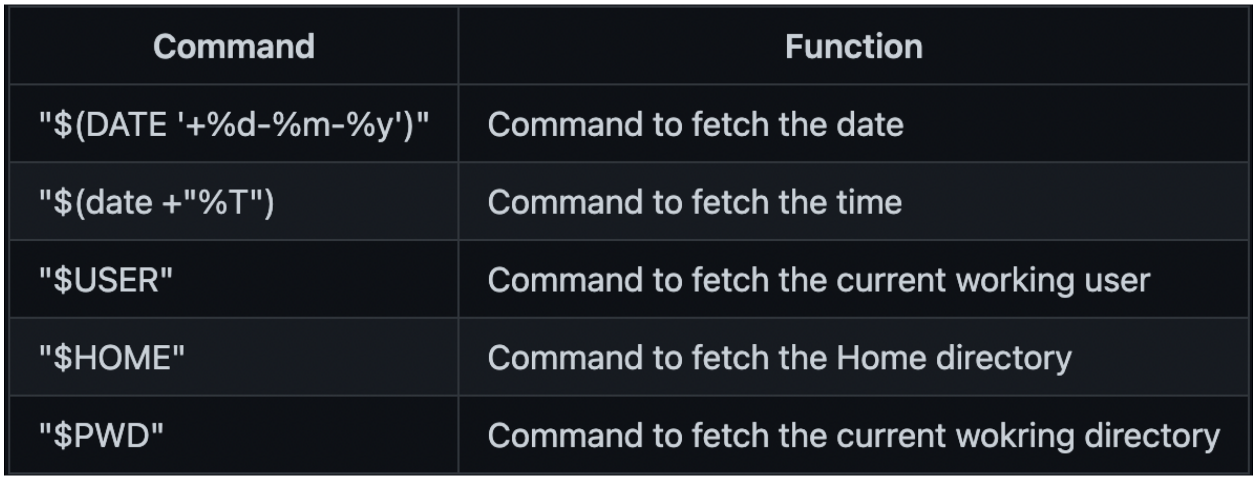
# PEER LEARNING- COMMAND LINE ASSIGNMENT

**QUESTION 1:**Write a bash script to get the current date, time, username, home directory and current working directory.

Commands used by me



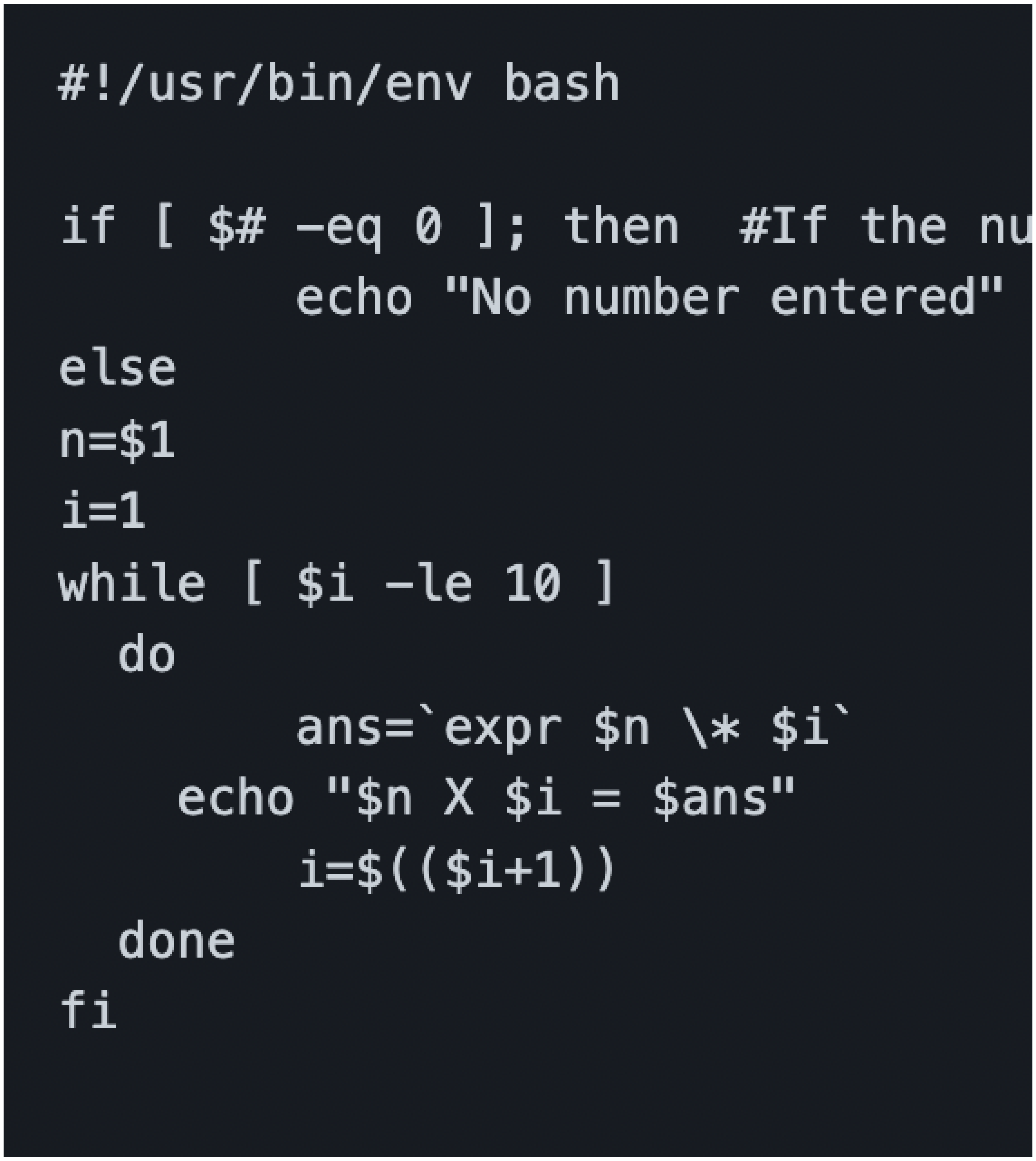
| Command | Function |
|---|---|
| "$(DATE '+%d-%m-%y')" | Command to fetch the date |
| "$(date +"%T") | Command to fetch the time |
| "$USER" | Command to fetch the current working user |
| "$HOME" | Command to fetch the Home directory |
| "$PWD" | Command to fetch the current wokring directory |

