- Successive write will start writing where the previous write stopped. Same idea as in read().
- You can also write multiple lines stored in a list, at once, with the **writelines()** function.

Reading and writing in files can be useful for your projects if:

- You need a save file system (saving the progress of your user/player)
- You need to keep track of a leaderboard with best scores
- You need to centralize data somewhere and save it for future use (poor man's database).
- Etc.

A friendly reminder to RTFM

As usual, many more operations on text files, so RTFM!: https://docs.python.org/3/library/pathlib.html#reading-and-writing-files



Conclusion (Chapter 13)

- What is a **file handle**?
- What are **opening modes** for files?
- How to **read** and **write strings to a text file**?
- Practice

slido

Please download and install the Slido app on all computers you use

What will be printed?

ⓘ Start presenting to display the poll results on this slide.

slido

Please download and install the Slido app on all computers you use

What will appear in the output.txt file?

ⓘ Start presenting to display the poll results on this slide.