BASH SCRIPTING

WHAT IS A BASH SCRIPT?

- Basic: listing of commands to execute with Bash shell
- Bash is also a programming language - scripts can contain:
  - control structures -- loops, conditionals
  - variables
  - functions
  - arguments (parameters)
  - arrays

WHO USES THEM AND WHY?

- Make life easier: automate or simplify tasks run regularly
- Efficiency - create script to perform repetitive tasks
- Examples:
  - sysadmins needing to check status and running the same commands on a regular basis
  - script to build and deploy personal website

CREATING AND RUNNING

- Open, edit, save file with program/list of commands
- Convention for bash is to use `.sh` extension
- Change the permissions to make executable
  - `chmod u+x filename`
  - `chmod +x filename`
- `./ filename`

POUND-BANG

- aka shebang, hashbang
- first line tells the kernel what to program to use to run the script
- Q. Why bother adding to script?
- A. Portability - users don't need to know what to use to call script
- easily run bash scripts from other shells

POUND-BANG (CONT.)

- Examples with bash:
  - `#! /bin/bash`
  - `#! /usr/bin/env bash`
- Can also use with others like Python:
  - `#! /usr/bin/env python3`
- `/usr/bin/env bash` vs `/bin/bash`
  - `env` uses whatever version of the executable comes first in `$PATH`
  - `env` - users can have different behavior

COMMENTS

- # begins a comment from there until end of line
- Exception:
  - pound-bang/shebang on first line of script

ARGUMENTS

- aka positional parameters
- reference by `${n}` - where `n` is the position
- `$0` - expands to command used to call program
- `$1`, `$2`, etc. are 1st, 2nd, etc. arguments on the command line
- need to use braces for numbers with more than 1 digit, i.e. `${10}`

MINILAB 08