M.A.N.U

2/21/2023

Mathematics Association of Nairobi University
isaak@students.uonbi.ac.ke
https://github.com/Isaakkamau

python modules

A `python module` can be defined as a python program file which contains a python code including python functions, class, or variables.

In other words, we can say that our python code file saved with the extension (`.py`) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provides us the flexibility to organize the code in a logical way.

Loading the module in our python code:

We use:

1. The `import` statement
2. The `from-import` statement

The import statement

The `import` statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file.

We can `import` multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

The from-import statement

Instead of importing the whole module into the namespace,python provides the flexibility to import only the specific attributes of a module. This can be done by using `from < module-name> import <name 1>, <name 2>..,<name n>` statement.

Re-naming a Module

You can create an alias when you import a module, by using the `as` keyword:

`import mymodule as mx`

Built-in Modules

There are several built-in modules in Python, which you can import whenever you like.

Reading and Writing to text files in Python

File handling is an important part of any web application.

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:

1. `"r"` - Read - Default value. Opens a file for reading, error if the file does not exist
2. `"a"` - Append - Opens a file for appending, creates the file if it does not exist
3. `"w"` - Write - Opens a file for writing, creates the file if it does not exist
4. `"x"` - Create - Creates the specified file, returns an error if the file exists

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

Open a File on the Server

Assume we have the following file, located in the same folder as Python:

- demofile.txt ```Hello! Welcome to demofile.txt
  ```
  This file is for testing purposes.
  Good Luck!
  ```
  ```

To open the file, use the built-in `open()` function.

The `open()` function returns a file object, which has a `read()` method for reading the content of the file:

In [ ]:
```
# Example

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

If the file is located in a different location, you will have to specify the file path, like this:

In [ ]:
```
f = open("D:\\myfiles\welcome.txt", "r")
print(f.read())
```

### Read Only Parts of the File

By default the `read()` method returns the whole text, but you can also specify how many characters you want to return:

Example: Return the 10 first characters of the file:

In [ ]:
```
f = open("demofile.txt", "r")
print(f.read(10))
```

By looping through the lines of the file, you can read the whole file, line by line:

In [ ]:
```
#Loop through the file line by line:

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

# Writing Files

### Write to an Existing File

To write to an existing file, you must add a parameter to the open() function:

1. `"a"` - Append - will append to the end of the file
2. `"w"` - Write - will overwrite any existing content

In [ ]:
```
#Open the file "demofile2.txt" and append content to the file:

f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```
```

```
#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

In [ ]:
```
#Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

Delete a File

To delete a file, you must import the `OS` module, and run its `os.remove()` function:

In [ ]:
```
#Remove the file "demofile.txt":

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

Delete Folder

To delete an entire folder, use the `os.rmdir()` method:

In [ ]:
```
#Remove the folder "myfolder":
#Note: You can only remove empty folders.

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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js