# MINI SHELL IN PYTHON

## Team Member Details

| | | | |
|---|---|---|---|
| Saumya Shah | 60004150109 | +91-9769953291 | saumya.shah@hotmail.com |
| Shikhar Shah | 60004150110 | +91-7666550475 | sb.shah14@gmail.com |
| Niti Shah | 60004150102 | +91-9769121929 | nits2097@gmail.com |

## Description

A shell is an ordinary program that reads commands from the user and executes them, and is the primary user interface to traditional Unix-like systems. The fact that the shell is a user program, and not a part of the kernel, illustrates the power of the system call interface.

We have tried to implement a simple shell using python. Two modules have been imported for the same - the os module and the cmd class from the cmd2 module.

*The os module*

This module provides a portable way of using operating system dependent functionality.The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

Some important functions that can be implemented using an os module:

```
import os


#Executing a shell command
os.system()


#Get the users environment
os.environ()


#Returns the current working directory.
os.getcwd()


#Return the real group id of the current process.
```

```
os.getgid()


#Return a list of the entries in the directory given by path.
os.listdir(path)

#Create a directory named path with numeric mode mode.
os.mkdir(path)

#Remove (delete) the directory path.
os.rmdir(path)
```

*The cmd class and the cmd2 module*
cmd2 is a tool for building interactive command line applications in Python. Its goal is to make it quick and easy for developers to build feature-rich and user-friendly interactive command line applications. It provides a simple API which is an extension of Python's built-in cmd module.
The Cmd class provides a simple framework for writing line-oriented command interpreters. These are often useful for test harnesses, administrative tools, and prototypes that will later be wrapped in a more sophisticated interface.

*The cmd object instance used in our project:*
Cmd.cmdloop() - Repeatedly issue a prompt, accept input, parse an initial prefix off the received input, and dispatch to action methods, passing them the remainder of the line as argument.
We defined our own functions for basic Linux commands. The syntax used was do_xxx().
The do_xxx method takes one extra parameter. This parameter corresponds to the part of the string entered by the user after the command name. The job of do_xxx is to parse this string and to find the command parameter's values.

**Code**

```
from cmd2 import Cmd
import os

class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

class Shell(Cmd):
        prompt = ">"

        def preloop(self):
                print 'A simple python Shell created by Saumya, Niti and Shikhar.\nUse help or
"?" to see list of commands.\nUse help (command name) for help related to the command'
                print 'Version 1.0'
                print'==============================================\n\n
\t\tWelcome to the shell\n\n=============================================='

        def do_lf(self, arg):
                'Lists files in the present directory.'
                for f in os.listdir('.'):
                        if os.path.isfile(f):
                                print f
                        else:
                                print bcolors.OKBLUE + f + bcolors.ENDC

        def do_exit(self, arg):
                'Exits the shell.'
                print 'Thanks for using this simple shell.'
                return True

        def do_crdr(self, arg):
                'Creates a directory.'
                os.mkdir(arg)
                print bcolors.BOLD + 'Directory named {0} created'.format(arg) + bcolors.ENDC

        def do_rmdr(self, arg):
                'Removes a directory.'
                os.rmdir(arg)
                print bcolors.FAIL +'Directory named {0} deleted'.format(arg) + bcolors.ENDC

        def do_rnm(self, arg):
```

```
		'Renames a directory or file.'
		params = arg.split(' ')
		#print params
		os.rename(params[0], params[1])
		print bcolors.OKGREEN + '{0} changed to {1}'.format(params[0],params[1]) +
bcolors.ENDC

	def do_cd(self, arg):
		'Changes the current working directory.'
		params = arg.split(' ')
		os.chdir(os.path.abspath(os.getcwd()) + '/' + params[0])
		print "Changed the current working directory to " + bcolors.BOLD + os.getcwd()
+ bcolors.ENDC

	def do_cwd(self, arg):
		'Shows the current working directory.'
		print os.getcwd()

	def do_nfile(self, arg):
		'Creates a new file.'
		os.mknod(arg)
		print bcolors.BOLD + "Created a file named {0}".format(arg) + bcolors.ENDC

	def do_rem(self, arg):
		'Removes file.'
		os.remove(arg)
		print bcolors.FAIL + "Deleted {0}".format(arg) + bcolors.ENDC

	def do_append(self, arg):
		'Appends to a file.\nappend file_name text_to_append'
		params = arg.split(' ')
		print params[0]
		stri = ""
		for x in params[1:]:
			stri += x + " "
		with open(params[0],"a") as file:
			file.write(stri)
```

