Drive Scanning w/ Logging and Argparse Features

•••

Project 1 - Group 11

Rich Giannetti | Saahiil Meswaanii | Alex ONeill | Xi Zhou

CS632P - Python Programming | Prof. Sarbanes

Features & Dependencies

What does our script help solve?

Capabilities

The list_content.py script operates as a command line utility capable of providing:

- Disk and Storage Information
- Folder and File Details
- Usage Details by File Type
- System-Agnostic
- Detailed Logging

Required Dependencies

- os
- shutil
- string
- time
- argparse
- logging
- textwrap
- Pathlib

| Argparse | Logging | Functions/Logic |

The argparse module allows us to enter arguments at the command line.

There are 4 functional modes, 2 printing modes, and a help mode.

- -d | --drv (drive information)
- -l | --fld (folder information)
- -f | --fil (file information)
- -t | --typ (file type information)

- -v | --verbose (verbose printing)
- -q | --quiet (quiet printing) *default

- -h | --help (help information)

| Argparse | Logging | Functions/Logic |

```python
# SECTION: MAIN()
def main():
    """Main argparse logic and function calls"""
    parser = argparse.ArgumentParser(
        add_help=False,
        formatter_class=argparse.RawDescriptionHelpFormatter,
        description=textwrap.dedent('''\
                    Here is the description of this app
                    *-------------------------------------------------------*
                        A Python app that will read the contents of any computer,
                        and produce an output based on the input provided.
                        The output will be written into a logging/logger log file,
                        for a view on demand and verification.
                        All the lines of the log file are accompanied by time stamps,
                        along with level message.
                    *-------------------------------------------------------*
                    '''))

    parser.add_argument('-h', '--help', action='help', default=argparse.SUPPRESS,
                        help='displays this screen and provides information on what options are available for use')
    # NOTE: -V / -Q MUTUALLY EXCLUSIVE GROUPING
    me_group = parser.add_mutually_exclusive_group()
    me_group.add_argument('-v', '--verbose', action='store_true')
    me_group.add_argument('-q', '--quiet', action='store_true')
    # NOTE: INPUT PARAMETER ARGUMENTS
    parser.add_argument('-d', '--drv',
                        help='lists drive details for the drive letter that is entered, eg: -d C:',
                        nargs='?', const=string.ascii_uppercase)
    parser.add_argument('-l', '--fld',
                        help='lists folder details for all folders in the path that is entered, eg: -l C:',
                        nargs='?')
    parser.add_argument('-f', '--fil',
                        help='lists file details for all files in the path that is entered, eg: -f C:\\path\\file',
                        nargs='?', const='all')
    parser.add_argument('-t', '--typ', help='lists file type details for the "file type" that is entered, eg: -t exe',
                        nargs='?', const='everything')
    args = parser.parse_args()
```

Help information / description

Mutually exclusive group for -v / -q

Function arguments

| Argparse | Logging | Functions/Logic |

```
485     # NOTE: VERBOSE MODE
486     if args.verbose:
487         if not any([args.drv, args.fld, args.fil, args.typ]):
488             print('No arguments entered -- please enter an argument')
489         else:
490             # NOTE: VERBOSE LOGGING AND CONSOLE PRINTING
491             console = logging.StreamHandler()
492             console.setLevel(logging.DEBUG)
493             formatter = logging.Formatter('%(asctime)s - %(levelname)s >>> %(message)s')
494             console.setFormatter(formatter)
495             logging.getLogger().addHandler(console)
496
497             if args.drv:...
508             if args.fld:...
511             if args.fil:...
525             if args.typ:...
```

Check for no parameters

-verbose additional logging config

Actions based on argument input (function calls)

| Argparse | Logging | Functions/Logic |

```python
# SECTION: LOGGING CONFIGURATION
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s - %(levelname)s >>> %(message)s',
                    datefmt='%m-%d %H:%M',
                    filename='info.log'
                    )
```

Initial logging config

```python
        else:
            # NOTE: VERBOSE LOGGING AND CONSOLE PRINTING
            console = logging.StreamHandler()
            console.setLevel(logging.DEBUG)
            formatter = logging.Formatter('%(asctime)s - %(levelname)s >>> %(message)s')
            console.setFormatter(formatter)
            logging.getLogger().addHandler(console)
```

