Sharif University of Technology
Department of Computer Engineering

Fundamentals of Programming

Python Language

Arman Malekzadeh
PhD Candidate in Artificial Intelligence

Table of contents

Working with Files

Files in Python

- Python has several functions for creating, reading, updating, and deleting files.
- The key function for working with files in Python is the **open()** function.
- The **open()** function takes two parameters; filename, and mode.
- There are four different methods (modes) for opening a file:
  - **"r"** - Read - Default value. Opens a file for reading, error if the file does not exist
  - **"a"** - Append - Opens a file for appending, creates the file if it does not exist
  - **"w"** - Write - Opens a file for writing, creates the file if it does not exist
  - **"x"** - Create - Creates the specified file, returns an error if the file exists

Files in Python

- In addition you can specify if the file should be handled as binary or text mode
- **"t"** - Text - Default value. Text mode
- **"b"** - Binary - Binary mode (e.g. images)

```
f = open("code.txt", "rt")
f = open("image.jpg", "rb")
```

Files in Python

- To read a file in Python, we must open the file in reading mode.
- There are various methods available for this purpose. We can use the **read(size)** method to read in size number of data. If size parameter is not specified, it reads and returns up to the end of the file.
- We can read the file line by line using a for loop. This is both efficient and fast.

```
f = open("demofile.txt", "r")
print(f.read())
```

Files in Python

- We can read the file line by line using a for loop. This is both efficient and fast.

```
f = open("demofile.txt", "r")
for x in f:
    print(x)
```

Files in Python

- We can read the file line by line using a for loop. This is both efficient and fast.

```
f = open("demofile.txt", "r")
print(f.readline())
```

Files in Python: Read only Parts of File

- By default the read() method returns the whole text, but you can also specify how many characters you want to return.
- Return the 10 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(10))
```

Files in Python: Read Lines

- You can return one line by using the readline() method.
- Read one line of the file:

```
f = open("demofile.txt", "r")
print(f.readline())
```

Closing Files in Python

- When we are done with performing operations on the file, we need to properly close the file.
- Closing a file will free up the resources that were tied with the file and is done using Python close() method.
- Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.
- It is done using the close() method available in Python.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Writing to Files in Python

- To write to an existing file, you must add a parameter to the open() function:
  - **"a"** - Append - will append to the end of the file
  - **"w"** - Write - will overwrite any existing content
- Write to an existing file:

```
f = open("demofile.txt", "a")
f.write("Now the file has more content!")
f.close()
```

Writing to Files in Python

- To write to an existing file, you must add a parameter to the open() function:
  - **"a"** - Append - will append to the end of the file
  - **"w"** - Write - will overwrite any existing content
- Write to an existing file:

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

Creating New Files in Python

- To create a new file in Python, use the open() method, with one of the following parameters:
  - **"x"** - Create - will create a file, returns an error if the file exist
  - **"a"** - Append - will create a file if the specified file does not exist
  - **"w"** - Write - will create a file if the specified file does not exist
- Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
f.close()
```

Deleting Files in Python

- To delete a file, you must import the OS module, and run its os.remove() function:
- Remove the file "demofile.txt":

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

Deleting Folders in Python

- To delete an entire folder, use the os.rmdir() method:
- Remove the folder "myfolder":

```
import os
os.rmdir("myfolder")
```

Check if File exist

- To avoid getting an error, you might want to check if the file exists before you try to delete it:
- Check if file exists, then delete it:

```
import os
if os.path.exists("demofile.txt"):
    os.remove("demofile.txt")
else:
    print("The file does not exist")
```

References

References I

[1] B Downey, A. (2015). Think Python: How to Think Like a Computer Scientist-2nd Edition.

[2] Deitel, H. M., & Deitel, P. J. (2004). C: How to program. Pearson Educacion.

Sharif University of Technology
Department of Computer Engineering

Arman Malekzadeh

Fundamentals of Programming

Python Language