COSC 2P05 — A2 — Through heavy air a demon shrieks

I crashed before the birth of Christ
Pterodactyls swarming
You died in 1989
Wanna get back to that morning in May

Don't you just hate it when you get stranded in time? Finding out where you are is the first step in getting back. Clearly, when hiding in a cave, the first step to reorienting is to inspect file systems. That's just common sense.

This assignment focuses on two things:

1. Writing a program with a modal UI — something that's waiting on the user before performing the next action, and can possibly have a single type of dialog at a time
    - This actually describes most/all programs you've written thus far, but this time it's by design
2. Learning libraries (modules) independently, as one needs to do when using a new language
    - Particularly, the common uses of the `os` module, as well as probably expanding on your understanding of strings

The assignment will be completed in **Python 2**. The marker will likely be testing on sandcastle, so it's advised to test on *at least* a Linux box.

- Note that windows uses different folder separators (`\` instead of `/`), so it's up to you to ensure the marker can somehow run your submission if you ignore this advice
    - (Hint: telling the marker to “just switch computers” doesn't qualify)

The task is simple: create a (very) simple shell environment. The user can enter commands, and through those commands navigate through a directory structure and perform very basic operations on the contents. Limited functionality is included for finding files based on filenames, inspecting file contents, note-taking, and minimal task automation.

You'll want to maintain a 'current working directory' at all times. Though several module commands can just infer the path, some operations won't be as easy to implement unless you keep an up-to-date string handy.

Note that terminal sizes matters. Several operations will require that output be trimmed to fit the available display. Trying to retrieve the terminal size programmatically can be tedious, so just assume the user is using the traditional terminal size: **80 columns by 24 rows**. (The number of rows will only matter for displaying notes; see below)

(I need to fill some space so everything else lines up, so…

uh…

how y'all doing? I hope you like this one. It's a bit weird at first, but then it should be pretty fun.

That enough space?

Yeh? Good!)

Requirements

To consolidate, you need:

- A working program (duh), with your name, username, and std number commented at the top
- Python docstrings for every method you write, as well as commenting where appropriate
- Proper separation of behaviours, procedural abstraction, etc.
- Reasonable design, multiple selection, etc.
- *Reasonably* stable, and resilient to user shenanigans
- The following commands:
  - `pwd` — prints the *current working directory*
  - `cd (path)` — changes the current directory
  - `ls [pattern]` — displays the contents of the current directory
    - If a *pattern* is supplied, only show file/folder names that contain that text
    - Lines are truncated if they'd exceed 80 characters
    - Folders are enclosed in [] (so truncation happens at 78)
    - The right-ends of names matter more than the left-ends
  - `md5 (filename)` — displays an md5 hash of the contents of a file
  - `tt (filename)` — infers the OS convention of the specified file
    - If the file only contains LFs, it's Unix; if it has CR and LF, it's Windows; otherwise it's binary
  - `find (pattern)` — recursively follows folders off current working directory to find files
    - Effectively a fancier version of `ls` *`pattern`*
  - `search` — recursively follows folders off working directory to find files with matching *content*
  - `head (filename) [# of lines]` — preview of a file's contents
    - Just displays the first however many lines of a file; default of 5 lines
    - Contents are cut off at 80 characters per line
  - `ln` — display names of all *notes* files
    - Notes are kept in *`filename`*`.notenote`, but the user only refers to the *`filename`* part
  - `tn (note name)` — create a new note, dictated from the user, to save in a `.notenote` file
  - `vn (note name)` — displays the contents of a note
  - `run (script name)` — uses a file to specify the commands to execute
    - Scripts are kept in *`filename`*`.scriptscript`, but the user just uses the *`filename`* part
    - To be clear: these are scripts for our own shell; *not python scripts*
  - `help [command]`
    - When calling `help`, displays the list of all commands
    - When calling `help` *`command`*, shows how to use that command
  - `exit` — Exits the program

Tips and Suggestions

- You don't need to worry about exceptions yet, but if that comes up outside of very isolated examples, you're probably doing something wrong
  - e.g. it's okay if pressing *ctrl+d* triggers an `EOFError`, or *ctrl+c* triggers a `KeyboardInterrupt`
- Having multiple possible commands qualifies as *multiple selection*. You know what that means… right?
- Slices can have overestimated boundaries and be fine, though variadic *args are also an option
- Since the current working directory is ubiquitous throughout the program, yes, it's acceptable to have the current path be a `global` variable
- The `os` and `md5` modules are the only ones you'll definitely need to `import`, though you're welcome to use more so long as they come prepackaged with python

- Some of the goodies for the `os` module are in `os.path` (e.g. `os.path.isdir`)
- Don't forget you have a built-in `help` command in python
- If you're reading text from a file and are getting *newline* characters you don't want at the end of each line, the `str`ing's `strip()` function trims off leading/trailing whitespace
- You shouldn't need it, but if you need to get the path of your 'home' folder for some reason, you can just use `os.path.expanduser('~')`

Grading

This assignment is out of **15**. Half-points are possible, but less than that is not.

