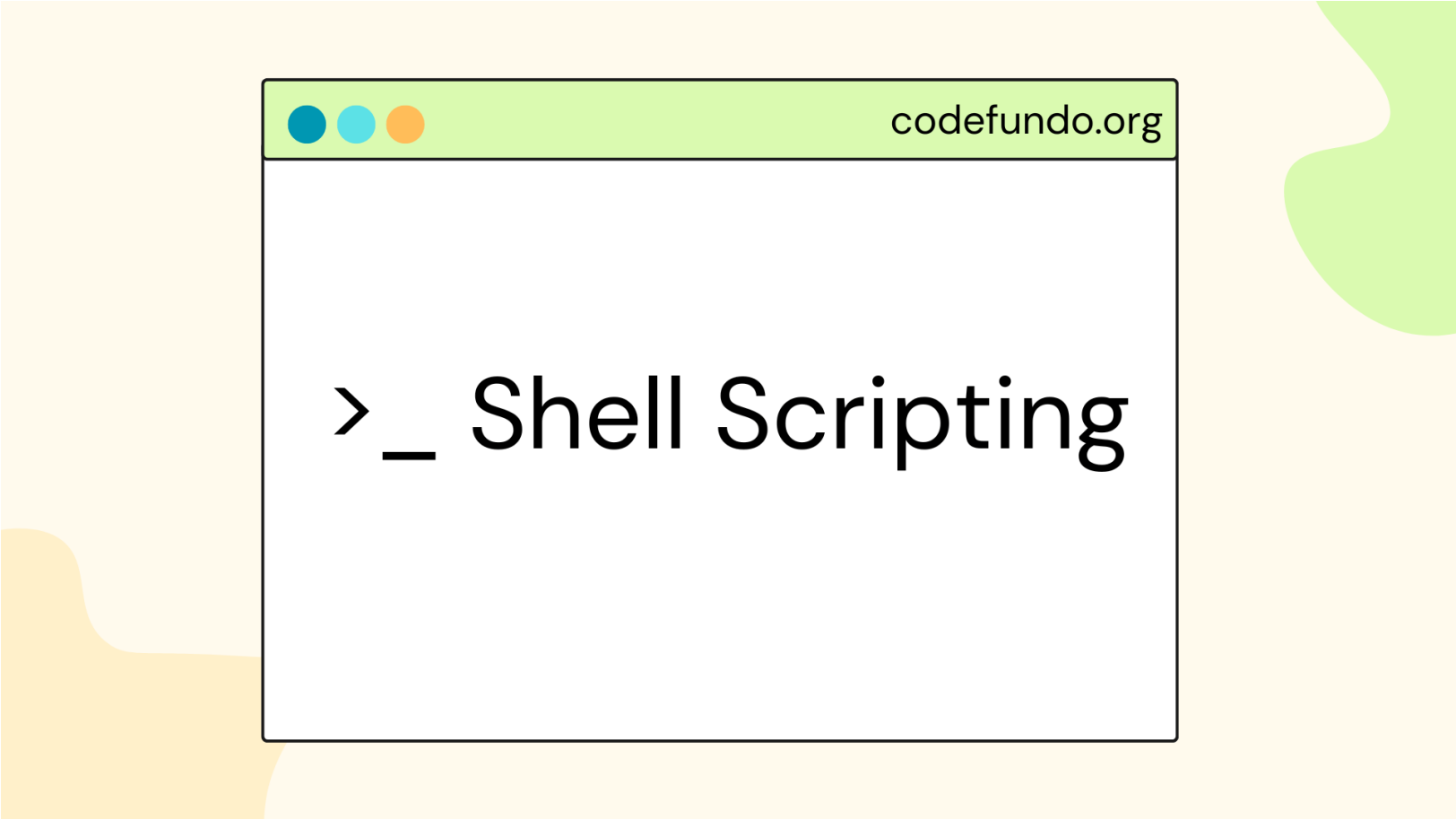




What is shell scripting?



Shell scripting is the process of creating and executing scripts written in a shell programming language. Shell scripts are used to automate tasks and execute command sequences in a Unix shell.

They can perform various tasks such as file manipulation, program execution, and system administration. Shell scripting is a powerful tool for managing systems and automating tasks in the Unix/Linux ecosystem.

The importance and usage of shell scripting:

Automation	Shell scripts can automate repetitive tasks, saving time and reducing human error.
Complex command sequences	Shell scripts can be used to create complex command sequences that are difficult to type manually.
System administration	Shell scripts are used for system administration tasks such as backups, user management, and software installation.
Portability	Shell scripts are portable across different Unix-like operating systems, making them useful for managing systems across different platforms.
Customization	Shell scripts can be customized to suit specific needs and requirements.
Rapid prototyping	Shell scripts can be used for rapid prototyping of command sequences before implementing them in a larger system.
Debugging	Shell scripts can be used for debugging and troubleshooting problems in Unix/Linux systems.

Overall, shell scripting is an essential tool for managing systems and automating tasks in the Unix/Linux ecosystem. It can improve productivity, reduce errors, and make system administration tasks more manageable.

That's all about the shell script and its importance. Now, let's explore some examples to better understand shell scripting.

How to create a shell script?

We can use the following steps to create a shell script in Linux / Ubuntu:

- Open a text editor (such as Vim, or Gedit) on your system.
- Create a new file with a .sh extension. For example, you can use the following command to create a file named "script.sh" in the current directory.

Code

```
vim script.sh
```

- Add the shebang line at the top of the script. This line tells the system which interpreter to use to run the script. For a bash script, use the following shebang line:

Code

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

- Write the commands you want the script to execute. For example, you can create a simple "Welcome to Shell Script" script by adding the following line:

Code

Shell Script

✕

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

Conditional Statements

Loops

Command Line Arguments

Functions

Useful Scripts

```
#!/bin/bash
echo 'Welcome to Shell Script'
```

How to execute or run the shell script?

1. Save the code in a file with a .sh extension, for example script.sh and make it executable using the following command.

Code

```
chmod +x script.sh
```

2. Run the script in your terminal by typing ./script.sh

Code

```
./script.sh
```

3. You can also run the script by typing sh script.sh in your terminal.

Code

```
sh script.sh
```

4. Alternatively, you can also run the script using the bash command, like this:

Code

```
bash script.sh
```

All three commands, './script.sh', 'sh script.sh', and 'bash script.sh', produce the same output.

Output

If we execute the script.sh file with the above commands in a terminal or shell, we can see the output 'Welcome to Shell Script' printed on the screen.

Output

```
Welcome to Shell Script
```

Example 1: Shell script to print "Hello World"

In a shell script, we can print a message to the terminal using the echo command followed by the message you want to print. So to print "Hello World" in a shell script, we can write the code something like below.

Code

```
echo 'Hello World'
```

When we run this command in a shell script and execute it in the terminal, the message "Hello World" will be printed to the terminal.

Output

```
Hello World
```

Example 2: Shell script to print your name

1. In a shell script, we can print a message to the terminal using the **echo** command followed by the message you want to print. So to **print your name** in a shell script, we can write the code something like below.

