

## Q1:

```
#!/bin/bash

PI=3.14
VAR_A=10
VAR_B=$VAR_A

echo "Lets print 3 variables:"
echo $VAR_A
echo $VAR_B
echo $VAR_C

echo "We know this will break:"
echo "0. The value of PI is $PIabc"

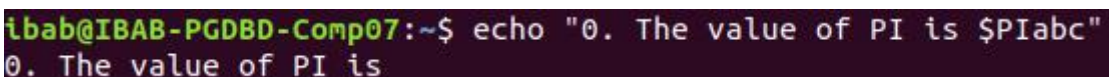
echo "And these will work:"
echo "1. The value of PI is $PI"
echo "2. The value of PI is ${PI}"
echo "3. The value of PI is" $PI

echo "And we can make a new string"
STR_A="Bob"
STR_B="Jane"
echo "${STR_A} + ${STR_B} equals Bob + Jane"
STR_C=${STR_A} + "${STR_B}"
echo "${STR_C} is the same as Bob + Jane too!"
echo "${STR_C} + ${PI}"

exit 0
```

Ans:

- 1) `#!/bin/bash` --> **means the script always run with BASH.**
- 2) `PI=3.14` --> **PI defined as a variable with the value 3.14.**
- 3) `VAR_A=10` --> **A declared as a variable and assigned value is 10 stored as a integer.**
- 4) `VAR_B=$VAR_A` --> **B declared as a variable and assigned value of 'A' stored as a integer.**
- 5) `VAR_C=${VAR_B}` --> **C declared as a variable and assigned value of 'B' due to '{}' the variable is expanded and force BASH to only interpret the VALUE inside the braces.**
- 6) `echo "Lets print 3 variables:"` --> **ECHO command is used to display line of existing text that are passed an argument.**
- 7) `echo $VAR_A` --> **returning the value of A which was assigned.**
- 8) `echo $VAR_B` --> **returning the value of B which was assigned.**
- 9) `echo $VAR_C` --> **returning the value of C which was assigned with forcing it.**
- 10) `echo "We know this will break:"` --> **Printing the text line in but it has no useful meaning.**

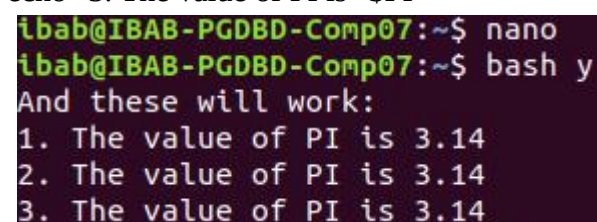


```
ibab@IBAB-PGDBD-Comp07:~$ echo "0. The value of PI is $PIabc"
0. The value of PI is
```

- 11) `echo "0. The value of PI is $PIabc"` --> **basically it is a error ECHO printing the line as is defined but \$PI didn't take the value as we want to set a new variable.**

12)

```
echo "And these will work:"
echo "1. The value of PI is $PI"
echo "2. The value of PI is ${PI}"
echo "3. The value of PI is" $PI
```



```
ibab@IBAB-PGDBD-Comp07:~$ nano
ibab@IBAB-PGDBD-Comp07:~$ bash y
And these will work:
1. The value of PI is 3.14
2. The value of PI is 3.14
3. The value of PI is 3.14
```

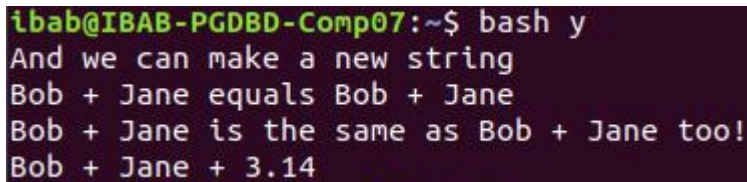
Here first line printing the input text.

Second line return the value of PI.

Third line force the BASH to interpret the value of PI inside the '{'.

Fourth line the BASH is returning the value of PI which was defined previously.

```
13)
echo "And we can make a new string"
STR_A="Bob"
STR_B="Jane"
echo "${STR_A} + ${STR_B} equals Bob + Jane"
STR_C=${STR_A}" + "${STR_B}
echo "${STR_C} is the same as Bob + Jane too!"
echo "${STR_C} + ${PI}"
```



```
ibab@IBAB-PGDBD-Comp07:~$ bash y
And we can make a new string
Bob + Jane equals Bob + Jane
Bob + Jane is the same as Bob + Jane too!
Bob + Jane + 3.14
```

Here first line printing the input text.