However my peers have used other alternatives that should also do the trick:

To get the current working user we could use $USER or also use $whoami command.

**QUESTION 2:**Write a bash script (name Table.sh) to print the Table of a number by using a while loop. It should support the following requirements.

- The script should accept the input from the command line.
- If you don't input any data, then display an error message to execute the script correctly.

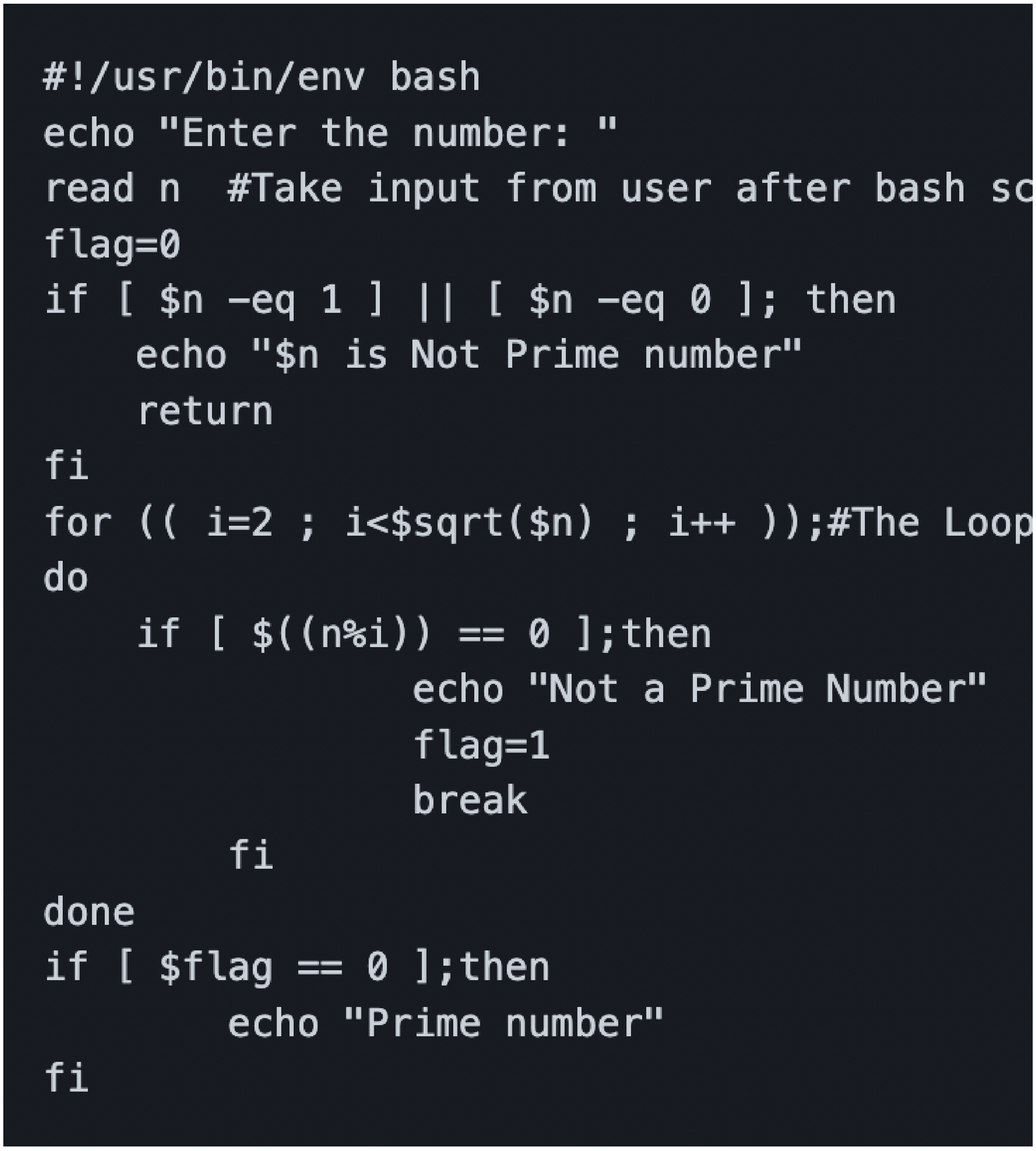


```
#!/usr/bin/env bash

if [ $# -eq 0 ]; then  #If the nu
        echo "No number entered"
else
n=$1
i=1
while [ $i -le 10 ]
  do
        ans=`expr $n \* $i`
    echo "$n X $i = $ans"
        i=$(($i+1))
  done
fi
```