Code

```
#!/bin/bash
echo 'My name is [YOUR NAME]'
```

Codefundo (/)	2. Replace [YOUR NAME] with your actual name, and save the script with a .sh file extension, say script.sh.	
Shell Script	Code	
Full Course	<pre>#!/bin/bash echo 'My name is Codefundo.org'</pre>	
Shell Script	3. You can then run the script in the terminal by typing ./script.sh.	
(/shell-script/shell-script-examples.html)	Code	
Basics	<pre>./script.sh</pre>	
Operators	4. The script will print your name to the terminal.	
Conditional Statements	Output	
Loops	<pre>My name is Codefundo.org</pre>	
Command Line Arguments		
Functions		
Useful Scripts		
	<div>Example 3: Read user input in the shell script</div> <p>To read user input in a shell script, we can use the read command. Here's an example:</p> <div>Code</div> <pre>#!/bin/bash echo "Enter your name:" read name echo "Welcome \$name"</pre> <div>Output</div> <pre>Enter your name: Codefundo Welcome Codefundo</pre> <div>How does it work?</div> <ol style="list-style-type: none">1. The echo command displays the message "Enter your name" on the screen.2. If we enter a name, the read command reads it and stores it in the variable 'name'.3. Finally, the echo command uses the name variable to print a welcome message. <div>Prompt message along with the read command</div> <p>We can also provide a prompt message along with the read command using -p option like this:</p> <div>Code</div> <pre>#!/bin/bash read -p "Enter your name: " name echo "Welcome \$name"</pre> <div>Output</div> <pre>Enter your name: Codefundo Welcome Codefundo</pre> <p>In this case, the prompt message "Enter your name: " will be displayed before the user is prompted to enter their name.</p> <div>Example 4: Shell script to add two numbers</div> <p>In this article, we will learn how to add two numbers in a shell script.</p> <div>Algorithm:</div> <ol style="list-style-type: none">1. Prompt the user to enter the first number.2. Read the input from the user using the read command and store it in a variable, say, 'a'.3. Prompt the user to enter the second number.4. Read the input from the user using the read command and store it in another variable, say, 'b'.5. Add the two numbers and store the result in a third variable	



Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

Conditional Statements

Loops

Command Line Arguments

Functions

Useful Scripts

```
#!/bin/bash
# Prompt the user to enter the first number
echo "Enter the first number: "
# Read the input from the user and store it in a variable
read a

# Prompt the user to enter the second number
echo "Enter the second number: "
# Read the input from the user and store it in a variable
read b

# Prompt the user to enter the third number
echo "Enter the third number: "
# Read the input from the user and store it in a variable
read c

# Use the $(( )) syntax to add the three numbers and store the result in a variable
sum=$((a + b + c))

# Display the result to the user
echo "The sum of $a, $b, and $c is: $sum"
```

Output

```
Enter the first number:
10
Enter the second number:
20
Enter the third number:
30
The sum of 10, 20, and 30 is: 60
```

Shell script to add three numbers using expr

Code

```
#!/bin/bash
# Prompt the user to enter the first number
echo "Enter the first number: "
# Read the input from the user and store it in a variable
read a

# Prompt the user to enter the second number
echo "Enter the second number: "
# Read the input from the user and store it in a variable
read b

# Prompt the user to enter the third number
echo "Enter the third number: "
# Read the input from the user and store it in a variable
read c

# Use the expr command to add the three numbers and store the result in a variable
sum=$(expr $a + $b + $c)

# Display the result to the user
echo "The sum of $a, $b, and $c is: $sum"
```

Output

```
Enter the first number:
10
Enter the second number:
20
Enter the third number:
30
The sum of 10, 20, and 30 is: 60
```

Example 6: Shell script to subtract two numbers

In this article, we will learn how to subtract two numbers in a shell script.

Algorithm:

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

1. Prompt the user to enter the first number.
2. Read the input from the user using the read command and store it in a variable, say, 'a'.

3. Prompt the user to enter the second number.
4. Read the input from the user using the read command and store it in another variable, say, 'b'.
5. Subtract the second number from the first number and store the result in a third variable.
6. Display the result to the user using the echo command.

Shell script to subtract two numbers using \$()

Code

```
#!/bin/bash
Prompt the user to enter the first number
echo "Enter the first number: "

Read the input from the user and store it in a variable
read a

Prompt the user to enter the second number
echo "Enter the second number: "

Read the input from the user and store it in another variable
read b

Subtract the second number from the first number and store the result in a variable
diff=$((a - b))

Display the result to the user
echo "The difference between $a and $b is: $diff"
```

Output

```
Enter the first number:
20
Enter the second number:
10
The difference between 20 and 10 is: 10
```

Shell script to subtract two numbers using expr

Code

```
#!/bin/bash
# Prompt the user to enter the first number
echo "Enter the first number: "
# Read the input from the user and store it in a variable
read a
# Prompt the user to enter the second number
echo "Enter the second number: "
# Read the input from the user and store it in another variable
read b
# Use the expr command to subtract the second number from the first number and store the result in a third variable
diff=`expr $a - $b`
# Display the result to the user
echo "The difference between $a and $b is: $diff"
```

Output

```
Enter the first number:
20
Enter the second number:
10
The difference between 20 and 10 is: 10
```

Example 7: Shell script to multiply two numbers


In this article, we will learn how to multiply two numbers in a shell script.


Algorithm:


1. Prompt the user to enter the first number.
2. Read the input from the user using the read command and store it in a variable, say, 'a'.
3. Prompt the user to enter the second number.
4. Read the input from the user using the read command and store it in another variable, say, 'b'.
5. Multiply the two numbers and store the result in a third variable.
6. Display the result to the user using the echo command.

Shell script to multiply two numbers using \$()

Code

Codefundo (/)		
Shell Script		<pre>#!/bin/bash Prompt the user to enter the first number echo "Enter the first number: "</pre>
Full Course		
Shell Script		<pre>Read the input from the user and store it in a variable read a Prompt the user to enter the second number echo "Enter the second number: "</pre>
(/shell-script/shell-script-examples.html)		
Basics		
Operators		<pre>Read the input from the user and store it in another variable read b</pre>
Conditional Statements		
Loops		<pre>Multiply the two numbers and store the result in a variable result=\$((a * b))</pre>
Command Line Arguments		<pre>Display the result to the user echo "The product of \$a and \$b is: \$result"</pre>
Functions		
Useful Scripts		<pre>Output Enter the first number: 10 Enter the second number: 20 The product of 10 and 20 is: 200</pre>
Shell script to multiply two numbers using expr		
Code		
<pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read a # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in another variable read b # Use the expr command to multiply the two numbers and store the result in a third variable result=`expr \$a * \$b` # Display the result to the user echo "The product of \$a and \$b is: \$result"</pre>		
Output		
<pre>Enter the first number: 10 Enter the second number: 20 The product of 10 and 20 is: 200</pre>		
Example 8: Shell script to divide two numbers		
In this article, we will learn how to divide two numbers in a shell script.		
Algorithm:		
<ol style="list-style-type: none">Prompt the user to enter the dividend.Read the input from the user using the read command and store it in a variable, say, 'dividend'.Prompt the user to enter the divisor.Read the input from the user using the read command and store it in another variable, say, 'divisor'.Perform the division and store the result in a variable, say, 'quotient'.Display the result to the user using the echo command.		
Shell script to divide two numbers using \$(())		
Code		

Codefundo (/)		
Shell Script		
Full Course	Shell Script	<pre>#!/bin/bash # Prompt the user to enter the dividend echo "Enter the dividend: " read dividend # Prompt the user to enter the divisor echo "Enter the divisor: " read divisor # Use the \$(()) operator to perform the division quotient=\$((dividend / divisor)) # Display the result to the user echo "The quotient of \$dividend and \$divisor is: \$quotient"</pre>
	(/shell-script/shell-script-examples.html)	
	Basics	
	Operators	
	Conditional Statements	
	Loops	
	Command Line Arguments	
	Functions	
	Useful Scripts	
Shell script to divide two numbers using expr		
Code		
<pre>#!/bin/bash # Prompt the user to enter the dividend echo "Enter the dividend: " # Read the input from the user and store it in a variable read dividend # Prompt the user to enter the divisor echo "Enter the divisor: " # Read the input from the user and store it in another variable read divisor # Use the expr command to perform the division and store the result in a variable quotient=`expr \$dividend / \$divisor` # Display the result to the user echo "The quotient of \$dividend and \$divisor is: \$quotient"</pre>		
Output		
<pre>Enter the dividend: 20 Enter the divisor: 5 The quotient of 20 and 5 is: 4</pre>		
Example 9: Shell script to perform arithmetic operations		
In this article, we will learn how to perform arithmetic operations in a shell script.		
Algorithm:		
1. Assign values to variables a and b.		
2. Perform arithmetic operations such as addition, subtraction, multiplication, division, and modulo using shell arithmetic expansion \$(()) or using the command-line utility expr.		
3. Finally, use the echo command to display the results.		
Arithmetic operations using \$(())		
Code		

