For loop: implementing a repeated task

For loops are useful if you want to repeat a block of Unix code N times.

For loops will go sequentially through a list of values and repeat execution of Unix commands for each item in the list. The list of values is defined as a series of items, separated by spaces.

The general format of the for loop in bash is:

```
for var in value1 value2 value3 … valueN
do
   block of Unix code  #the repeated task
done
```

Example:

```
for i in Hello ciao Hola
do
  echo $i
done
```

For loop can use braces

```
for i in {1..10} #use braces to generate a sequence of inter numbers
do
   echo $((i*2))
done
```

```
for n in {a..n} #use braces to generate a sequence of letters
 do
   echo $n > file$n
 done
```

Hardcoding example – we take points off in HWs and Exams

```
for i in 1 2 3 4 5 6 7 8 9 10
for i in a b c d e f g h i l m n
```

For loop can use wildcards

In a for loop, you can define a list of values to be matched by wildcards

```
for i in *.bash #list of all files in current directory ending with .bash
do
   echo $i
    head -3 $i
done
```

```
for i in data-temp/temp* #list of all files starting with temp, which
are within data-temp directory
do
    echo $i
     head -3 $i
done
```

For relative pathnames, a script starts from the directory where you run the script

Example

Make this for loop in a script called **numbers-example.bash**

```
for i in {1..10}
do
   echo $((i*2)) >> numbers.txt
done
```

- If you run this script multiple times, do the contents of numbers.txt change?
  Write your answer in a comment line, and explain why
- **Add one line of code (DO NOT MODIFY THE PROVIDED CODE)** so that if you run the script multiple times the contents of the file numbers.txt will not change.

Passing a list of arguments to a script from the command line

In your script

```
$1 will be expanded in the 1st  argument
$2 will be expanded in the 2nd  argument
and so on
$@ will be expanded in all the arguments
```

You would run the script by providing the arguments that will be passed to the script as variables

```
. script.bash arg1 arg2 arg3
```

In a script called pass-arg.bash write:

```
var1=$1
var2=$2
var_all=$@
echo this is argument1 $var1
echo this is argument2 $var2
echo all arguments in $var_all
```

Now run the script – try to change the argument values

```
. pass-arg.bash 1 2 4 5 6 7
```

Passing a list of arguments from command line

Passing a list of values (arguments) to a for loop from command line

Modify the script loop-arg.bash to:

```
for i in $@
do
    echo $i
done
```

And you can easily modify the input arguments form the command line

```
. loop-arg.bash 1 2 3 4 5
. loop-arg.bash {1..10}
. loop-arg.bash 1 2 three
```

Activity - Summing loop

You can sum numbers with a loop

Make a script called **A7-sum-example.bash**:

```
s=0      #variable s is initialized to 0
for i in {1..50}
do
  s=$((s+$i))#variable s is incremented by the value stored in i
done
echo The final sum is: $s
```

ex1. Edit **A7-sum-example.bash** to make a summing script that takes input arguments from the command line. The script should only output the total sum. When you run the script, you should input integer numbers from 1 to 1000.

Write in a comment line, at the end of the script, how you would run the script.

Activity - Counting loop

A common type of program **loop** is one that is controlled by an integer that counts up from an initial value to an upper limit. Such a **loop** is called a **counting loop**

Make a script called **A7-count-example.bash**:

```
c=0  #variable c is initialized to 0
for i in 1 3 5 7
do
  c=$((c+1))#variable c in incremented by 1 in each iteration
done
echo The final count is: $c
```

ex2. Edit the script **A7-count-example.bash** to count the arguments passed from the command line and print only the total count.   If you run the script like this:

```
count-example.bash 3 7 Hello
```

The output should be

```
The arguments are 3
```

Activity

ex3 Make a script called **A7-loop-letters.bash,** and in it write one for loop to create file-a.txt, which contains the letter A, file-b.txt, which contains the letter B, and so on until file-z.txt with the letter Z. Part of the code is reported here. Your task is to complete the code. In the repeated task of the loop use pipeline and the tr command

```
for n in {a..z}
do

done
```

ex4 Make a script called **A7-loop-dir.bash**, and in it write one for loop to create the following directories within your current working directory, and in each directory make a file. In directory red, make a file called red.dat, in directory blue, a file called blue.dat, and so on.
You would need to lines of code.

red
blue
yellow
green
purple

Hint: Open a terminal and make a directory called green, and within directory green make a file called green.dat. Then generalize the two lines of code to be a repeated task in your for loop by using the loop variable.

Activity

ex5 Start from home directory. Within your home directory you should have a directory called data-temp. In your home directory make a script called **A7-loop-var.bash.**

Make this variable that stores the state codes

```
var='AZ CA CO NV NM UT'
```

Make a for loop over the state codes by using variable var, and for each state print to screen the minimum temperature value and the state code.

The repeated task should be **one pipeline of commands.**

Use the data file **temp-clean1.dat**. Use relative path to the data file. Your script should be in home directory, and data-temp directory should be within your home. The output should be:

```
AZ 1.480707645
CA 1.138502927
CO 1.280255682
NV 1.297301136
NM 1.341059745
UT 1.385726584
```

Hint: At the terminal, write one line of code for, example AZ, and then generalize the code in your for loop to be a repeated task by using the loop variable

Activity

ex6 Optional - Make a script called **A7-mybak.bash** and in it make one for loop to make a backup of all .bash files that you have in your current working directory. Include a comment line in the back-up files.

For example, for single file named loop1.bash you would do:

```
cp loop1.bash loop1.bash.bak
echo \#Backup of loop1.bash >> loop-1.bash.bak
```

Activity

ex7 Optional - In your home directory make a script called A7-loop-dat.bash.

Directory data-temp should be within your home directory

In the script write one for loop that goes over all files starting with temp and ending with .dat, which are within data-temp. Use relative path to the files.

The repeated task is to display ONLY the name of each file and the minimum temperature anomaly of the state of Colorado (CO).

The output should be

```
temp-clean.dat
501 CO 1.280255682
temp-clean1.dat
501 CO 1.280255682
temp.dat
501 CO 1.280255682
```

Hint: In the repeated task, ***use one long pipeline of commands*** *to:*

- extract lines that do not contain the #

- change the field separator to be the same for all the files (from : to space)

- extract lines containing CO and continue with text processing.

To print only the filename, think about which field separator to use in awk or cut

Submit to Gradescope A7 the following scripts:

• A7-sum-example.bash	0.5 point
• A7-count-example.bash	0.5 point
• A7-loop-letters.bash	1 point
• A7-loop-dir.bash	0.5 point
• A7-loop-var.bash	0.5 point

- **A7-loop-dat.bash - optional**
- **A7-mybak.bash - optional**