ShellTools and Scripting

define a variable in shell:

variable_name=value (don’t add spaces)

eg: var=abc

echo $var (o/p: abc)

echo “the variable is $var” (o/p: the variable is abc”

echo ‘the variable is $var’ (o/p: the variable is $var, this is because only “ “ work the trick)

Defining and running a function:

- code mcd.sh #creates a mcd.sh and opens it in VS code studio
- inside mcd.sh:

```
mcd(){
    mkdir -p "$1"    #$1 is a reserved keyword, to access arg
    cd "$1"
}
```

- source mcd.sh #to load the script into the shell for use
- mcd test #creates a ‘test’ directory and changes directory to ‘test’, as defined in the script
- $0 - name of the script
- $1 to $9 - arguments
- $? - gives error from the previous command
- $_ - gives the last argument from the previous command

  eg: mkdir test

  cd $_ $cd into test
- rmdir <dir name> - removes empty directory only, cannot be used for populated directory
- rm -r <dir name> - removes the directory along with any other directories and files inside it.

- !! - copies the previous command and pastes at the “!!” position

  eg: pip install pandas

  sudo !! #runs sudo pip install pandas

  !! numpy # runs sudo pip install pandas numpy

- OR operator:

  || is used as an OR operator in shell
  eg: false || echo “process failed” #o/p: process failed

  eg: true || echo “won’t print this”

- AND operator:

  &&
  eg: true && echo “prints this coz true”

  eg: false && echo “wont print coz false”

- semicolon:

  eg: false ; echo “demo semicolon”
  concatenates two different commands in one line

- Globbing:
  - ls *.sh - lists all file/dir names that end with .sh
  - ls project? - lists all files/dir that match “project” + 1 variable character
    eg: project1 project 34 projectX → project1 and projectX match coz “project” + 1 variable char
- Typing shortcut:
  - say we want to write a command like : convert image.png image.jpg
  - we can use a shortcut like: convert image.{png,jpg} #this will turn into convert image.png image.jpg
  - eg: touch foo{,0,1,2} ⇒ touch foo foo0 foo1 foo2
-

```
echo "starting program at $(date)"
echo "running program $0 with $# arguments with pid $$"
for file in "$@"; do
```

```
    grep foobar "$file" > /dev/null 2>/dev/null
    if [[ "$?" -ne 0 ]]; then
        echo "File $file does not have any foobar, adding one
        echo "# foobar" >> "$file"
    fi
done
```

- Running a python script from shell
- code pyscript.sh

```
#!/usr/local/bin/env python #this is a shebang cmd, lets sh
ell know to use pyton
import sys
for argin reversed(sys.argv[1:]):
print(arg)
```

- DEBUG:
  - shellcheck <filename>
- Good Tools to install to work better in shell:
  - tldr <cmd> - gives examples of how the command can be run from the internet
  -

technoidentity@TI:~$ tldr cd
warning: 1 page(s) found for other platforms:

1. windows (tldr --platform windows cd)

cd

Change the current working directory.
More information:
https://manned.org/cd.

Go to the specified directory:

```
cd path/to/directory
```

Go up to the parent of the current directory:

```
cd ..
```

Go to the home directory of the current user:

```
cd
```

Go to the home directory of the specified user:

```
cd ~username
```

Go to the previously chosen directory:

```
cd -
```

Go to the root directory:

```
cd /
```

- ripgrep