

Basic Of Shell Scripting

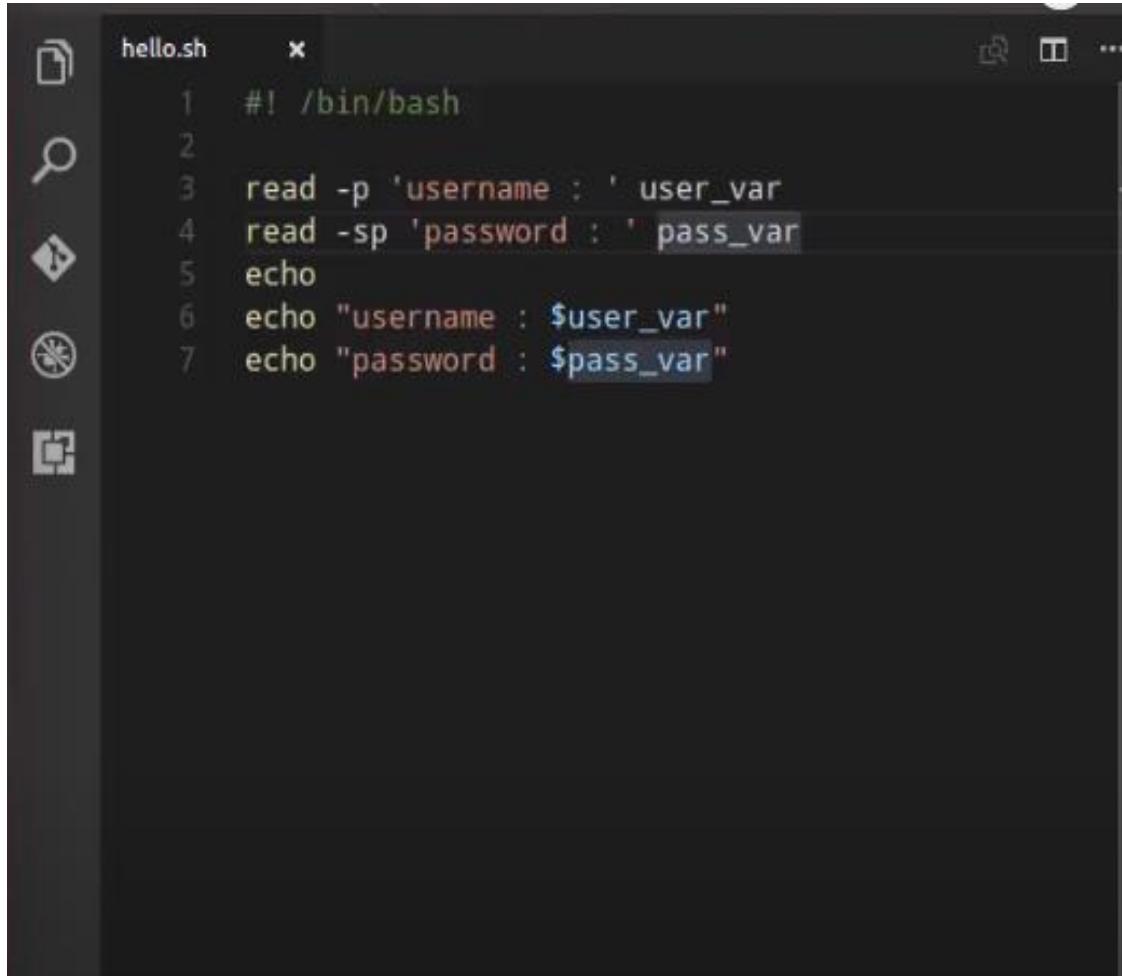
||

read Command

- Syntax for taking multiple user input using read keyword.