Codefundo (/)		
Shell Script		
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
		<pre>#!/bin/bash # Assign values to variables a and b a=100 b=10 # Perform arithmetic operations sum=\$((a + b)) diff=\$((a - b)) prod=\$((a * b)) quot=\$((a / b)) rem=\$((a % b)) # Display the results echo "The sum of \$a and \$b is: \$sum" echo "The difference between \$a and \$b is: \$diff" echo "The product of \$a and \$b is: \$prod" echo "The quotient of \$a and \$b is: \$quot" echo "The remainder of \$a and \$b is: \$rem"</pre> <p>Output</p> <p>The sum of 100 and 10 is: 110 The difference between 100 and 10 is: 90 The product of 100 and 10 is: 1000 The quotient of 100 and 10 is: 10 The remainder of 100 and 10 is: 0</p> <p>Arithmetic operations using expr</p> <p>Code</p> <pre>#!/bin/bash # Assign values to variables a and b a=100 b=10 # Perform arithmetic operations sum=\$(expr \$a + \$b) diff=\$(expr \$a - \$b) prod=\$(expr \$a * \$b) quot=\$(expr \$a / \$b) rem=\$(expr \$a % \$b) # Display the results echo "The sum of \$a and \$b is: \$sum" echo "The difference between \$a and \$b is: \$diff" echo "The product of \$a and \$b is: \$prod" echo "The quotient of \$a and \$b is: \$quot" echo "The remainder of \$a and \$b is: \$rem"</pre> <p>Output</p> <p>The sum of 100 and 10 is: 110 The difference between 100 and 10 is: 90 The product of 100 and 10 is: 1000 The quotient of 100 and 10 is: 10 The remainder of 100 and 10 is: 0</p> <p>Note that we need to use the backslash character (\) to escape the multiplication operator (*) so that it's interpreted correctly by the shell.</p>
Example 10: Shell script to perform arithmetic operations on given two numbers		
In this article, we will learn how to perform arithmetic operations on two given numbers in a shell script.		
Algorithm:		
1. Prompt the user to enter two numbers using the echo command.		
2. Store the given numbers in the variables a and b.		
3. Perform arithmetic operations such as addition, subtraction, multiplication, division, and modulo using shell arithmetic expansion \$(()) or using the command-line utility expr.		
4. Finally, use the echo command to display the results of the arithmetic operations.		
Arithmetic operations on given two numbers using \$(())		
Code		

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

```
#!/bin/bash
# Prompt the user to enter two numbers
echo "Enter the first number:"
read a
echo "Enter the second number:"
read b

# Perform arithmetic operations
sum=$((a + b))
diff=$((a - b))
prod=$((a * b))
quot=$((a / b))
rem=$((a % b))

# Display the results
echo "The sum of $a and $b is: $sum"
echo "The difference between $a and $b is: $diff"
echo "The product of $a and $b is: $prod"
echo "The quotient of $a and $b is: $quot"
echo "The remainder of $a and $b is: $rem"
```

Output

```
Enter the first number:
100
Enter the second number:
10
The sum of 100 and 10 is: 110
The difference between 100 and 10 is: 90
The product of 100 and 10 is: 1000
The quotient of 100 and 10 is: 10
The remainder of 100 and 10 is: 0
```

Arithmetic operations on given two numbers using expr

Code

```
#!/bin/bash
# Prompt the user to enter two numbers
echo "Enter the first number:"
read a
echo "Enter the second number:"
read b

# Perform arithmetic operations
sum=$(expr $a + $b)
diff=$(expr $a - $b)
prod=$(expr $a \* $b)
quot=$(expr $a / $b)
rem=$(expr $a % $b)

# Display the results
echo "The sum of $a and $b is: $sum"
echo "The difference between $a and $b is: $diff"
echo "The product of $a and $b is: $prod"
echo "The quotient of $a and $b is: $quot"
echo "The remainder of $a and $b is: $rem"
```

Output

```
Enter the first number:
100
Enter the second number:
10
The sum of 100 and 10 is: 110
The difference between 100 and 10 is: 90
The product of 100 and 10 is: 1000
The quotient of 100 and 10 is: 10
The remainder of 100 and 10 is: 0
```

Note that we need to use the backslash character (\) to escape the multiplication operator (*) so that it's interpreted correctly by the shell.

Example 11: Shell script to swap two numbers

In this article, we will learn how to swap two numbers in a shell script.

Algorithm:

- 1. Prompt the user to enter the first number.
- 2. Read the input from the user using the read command and store it in a variable, say, 'num1'.

Codefundo (/)		<div>3. Prompt the user to enter the second number.</div> <div>4. Read the input from the user using the read command and store it in another variable, say, 'num2'.</div>
Shell Script	✕	<div>5. Swap the values of 'num1' and 'num2' using a temporary variable.</div> <div>6. Display the swapped values of 'num1' and 'num2' to the user using the echo command.</div>
Full Course		<div>Shell script to swap two numbers</div> <div>Code<div><pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read num1 # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in another variable read num2 # Swap the values of num1 and num2 temp=num1 num1=num2 num2=temp # Display the swapped values of num1 and num2 echo "After swapping, num1 is \$num1 and num2 is \$num2"</pre></div><div>Output<div>Enter the first number: 10 Enter the second number: 20 After swapping, num1 is 20 and num2 is 10</div></div><div>In the above code, we have used a temporary variable to swap the values of 'num1' and 'num2'. After swapping, we have displayed the swapped values using the echo command.</div></div>
		<div>Example 12: Shell script to swap two numbers without using a third variable</div> <div>In this article, we will learn how to swap two numbers without using a third variable in a shell script.</div> <div>Algorithm:</div> <div>1. Prompt the user to enter the first number.</div> <div>2. Read the input from the user using the read command and store it in a variable, say, 'num1'.</div> <div>3. Prompt the user to enter the second number.</div> <div>4. Read the input from the user using the read command and store it in another variable, say, 'num2'.</div> <div>5. Use arithmetic operations to swap the two numbers without using a third variable.</div> <div>6. Display the swapped values to the user using the echo command.</div> <div>Shell script to swap two numbers without using a third variable</div> <div>Code<div><pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read num1 # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in another variable read num2 # Swap the values without using a third variable num1=\$((num1+num2)) num2=\$((num1-num2)) num1=\$((num1-num2)) # Display the swapped values echo "After swapping:" echo "First number: \$num1" echo "Second number: \$num2"</pre></div><div>Output</div></div>

Codefundo (/)

Shell Script

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

Conditional Statements

Loops

Command Line Arguments

Functions

Useful Scripts

✕

Enter the first number:
10
Enter the second number:
20
After swapping:
First number: 20
Second number: 10

Explanation:

In the above code, we have used arithmetic operations to swap the two numbers without using a third variable. Here, we have used the following formula:

num1 = num1 + num2
num2 = num1 - num2
num1 = num1 - num2

First, we add the two numbers and store the result in 'num1'. Then, we subtract the value of 'num2' from 'num1' and store the result in 'num2'. Finally, we subtract the value of 'num2' from the new value of 'num1' to get the original value of 'num1' and store the result in 'num1'.
After swapping, we display the values of 'num1' and 'num2' to the user.