-verbose additional logging config

Logging messages embedded throughout code

```python
        except FileNotFoundError as fnf:
            logging.warning('{} not found {}'.format(path, fnf))
        except OSError as ose:
            logging.critical('Cannot access {} .Probably a permissions error {}'.format(path, ose))
```

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -d drv FUNCTIONS
def list_drives(drive: str) -> None:
    """Identifies OS version then calls proper -d function"""
    if os.name == 'posix':
        list_drives_mac()
    else:
        list_drives_win(drive)
```

Initial drive function filtering

```python
def list_drives_win(drive: str = string.ascii_uppercase) -> None:
    """Returns drive information including directory and file count for Windows OS"""
    logging.info('#' * 50)
    logging.info('-drv: This will take a while please wait.')
    drive = drive

    progress = ProgressBar()
    for each_drive in drive:
        if os.path.exists(each_drive + ":\\"):
            path = each_drive + ":\\"
            usage = shutil.disk_usage(path)
            dirnum = 0
            filenum = 0
            logging.info('#' * 50)
            logging.info('In Drive {}'.format(each_drive))
            logging.info('Drives total size: {}'.format(sizeConvert(usage.total)))
            logging.info('Drives used size: {}'.format(sizeConvert(usage.used)))
            logging.info('Drives free size: {}'.format(sizeConvert(usage.free)))
            logging.debug('counting files and directories now please wait.')
            for root, dirs, files in os.walk(path):
                try:
                    for file in files:
                        filepath = os.path.join(root, file)
                        if os.path.isfile(filepath):
                            filenum += 1
                            progress.current_file = filenum
                            progress()
                    for dir in dirs:
                        dirpath = os.path.join(root, dir)
                        if os.path.isdir(dirpath):
                            dirnum += 1
                except FileNotFoundError as fnf:
                    logging.warning('{} not found {}'.format(path, fnf))
                except OSError as ose:
                    logging.critical('Cannot access {} .Probably a permissions error {}'.format(path, ose))
            logging.info('The total number of directories is: {}'.format(dirnum))
            logging.info('The total number of files is: {}'.format(filenum))
```

Function details to scan drives and collect details

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -l fld FUNCTION
def get_folder_info(dir: str) -> None:
    """Returns information about the file path that is entered"""
    logging.info('#' * 50)
    logging.info('-fld')
    if os.path.exists(dir):
        total_size = sum(f.stat().st_size for f in Path(dir).glob('**/*') if f.is_file())
        for item in os.listdir(dir):
            itempath = os.path.join(dir, item)
            try:
                if os.path.isfile(itempath):
                    continue
                elif os.path.isdir(itempath):
                    filenum = len(os.listdir(itempath))
                    root_directory = Path(itempath)
                    filesize = sum(f.stat().st_size for f in root_directory.glob('**/*') if f.is_file())
                    logging.info('folder: {:20}, number of files: {:4}, folder size: {},'.format(itempath, filenum,
                                                                                                 sizeConvert(filesize)))
            except FileNotFoundError as fnf:
                logging.warning('{} not found {}'.format(itempath, fnf))
            except OSError as ose:
                logging.critical('Cannot access {} .Probably a permissions error {}'.format(itempath, ose))
        t_s_format = sizeConvert(total_size)
        logging.info('Total Storage of all files: ' + t_s_format)
    else:
        logging.warning('{} is not a valid path'.format(dir))
```

Function to gather folder details based on file path argument

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -f file FUNCTIONS
def get_all_files(fil: str) -> None:
    """Identifies OS version then calls proper -f function"""
    if os.name == 'posix':
        get_all_files_mac(fil)
    else:
        get_all_files_win(fil)
```

Initial file details function filtering

```python
    # NOTE: GETTING ALL FILES NAMES
    if fil == 'all':
        progress = ProgressBar()
        filenum = 0
        for each_drive in drive:
            if os.path.exists(drive + each_drive):
                dir = drive + each_drive
                try:
                    for root, dirs, files in os.walk(dir):
                        for file in files:
                            filepath = os.path.join(root, file)
                            if os.path.isfile(filepath):
                                filename = file
                                filetype = os.path.splitext(file)[1]
                                filesize = sizeConvert(os.stat(filepath).st_size)
                                filetime = time.strftime('%Y-%m-%d %H:%M:%S',
                                                         time.localtime(os.stat(filepath).st_ctime))
                                filenum += 1
                                progress.current_file = filenum
                                progress()
                                logging.info(
                                    'filename: {:30}, filetype: {:7} filesize: {:10}, time stamp: {}'.format(filename,
                                                                                                             filetype,
                                                                                                             filesize,
                                                                                                             filetime))
                except FileNotFoundError as fnf:
                    logging.warning('{} not found {}'.format(dir, fnf))
                except OSError as ose:
                    logging.critical('Cannot access {} .Probably a permissions error {}'.format(dir, ose))
```

Gathering file details with NO arguments

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -f file FUNCTIONS
def get_all_files(fil: str) -> None:
    """Identifies OS version then calls proper -f function"""
    if os.name == 'posix':
        get_all_files_mac(fil)
    else:
        get_all_files_win(fil)
```