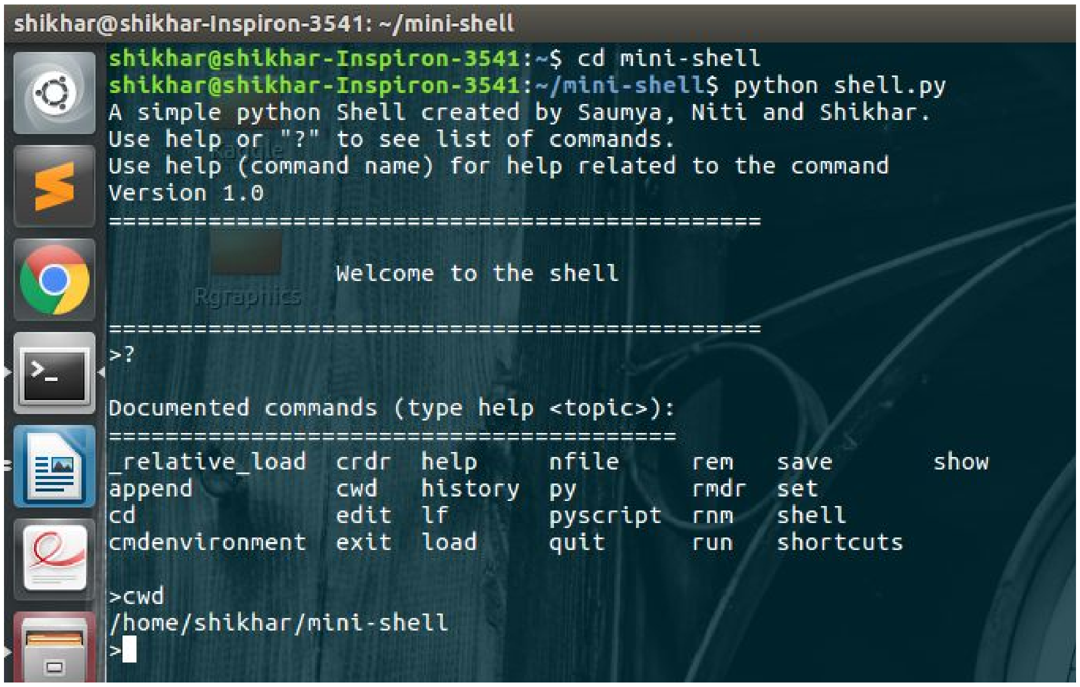
```
shell = Shell()
shell.cmdloop()
```

**Review of Literature**

The terminal is one of the most integral part of a Linux operating system. Linux sometimes has no UI. But it has command line interface. So exploring the commands of CLI was our main aim. Thus we came up with the idea of developing a mini shell in Python.

**Results**

We successfully created a mini shell using Python. Some of the functions we defined are functions for listing files, creating a file, creating a directory, removing a directory, appending to a file and the exit function. Here are screenshots of the output.



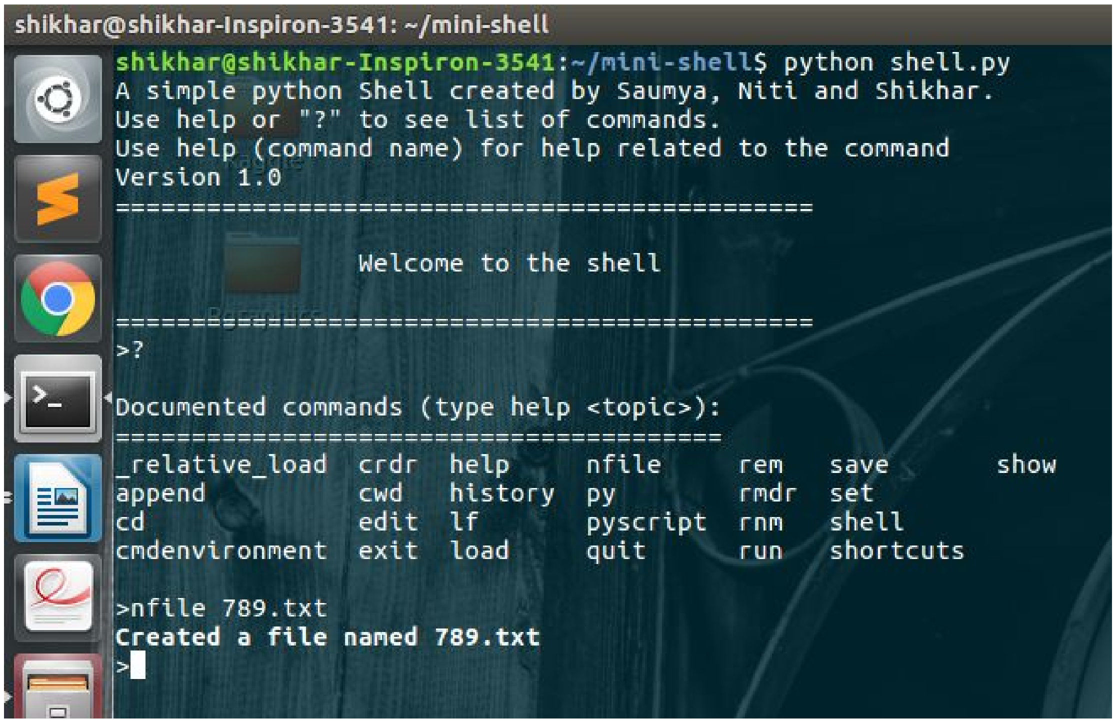
```
shikhar@shikhar-Inspiron-3541: ~/mini-shell
shikhar@shikhar-Inspiron-3541:~$ cd mini-shell
shikhar@shikhar-Inspiron-3541:~/mini-shell$ python shell.py
A simple python Shell created by Saumya, Niti and Shikhar.
Use help or "?" to see list of commands.
Use help (command name) for help related to the command
Version 1.0
==============================================

                Welcome to the shell

==============================================
>?

Documented commands (type help <topic>):
========================================
_relative_load  crdr  help     nfile     rem   save       show
append          cwd   history  py        rmdr  set
cd              edit  lf       pyscript  rnm   shell
cmdenvironment  exit  load     quit      run   shortcuts

>cwd
/home/shikhar/mini-shell
>
```