Example 13: Shell program to find the given number is odd or even

In this article, we will learn how to write a shell program to determine whether a given number is odd or even.

Algorithm:

1. Prompt the user to enter a number

2. Read the input number using the read command and store it in a variable, say, 'num'

3. Use the modulus operator (%) to find the remainder when the number is divided by 2. If the remainder is 0, the number is even. If the remainder is 1, the number is odd.

4. Print the result to the user using the echo command.

Pictorial explanation:

Number	Number % 2	Result
1	1	Odd
2	0	Even
3	1	Odd
4	0	Even
5	1	Odd
6	0	Even
7	1	Odd
8	0	Even

codefundo.org

codefundo.org

codefundo.org

Flow chart:

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

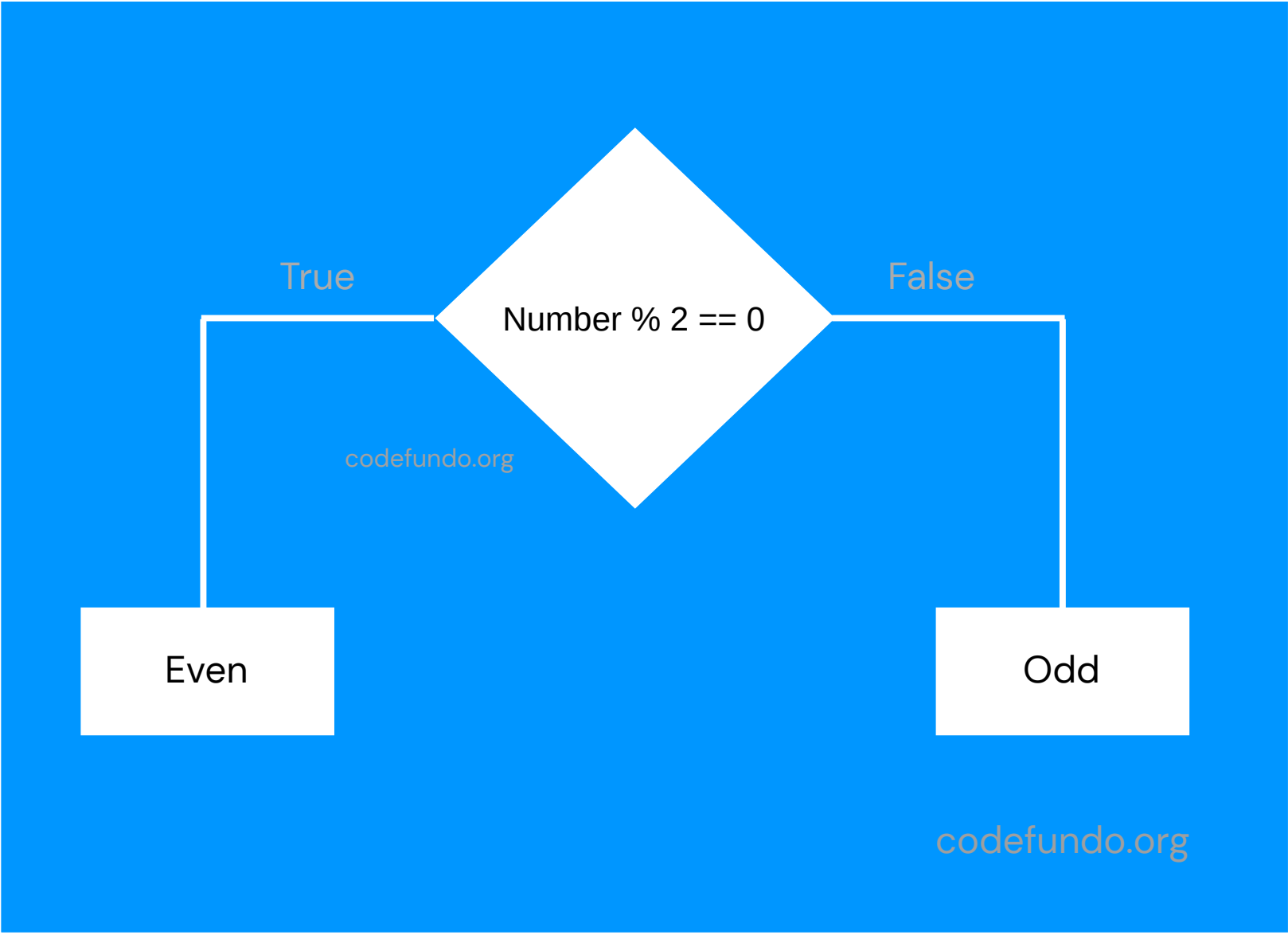
Conditional Statements

Loops

Command Line Arguments

Functions

Useful Scripts



Shell script to find whether a number is odd or even

Code:

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number: "
# Read the input from the user and store it in a variable
read num

# Use the modulus operator to check whether the number is even or odd
if [  $$(num \% 2)$  -eq 0 ]
then
echo "$num is an even number"
else
echo "$num is an odd number"
fi
```

Output:

