Bash

by Deborah R. Fowler

KEY CONCEPTS

- ✓ variables
- ✓ truth statements
- ✓ looping
- ✓ functions
- ✓ I/O
- ✓ lists
- ✓ classes/objects
- ✓ OOP

What is bash? “Bourne-Again Shell”

- is a shell

Great – what is a shell?

- Command language interpreter

“Bourne-Again Shell”

Stephen Bourne – author of direct ancestor of Unix shell sh

Other shells you may hear of: sh, ksh (Korn shell), csh (C shell)

Bash is the default shell

Shells used:

Interactively – type from keyboard (you are already doing this)

Non-interactively – a script

Shells like any high-level language you have

Variables

Flow control contructs (if, for, while)

Functions

Shells offer easy job control, command line duties

Sites you may find useful:

https://help.ubuntu.com/community/Beginners/BashScripting

https://www.udemy.com/bash-scripting-for-beginners
(just the free previews are useful)

So if you are in the linux shell (bash) type:

```
pwd
date
ls
cal
```

Let’s put that into a script:

```
#!/bin/bash
pwd
date
ls
cal
```

Try running it by typing ./nameOfFile

It will fail – the reason is you don’t have **permission** to execute the code

chmod +x nameOfFile

In-class

Go ahead and try this

```
#!/bin/bash
pwd
date
ls
cal
```

The power of bash is that you can do many things within the script

Let’s run thru a few examples …

```
#!/bin/bash
echo “Hello world”
```

```
#!/bin/bash

# arguments can be used
echo “My first name is $1”
echo “My last name is $2”
```

Same this is a file called test, chmod +x test

```
./test  Kermit Frog
```

```
#!/bin/bash

exec < $1
while read LINE
do
        echo $LINE
done
```

./test filename

prints the lines of the file

```
#!/bin/bash

exec < $1
let count=0
while read LINE
do
        ((count=count+1))
done
echo “Number of lines: $count”
```

./test filename

count the lines

We can continue expanding on this – it is another syntax to get used to but the key concepts of variables, selection, looping, functions are the same

```
#!/bin/bash

echo “hello, $USER”
echo “Here are your files in directory, $PWD”
ls
```

Variables:

x=“hello” NOTE no spaces on either side of =

refer to it as

$x

If statements:

```
#!/bin/bash
x=3
y=4
if [  $x  -lt  $y  ]
then
        echo “It is true”
fi
```

For loops:

```
#!/bin/bash
for x in red green blue
do
        echo $x
done
```

while loops:

```
#!/bin/bash
x=0
while [  $x  -lt  20  ]
do
        echo $x
        ((x=x+1))
done
```

functions:

```
#!/bin/bash
function kermit()
{
        echo “Same concepts, different syntax”
}

kermit
```

Functions with parameters (they are positional):

```
#!/bin/bash
function kermit()
{
        echo “Same concepts, different syntax with $1”
}

kermit 10
```

Bash scripting can be useful – however it does not support OOP – so back to python