This was done by taking input from the command line by using the '$1' operator and then to check for if no numbers were entered in the command line we can use '$# ==0' condition and then we can loop from 1 to 10 to print the table which is pretty similar to all the peers.

**Question - 3**
Write a Function in bash script to check if the number is prime or not? It should support the following requirement. - The script should accept the input from the User.



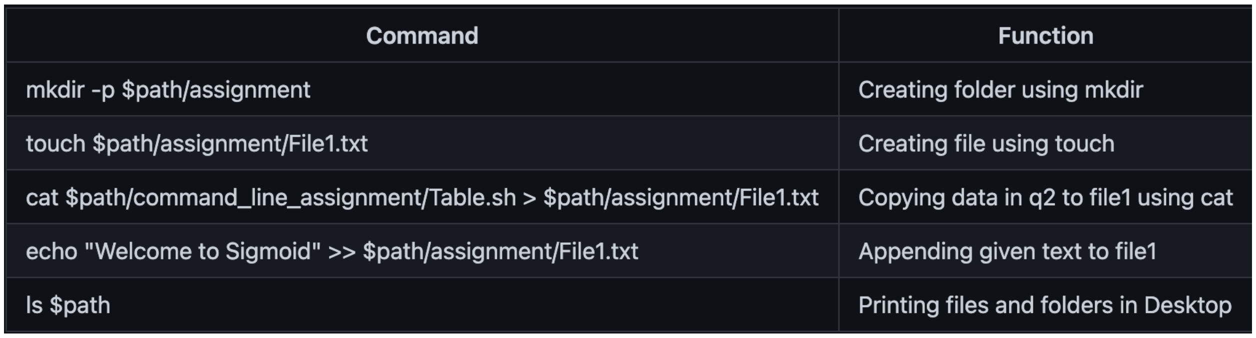
```
#!/usr/bin/env bash
echo "Enter the number: "
read n  #Take input from user after bash sc
flag=0
if [ $n -eq 1 ] || [ $n -eq 0 ]; then
    echo "$n is Not Prime number"
    return
fi
for (( i=2 ; i<$sqrt($n) ; i++ ));#The Loop
do
    if [ $((n%i)) == 0 ];then
                echo "Not a Prime Number"
                flag=1
                break
        fi
done
if [ $flag == 0 ];then
        echo "Prime number"
fi
```

So I first take input of the number to tested whether it is a prime number or not then , check if the number entered is less than 2 then it is not a prime number if the number is greater than 2 then i loop from 2 to the square root of  the number to see if there are any numbers that divide the number, if the number gets divided it is not a prime number or else it is a prime number. The rest of my peers have done the same although Shikhar and Arin have looped till n which would result in the same number however would increase the time complexity.