Second line define a String A name BOB.

Third line define a String B name JANE.

Fourth line the BASH is returning the value of A & B with the help of variable set expression \$ but '+' not concatenate the string.

Fifth line the string value of A & B is passing to String C.

Sixth line String C value is printing.

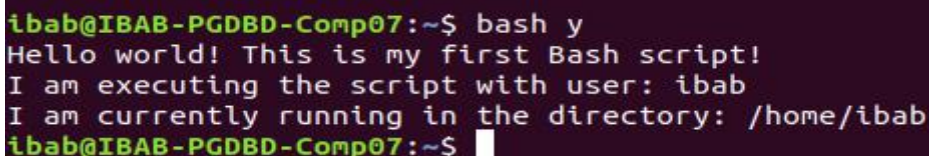
Seventh line along with the string C, forcing the BASH to call the PI value by defining the \${PI}.

14) exit 0 --> exit from terminal.

Q2.

```
#!/bin/bash
```

```
echo "Hello world! This is my first Bash script!"
echo -n "I am executing the script with user: "
whoami
echo -n "I am currently running in the directory: "
PwD
exit 0
```



```
ibab@IBAB-PGDBD-Comp07:~$ bash y
Hello world! This is my first Bash script!
I am executing the script with user: ibab
I am currently running in the directory: /home/ibab
ibab@IBAB-PGDBD-Comp07:~$
```

Line one returning the same text which is declared in the "" after ECHO.

Line two first '-' check if the given string operand size is non-zero; if it is nonzero length, then it returns true, then returning name of the admin along with the text inside the "".

Line three also check the if its true or not then returning the path of the current working directory.

Then exit from the terminal

### Q3.

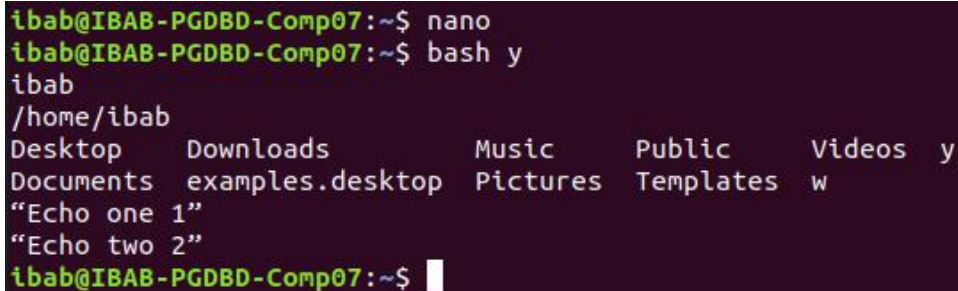
```
#!/bin/bash
```

```
whoami
```

```
pwd
```

```
ls
```

```
echo "Echo one 1"; echo "Echo two 2"
```



```
ibab@IBAB-PGDBD-Comp07:~$ nano
ibab@IBAB-PGDBD-Comp07:~$ bash y
ibab
/home/ibab
Desktop    Downloads      Music    Public    Videos  y
Documents  examples.desktop  Pictures  Templates w
"Echo one 1"
"Echo two 2"
ibab@IBAB-PGDBD-Comp07:~$
```

in first line returning the name of the user in bash its inbuilt command.

Pwd shows the path of the current working directory.

LS is shell command that lists files and directories within the working directory.

In last line we just printing the text inside the "" with the help of ECHO and for ';' we get two different lines.

### Q4.

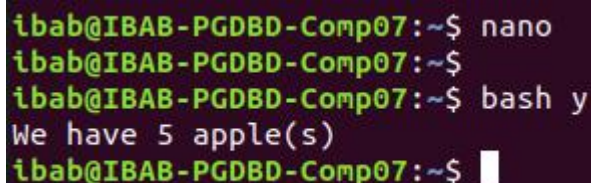
```
#!/bin/bash
```

```
#Filename :variables.sh
```

```
fruit=apple
```

```
count=5
```

```
echo "We have $count ${fruit}(s)"
```



```
ibab@IBAB-PGDBD-Comp07:~$ nano
ibab@IBAB-PGDBD-Comp07:~$
ibab@IBAB-PGDBD-Comp07:~$ bash y
We have 5 apple(s)
ibab@IBAB-PGDBD-Comp07:~$
```

file name is variables.sh means its a shell scripting file.

Define variable Fruit with Apple as a variable value.

Another variable Count define with value 5.

in last line \$count returning the value of count =5 and \${fruit} explicitly returning the value of fruit =apple and (s) is printing as simple text.