Commands to cover

- **Cat & tac**

```
>>cat sample_data.txt
```

- **sort**

```
>>cat sample_data.txt
>> sort sample_data.txt
>> sort -k 3 sample_data.txt
```

- **Uniq**

```
>> uniq sample_data_dupli.txt
```

- **Grep**

```
>>egrep -n 'mellon' mysampledata.txt
        3:Mark watermellons 12
        11:Oliver rockmellons 2
```

• Bash script

```
-------------------------------------------------------------
#!/bin/bash
directory="./BashScripting"

# bash check if directory exists
if [ -d $directory ]; then
	echo "Directory exists"
else
	echo "Directory does not exists"
fi

-----------------------------------------------------------------
```

Anything you can run on the command line you may place into a script and they will behave exactly the same. Vice versa, anything you can put into a script, you may run on the command line and again it will perform exactly the same.

Shebang (#!)

identifies which interpreter should be used

```
#!/bin/sh
#!/bin/csh
```

#!/usr/bin/env python – Execute using Python by looking up the path to the Python interpreter automatically via env

```
>> which bash
```

Linux is an extensionless system, common to put **.sh** to make them easy to identify

Bash script to use variables

```
-----------------------------------------------------------------------------------------------------
#!/bin/bash
# A simple demonstration of variables
#

name=Amit
echo Hello $name

# you can use environment users as well
#
echo Hello $USER

operations=`cat $1 | wc -l`
echo The number of operations in the file $1 is $operations
```

Backup script

```
#!/bin/bash
# Backs up project directory
#

date=`date +%F`
mkdir ~/projectbackups/$1_$date
cp -R ~/projects/$1 ~/projectbackups/$1_$date
echo Backup of $1 completed
```

```
>> ./projectbackup.sh msd_cache
```

- **Makefile**

- **Timestamp**

- **Top**

- **gdb**

  Break points
  Stepping
  Assembly code (objdump)

Regex (regular expressions)

. (dot) - a single character.
? - the preceding character matches 0 or 1 times only.
* - the preceding character matches 0 or more times.
[agd] - the character is one of those included within the square brackets.

+ - the preceding character matches 1 or more times.
{n} - the preceding character matches exactly n times.
{n,m} - the preceding character matches at least n times and not more than m times.
[^agd] - the character is not one of those included within the square brackets.
[c-f] - the dash within the square brackets operates as a range. In this case it means either the letters c, d, e or f.
() - allows us to group several characters to behave as one.
| (pipe symbol) - the logical OR operation.
^ matches the beginning of the line.
$ - matches the end of the line.

```
>>egrep '2$' mysampledata.txt
        Mark watermellons 12
        Susy oranges 12
        Oliver rockmellons 2
```

hw3
homework2

??[0-9]
hw[0-9]

hw[^0-9] --> by mistake saved by different name

homework*

*.v *.sv
*.{v,sv}

```
ls -lh /home/*/.bashrc
ls -lh /home/*/.bash_history

>>ls *.???
    barry.txt blah.txt example.png frog.png



>> file /home/ryan/*
          bin: directory
          Documents: directory
          frog.png: PNG image data
```