```
Enter a number: 24
24 is an even number
```

In the above code, we have used the modulus operator (%) to check whether the given number is even or odd. If the remainder is 0, the number is even, and if the remainder is 1, the number is odd. We have used the '-eq' option to check whether the remainder is equal to zero or not.

Example 14: Shell script to check whether a number is positive or negative

In this article, we will learn how to write a shell script to determine whether a given number is positive or negative.

Algorithm:

1. Prompt the user to enter a number.
2. Read the input number using the read command and store it in a variable, say, 'num'.
3. Check if the number is greater than zero. If yes, then the number is positive. If no, then the number is negative.
4. Print the result to the user using the echo command.

Codefundo (/)

Shell Script

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

Conditional Statements

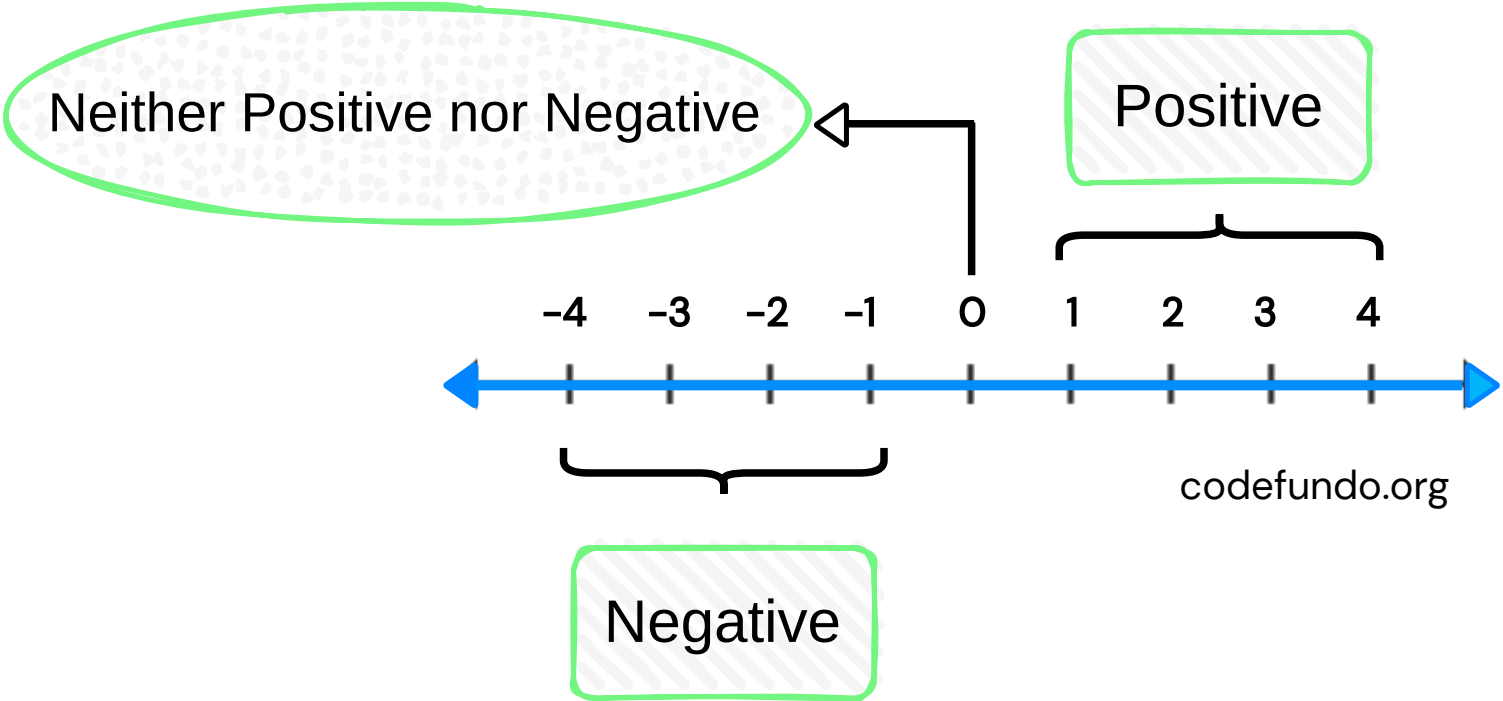
Loops

Command Line Arguments

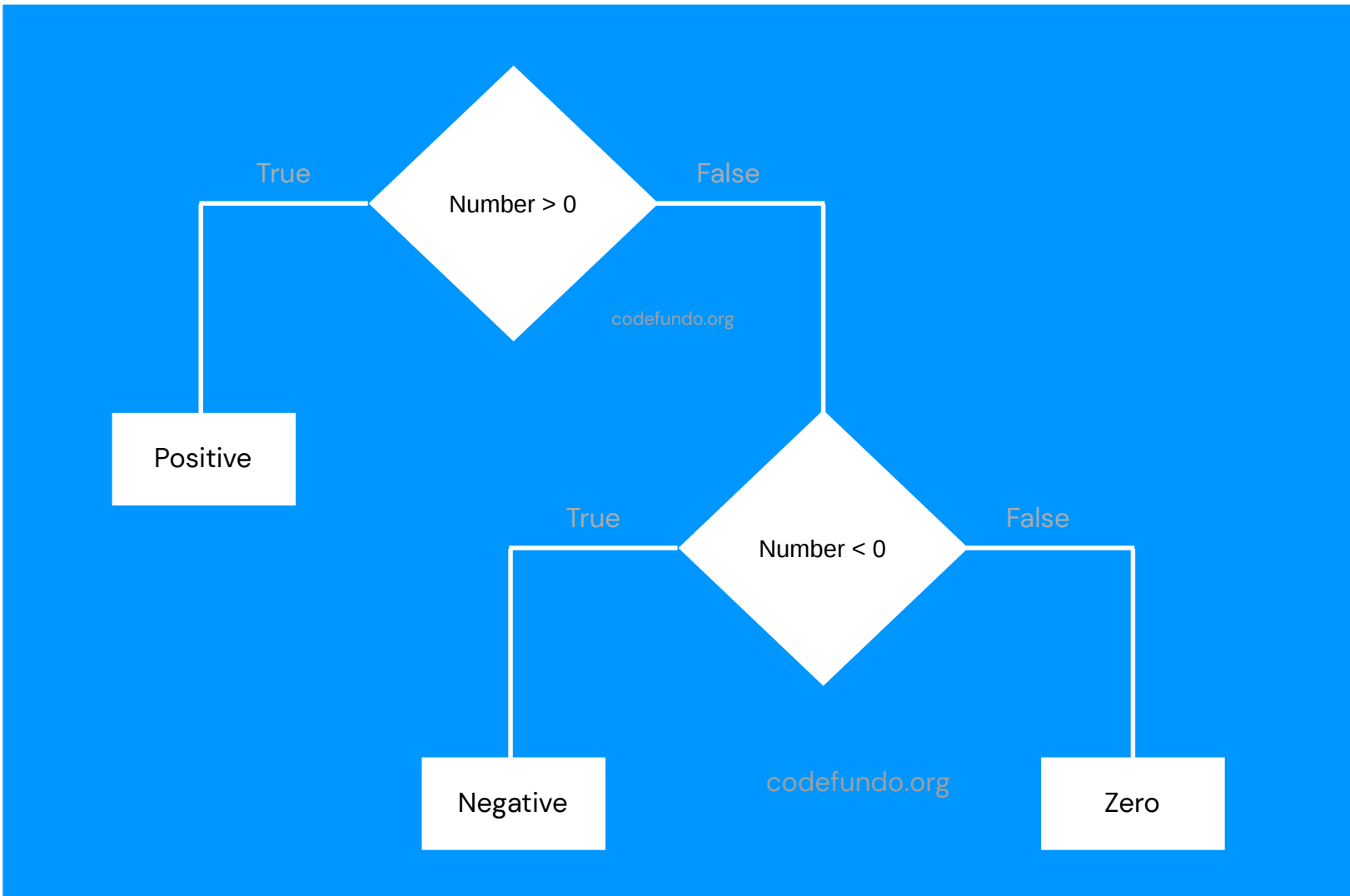
Functions

Useful Scripts

Pictorial explanation:



Flow chart:



Shell script to find whether a number is positive or negative

Code:

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number: "
# Read the input from the user and store it in a variable
read num
# Check if the number is positive or negative
if [ $num -gt 0 ]
then
echo "$num is a positive number"
elif [ $num -lt 0 ]
then
echo "$num is a negative number"
else
echo "The number is zero. It is neither positive nor negative."
fi
```

Output:

```
Enter a number: 5
5 is a positive number

-----

Enter a number: -3
-3 is a negative number

-----

Enter a number: 0
The number is zero. It is neither positive nor negative.
```


