GERMANNA
ACADEMIC CENTER
FOR EXCELLENCE

Python Programming: File Handling Handout

File handling is a common practice in programming that enables the programmer to read, write, and access content stored externally. Students in ITP 100 often find file handling to be a challenge when setting up their program and using essential functions.

This handout provides simple steps for accessing, locating, reading, and writing files in Python. Each of these topics have been divided into sections that include Python and coding steps, as well as examples.

Please use the following links to navigate between the sections:

Locating Files

When working with files in programming, it is important to know the file's location for reference. In Python, there are two ways to do this: by using the **relative path** or the **absolute path** of the file.

The **relative path** of a file consists of the file's name and extension. This path can only be used when the program is working from the same directory, folder, as the file that is being accessed.

Example:

Console Relative Path
with open("example.txt") as example_file:

In this case, the file "example.txt" is in the same directory as the python script file.

Alternatively, if the program is not located in the same directory as the file, a file's **absolute path** can be used instead. The **absolute path** details the exact location of where the file is stored on the computer.

Example:

Console File Path
with open("C:\\Users\\a_folder\\example.txt") as file:

In the example, the **absolute path** starts from the root directory "C:" and then lists all folders leading to a file. Each folder is separated by a double backslash " \\ " , until it locates 'example.txt'.

Accessing Files

Files must be created and opened before the information can be accessed or used. A file object can be assigned to a variable using the equal sign and the "open()" function.

Example:

Console
file = open("example.txt", "r")

In the example above, the variable is set with the mode value of "r" meaning that the file is to be read. The file's mode in Python dictates how the programmer intends to use the file.

Common Modes	
"r"	File open for reading
"w"	File open for writing, information will be overwritten
"a"	File open for appending, will add text on pre-existing files
"r+" or "w+"	Update mode that allows for reading and writing

Another option for opening a file is to use the '*with*' and '*as*' keywords and a colon to declare the file as a variable. Then the file variable will be accessed with a function such as "read()" or "open()".

Example:

Console
with open("example.txt", "r") as file: file.read()

Reading Files

Reading a file's contents in Python allows the programmer to gain access to view the information on the file. While the contents cannot be changed, the information can be displayed and used. The file must be opened and set with the 'r' or 'r+' mode parameter.

Files can be read in different ways; however, this handout will focus solely on the contents of ITP 100. The basic "read()" method in a program will read all of a file's contents but will not display the contents of the file to the console without being used in a print statement.

To access information line by line, students can use a 'for loop' or the "readlines()" function.

Example: Example.txt

```
Hello!
Welcome to our resources.
```

Example:

Console

```
file = open("Example.txt", "r")
for line in file:
  print(line)
```

Output

```
Hello!
Welcome to our resources.
```

When using a 'for loop,' the program checks each line of the document for data and exits the loop once there is not any data on the following line.

Example:

Console

```
file = open("Example.txt", "r")
print(file.readlines())
file.close()
```

Output

```
Hello!
Welcome to our resources.
```

The "readlines()" method accomplishes the same output as the for loop, but it does not allow options for using the data. Alternatively, a programmer can use the "readline()" method to read a single line from a document.

Writing and Amending Files

When altering files, it is important to open the file with the appropriate mode parameter. When choosing to add more information to the end of a file, the 'a' or "appended" parameter should be used.

Example: Example.txt

Hello! Welcome to our resources.

Console
file = open("Example.txt", "w") file.write("This a great resource.")
File Change
This is a great resource.

When using the "w" mode, if the file already has contents inside, the file will start over by clearing the previous data inside and writing the new value inside the document.

Example:

Console
file= open("Example.txt", "a") file.write(" This a great resource.")
File Change
Hello! Welcome to our resources. This is a great resource.

When appending files, it is important to include a space before the new statement, since it will be added directly on the last sentence in the file. If a space is not included, the new content will attach itself to the old content.

Example:

Console
f = open(“Example.txt”, “a”) f.write(“This a great resource.”)
File Change
Hello! Welcome to our resources.This is a great resource.

Once the information in the file is no longer needed, the file should be closed at the end to promote good coding practice and to prevent file corruption. If the file was opened using a *with-as* statement, the file will be automatically closed when the program exits that code block.

If the file object has been assigned to a variable, this is accomplished by calling the “close()” method on the object. Using the “close()” function only applies when the file uses the “=” operator.

Example:

Console
file = open(“Example.txt”, “a”) file.close()

Alternatively, when using the file with the keyword ‘with’ the file is automatically closed once the section has been completed.

Example:

Console
with open(“example.txt”, “w”) as file: file.read()

The Academic Center for Excellence (ACE) offers free on-campus and online tutoring appointments for software design. For further assistance with programming concepts, please call ACE at (540) 891-3017 or email ACE@germanna.edu.