Python
WEEK 7
File Handling

AI Academy

COMPUTER PROGRAMMING WITH PYTHON

Instructor - James E. Robinson, III

Teaching Assistant - Travis Martin

LIGHTNING REVIEW

- Variables
- Input / Output
- Expressions
- Functions
- Conditional Control
- Looping
- Data Types
- Logging
- Functions
- Scope
- Recursion
- Decorators
- Dynamic Prg
- Exceptions
- Classes
- Objects
- Encapsulation
- Public v/s Private
- Dunder Methods
- Instances
- Inheritance
- Types of Inheritance
- Polymorphism
- Method Overriding
- Queue
- Stacks
- Graphs
- Trees
- Binary Trees
- Traversal Methods
- Searching

TOPICS COVERED

- Files
- Opening / Closing
- Reading / Writing
- Context Managers
- File Exceptions
  - Handling Exceptions
- File Formats

FILES

RECORDED DATA RESOURCES STORED IN A DEVICE

File extension
Suffix to the name of a computer file.
Indicates a characteristic of the file contents or its intended use.
Example : .txt, .docx, .md

File path
String of characters used to uniquely identify a location in a directory structure.
Usually indicated by hierarchy structure of folders
Example: home/documents/file.txt

OPENING / CLOSING

TO HANDLE FILES, THEY NEED TO BE OPENED AND CLOSED WITHIN CODE

Syntax

```
f = open("filename.txt", "mode")
# file operations
f.close()
```

Example

```
f = open("home/exp.txt", "r")
data = f.read()
f.close()
```

Modes

"r" - Read (default)
Opens a file for reading
"a" - Append
Opens a file for appending
"w" - Write
Opens a file for writing
"+" -
Adds reading privileges

Note:

'r' mode returns error if file does not exist, while 'a' & 'w' create the file.

filename can include a relative or absolute path.

Cursor

File object returned that indicated the current position of the file handling process
i.e variable f from example

READING / WRITING

METHODS FOR READING AND WRITING THE CONTENT OF THE FILE

Example

```
f = open("file.txt",'w')
f.write("Hello World")
f.write("\nHow are you?")
f.close()
f = open("file.txt",'r')
data = f.read()
f.close()
print(data)
```

OUTPUT

Hello World
How are you?

Reading whole file

```
data = f.read()
```

Reading single line

```
data = f.readline()
```

Using for loop

```
for x in f:
        data += x
```

Writing in file

f.write('demo text')

Note:

Once you have read a file, the cursor points at the end of the file. You will have to first close and open the file again to access the data from the start.

CONTEXT MANAGER

THE 'WITH' KEYWORD

Context Managers help the code allocate and release resources without explicit mention in the code.

Syntax

```
with open('filename.txt', 'w') as file:
    # operations
    file.write('Demo Text')
```

Example

```
with open('double.txt', 'w') as file:
    x = int(input('Number:'))
    file.write('Double = '+str(x*2))
```

Equivalent Example

```
f = open('double.txt', 'w')
x = int(input('Number:'))
f.write('Double = '+str(x*2))
f.close()
```

Note:

Context managers close file once outside the scope of 'with'

There are methods to customize context managers here

FILE EXCEPTIONS

EXCEPTIONS THAT OCCUR DURING FILE HANDLING PROCEDURES

FileNotFoundError

When trying to operate on a file that doesn't exist (or incorrect name/path is given)

FileExistsError

When trying to create a new file that already exists in the directory

PermissionError

When trying to access or alter file for which necessary system permissions are not present

IsADirectoryError

When trying to interact with a directory as a file

ValueError

When trying to write a closed file

UnsupportedOperation

When trying to write a file that is opened in ‘read’ mode

HANDLING FILE EXCEPTIONS

USING TRY - EXCEPT - FINALLY

Example

```
filename = input("Enter FileName :")
try:
        with open(filename, "w") as fp:
                fp.write("hello, world\n")
        with open(filename, "r") as fp:
                print(fp.read())
except Exception as e:
        print(e)
finally:
        print("File Handling Complete")
```

Note:

This is similar to Exception Handling as discussed in Week 3

Some Errors are mode specific, therefore exceptions can be more detailed based on the mode used in the code.

FILE FORMATS

DATA IS STORED IN A STRUCTURED MANNER IN FILES IN SPECIFIC FORMATS

XML
Extensible Markup Language

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this
weekend!</body>
</note>
```

Format based on tags & directory like structure

HTML
Hyper Text Markup Language

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

File format suitable for web content

FILE FORMATS

DATA IS STORED IN A STRUCTURED MANNER IN FILES IN SPECIFIC FORMATS

For Data Mining and AI applications, the most used file formats are JSON and CSV

JSON
JavaScript Object Notation

{
'1' : {"name":"John", "age":30, "car":null}
'2': {"name":"Sam", "age":25, "car":Sedan}
}

JSON files are based on Key Value Pairing

CSV
Comma Separated Values

1, John, 30, null
2, Sam, 25, Sedan

CSV Formats are based on rows and columns

NEW MODULE INTRODUCTION

CSV

JSON

XML

CSV FILE EXTRACTION

USING CSV MODULE

Example File
(test.csv)
NAME,AGE,CAR
Sam,25,Sedan
Kim,30,Crossover
Jane,20,Sedan

Example Code

```
import csv
rows = [ ]
with open("test.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)
print(header)
print(rows)
```

OUTPUT

```
['"NAME"', '"AGE"', '"CAR"']
[['"Sam"', '25', '"Sedan"'], ['"Kim"',
'30', '"Crossover"'], ['"Jane"', '20',
'"Sedan"']]
```

WEEK SUMMARY

- Learned to open, read, write, and close files
- Learned use of context managers
- Understood different file handling exceptions
- Learned to deal with exceptions occurring in file handling procedures
- Learned different types of file formats
- Learned to parse csv files

THANK YOU

FOR ADDITIONAL QUERIES OR DOUBTS
CONTACT:
jerobins@ncsu.edu

AI Academy