Initial file details function filtering

Gathering file details WITH an argument

```python
# NOTE: SEARCHING FOR SPECIFIC FILE NAME
else:
    progress = ProgressBar()
    filenum = 0
    found_file = False
    for each_drive in drive:
        if os.path.exists(drive + each_drive):
            dir = drive + each_drive
            try:
                for root, dirs, files in os.walk(dir):
                    for file in files:
                        filepath = os.path.join(root, file)
                        if os.path.isfile(filepath):
                            filename = file
                            filetype = os.path.splitext(file)[1]
                            filesize = sizeConvert(os.stat(filepath).st_size)
                            filetime = time.strftime('%Y-%m-%d %H:%M:%S',
                                                     time.localtime(os.stat(filepath).st_ctime))
                            filenum += 1
                            progress.current_file = filenum
                            progress()
                            if filename.lower() == fil.lower():
                                found_file = True
                                logging.info('File found at path: {}'.format(filepath))
                                logging.info(
                                    'filename: {:30}, filetype: {:7} filesize: {:10}, time stamp: {}'.format(filename,
                                                                                                             filetype,
                                                                                                             filesize,
                                                                                                             filetime))

            except FileNotFoundError as fnf:
                logging.warning('{} not found {}'.format(dir, fnf))
            except OSError as ose:
                logging.critical('Cannot access {} .Probably a permissions error {}'.format(dir, ose))
    if not found_file:
        logging.warning('Unable to find file: {}'.format(fil))
```

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -t typ FUNCTIONS
def get_all_types(typ: str) -> None:
    """Identifies OS version then calls proper -t functio
    if os.name == 'posix':
        everything_mac(typ)
    else:
        everything_win(typ)
```

Initial file type function filtering

Gathering all file type details

```python
def everything_win(typ: str) -> None:
    """Returns information about the file type that is entered or all types for Windows OS"""
    logging.info('#' * 50)
    logging.info('-typ: This will take a while please wait.')
    drive = string.ascii_uppercase
    type_dicts = {}
    progress = ProgressBar()
    filenum = 0
    for each_drive in drive:
        if os.path.exists(each_drive + ":\\"):
            path = each_drive + ":\\"
            try:
                for root, dirs, files in os.walk(path):
                    for file in files:
                        filepath = os.path.join(root, file)
                        type = os.path.splitext(file)[-1].lower()
                        size = os.stat(filepath).st_size
                        filenum += 1
                        progress.current_file = filenum
                        progress()
                        if type in type_dicts.keys():
                            type_dicts[type]['file_count'] += 1
                            type_dicts[type]['total_size'] = type_dicts[type]['total_size'] + size
                        else:
                            type_dicts[type] = {'file_count': 1, 'total_size': size}
            except FileNotFoundError as fnf:
                logging.warning('{} not found {}'.format(path, fnf))
            except OSError as ose:
                logging.critical('Cannot access {} .Probably a permissions error {}'.format(path, ose))
```

| Argparse | Logging | Functions/Logic |

```python
# SECTION: -t typ FUNCTIONS
def get_all_types(typ: str) -> None:
    """Identifies OS version then calls proper -t function"""
    if os.name == 'posix':
        everything_mac(typ)
    else:
        everything_win(typ)
```

Initial file type function filtering

Print file type details based on arguments