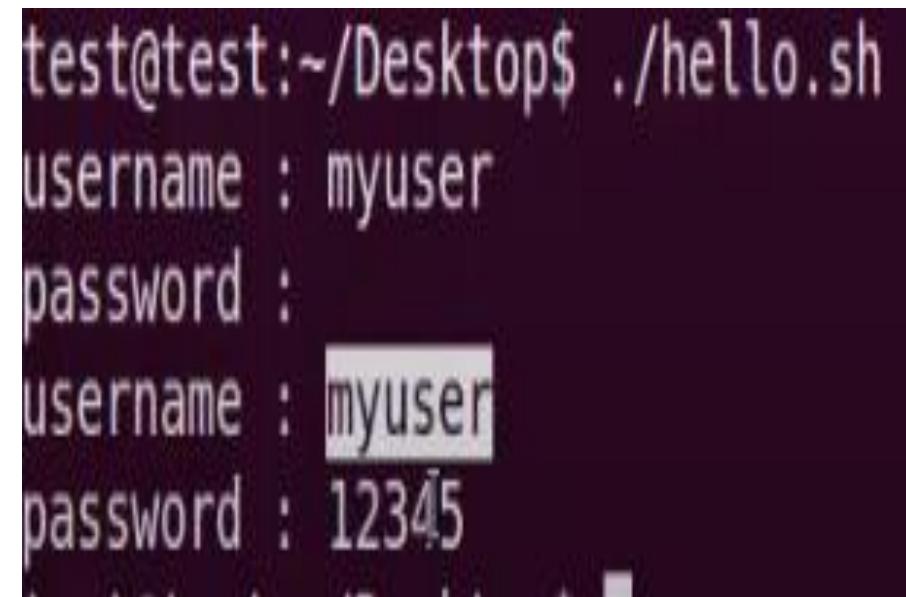
```
#!/bin/bash
echo "Enter three course name"
read name1 name2 name3
echo "names are: $name1, $name2, $name3"
```

Example



A screenshot of a dark-themed code editor window. The title bar says "hello.sh". The code in the editor is:

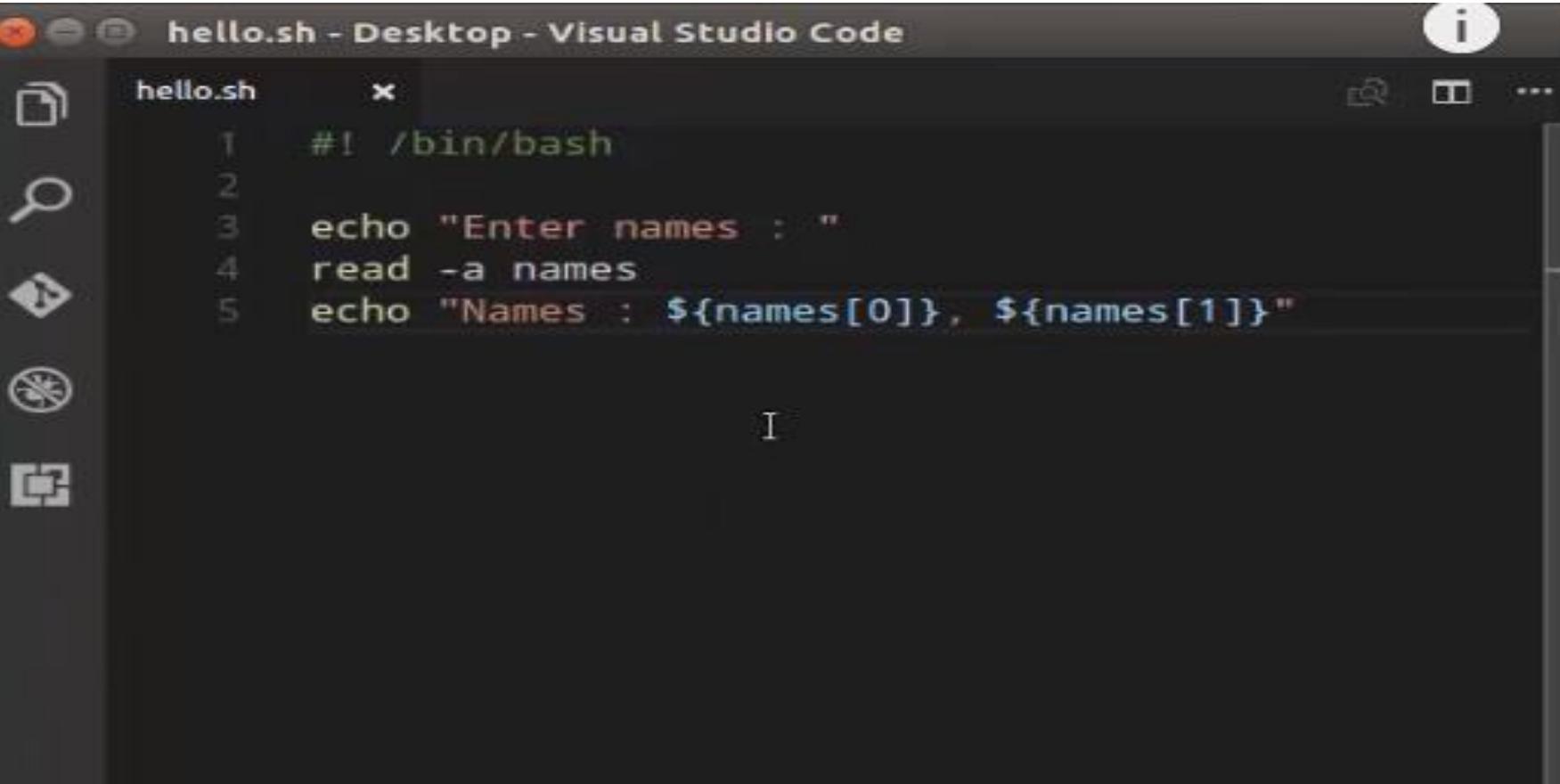
```
1 #! /bin/bash
2
3 read -p 'username : ' user_var
4 read -sp 'password : ' pass_var
5 echo
6 echo "username : $user_var"
7 echo "password : $pass_var"
```



