Lecture 10

File IO

Burkay Genç, Ahmet Selman Bozkır, and Selma Dilek

17/05/2023

PREVIOUS LECTURE

- exceptions
- assertions

TODAY

- file IO
- how to read from a file on disk
- how to write to a file on disk

File Types

- The type of a file is usually understood from its **extension**
- The extension of a file is the **substring after the last '.'**
- There are two types of files on your harddisk
  - **Binary files**: contains characters unreadable by humans (only meaningful for a computer)
    - Image files : .JPG, .GIF, .PNG, .BMP
    - Audio files : .OGG, .MP3, .M4A, .WAV
    - Video files : .MP4, .MPEG, .AVI, .MKV
    - Data files : .SAV, .DAT, .BIN
    - Executable files : EXE
  - **Text files**: contain multiple lines of strings
    - Usually ends with .TXT

Text Files

A text file can be thought of as a sequence of lines

```
MUH101 Introduction to Programming
A Course By Burkay Genc
2019-2020 Spring

Topics:
1 - Introduction
2 - Branching and Iteration
3 - Iterations
...
```

You can download this file from here

- Download this file and place it into your work folder
- If you are working in Colab, then you must upload the file to Colab

Opening a File

- Before we can read the contents of a file, we must tell Python which file we are going to work with and what we will be doing with the file
- This is done with the `open()` function
- `open()` returns a **file handle** - a variable used to perform operations on the file
- Similar to “File -> Open” in a Word Processor

Using `open()`

```
handle = open(filename, mode)
```

returns a handle used to manipulate the file

- `filename` is a string
  - it must clearly show where the file is located on the computer
- `mode` is optional and should be
  - **'r'** if we are planning to **read** from the file
  - **'w'** if we are going to **write** to the file

Example

```
muh101_file = open("muh101.txt", "r")
muh101_file
```

```
## <_io.TextIOWrapper name='muh101.txt' mode='r' encoding='cp65001'>
```

```
muh101_file.close()
```

- Never forget to close a file when you are done
  - `handle.close()`

Wrong Filename

- If you try to open a handle to a non-existent file, you will get an error:

```
muh101_file = open("muh100.txt", "r")
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'muh100.txt'
```

The `newline` Character

- We use a special character called the **newline** to indicate when a line ends
- We represent it as `\n` in strings
- **newline** is still one character - not two
- newline only creates a new line when **printed**

```
stuff = "Hello\nWorld!"
stuff
```

```
## 'Hello\nWorld!'
```

```
print(stuff)
```

```
## Hello
## World!
```

```
len(stuff)
```

```
## 12
```

Processing Text Files

A text file can be thought of as a sequence of lines

```
MUH101 Introduction to Programming
A Course By Burkay Genc
2019-2020 Spring

Topics:
1 - Introduction
2 - Branching and Iteration
3 - Iterations
...
```

Processing Text Files

- A text file has **newlines** at the end of each line
  - But they are not normally visible in a text editor

```
MUH101 Introduction to Programming\n
A Course By Burkay Genc\n
2019-2020 Spring\n
\n
Topics:\n
1 - Introduction\n
2 - Branching and Iteration\n
3 - Iterations\n
...
```

READING TEXT FILES

File Handle as a Sequence

- A **file handle** open for read can be treated as a **sequence** of strings where each line in the file is a string in the sequence
- We can use the **for** statement to iterate through a **sequence**
- Remember - a **sequence** is an ordered set

```
xfile = open('muh101.txt', 'r')
for line in xfile:
    print(line, end="")
```

```
## MUH101 Introduction to Programming
## A Course By Burkay Genc
## 2019-2020 Spring
##
## Topics:
## 1 - Introduction
## 2 - Branching and Iteration
## 3 - Iterations
## 4 - Functions and Scope
## 5 - Functions and Recursion
## 6 - Tuples and Lists
## 7 - Dictionaries
## 8 - Testing and Debugging
## 9 - Exceptions and Assertions
## 10 - File IO
```

```
xfile.close()
```

Counting Lines in a File

- Open a **file** read-only
- Use a **for** loop to read each line
- **Count** the lines and print out the number of lines

```
fhand = open('muh101.txt', 'r')
count = 0
for line in fhand:
  count = count + 1
print('Line Count:', count)
```

```
## Line Count: 15
```

```
fhand.close()
```

Reading the Whole File

We can **read** the whole file (newlines and all) into a **single string** using the `handle.read()` function

```
fhandle = open('muh101.txt', 'r')
wholeContent = fhandle.read()
print(wholeContent)
```

```
## MUH101 Introduction to Programming
## A Course By Burkay Genc
## 2019-2020 Spring
##
## Topics:
## 1 - Introduction
## 2 - Branching and Iteration
## 3 - Iterations
## 4 - Functions and Scope
## 5 - Functions and Recursion
## 6 - Tuples and Lists
## 7 - Dictionaries
## 8 - Testing and Debugging
## 9 - Exceptions and Assertions
## 10 - File IO
```

```
fhandle.close()
```

You can provide an integer to read that many characters from the file:

```
fhandle = open('muh101.txt', 'r')
wholeContent = fhandle.read(100)
print(wholeContent)
```

```
## MUH101 Introduction to Programming
## A Course By Burkay Genc
## 2019-2020 Spring
##
## Topics:
## 1 - Introducti
```

```
fhandle.close()
```