Codefundo (/)		
Shell Script	✕	<div>Explanation:</div> <div>In the above script, we first prompt the user to enter a number and read the input using the read command. We then use an if-else statement to check whether the number is positive, negative, or zero. We use the '-gt' option to check whether the number is greater than zero and the '-lt' option to check whether the number is less than zero. If the number is equal to zero, we print the result accordingly.</div>
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
		<div>Example 15: Shell script to compare two numbers</div> <div>In this article, we will learn how to compare two numbers in a shell script.</div> <div>Algorithm:</div> <div><div>1. Prompt the user to enter the first number.</div><div>2. Read the input from the user using the read command and store it in a variable, say, 'num1'.</div><div>3. Prompt the user to enter the second number.</div><div>4. Read the input from the user using the read command and store it in another variable, say, 'num2'.</div><div>5. Use an if-else statement to compare the two numbers. If 'num1' is greater than 'num2', print a message saying so. If 'num2' is greater than 'num1', print a message saying so. If both numbers are equal, print a message saying so.</div><div>6. Display the result to the user using the echo command.</div></div> <div>Shell script to compare two numbers</div> <div><div>Code</div><div><pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read num1 # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in another variable read num2 # Use if-else statements to compare the two numbers if [\$num1 -gt \$num2] then echo "\$num1 is greater than \$num2" elif [\$num2 -gt \$num1] then echo "\$num2 is greater than \$num1" else echo "Both numbers are equal" fi</pre></div><div>Output</div><div><pre>Enter the first number: 10 Enter the second number: 20 20 is greater than 10</pre></div></div> <div>In the above code, we have used the if-else statement to compare two numbers. The '-gt' option is used to check if the first number is greater than the second number. Similarly, '-lt' can be used to check if the first number is less than the second number.</div>
		<div>Example 16: Shell script to find the greatest of two numbers</div> <div>In this article, we will learn how to find the greatest of two numbers in a shell script.</div> <div>Algorithm:</div> <div><div>1. Prompt the user to enter the first number.</div><div>2. Read the input from the user using the read command and store it in a variable, say, 'num1'.</div><div>3. Prompt the user to enter the second number.</div><div>4. Read the input from the user using the read command and store it in another variable, say, 'num2'.</div><div>5. Use an if-else statement to compare the two numbers. If 'num1' is greater than 'num2', print 'num1' as the greatest number. If 'num2' is greater than 'num1', print 'num2' as the greatest number. If both numbers are equal, print a message saying 'Both numbers are equal'.</div><div>6. Display the result to the user using the echo command.</div></div> <div>Shell script to find the greatest of two numbers</div> <div><div>Code</div></div>

Codefundo (/)		
Shell Script	✕	
Full Course	Shell Script	<pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " #Read the input from the user and store it in a variable read num1</pre>
	(/shell-script/shell-script-examples.html)	<pre># Prompt the user to enter the second number echo "Enter the second number: "</pre>
	Basics	<pre># Read the input from the user and store it in another variable read num2</pre>
	Operators	
	Conditional Statements	<pre># Use if-else statements to find the greatest of two numbers if [\$num1 -gt \$num2] then echo "\$num1 is the greatest number" elif [\$num2 -gt \$num1] then echo "\$num2 is the greatest number" else echo "Both numbers are equal" fi</pre>
	Loops	
	Command Line Arguments	
	Functions	
	Useful Scripts	
		<p>Output</p> <pre>Enter the first number: 10 Enter the second number: 20 20 is the greatest number</pre>
<h2>Example 17: Shell script to find the largest of three numbers</h2> <p>In this article, we will learn how to write a shell script to find the largest of three numbers.</p> <p>Algorithm 1:</p> <ol style="list-style-type: none">Prompt the user to enter the first number.Read the input from the user using the read command and store it in a variable, say, 'a'.Prompt the user to enter the second number.Read the input from the user using the read command and store it in another variable, say, 'b'.Prompt the user to enter the third number.Read the input from the user using the read command and store it in another variable, say, 'c'.Compare the three numbers and find the largest one.Display the result to the user using the echo command. <p>Shell script to find the largest of three numbers</p> <p>Code</p> <pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read a # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in another variable read b # Prompt the user to enter the third number echo "Enter the third number: " # Read the input from the user and store it in another variable read c # Compare the three numbers and find the largest one if [\$a -gt \$b] && [\$a -gt \$c] then largest=\$a elif [\$b -gt \$a] && [\$b -gt \$c] then largest=\$b else largest=\$c fi # Display the result to the user echo "The largest of \$a, \$b, and \$c is: \$largest"</pre> <p>Output</p>		

```
Enter the first number:
10
Enter the second number:
20
Enter the third number:
15
The largest of 10, 20, and 15 is: 20
```

Algorithm 2:

1. Prompt the user to enter the first number.
2. Read the input from the user using the read command and store it in a variable, say, 'a'.
3. Prompt the user to enter the second number.
4. Read the input from the user using the read command and store it in another variable, say, 'b'.
5. Prompt the user to enter the third number.
6. Read the input from the user using the read command and store it in another variable, say, 'c'.
7. Compare 'a' and 'b', and store the larger number in a variable, say, 'max'.
8. Compare 'max' and 'c', and store the larger number in 'max'.
9. Display the largest number to the user using the echo command.

Shell script to find the largest of three numbers using if-else

Code

```
#!/bin/bash
# Prompt the user to enter the first number
echo "Enter the first number: "

# Read the input from the user and store it in a variable
read a

# Prompt the user to enter the second number
echo "Enter the second number: "

# Read the input from the user and store it in another variable
read b

# Prompt the user to enter the third number
echo "Enter the third number: "

# Read the input from the user and store it in another variable
read c

# Initialize the maximum value to the first number
max=$a

# Compare 'max' and 'b', and update 'max' if 'b' is greater
if [ $b -gt $max ]
then
max=$b
fi

# Compare 'max' and 'c', and update 'max' if 'c' is greater
if [ $c -gt $max ]
then
max=$c
fi

# Display the largest number to the user
echo "The largest number among $a, $b, and $c is: $max"
```

Output

```
Enter the first number:
10
Enter the second number:
30
Enter the third number:
20
The largest number among 10, 30, and 20 is: 30
```

Example 18: Shell script to find the smallest of three numbers

In this article, we will learn how to find the smallest of three numbers using a shell script. The script will prompt the user to enter three numbers and then output the smallest number among them.

Algorithm:

The following is the algorithm to find the smallest of three numbers:

1. Prompt the user to enter the first number.
2. Read the input from the user using the read command and store it in a variable, say, 'num1'.
3. Prompt the user to enter the second number.

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

4. Read the input from the user using the read command and store it in another variable, say, 'num2'.
5. Prompt the user to enter the third number.
6. Read the input from the user using the read command and store it in another variable, say, 'num3'.
7. Use the if-else statement to compare the three numbers. If 'num1' is less than 'num2' and 'num1' is less than 'num3', then 'num1' is the smallest. If 'num2' is less than 'num1' and 'num2' is less than 'num3', then 'num2' is the smallest. Otherwise, 'num3' is the smallest.
8. Display the result to the user using the echo command.

Shell script to find the smallest of three numbers

Code

```
#!/bin/bash
# Prompt the user to enter the first number
echo "Enter the first number: "
# Read the input from the user and store it in a variable
read num1
# Prompt the user to enter the second number
echo "Enter the second number: "
# Read the input from the user and store it in another variable
read num2
# Prompt the user to enter the third number
echo "Enter the third number: "
# Read the input from the user and store it in another variable
read num3
# Use if-else statements to find the smallest number
if [ $num1 -lt $num2 ] && [ $num1 -lt $num3 ]
then
echo "$num1 is the smallest number"
elif [ $num2 -lt $num1 ] && [ $num2 -lt $num3 ]
then
echo "$num2 is the smallest number"
else
echo "$num3 is the smallest number"
fi
```

Output

```
Enter the first number:
15
Enter the second number:
25
Enter the third number:
10
10 is the smallest number
```

Example 19: Shell script to find largest and smallest of three numbers

In this article, we will learn how to write a shell script to find the largest and smallest of three numbers.


Algorithm:


The following steps are required to write a shell script to find the largest and smallest of three numbers:

1. Prompt the user to enter the three numbers.
2. Read the input from the user using the read command and store it in three variables, say, 'num1', 'num2', and 'num3'.
3. Use the 'if-else' statement to compare the three numbers to find the largest and smallest number.
4. Display the largest and smallest number to the user using the echo command.

