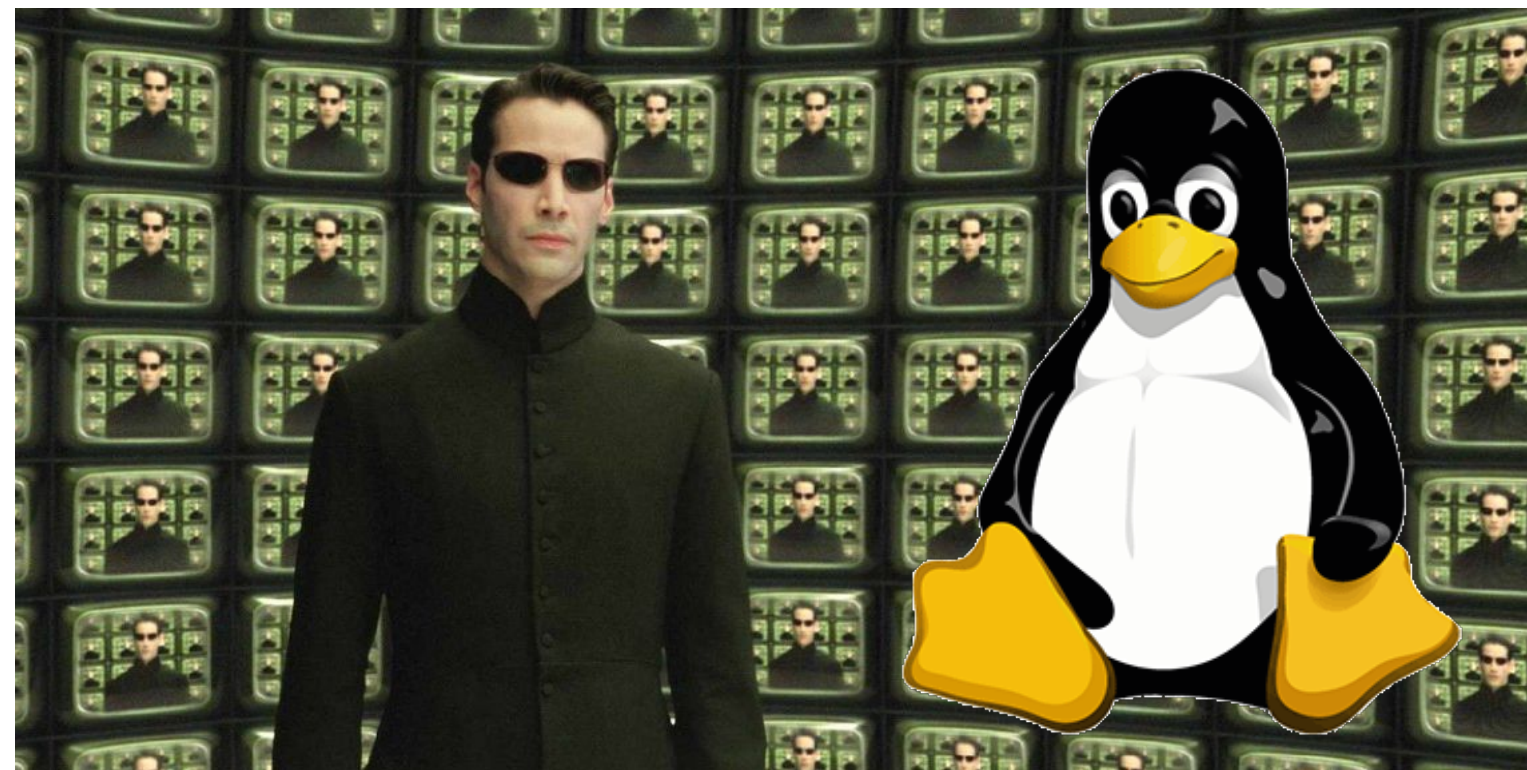




CSCI 120

Introduction to CompSci and Programming I

Lecture 6: File IO



Jetic Gū

Overview

- Focus: Python Programming
- Architecture: von Neumann
- Core Ideas:
 1. File IO

File I/O

Reading and Writing to file

Computer I/O

- Keyboard/Onscreen
 - `input / print` functions in python
- Modern Computers store files on storage devices
 - Programmes access files on these devices through File I/O