A screenshot of a terminal window. The command entered is "test@test:~/Desktop\$./hello.sh". The output shows the script prompting for a username and password, and then echoing them back:

```
username : myuser
password :
username : myuser
password : 12345
```

Example



A screenshot of the Visual Studio Code interface. The title bar shows "hello.sh - Desktop - Visual Studio Code". The left sidebar has icons for file operations, search, and other tools. The main editor area contains the following bash script:

```
hello.sh      x
1  #! /bin/bash
2
3  echo "Enter names : "
4  read -a names
5  echo "Names : ${names[0]}, ${names[1]}"
```

The cursor is positioned at the end of the fifth line.

Adding Basic Options

- Let's take a look at some of the most basic options we can use:
- *-a array*: stores the results of the word split operation in an array rather than separate variables
- *-s*: does not echo the input line to the standard output stream
- *-p prompt*: print the prompt text before requesting the input from the standard input stream without a *<newline>* character
- *-t timeout*: attempt to read the input for a given period of seconds
- *-N*: read exactly N characters from the input unless a timeout occurs or *EOF* is reached

Conditional Statement

If-else Statement

- Bash if conditionals can have different forms. The most basic if statement takes the following form:

if TEST-COMMAND

then

STATEMENTS

Else

STATEMENTS

fi

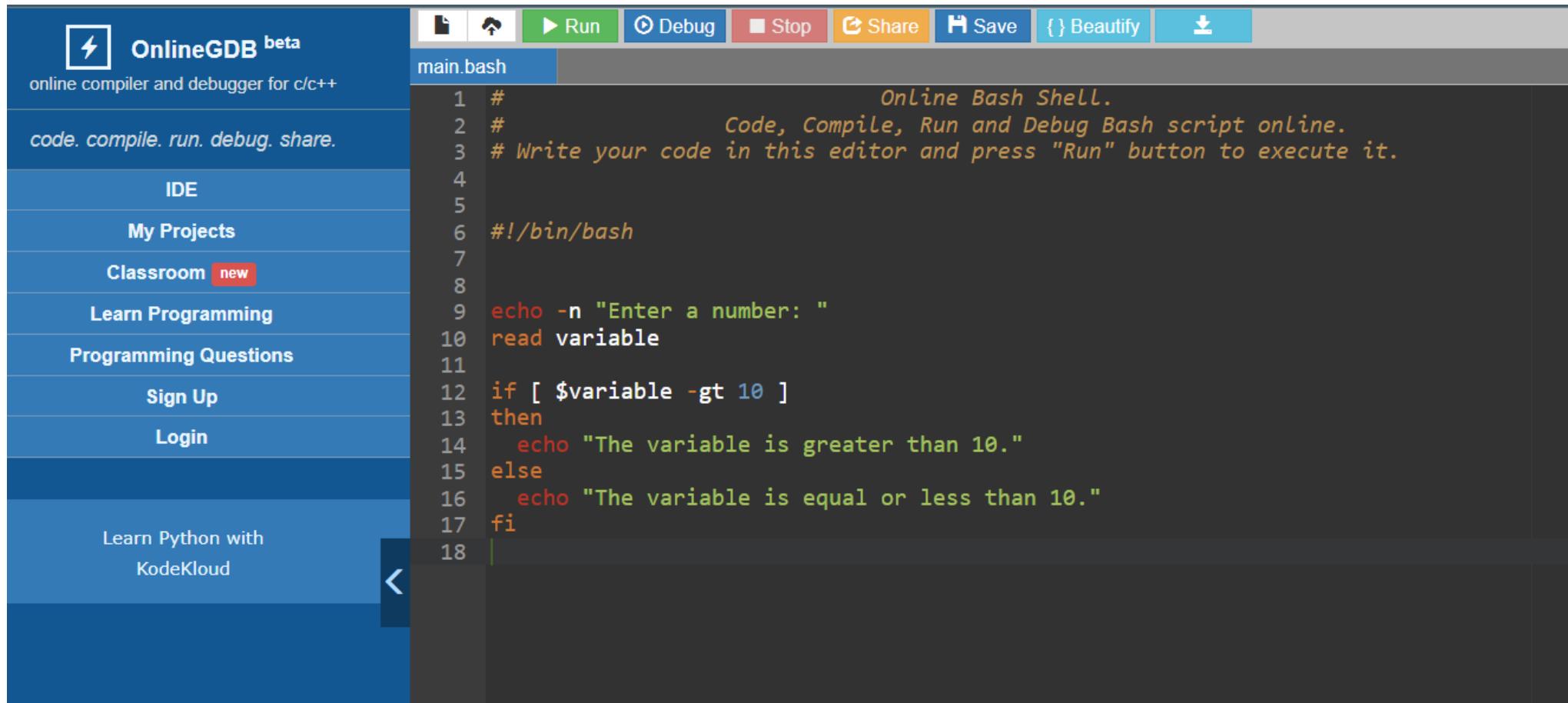
- The if statement starts with the if keyword followed by the conditional expression and the then keyword. The statement ends with the fi keyword.

