Advanced Python Programming

Assignment 1

'Command Line Hero'

Each APP assignment consists of 5 tasks. This one isn't exception. Be patient and attentive (!), some of them may not be as straightforward as you wish them to see. Remember, this is an **advanced programming course**! So the things can be very tricky at first! Nevertheless, since our kindness has no boundaries you can always contact us for any appropriate help.

To submit your assignment:

- Upload your assignment to MS Teams as f.lastname.zip with all the code (or written answers) packed within.
- Commit your solution to the GitHub repository of your team.

Your code will probably not be ~~compiled~~ executed on real machine (but brain). However, if it is the case, put the Python's version at the top of your code (`python -V`). Just be honest and grades bless you in the sky. Regarding the grades.. Since we all live in the binary world each task is **2 points max** (10 for the whole assignment). One point – "well..that's ok", 2 – "well done!", 0 – "where is the task n btw?". The last note: **python3** is preferred.

DEADLINE next Monday, 3:30pm (just before the lecture)

That's all and we are ready to start..

1. **'Shell creator'**.

   Write a program that runs "shell" commands **within** your program and immediately throws what's on its stdout; line-by-line. Once command is done, it waits for another command to be typed. (hint: `cd` will not change your working directory in the most simple implementation). Example:

   ```
   ./myshell
   myshell: ls
   file1.txt
   file2.txt

   myshell: python
   Python 2.7.15+ (default, Nov 27 2018, 23:36:35)
   [GCC 7.3.0] on linux2
   >>> exit()

   myshell: exit or ^D
   Goodbye!
   # Now we're back in normal shell
   ```

2. **'Path screwer'**.

   Extend your shell so that `cd` works as expected (hint: `cd` might not support any flags or options, just "keep it simple, stupid"). Current working directory should appear in your shell prompt where each dirname is shortened to one character (or two, if dir's name start with .). Example:

```
./myshell
myshell [/h/t]: <- shows where the script was executed
myshell [/h/t]: cd /usr/lib
myshell [/u/l]: cd /home/timfayz/.config
myshell [/h/t/.c]: exit or ^D
Goodbye!
```

3. **'Hard-boiled sysadmin'**.
   Extend your shell so that it logs all the actions and useful information back into `myshell.log`. That includes: timestamp, command name, its arguments, lines of output, pid and exit code. Example:

```
./myshell
myshell [/h/t]: echo "TEST"
TEST
myshell [/h/t]: cat myshell.log
[2018-11-18 09:32:36] cmd: echo, args: TEST, stdout: 1, pid: 25160, exit: 0
myshell [/h/t]: bla
/bin/sh: 1: bla: not found
myshell [/h/t]: cat myshell.log
[2018-11-18 09:34:55] cmd: bla, args: , stdout: 1, pid: 25364, exit: 32512
[2018-11-18 09:33:33] cmd: cat, args: myshell.log, stdout: 0, pid: 25163, exit: 0
[2018-11-18 09:32:36] cmd: echo, args: TEST, stdout: 1, pid: 25160, exit: 0
myshell [/h/t/.c]: exit or ^D
Goodbye!
```

4. **'CCleaner'**. (first C from CLI)
   Again, extend your shell so that if a command encounters any error it wouldn't put it back into stdout. Instead, pipe the error stream into `myshell.stderr` file and keep stdout clean. Example:

```
./myshell
myshell [/h/t]: cd nowhere
myshell [/h/t]: cat myshell.stderr
/bin/sh: 1: cd: can't cd to nowhere
myshell [/h/t/.c]: exit or ^D
Goodbye!
```

5. **'Pedant coder'**
   Whatever you've written or googled on the web, rewrite your shell using `subprocess` module which is (obviously) part of the standard library and supposed to replace several older modules for process management. We encourage you to read the following official pages:
   1. https://docs.python.org/3/library/subprocess.html
   2. https://docs.python.org/2/library/subprocess.html