Points are earned as follows:

Current folder /1
- pwd and cd

File inspection /1
- md5 and tt

ls (*without* pattern) /1
- Includes truncation for terminal width, and [bracketing] for folders

Filename matching /1
- [pattern] feature of ls, and find command

Content searching /1
- search command (needs to recurse folders)

head /1
- Including always truncating for width, and displaying (up to) requested number of lines
- If no number of lines requested, defaults to 5

Notes /3
- 0.5 for displaying notes (`ln`); 0.5 for properly filtering out `.notenote` portion of filenames
- 1 for tn; 1 for vn

run a script /1
- Should work with or without an `exit`; should be safe to put *anything* after an `exit`

help /1
- 0.5 for displaying commands; 0.5 for being able to access help for each command

Proper documentation /1
- Include both docstrings *and* commenting

Clean and efficient code /1
- Actually write it like someone will be reading it

Additional: /2
- There are several things that couldn't fit into one of the above, including style, coding and commenting conventions, etc.
- This is basically a catch-all for being thorough and reasonable in your designs

Warning: additional points can also be deducted!
For example, clearly when starting the script, it should… start. Put in a *main* or something, and then call it! However, deductions outside of the above will only be made for things that are **objectively wrong**.

Submission

You'll be submitting through Sakai.
Bundle everything up into a single **`.zip`** file, and submit that.
Note: `.zip`.
Not `.rar`.
Not .7z.
This is not 1993.

Ideally, your submission will be simple. If not, the burden is on you to ensure that the marker can compile and run it. It *must* be able to be run on sandcastle.

Sample Execution

Consider the following brief example.

```
>help
---
head
run
help
vn
cd
pwd
find
md5
search
ln
tt
tn
ls
exit
---
>pwd
/home/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter
>ls
[A1]
[A2]
[code]
[slides]
1337wisdom.txt
2P05outline.odt
2P05outline.pdf
Concepts of Programming Languages 11th Ed.pdf
huh.txt
letthesunshinein.txt
realdata.bin
trivial.scriptscript
>ls es
[slides]
Concepts of Programming Languages 11th Ed.pdf
letthesunshinein.txt
>ls des
[slides]
```

```
>find
Invalid pattern
>find r.java
/home/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/A1/A1/Survivor.java
e/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/A1/Zombies/src/Plinker.java
me/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/A1/Zombies/src/Sniper.java
/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/A1/Zombies/src/Survivor.java
me/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/A1/Zombies/src/Flamer.java
>search
Invalid pattern
>search grobble
ount/WorkSandrock/2P05-2019-2020-winter/code/05/slide35/grobble/Mooltipass.class
oxwell/xMount/WorkSandrock/2P05-2019-2020-winter/code/05/slide35/Mooltipass.java
/home/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter/code/02/procedural.py
>md5
File not found
>md5 2P05outline.pdf
fd1027d703a857abaf19b44068a16fbc
>tt
File not found
>tt realdata.bin
binary
>tt 1337wisdom.txt
Unix
>tt letthesunshinein.txt
Windows
>head
File not found
>help head
head (filename) [# of lines]  ---  Displays beginning of file

     # of lines defaults to 5 unless otherwise specified.

>head 1337wisdom.txt
Y'know, they call'em fingers, but I never see'em fing.

...
Whoa! There they go!
>head 1337wisdom.txt 1
Y'know, they call'em fingers, but I never see'em fing.
>head 1337wisdom.txt 50
Y'know, they call'em fingers, but I never see'em fing.

...
Whoa! There they go!
>head 1337wisdom.txt 0
>head 1337wisdom.txt 1 2 3 4 5
Y'know, they call'em fingers, but I never see'em fing.
>head letthesunshinein.txt 6
Grobble groobles?

Yup. totleslly real data.

Such data, very neat. also doing the thing for the stuff with the dealies and wh
yup.
>ls .notenote
```

```
>ln
>tn
Nah.
>tn demo
Enter as many lines as needed. Start a line with . to stop.
Enter as many lines as needed. Start a line with . to stop.
This is a line of text.
We don't need to worry about terminal dimensions here, because we aren't policin
g whether the user probably meant to do.
.... I think? BTW, that only took down 3 periods. The first pair acted as an esc
aped period.
.this line won't be registered, as the leading period ends dictation
>ls .notenote
demo.notenote
>ln
demo
>vn demo.notenote
File not found
>vn demo
Enter as many lines as needed. Start a line with . to stop.
This is a line of text.
We don't need to worry about terminal dimensions here, because we aren't policin
g whether the user probably meant to do.
... I think? BTW, that only took down 3 periods. The first pair acted as an esca
ped period.
>run
File not found
>run trivial
/home/efoxwell/xMount/WorkSandrock/2P05-2019-2020-winter
>exit
```

Of course, all the text will actually be the same colour. I just colour-coded input vs output to make it easier to see what's going on.

The contents of `trivial.scriptscript` are simply:

```
pwd
exit
pwd
doesn't really matter what I put here, eh?
```

The only major thing not demonstrated above is the 'page-breaks' for displaying notes. That's because (with no line-truncation for over 80 characters) there's no way to know how many lines will *always* fit. Just make sure to have it wait for user input after ~20 lines, before resuming.
e.g. I had it display a `:::>` to indicate to press *Enter* to continue.