```python
    # NOTE: PRINTING ALL RESULTS
    if typ == 'everything':
        for ft in sorted_tups:
            file_count = dict(ft[1])['file_count']
            total_size = sizeConvert(dict(ft[1])['total_size'])
            logging.info('Type: {:10}, File-Count: {}, Total-Size: {}'.format(ft[0], file_count, total_size))
    # NOTE: PRINTING MATCHING FILE TYPES
    else:
        if typ[0] != '.':
            typ = '.' + typ
        for ft in sorted_tups:
            if ft[0] == typ:
                file_count = dict(ft[1])['file_count']
                total_size = sizeConvert(dict(ft[1])['total_size'])
                logging.info('Type: {:10}, File-Count: {}, Total-Size: {}'.format(ft[0], file_count, total_size))
```

Wrap up & Demo

Challenges

- OS compatibility
- Identifying and meeting project requirements causing code re-work
- Collaborative coding

For Next Time

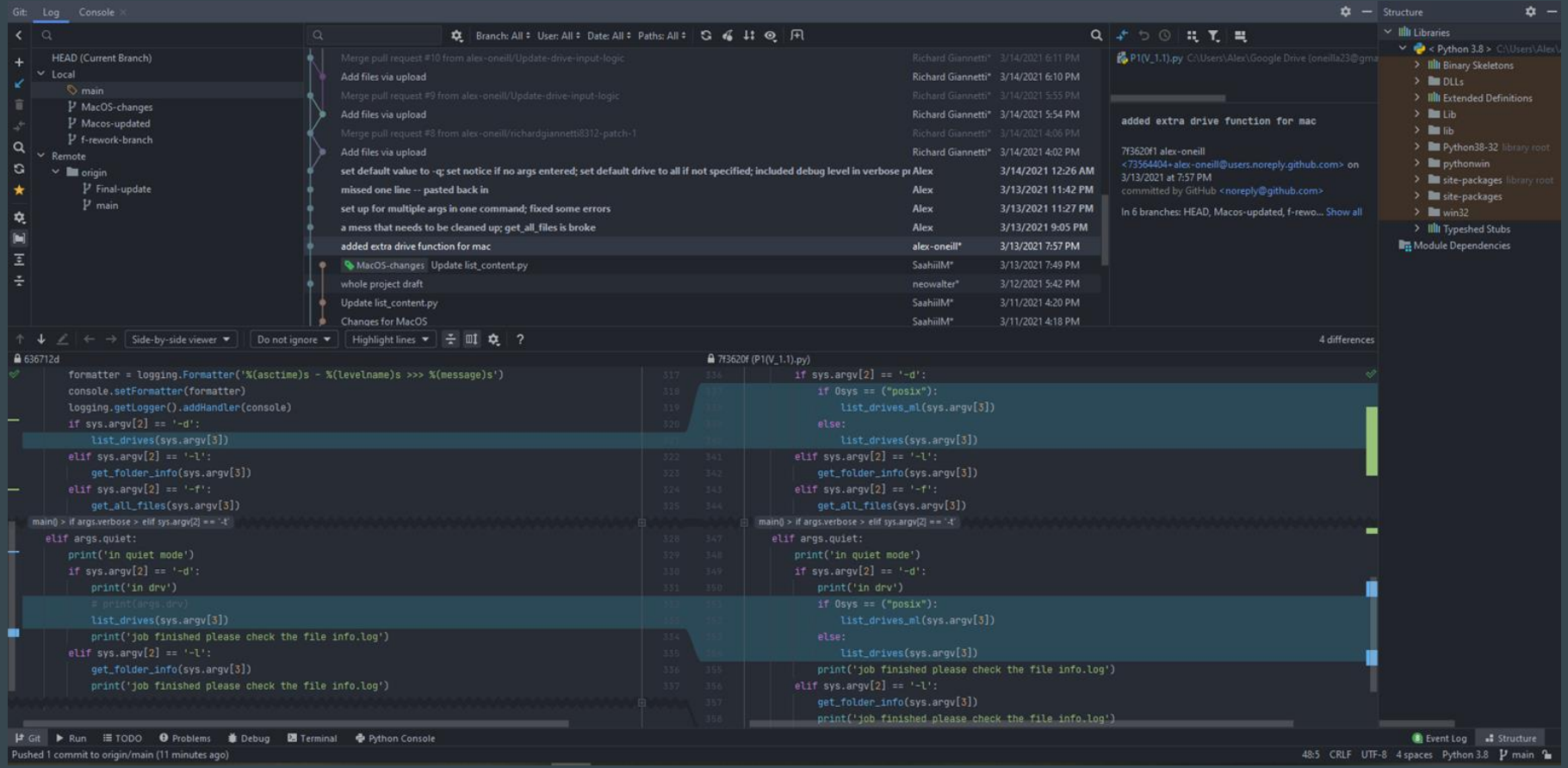
- Team kick-off meeting with a focus on identifying project requirements
- Translate these requirements into pseudo-code with everyone present
- Continue to use Slack and Github to collaborate
- Be more deliberate in allocating areas of work

Wrap up & Demo

Questions?

Demo Results

Git Demo


Git: Log Console
HEAD (Current Branch)
Local
main
MacOS-changes
Macos-updated
f-rework-branch
Remote
origin
Final-update
main
Branch: All User: All Date: All Paths: All
Merge pull request #10 from alex-oneill/Update-drive-input-logic Richard Giannetti* 3/14/2021 6:11 PM
Add files via upload Richard Giannetti* 3/14/2021 6:10 PM
Merge pull request #9 from alex-oneill/Update-drive-input-logic Richard Giannetti* 3/14/2021 5:55 PM
Add files via upload Richard Giannetti* 3/14/2021 5:54 PM
Merge pull request #8 from alex-oneill/richardgiannetti8312-patch-1 Richard Giannetti* 3/14/2021 4:06 PM
Add files via upload Richard Giannetti* 3/14/2021 4:02 PM
set default value to -q; set notice if no args entered; set default drive to all if not specified; included debug level in verbose pr Alex 3/14/2021 12:26 AM
missed one line -- pasted back in Alex 3/13/2021 11:42 PM
set up for multiple args in one command; fixed some errors Alex 3/13/2021 11:27 PM
a mess that needs to be cleaned up; get_all_files is broke Alex 3/13/2021 9:05 PM
