THE UNIVERSITY OF WESTERN AUSTRALIA

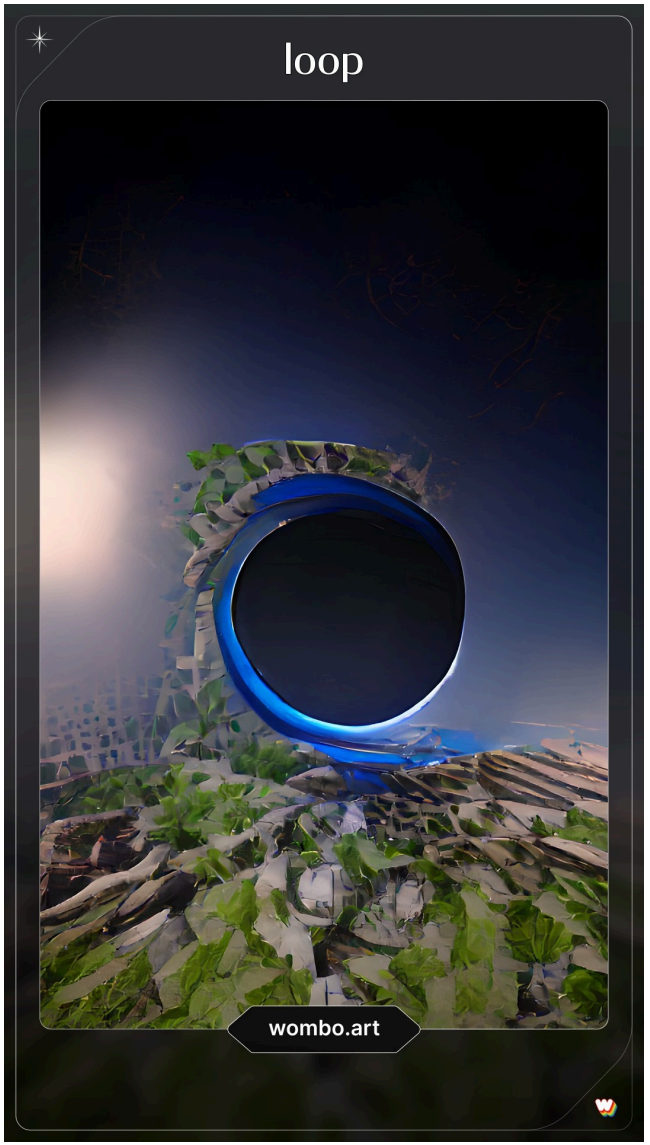
Break, while, shift

Lecture 8

Michael J Wise

Loop and a half?

- Given a set of things to be gone through, e.g. files in the current directory, a `for` loop starts at first item, then second, etc., till the end.
- What if you found what you were seeking part way through the loops?



All against all

```
#This program loops through all the files
# in the directory comparing all vs all
for i in *
do
   for j in *
   do
      echo Compare $i $j
      diff $i $j  #lists differences
      echo
   done
done
```

- Note the indenting (for style and readability).

All against all mechanism

- Say the list of files is: `a b c d`
- First `i=a`, and `j=a`, then `b`, then `c` …
- The `i=b` and `j=a`, ……

- There is flaw with the code. What is it?

All against all

```
#!/usr/bin/env bash
# Assume current directory
for i in *
do
  for j in $(ls -r . ) #from other end
  do
    if [[ $j == $i ]] #seen it before from
    then                # from this point
      break
    fi
    echo Compare $i $j
    diff $i $j
     echo
  done
done
```

While

- The `for` loop is fine if you want to loop through a set of things that is known up front, one item at a time, e.g. files in a directory:

```
for i in *
```

- What if you need to work through a set of things taking a variable number each time round, or that loops until some event occur (event loop)?

Demo

- In the computational biology work that I do, I run jobs that can take hours on a super-computer at Pawsey, sometimes days.
- I know how long the process *should* take, but what if it falls over part way through?
- I want to write a small script that monitors a process every few seconds. I supply the monitored process' PID.
- First need to mimic a long running process using a script with an infinite loop and `sleep` command (to not waste computing resources).

Shift

- `shift` shifts the arguments to the current script one place to the left (reducing argument count)

```
#!/usr/bin/env bash
echo "$# : $*"
shift
echo  "$# : $*”
```

```
% testshift 1 two "two 1/2" 2.99994
```

4 : 1 two two 1/2 2.99994

3 : two two 1/2 2.99994

```
1 two "two 1/2" 2.99994
```

```
two "two 1/2" 2.99994
```

What is shift good for?

- Many UNIX command line arguments are of the form:
  - -<*tag*> <*value*>
- That is, options can have alternatives. For example:

```
sort –u –t , -k 4nr file.csv
```

Sort on fourth column, removing duplicates, contents of csv file. The command script would

- *get the first argument;* `shift`
- *get the first (was second) and then look ahead to qualifier;* `shift`; `shift`
- *get the first (was) third and then look ahead to qualifier;* `shift`; `shift`
- *get the first (was fourth) argument*

Demo

- Want to generalise count_occurences to have optional `-topN` argument (if absent, assume all), .e.g

```
count_occurences -topN 100
Alice_in_Wonderland.txt
```

```
count_occurences Alice_in_Wonderland.txt
```

But what about

```
count_occurences -top 100
Alice_in_Wonderland.txt
```

Or

```
count_occurences -topN 100
Alice_in_Wonderland.tx
```