Opening/Closing a File

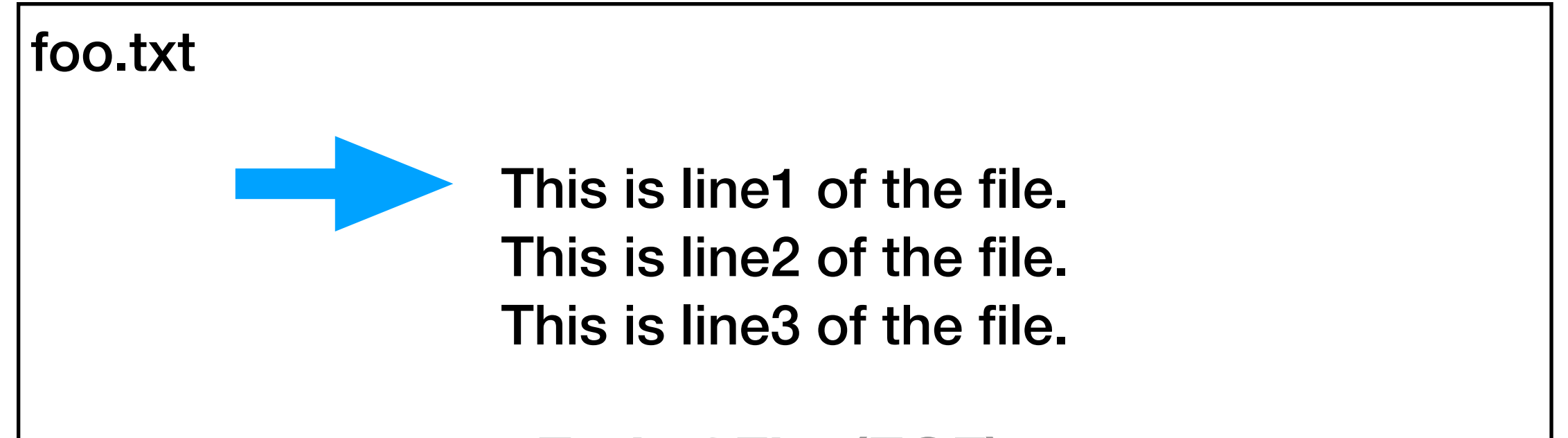
- Python provides basic functions and methods necessary to manipulate files through **file** objects
- `open(fileName, [, access_mode])`
 - Before you can read or write a file, you have to **open** it. This creates a file object
- Example:
 - `myFile = open("foo.txt", "r")`
This opens the file `foo.txt` in **read** mode
 - When you are done, close it
`myFile.close()`

Opening/Closing a File

- A better way:
 - `with open("foo.txt", "r") as myFile:`
 # do stuff here
- This way, the programme closes the file as soon as you exit the subroutine

Reading a File

- file pointer
- a file is like a map
- the file pointer points to the place you are currently at
- when you read a line the pointer will move forward
- you can also reset the pointer



End of File (EOF)

```
> f = open("foo.txt", "r")
> print(f.readline())
> print(f.readline())
> print(f.readline())
> f.seek(0)
# reset pointer to position 0
# which is the very beginning
```

Reading a File

- Let's say you want to read the file `shakespeare-hamlet.txt`
<https://raw.githubusercontent.com/teropa/nlp/master/resources/corpora/gutenberg/shakespeare-hamlet.txt>
- `myFile = open("Hamlet.txt", "r")`
- Iterating
 - **`for line in myFile:`**
`print(line)`
prints each line
each line is already ending with `'\n'`, so if you do it this way, there will be empty lines

Example

Reading a File

- Read all-in-one
 - `content = myFile.read()`
this returns the whole file as a single string
 - `list_content = list(myFile)`
this returns the whole file as a **list**
each element is a single line **string ending with '\n'**

Access Modes (Short)

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer will be at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

Technical

file Object

- attributes
 - `myFile.closed`
Returns true if file is closed, false otherwise.
 - `myFile.mode`
Returns access mode with which file was opened.
 - `myFile.name`
Returns name of the file.

file Object

- **Example:**

```
fo = open("foo.txt", "wb")  
print("Name of the file: ", fo.name)  
print("Closed or not : ", fo.closed)  
print("Opening mode : ", fo.mode)  
print("Softspace flag : ", fo.softspace)
```

- **This would produce following result:**

```
Name of the file: foo.txt  
Closed or not : False  
Opening mode : wb  
Softspace flag : 0
```

Writing to File

The `write()` Method:

- The `write()` method writes any string to an open file
- The `write()` method does not add a newline character (`'\n'`) to the end of the string

Example:

```
with open("output.txt", "w") as f:  
    # file opened in write mode  
    f.write("I like cheese")
```

Writing to File

- **Example:**

```
> f = open("foo.txt", "wb")
```

```
> f.write("Python is a great language.\r\nYeah its great!!\r\n");
```

```
> f.close()
```

- The above method would create `foo.txt` file and write the following in it
Python is a great language.
Yeah its great!!

File System Management

- OS provides file-processing operations, such as **renaming** and **deleting** files.
- The `rename()` method takes two arguments, the current filename and the new filename.
- Syntax:

```
os.rename(current_file_name, new_file_name)
```
- Example:

```
import os  
os.rename("test1.txt", "test2.txt")
```

File System Management

- You can use the `delete()` method to delete files by supplying the name of the file to be deleted as the argument

- Syntax:

```
os.remove(file_name)
```

- Example:

```
import os  
os.remove("test2.txt")
```


Directories in Python

- Use the `mkdir()` method of the `os` module to create directories in the current directory. You need to supply an argument to this method, which contains the name of the directory to be created.

- Syntax:

```
os.mkdir("newdir")
```

- Example:

```
import os  
os.mkdir("test") # Create a directory "test"
```

Directories in Python

- Use the `chdir()` method to change the current directory. The `chdir()` method takes an argument, which is the name of the directory that you want to make the current directory.

- Syntax:

```
os.chdir("newdir")
```

- Example:

```
import os  
os.chdir("test")
```

Directories in Python

- The `getcwd()` method displays the current working directory

- Syntax:

```
os.getcwd()
```

- Example:

```
import os  
os.getcwd()
```

Directories in Python

- The `rmdir()` method deletes the directory, which is passed as an argument in the method. Before removing a directory, all the contents in it **MUST** be removed.
- **Syntax:**

```
os.rmdir('dirname')
```
- **Example:**

```
import os  
os.rmdir("/tmp/test")
```