L-05

CSCI - P436 | *Brandon Young*

Source Code

```
# Imports
import sys
import os
import fileinput
import argparse
import fileinput
import re

# Global variables
version = "1.1.2"

# Class
class Disk:

    def __init__(self):
       folder = 1

# Methods

## Parse the input flags using argparse
## Void input, returns a argument parser with data
def parseInput():

    # Create parser
    parser = argparse.ArgumentParser(add_help=False,description='Takes a disk and
prints its contents.')

    # Add args
    parser.add_argument('-v', '--version', action="version",version="shell.py
version --> " + version,help="Display current program version")
    parser.add_argument('-d','-dir','--directory',action="store_true",help="List
file contents of the input drive")
    parser.add_argument('-h','-H', '--help','-?', action='help',
default=argparse.SUPPRESS,help='Show this help message and exit.')
    parser.add_argument('-f','--file', type=str, help="Specify a formatted drive
file")

    # Check if no args were passed
    if len(sys.argv)==1 and sys.stdin.isatty():
        parser.print_help(sys.stderr)
        sys.exit(1)
    args = parser.parse_args()

    #Return parser with args
    return args

## Takes in an argParser with data and executes main program logic
def evalArgs(argsData):
    try:
        #print("test")
        # Predefine vars
        fileName = None
        diskFile = None
        # Check for stdin
        if not sys.stdin.isatty():
            diskFile = sys.stdin#fileinput.input()

        # Check for file input flag
        if argsData.file != None:
            fileName = argsData.file
            diskFile = open(fileName)

        # Check if either stdin or -f has been passed in
        if diskFile != None:
            data = collectRawDisk(diskFile)
            printRawDisk(data)

        # Checks for directory file
        if argsData.directory or not sys.stdin.isatty():
            crawlRawDisk(data)

    except:
        # Check if file does not exist
        if (fileName != None) and not(verifyFile(fileName)):
            print("Error: The disk is broken or does not exist, please provide a
correct drive file.")
        # Check if dir flag was set and no other args
        elif argsData.directory and fileName == None and diskFile == None and data
== []:
            print("Error: -dir is useless without a specified disk file")
        else:
            pass
        # Catch all other errors
        # else:
        #     print(Exception + "Error: you broke the thing... good job")

## Inputs a fileName and checks if that file exists in the CWD
## Returns true or false
def verifyFile(fileName):
    return os.path.exists(os.path.join(os.getcwd(), fileName))

## Void method inputs a list and prints the contents
def printRawDisk(disk):
    for line in disk:
        print(line)

## inputs empty list and file, and extracts contents from file to disk
## Returns filled list
```

```python
def collectRawDisk(file):
    data = []
    try:
        count = 0
        for line in file:
            if count > 1:
                data.append(str(line[3:64]))
            count += 1
        return data
    except:
        pass

## Void method takes in a list and recursively iterates through all the clusters
in the list
def crawlRawDisk(*args):
    if len(args) == 1:
        currentCluster = args[0][0]
        startOfVolume = int(currentCluster[5:7],16)
        volumeName = currentCluster[7:64]
        data = readClusterData(volumeName)
        print("Volume: " + data)
        crawlRawDisk(args[0],startOfVolume)
    if len(args) == 2:
        currentCluster = args[0][args[1]]
        nextCluster = int(currentCluster[1:3],16)
        filename = readClusterData(currentCluster[5:64])
        print("\tFile: " + filename)
        if nextCluster != 00:
            crawlRawDisk(args[0],nextCluster)

def hexToStr(hex):
    hex = re.sub("00.*","",hex)
    return bytearray.fromhex(hex).decode()

## Void method takes in a string and evaluates the sector
def readClusterData(clusterData):
    return hexToStr(clusterData)

# Main
parsed = parseInput()
evalArgs(parsed)
```

Running the code

1. `python shell.py`
2. `python shell.py -?`
3. `python shell.py -v`
4. `python shell.py -f`
5. `python shell.py -f diskname.txt`

6. `python shell.py -f formatted_disk_IUS.txt -dir`

7. `cat formatted_disk_IUS.txt | python shell.py -dir`

Outputs

1.
```
usage: shell.py [-v] [-d] [-h] [-f FILE]
Takes a disk and prints its contents.
optional arguments:
  -v, --version         Display current program version
  -d, -dir, --directory
                        List file contents of the input drive
  -h, -H, --help, -?    Show this help message and exit.
  -f FILE, --file FILE  Specify a formatted drive file
```