Shell script to find largest and smallest of three numbers:

Code

Codefundo (/)		
Shell Script		
Full Course Shell Script (/shell-script/shell-script-examples.html) Basics Operators Conditional Statements Loops Command Line Arguments Functions Useful Scripts	<pre>#!/bin/bash # Prompt the user to enter three numbers echo "Enter the first number: " read num1 echo "Enter the second number: " read num2 echo "Enter the third number: " read num3 # Find the largest number if [\$num1 -gt \$num2] && [\$num1 -gt \$num3] then largest=\$num1 elif [\$num2 -gt \$num1] && [\$num2 -gt \$num3] then largest=\$num2 else largest=\$num3 fi # Find the smallest number if [\$num1 -lt \$num2] && [\$num1 -lt \$num3] then smallest=\$num1 elif [\$num2 -lt \$num1] && [\$num2 -lt \$num3] then smallest=\$num2 else smallest=\$num3 fi # Display the largest and smallest number echo "Largest number is \$largest" echo "Smallest number is \$smallest"</pre>	
	<h2>Output</h2>	
	<p>Sample Output</p>	
	<pre>Enter the first number: 10 Enter the second number: 20 Enter the third number: 15 Largest number is 20 Smallest number is 10</pre>	
	<p>In the above code, we have used the 'if-else' statement to compare the three numbers to find the largest and smallest number. The '-gt' option is used to check if a number is greater than the other number. Similarly, '-lt' can be used to check if a number is less than the other number.</p>	
	<h2>Example 20: Shell script to find the largest of four numbers</h2>	
	<p>In this article, we will learn how to write a shell script to find the largest of four numbers.</p>	
	<p>Algorithm:</p>	
	<ol style="list-style-type: none">Prompt the user to enter the first number.Read the input from the user using the read command and store it in a variable, say, 'num1'.Prompt the user to enter the second number.Read the input from the user using the read command and store it in another variable, say, 'num2'.Prompt the user to enter the third number.Read the input from the user using the read command and store it in another variable, say, 'num3'.Prompt the user to enter the fourth number.Read the input from the user using the read command and store it in another variable, say, 'num4'.Use if-else statements to compare the four numbers and find the largest among them.Display the largest number to the user using the echo command.	
	<p>Shell script to find the largest of four numbers</p>	
	<p>Code</p>	

Codefundo (/)		
Shell Script		
Full Course	Shell Script	(/shell-script/shell-script-examples.html)
	Basics	
	Operators	
	Conditional Statements	
	Loops	
	Command Line Arguments	
	Functions	
	Useful Scripts	
		<pre>#!/bin/bash # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read num1 # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in a variable read num2 # Prompt the user to enter the third number echo "Enter the third number: " # Read the input from the user and store it in a variable read num3 # Prompt the user to enter the fourth number echo "Enter the fourth number: " # Read the input from the user and store it in a variable read num4 # Use if-else statements to compare the four numbers if [\$num1 -gt \$num2] && [\$num1 -gt \$num3] && [\$num1 -gt \$num4] then echo "\$num1 is the largest number." elif [\$num2 -gt \$num1] && [\$num2 -gt \$num3] && [\$num2 -gt \$num4] then echo "\$num2 is the largest number." elif [\$num3 -gt \$num1] && [\$num3 -gt \$num2] && [\$num3 -gt \$num4] then echo "\$num3 is the largest number." else echo "\$num4 is the largest number." fi</pre> <div>Output</div> <pre>Enter the first number: 12 Enter the second number: 45 Enter the third number: 33 Enter the fourth number: 67 67 is the largest number.</pre>
		<div>Example 21: Shell script to perform arithmetic operations using switch case</div> <p>In this article, we will learn how to perform arithmetic operations using a case statement in the shell script.</p> <p>Algorithm:</p> <ol style="list-style-type: none">Prompt the user to enter two numbers using the echo command.Store the given numbers in the variables a and b.Prompt the user to choose an arithmetic operation and store the given input in the variable choice.Use a case statement to perform the selected arithmetic operation based on the value of choice. Store the result in the variable result and the arithmetic operator in the variable operator.Finally, use the echo command to display the results of the arithmetic operation. <p>Arithmetic operations using case statement</p> <div>Code</div>

Codefundo (/)		
Shell Script		
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
		<pre>#!/bin/bash # Prompt the user to enter two numbers echo "Enter the first number:" read a echo "Enter the second number:" read b # Prompt the user to choose an arithmetic operation echo "Choose an arithmetic operation:" echo "1. Addition" echo "2. Subtraction" echo "3. Multiplication" echo "4. Division" echo "5. Modulo" read choice # Perform arithmetic operation based on user's choice case \$choice in 1) result=\$((a + b)) operator="+" ;; 2) result=\$((a - b)) operator="-" ;; 3) result=\$((a * b)) operator="*" ;; 4) result=\$((a / b)) operator="/" ;; 5) result=\$((a % b)) operator="%" ;; *) echo "Invalid choice" exit 1 ;; esac # Display the result echo "\$a \$operator \$b = \$result"</pre> <p>Output</p> <pre>Enter the first number: 100 Enter the second number: 10 Choose an arithmetic operation: 1. Addition 2. Subtraction 3. Multiplication 4. Division 5. Modulo 1 100 + 10 = 110 ----- Enter the first number: 100 Enter the second number: 10 Choose an arithmetic operation: 1. Addition 2. Subtraction 3. Multiplication 4. Division 5. Modulo 3 100 * 10 = 1000 ----- Enter the first number: 100 Enter the second number: 10 Choose an arithmetic operation: 1. Addition 2. Subtraction 3. Multiplication 4. Division 5. Modulo 6 Invalid choice</pre>
Example 22: Shell script to find the power of a given number		

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

Example 22: Shell script to find the power of a given number.

In this article, we will learn how to find the power of a given number in a shell script.

Algorithm:

1. Prompt the user to enter the base number.
2. Read the input from the user using the read command and store it in a variable, say, 'base'.
3. Prompt the user to enter the exponent value.
4. Read the input from the user using the read command and store it in another variable, say, 'exp'.
5. Use a for loop to calculate the power of the base number. Initialize a variable, say, 'result' to 1. Loop through the exponent value and multiply the result with the base number for each iteration.
6. Display the result to the user using the echo command.

Shell script to find the power of a given number

Code

```
#!/bin/bash
# Prompt the user to enter the base number
echo "Enter the base number: "
# Read the input from the user and store it in a variable
read base

# Prompt the user to enter the exponent value
echo "Enter the exponent value: "

# Read the input from the user and store it in another variable
read exp

# Calculate the power of the base number using a for loop
result=1
for ((i=0; i<$exp; i++))
do
    result=$((result*$base))
done

# Display the result to the user
echo "$base raised to the power of $exp is: $result"
```

Output:

Sample Output

```
Enter the base number:
2
Enter the exponent value:
5
2 raised to the power of 5 is: 32
```

In the above code, we have used a for loop to calculate the power of the base number. The loop runs for the number of times specified by the exponent value, and in each iteration, the result is multiplied with the base number.

Example 23: Shell script to print numbers from 1 to 100

In this article, we will learn how to write a shell script to print numbers from 1 to 100.

Algorithm:

1. Use a for loop to iterate from 1 to 100.
2. Display each number using the echo command.

Shell script to print numbers from 1 to 100

Code

```
#!/bin/bash
# Use a for loop to iterate from 1 to 100
for ((i=1;i<=100;i++))
do
    # Display each number using the echo command
    echo $i
done
```

Output

```
1
2
3
...
97
98
99
100
```

In the above code, we have used a for loop to iterate from 1 to 100 and the echo command to display each

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

In the above code, we have used a for loop to iterate from 1 to 100 and the echo command to display each number. The loop starts with 'i=1' and continues until 'i<=100'. The increment operator 'i++' is used to increase the value of 'i' by 1 in each iteration.

Example 24: Shell script to display even numbers from 1 to 100

In this article, we will learn how to display even numbers from 1 to 100 in a shell script.

Algorithm:

1. Use a for loop to iterate through numbers 1 to 100.
2. Use the modulus operator to check if the current number is even.
3. If the number is even, display it to the user using the echo command.