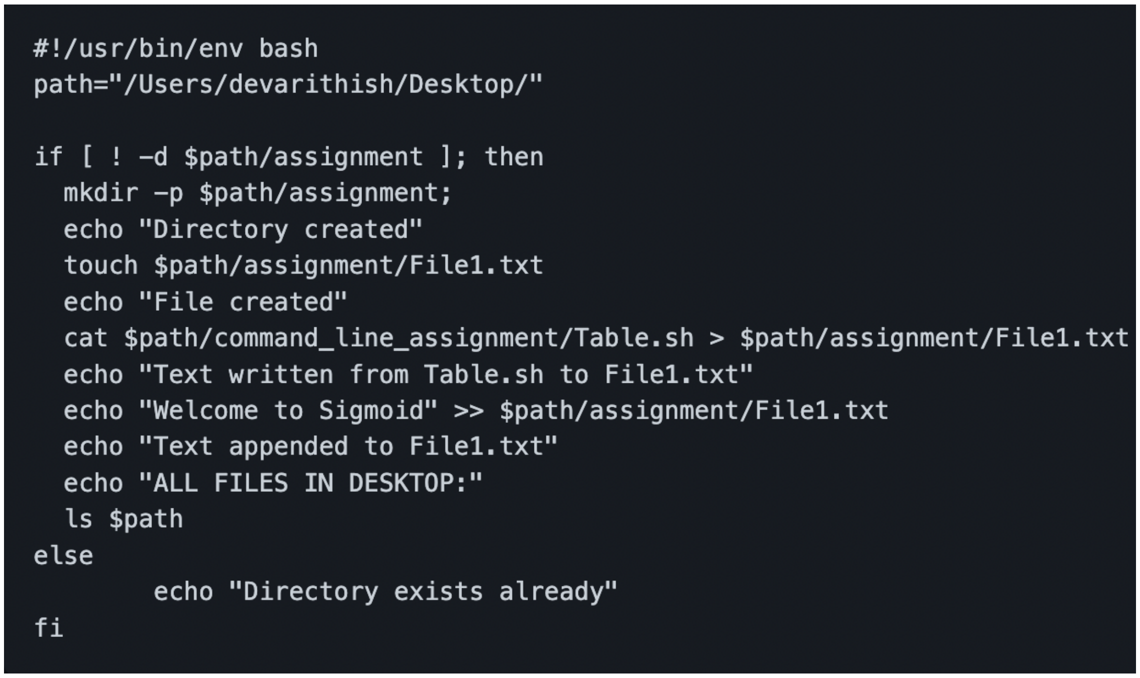
**Question 4:**
Create a bash script that supports the following requirement.

Create a folder ‘Assignment’.
Create a file ‘File1.txt’ inside ‘Assignment’ Folder.
Copy all the content of Table.sh(2nd script) in ‘File1.txt’ without using ‘cp’ and ‘mv’ command.
Append the text Welcome to Sigmoid’ to the ‘File1.txt’ file.
List all the directories and files present inside Desktop Folder.

Commands used by me are listed below which are the same as my peers.



| Command | Function |
|---|---|
| mkdir -p $path/assignment | Creating folder using mkdir |
| touch $path/assignment/File1.txt | Creating file using touch |
| cat $path/command_line_assignment/Table.sh > $path/assignment/File1.txt | Copying data in q2 to file1 using cat |
| echo "Welcome to Sigmoid" >> $path/assignment/File1.txt | Appending given text to file1 |
| ls $path | Printing files and folders in Desktop |