2.
```
 usage: shell.py [-v] [-d] [-h] [-f FILE]

 Takes a disk and prints its contents.

 optional arguments:
   -v, --version         Display current program version
   -d, -dir, --directory
                         List file contents of the input drive
   -h, -H, --help, -?    Show this help message and exit.
   -f FILE, --file FILE  Specify a formatted drive file
```

3.
```
 shell.py version --> 1.1.2
```

4.
```
 usage: shell.py [-v] [-d] [-h] [-f FILE]
 shell.py: error: argument -f/--file: expected one argument
```

5.
```
 Error: The disk is broken or does not exist, please provide a correct drive
file.
```

6.
```
 0010007495553000000000000000000000000000000000000000000000000
 1020000000000000000000000000000000000000000000000000000000000
 1030000000000000000000000000000000000000000000000000000000000
 1040000000000000000000000000000000000000000000000000000000000
 1050000000000000000000000000000000000000000000000000000000000
 1060000000000000000000000000000000000000000000000000000000000
 1090000000000000000000000000000000000000000000000000000000000
```

```
3080046524544004150504C45000000000000000000000000000000000000
31D004632004920414d204652454420464154000000000000000000000000
10A0000000000000000000000000000000000000000000000000000000000
10B0000000000000000000000000000000000000000000000000000000000
10C0000000000000000000000000000000000000000000000000000000000
10D0000000000000000000000000000000000000000000000000000000000
10E0000000000000000000000000000000000000000000000000000000000
10F0000000000000000000000000000000000000000000000000000000000
1100000000000000000000000000000000000000000000000000000000000
11F0000000000000000000000000000000000000000000000000000000000
1120000000000000000000000000000000000000000000000000000000000
1130000000000000000000000000000000000000000000000000000000000
1140000000000000000000000000000000000000000000000000000000000
1150000000000000000000000000000000000000000000000000000000000
1160000000000000000000000000000000000000000000000000000000000
1170000000000000000000000000000000000000000000000000000000000
1180000000000000000000000000000000000000000000000000000000000
1190000000000000000000000000000000000000000000000000000000000
11B0000000000000000000000000000000000000000000000000000000000
400504542424C455320414E442044494E4f2E000000000000000000000000
1000000000000000000000000000000000000000000000000000000000000
3000057494C4D41004452455353000000000000000000000000000000000
31C1A42414D42414D004D592046414D494C59204953204652454420414E44
1110000000000000000000000000000000000000000000000000000000000
11E0000000000000000000000000000000000000000000000000000000000
Volume: IUS
        File: FRED
        File: F2
        File: BAMBAM
        File: WILMA
```

7.

```
0010007495553000000000000000000000000000000000000000000000000
1020000000000000000000000000000000000000000000000000000000000
1030000000000000000000000000000000000000000000000000000000000
1040000000000000000000000000000000000000000000000000000000000
1050000000000000000000000000000000000000000000000000000000000
1060000000000000000000000000000000000000000000000000000000000
1090000000000000000000000000000000000000000000000000000000000
3080046524544004150504C45000000000000000000000000000000000000
31D004632004920414d204652454420464154000000000000000000000000
10A0000000000000000000000000000000000000000000000000000000000
10B0000000000000000000000000000000000000000000000000000000000
10C0000000000000000000000000000000000000000000000000000000000
10D0000000000000000000000000000000000000000000000000000000000
10E0000000000000000000000000000000000000000000000000000000000
10F0000000000000000000000000000000000000000000000000000000000
1100000000000000000000000000000000000000000000000000000000000
11F0000000000000000000000000000000000000000000000000000000000
1120000000000000000000000000000000000000000000000000000000000
1130000000000000000000000000000000000000000000000000000000000
1140000000000000000000000000000000000000000000000000000000000
```

```
1150000000000000000000000000000000000000000000000000000000000
1160000000000000000000000000000000000000000000000000000000000
1170000000000000000000000000000000000000000000000000000000000
1180000000000000000000000000000000000000000000000000000000000
1190000000000000000000000000000000000000000000000000000000000
11B0000000000000000000000000000000000000000000000000000000000
400504542424C455320414E442044494E4f2E000000000000000000000000
1000000000000000000000000000000000000000000000000000000000000
3000057494C4D41004452455353000000000000000000000000000000000
31C1A42414D42414D004D592046414D494C59204953204652454420414E44
1110000000000000000000000000000000000000000000000000000000000
11E0000000000000000000000000000000000000000000000000000000000
Volume: IUS
        File: FRED
        File: F2
        File: BAMBAM
        File: WILMA
```