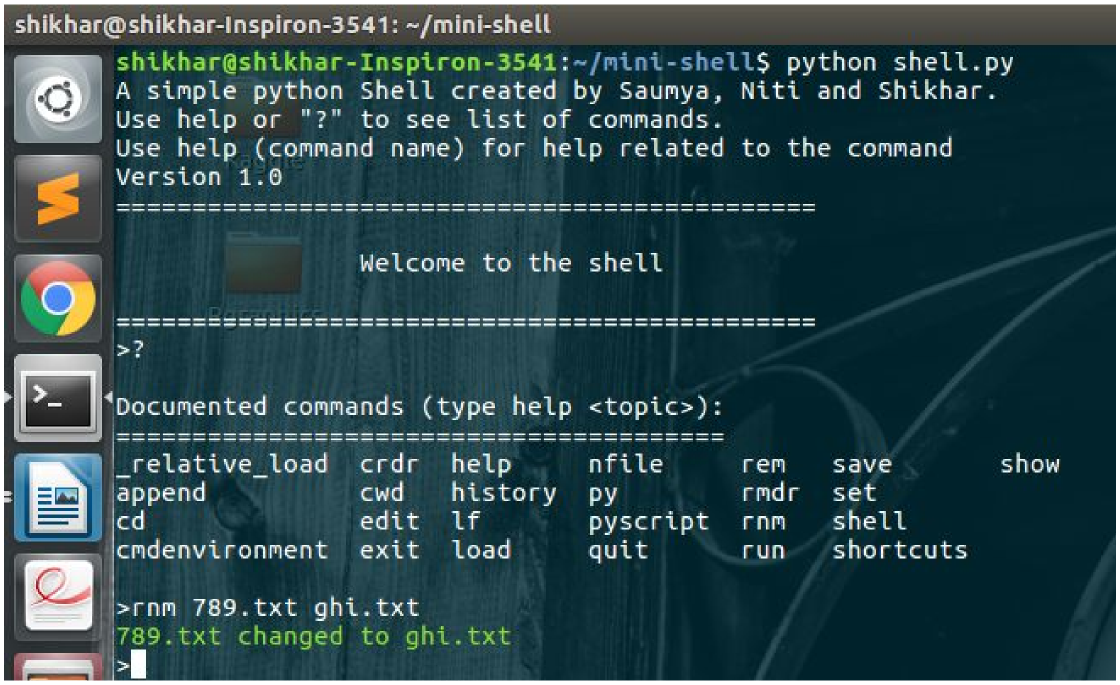
```
shikhar@shikhar-Inspiron-3541: ~/mini-shell
shikhar@shikhar-Inspiron-3541:~/mini-shell$ python shell.py
A simple python Shell created by Saumya, Niti and Shikhar.
Use help or "?" to see list of commands.
Use help (command name) for help related to the command
Version 1.0
==============================================

              Welcome to the shell

==============================================
>?

Documented commands (type help <topic>):
========================================
_relative_load  crdr  help     nfile     rem   save       show
append          cwd   history  py        rmdr  set
cd              edit  lf       pyscript  rnm   shell
cmdenvironment  exit  load     quit      run   shortcuts

>nfile 789.txt
Created a file named 789.txt
>
```



```
shikhar@shikhar-Inspiron-3541: ~/mini-shell
shikhar@shikhar-Inspiron-3541:~/mini-shell$ python shell.py
A simple python Shell created by Saumya, Niti and Shikhar.
Use help or "?" to see list of commands.
Use help (command name) for help related to the command
Version 1.0
==============================================

              Welcome to the shell

==============================================
>?

Documented commands (type help <topic>):
========================================
_relative_load  crdr  help     nfile     rem   save       show
append          cwd   history  py        rmdr  set
cd              edit  lf       pyscript  rnm   shell
cmdenvironment  exit  load     quit      run   shortcuts

>rnm 789.txt ghi.txt
789.txt changed to ghi.txt
>
```

**Conclusion**

Thus, on developing the shell, we understood the how the shell acts like an interface and provides a way for the user to communicate with the Operating System. We also understood the usage of Python and how the different modules such as os and cmd work.