Python for Class XII _- Part 9

Topics:

File handling: Need for a data file, Types of file: Text files, Binary files and CSV (Comma separated values) files. *(CSV will be discussed separately)*

Text File: Basic operations on a text file: Open (filename – absolute or relative path, mode) / Close a text file, Reading and Manipulation of data from a text file, Appending data into a text file, standard input /output and error streams, relative and absolute paths.

Binary File: Basic operations on a binary file: Open (filename – absolute or relative path, mode) / Close a binary file, Pickle Module – methods load and dump; Read, Write/Create, Search, Append and Update operations in a binary file. (continued)

Python – stdin, stdout, and stderr

Before going through this tutorial, let us understand what the terms stdin, stdout and stderr are.

Standard input – This is the *file-handle* that a user program reads to get information from the user. We give input to the standard input (**stdin**).

Standard output – The user program writes normal information to this file-handle. The output is returned via the Standard output (**stdout**).

Standard error – The user program writes error information to this file-handle. Errors are returned via the Standard error (**stderr**).

Python provides us with **file-like objects** that represent **stdin**, **stdout,** and **stderr**. Let us look at how we could work with these objects to work with the input and output of our program.

1. sys.stdin

Python’s sys module provides us with all three file objects for stdin, stdout, and stderr. For the input file object, we use sys.stdin. This is similar to a file, where you can open and close it, just like any other file.

Let us understand this through a basic example:

```python
import sys

stdin_fileno = sys.stdin

# Keeps reading from stdin and quits only if the word 'exit' is there
# This loop, by default does not terminate, since stdin is open
for line in stdin_fileno:
    # Remove trailing newline characters using strip()
    if 'exit' == line.strip():
        print('Found exit. Terminating the program')
        exit(0)
    else:
        print('Message from sys.stdin: ---> {} <---'.format(line))
```

Output:

```
Hi
Message from sys.stdin: ---> Hi
 <---
Hello from RPVV Raj Niwas Marg
Message from sys.stdin: ---> Hello from RPVV Raj Niwas Marg
 <---
exit
Found exit. Terminating the program
```

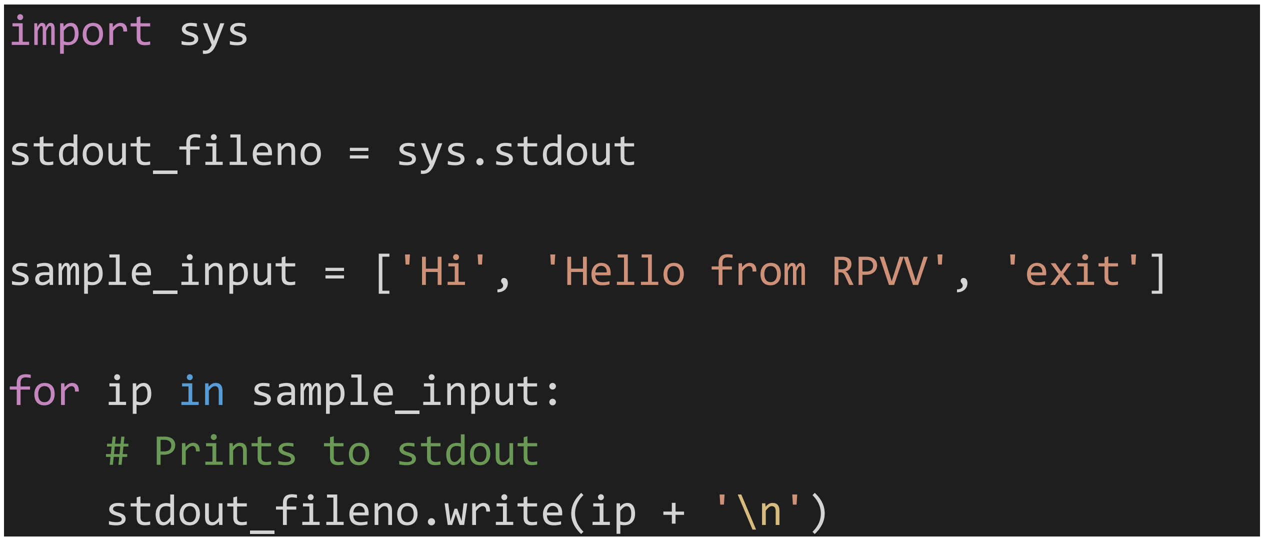
The above snippet keeps reading input from stdin and prints the message to the Console (stdout) until the word exit is encountered.
NOTE: We do not normally close the default stdin file object, although it is allowed.
So stdin_fileno.close() is valid Python code.