added extra drive function for mac alex-oneill* 3/13/2021 7:57 PM
MacOS-changes Update list_content.py SaahiilM* 3/13/2021 7:49 PM
whole project draft neowalter* 3/12/2021 5:42 PM
Update list_content.py SaahiilM* 3/11/2021 4:20 PM
Changes for MacOS SaahiilM* 3/11/2021 4:18 PM
P1(V_1.1).py C:\Users\Alex\Google Drive (oneilla23@gma
added extra drive function for mac
7f3620f1 alex-oneill
<73564404+alex-oneill@users.noreply.github.com> on 3/13/2021 at 7:57 PM
committed by GitHub <noreply@github.com>
In 6 branches: HEAD, Macos-updated, f-rewo... Show all
Structure
Libraries
< Python 3.8 > C:\Users\Alex\
Binary Skeletons
DLLs
Extended Definitions
Lib
lib
Python38-32 library root
pythonwin
site-packages library root
site-packages
win32
Typeshed Stubs
Module Dependencies
Side-by-side viewer Do not ignore Highlight lines
4 differences
636712d
7f3620f (P1(V_1.1).py)
formatter = logging.Formatter('%(asctime)s - %(levelname)s >>> %(message)s')
console.setFormatter(formatter)
logging.getLogger().addHandler(console)
if sys.argv[2] == '-d':
list_drives(sys.argv[3])
elif sys.argv[2] == '-l':
get_folder_info(sys.argv[3])
elif sys.argv[2] == '-f':
get_all_files(sys.argv[3])
main() > if args.verbose > elif sys.argv[2] == '-t'
elif args.quiet:
print('in quiet mode')
if sys.argv[2] == '-d':
print('in drv')
# print(args.drv)
list_drives(sys.argv[3])
print('job finished please check the file info.log')
elif sys.argv[2] == '-l':
get_folder_info(sys.argv[3])
print('job finished please check the file info.log')
if sys.argv[2] == '-d':
if Osys == ("posix"):
list_drives_ml(sys.argv[3])
else:
list_drives(sys.argv[3])
elif sys.argv[2] == '-l':
get_folder_info(sys.argv[3])
elif sys.argv[2] == '-f':
get_all_files(sys.argv[3])
main() > if args.verbose > elif sys.argv[2] == '-t'
elif args.quiet:
print('in quiet mode')
if sys.argv[2] == '-d':
print('in drv')
if Osys == ("posix"):
list_drives_ml(sys.argv[3])
else:
list_drives(sys.argv[3])
print('job finished please check the file info.log')
elif sys.argv[2] == '-l':
get_folder_info(sys.argv[3])
print('job finished please check the file info.log')
Git Run TODO Problems Debug Terminal Python Console
Event Log Structure
Pushed 1 commit to origin/main (11 minutes ago)
48:5 CRLF UTF-8 4 spaces Python 3.8 main