Syntax

```
untitled — Atom
untitled • Telemetry Consent   Welcome

1 integer comparison
2
3 -eq - is equal to - if [ "$a" -eq "$b" ]
4 -ne - is not equal to - if [ "$a" -ne "$b" ]
5 -gt - is greater than - if [ "$a" -gt "$b" ]
6 -ge - is greater than or equal to - if [ "$a" -ge "$b" ]
7 -lt - is less than - if [ "$a" -lt "$b" ]
8 -le - is less than or equal to - if [ "$a" -le "$b" ]
9 < - is less than - (( "$a" < "$b" ))
10 <= - is less than or equal to - (( "$a" <= "$b" ))
11 > - is greater than - (( "$a" > "$b" ))
12 >= - is greater than or equal to - (( "$a" >= "$b" ))
13
14 string comparison
15 = - is equal to - if [ "$a" = "$b" ]
16 == - is equal to - if [ "$a" == "$b" ]
17 != - is not equal to - if [ "$a" != "$b" ]
18 < - is less than, in ASCII alphabetical order - if [[ "$a" < "$b" ]]
19 > - is greater than, in ASCII alphabetical order - if [[ "$a" > "$b" ]]
20 -z - string is null, that is, has zero length
```

Example Code



The screenshot shows the OnlineGDB beta interface. The left sidebar includes links for IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. A banner at the bottom left says "Learn Python with KodeKloud". The main area has a toolbar with Run, Debug, Stop, Share, Save, and Beautify buttons. The code editor window is titled "main.bash" and contains the following Bash script:

```
1 #                                     Online Bash Shell.
2 #                                     Code, Compile, Run and Debug Bash script online.
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6 #!/bin/bash
7
8
9 echo -n "Enter a number: "
10 read variable
11
12 if [ $variable -gt 10 ]
13 then
14     echo "The variable is greater than 10."
15 else
16     echo "The variable is equal or less than 10."
17 fi
18
```

if..elif..else Statement

```
if TEST-COMMAND1  
then  
    STATEMENTS1  
elif TEST-COMMAND2  
then  
    STATEMENTS2  
else  
    STATEMENTS3  
fi
```

If the TEST-COMMAND1 evaluates to True, the STATEMENTS1 will be executed. If the TEST-COMMAND2 evaluates to True, the STATEMENTS2 will be executed. If none of the test commands evaluate to True, the STATEMENTS2 is executed.

Example Code



The screenshot shows the OnlineGDB beta interface. The left sidebar has links for IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. A banner at the bottom left says "Learn Python with KodeKloud". The main area has tabs for "main.bash" and "Online Bash Shell". The code editor contains the following Bash script:

```
1 #                                     Online Bash Shell.
2 #                                     Code, Compile, Run and Debug Bash script online.
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6#!/bin/bash
7
8
9echo -n "Enter a number: "
10read Variable
11
12if [ $Variable -gt 10 ]
13then
14    echo "The variable is greater than 10."
15elif [ $Variable -eq 10 ]
16then
17    echo "The variable is equal to 10."
18else
19    echo "The variable is less than 10."
20fi
```