Now that we know a little bit about stdin, let us move to stdout.

2. sys.stdout

For the output file object, we use sys.stdout. It is similar to sys.stdin, but it directly displays anything written to it to the Console.

The below snippet shows that we get the Output to the Console if we write to sys.stdout.



```python
import sys

stdout_fileno = sys.stdout

sample_input = ['Hi', 'Hello from RPVV', 'exit']

for ip in sample_input:
    # Prints to stdout
    stdout_fileno.write(ip + '\n')
```

Output:

```
Hi
Hello from RPVV
exit
```

3. sys.stderr

This is similar to sys.stdout because it also prints directly to the Console. But the difference is that it *only* prints **Exceptions** and **Error messages**. (That is why it is called **Standard Error**).

Let us illustrate this with an example.

```python
import sys

stdout_fileno = sys.stdout
stderr_fileno = sys.stderr

sample_input = ['Hi', 'Hello from RPVV', 'exit']

for ip in sample_input:
    # Prints to stdout
    stdout_fileno.write(ip + '\n')
    # Tries to add an Integer with string. Raises an exception
    try:
        ip = ip + 100
    # Catch all exceptions
    except:
        stderr_fileno.write('Exception Occurred!\n')
```

Output:

```
Hi
Exception Occurred!
Hello from RPVV
Exception Occurred!
exit
Exception Occurred!
```

As you can observe, for all the input strings, we try to add to an Integer, which will raise an Exception. We catch all such exceptions and print another debug message using sys.stderr.

Redirection to a file

We can redirect the stdin, stdout and stderr file handles to any other file (file-handle). This may be useful if you want to log events to a file without using any other module such as Logging.
The below snippet redirects output(stdout) to a file called Output.txt.

So, we will not see anything printed to the Console, because it is now being printed to the file itself! This is the essence of Output redirection. You ‘redirect’ the Output to some other place. (This time, to Output.txt, instead of the Console)

```python
import sys

# Save the current stdout so that we can
# revert sys.stdou after we complete our redirection
stdout_fileno = sys.stdout

sample_input = ['Hi', 'Hello from RPVV', 'exit']

# Redirect sys.stdout to the file
sys.stdout = open('Output.txt', 'w')

for ip in sample_input:
    # Prints to the redirected stdout (Output.txt)
    sys.stdout.write(ip + '\n')
    # Prints to the actual saved stdout handler
    stdout_fileno.write(ip + '\n')

# Close the file
sys.stdout.close()
# Restore sys.stdout to our old saved file handler
sys.stdout = stdout_fileno
```

Output:

```
Hi
Hello from RPVV
```

```
exit
```

Contents of Output.txt:

```
Hi
Hello from RPVV
exit
```

As you can see, we have printed the output to both the Console, as well as to Output.txt.
We first save the original sys.stdout file handler object to another Variable. We not only need this to restore sys.stdout to the old handler (pointing to the Console), but we can also print to the console using this Variable!
Note that after writing to the file, we close it, similar to how we close a file because that file was still open.
We finally restore the handler of sys.stdout to the Console, using the variable stdout_fileno.
A similar process can be followed for Input and Error redirection, replacing sys.stdout with sys.stdin or sys.stderr and working with Inputs and Exceptions instead of Output.

Summary

In this tutorial, we learned about using stdin, stdout and stderr in Python, using the sys module. We also learned how to manipulate the corresponding file handlers for redirection to/from a file.