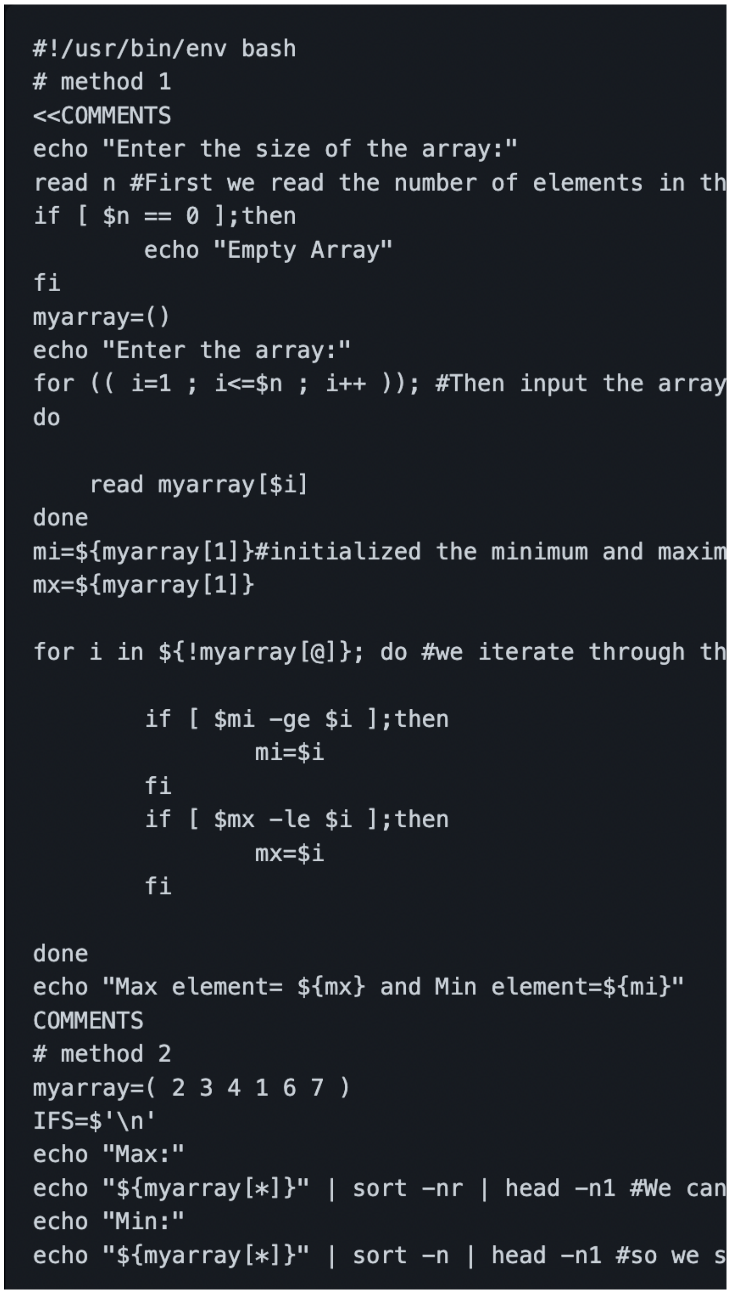


```
#!/usr/bin/env bash
path="/Users/devarithish/Desktop/"

if [ ! -d $path/assignment ]; then
  mkdir -p $path/assignment;
  echo "Directory created"
  touch $path/assignment/File1.txt
  echo "File created"
  cat $path/command_line_assignment/Table.sh > $path/assignment/File1.txt
  echo "Text written from Table.sh to File1.txt"
  echo "Welcome to Sigmoid" >> $path/assignment/File1.txt
  echo "Text appended to File1.txt"
  echo "ALL FILES IN DESKTOP:"
  ls $path
else
        echo "Directory exists already"
fi
```

Here I have specified a variable called path which is user dependent, to specify where the folder and where the file exists and where edits take place however shikhar and arin's code doesn't take a path variable and will work on any machine regardless the path.

**Question 5:**You have given an array. Using Bash script, print its length, maximum element and minimum element.arr=( 2 3 4 1 6 7).
My code:



```
#!/usr/bin/env bash
# method 1
<<COMMENTS
echo "Enter the size of the array:"
read n #First we read the number of elements in th
if [ $n == 0 ];then
        echo "Empty Array"
fi
myarray=()
echo "Enter the array:"
for (( i=1 ; i<=$n ; i++ )); #Then input the array
do

    read myarray[$i]
done
mi=${myarray[1]}#initialized the minimum and maxim
mx=${myarray[1]}

for i in ${!myarray[@]}; do #we iterate through th

        if [ $mi -ge $i ];then
                mi=$i
        fi
        if [ $mx -le $i ];then
                mx=$i
        fi

done
echo "Max element= ${mx} and Min element=${mi}"
COMMENTS
# method 2
myarray=( 2 3 4 1 6 7 )
IFS=$'\n'
echo "Max:"
echo "${myarray[*]}" | sort -nr | head -n1 #We can
echo "Min:"
echo "${myarray[*]}" | sort -n | head -n1 #so we s
```

Here I have taken an array input and traversed the array to find the smallest and largest element which involves storing alias variables to store the min and max values. However I've used another method using pipelines to sort(asc for min , desc for max) and return the starting element without creating any temporary variable. However my peers have used the first method which results in the right output but can save the temporary space.