Searching Through a File

We can put an **if** statement in our **for** loop to only print lines that meet some criteria

```
fhandle = open('muh101.txt')
for line in fhandle:
  if line.startswith('3') or line.startswith('5'):
    print(line)
```

```
## 3 - Iterations
##
## 5 - Functions and Recursion
```

```
fhandle.close()
```

Removing newline Characters

- When you print a string read from a file you get extra newlines printed
  - This is because `print()` adds a newline by default
  - So it becomes two newlines
- We can avoid this by using `rstrip()` function to remove the newlines from each line

```
fhandle = open('muh101.txt')
for line in fhandle:
  line = line.rstrip()
  if line.startswith('3') or line.startswith('5'):
    print(line)
```

```
## 3 - Iterations
## 5 - Functions and Recursion
```

```
fhandle.close()
```

- `rstrip` removes extra whitespaces from strings
  - including the newline character at the end

Skipping With Continue

We can conveniently skip a line by using the **continue** statement

```
fhandle = open('muh101.txt')
i = 0
for line in fhandle:
  i += 1
  line = line.rstrip()
  if i < 5:                # Skip the first 4 lines
    continue
  print(line)
```

```
## Topics:
## 1 - Introduction
## 2 - Branching and Iteration
## 3 - Iterations
## 4 - Functions and Scope
## 5 - Functions and Recursion
## 6 - Tuples and Lists
## 7 - Dictionaries
## 8 - Testing and Debugging
## 9 - Exceptions and Assertions
## 10 - File IO
```

```
fhandle.close()
```

Using in to Select Lines

We can look for a string anywhere in a line as our selection criteria

```
fhandle = open('muh101.txt')
for line in fhandle:
  line = line.rstrip()
  if 'and' in line:          # Print the line only if it contains 'and'
    print(line)
```

```
## 2 - Branching and Iteration
## 4 - Functions and Scope
## 5 - Functions and Recursion
## 6 - Tuples and Lists
## 8 - Testing and Debugging
## 9 - Exceptions and Assertions
```

```
fhandle.close()
```

IO Exceptions

- It is very likely to get IO Exceptions when dealing with file input and output
- So, always surround file operations with try/except blocks

```
try:
  fhandle = open("muh100.txt")
  print("File contains", len(fhandle.read()), "characters.")
except:
  print("File not found!")
```

```
## File not found!
```

Writing To A File

- Writing is similar to reading
  - Use ‘w’ instead of ‘r’

```
fhandle = open("muh102.txt", "w")
fhandle.write("test string")        # Outputs number of written characters
```

```
## 11
```

```
fhandle.close()
```

We can now read from newly created `muh102.txt`:

```
fhandle = open("muh102.txt", "r")
print(fhandle.read())
```

```
## test string
```

```
fhandle.close()
```

Writing To A File

- If you write on an existing file, it gets **overwritten**:

```
fhandle = open("muh102.txt", "w")
fhandle.write("another string")
```

```
## 14
```

```
fhandle.close()
fhandle = open("muh102.txt", "r")
print(fhandle.read())
```

```
## another string
```

```
fhandle.close()
```

- You can use the append, `'a'`, mode to avoid this behaviour.

Appending To A File

- If you write on an existing file, it gets **overwritten**:

```
fhandle = open("muh102.txt", "a")
fhandle.write("appended string")
```

```
## 15
```

```
fhandle.close()
fhandle = open("muh102.txt", "r")
print(fhandle.read())
```

```
## another stringappended string
```

```
fhandle.close()
```

- Use `\n` to write to a new line
  - `fhandle.write("\nappended string")`

File Open Modes

There are many other modes besides ‘r’, ‘w’ and ‘a’

- ‘rb’: Opens a file for reading only in binary format.
- ‘r+’: Opens a file for both reading and writing.
- ‘rb+’: Opens a file for both reading and writing in binary format.
- ‘wb’: Opens a file for writing only in binary format. Overwrites the file if the file exists.
- ‘w+’: Opens a file for both writing and reading. Overwrites the existing file if the file exists.
- ‘wb+’: Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists.
- ‘ab’: Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists.
- ‘a+’: Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists.
- ‘ab+’: Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists.

Example

- Create a file containing squares of the first 10 positive integers, line by line

Example

- Create a file containing squares of the first 10 positive integers, line by line

```
fhandle = open("squares.txt", "w")
for i in range(1, 11):
  fhandle.write(str(i**2) + "\n")
```

```
## 2
## 2
## 2
## 3
## 3
## 3
## 3
## 3
## 3
## 4
```

```
fhandle.close()
```

Example

Check the file:

```
fhandle = open("squares.txt", "r")
print(fhandle.read())
```

```
## 1
## 4
## 9
## 16
## 25
## 36
## 49
## 64
## 81
## 100
```

```
fhandle.close()
```

Opening a file using `with open`

The `with` statement works with the `open()` function to open a file. Unlike `open()` where you have to close the file with the `close()` method, the `with` statement closes the file for you.

```
with open("squares.txt", "r") as fhandle:
  lines = fhandle.readlines()
print(lines)
```

```
## ['1\n', '4\n', '9\n', '16\n', '25\n', '36\n', '49\n', '64\n', '81\n', '100\n']
```

- `readlines()` returns all lines in the file, as a list where each line is an item in the list object.

Copyright Information