Shell script to display even numbers from 1 to 100

Code

```
#!/bin/bash
# Use a for loop to iterate through numbers 1 to 100
for ((i=1; i<=100; i++))
do
    # Use the modulus operator to check if the current number is even
    if [ $((i%2)) -eq 0 ]
    then
        # If the number is even, display it to the user
        echo $i
    fi
done
```

Output

2
4
6
8
10
...
94
96
98
100

Example 25: Shell script to display odd numbers from 1 to 100

In this article, we will learn how to write a shell script to display odd numbers from 1 to 100.

Algorithm:

1. Use a for loop to iterate through the numbers from 1 to 100.
2. Use an if statement to check if the current number is odd.
3. If the number is odd, print it to the console using the echo command.

Shell script to display odd numbers from 1 to 100

Code

```
#!/bin/bash
# Use for loop to iterate through numbers from 1 to 100
for (( i=1; i<=100; i++ ))
do
    # Use if statement to check if the number is odd
    if [ $((i%2)) -ne 0 ]
    then
        # If the number is odd, print it to the console
        echo $i
    fi
done
```

Output

1
3
5
7
9
11
...
95
97
99

Example 26: Shell script to find the smallest of n numbers

In this article, we will learn how to find the smallest of n numbers in a shell script without using an array.



Full Course
Shell Script
(/shell-script/shell-script-examples.html)
Basics
Operators
Conditional Statements
Loops
Command Line Arguments
Functions
Useful Scripts

Algorithm:

1. Prompt the user to enter the number of elements (n).
2. Read the input from the user using the read command and store it in a variable, say, 'n'.
3. Initialize a variable, say, 'min' with the maximum possible value (2^31-1).
4. Use a loop to read n numbers from the user and compare them with the current minimum value. If the current number is smaller than the current minimum value, set the minimum value to the current number.
5. Display the result to the user using the echo command.

Shell script to find the smallest of n numbers

Code

```
#!/bin/bash
# Prompt the user to enter the number of elements
echo "Enter the number of elements: "
# Read the input from the user and store it in a variable
read n

# Initialize the minimum value with the maximum possible value
min=2147483647

# Loop to read n numbers from the user and find the minimum value
for ((i=1;i<=n;i++))
do
echo "Enter number $i: "
read num
if [ $num -lt $min ]
then
min=$num
fi
done

# Display the minimum value to the user
echo "The smallest number is: $min"
```

Output

```
Enter the number of elements:
5
Enter number 1:
100
Enter number 2:
50
Enter number 3:
20
Enter number 4:
80
Enter number 5:
10
The smallest number is: 10
```

Example 27: Shell script to find the largest of n numbers


In this article, we will learn how to find the largest of n numbers using a shell script.

Algorithm:

1. Prompt the user to enter the value of 'n', i.e., the total number of elements to be compared.
2. Read the input from the user using the read command and store it in a variable, say, 'n'.
3. Initialize a variable 'largest' to 0.
4. Use a for loop to iterate 'n' times.
5. Inside the loop, prompt the user to enter the value of the current element.
6. Read the input from the user and store it in a variable, say, 'num'.
7. Use an if statement to compare 'num' with 'largest'. If 'num' is greater than 'largest', assign the value of 'num' to 'largest'.
8. After the loop ends, display the value of 'largest' using the echo command.

Shell script to find the largest of n numbers

Code

Codefundo (/)		
Shell Script		
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
		<pre>#!/bin/bash # Prompt the user to enter the total number of elements echo "Enter the total number of elements: " # Read the input from the user and store it in a variable read n # Initialize a variable 'largest' to 0 largest=0 # Use a for loop to iterate 'n' times for ((i=1; i<=n; i++)) do # Prompt the user to enter the value of the current element echo "Enter element \$i: " # Read the input from the user and store it in a variable read num # Use an if statement to compare 'num' with 'largest' if [\$num -gt \$largest] then # If 'num' is greater than 'largest', assign the value of 'num' to 'largest' largest=\$num fi done # Display the value of 'largest' using the echo command echo "The largest number is: \$largest"</pre> <div>Output</div> <div>Enter the total number of elements: 5 Enter element 1: 10 Enter element 2: 20 Enter element 3: 30 Enter element 4: 40 Enter element 5: 50 The largest number is: 50</div>
Example 28: Shell script to find the factorial of a number		
In this article, we will learn how to find the factorial of a given number using a shell script.		
Algorithm:		
1. Prompt the user to enter a number.		
2. Read the input from the user using the read command and store it in a variable, say, 'num'.		
3. Initialize a variable, say, 'factorial' to 1.		
4. Use a for loop to iterate from 1 to 'num'. For each iteration, multiply the 'factorial' variable with the loop counter.		
5. Display the result to the user using the echo command.		
Shell script to find the factorial of a number		
Code		<pre>#!/bin/bash # Prompt the user to enter a number echo "Enter a number: " # Read the input from the user and store it in a variable read num # Initialize the 'factorial' variable to 1 factorial=1 # Use a for loop to find the factorial of the number for ((i=1;i<=num;i++)) do factorial=\$((factorial*i)) done # Display the result to the user echo "The factorial of \$num is \$factorial"</pre>
Output		
Enter a number: 5		

Shell Script

✕

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Operators

Conditional Statements

Loops

Command Line Arguments

Functions

Useful Scripts

Example 29: Shell script to find the sum of the digits of a number

In this article, we will learn how to find the sum of the digits of a number using a shell script.

Algorithm:

- Prompt the user to enter a number.
- Read the input from the user using the read command and store it in a variable, say, 'num'.
- Initialize a variable, say, 'sum' to 0.
- Use a while loop to find the sum of the digits of the number. Inside the loop, extract the last digit of the number using the modulo operator (%) and add it to the sum. Then, remove the last digit of the number using integer division (/).
- Repeat step 4 until the number becomes 0.
- Display the sum to the user using the echo command.

Shell script to find the sum of the digits of a number

Code

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number: "
# Read the input from the user and store it in a variable
read num

# Initialize the sum variable to 0
sum=0

# Use a while loop to find the sum of the digits
while [ $num -gt 0 ]
do

# Get the last digit of the number and add it to the sum
digit=$((num % 10))
sum=$((sum + digit))

# Remove the last digit of the number
num=$((num / 10))
done

# Display the sum to the user
echo "The sum of the digits is: $sum"
```

Output

```
Enter a number:
12345
The sum of the digits is: 15
```

Example 30: Shell script to reverse a number

In this article, we will learn how to reverse a number in a shell script.

Algorithm:

- Prompt the user to enter a number.
- Read the input from the user using the read command and store it in a variable, say, 'num'.
- Initialize a variable, say, 'rev' to 0.
- Use a while loop to reverse the number. In each iteration of the loop, extract the last digit of the number using the modulo operator and add it to 'rev'. Multiply 'rev' by 10 and add the result to the last digit. Divide the number by 10 and store the result in the same variable, i.e., 'num'. Repeat until 'num' becomes 0.
- Display the reversed number to the user using the echo command.

Shell script to reverse a number

Code

Codefundo (/)		
Shell Script	✕	
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
<div>#!/bin/bash# Prompt the user to enter a numberecho "Enter a number: "# Read the input from the user and store it in a variableread num# Initialize the reverse variable to 0rev=0# Use a while loop to reverse the numberwhile [\$num -gt 0]do# Extract the last digit of the number using the modulo operatordigit=\$((\$num % 10))# Add the last digit to the reverse variablerev=\$((\$rev * 10 + \$digit))# Divide the number by 10 and store the result in the same variablenum=\$((\$num / 10))done# Display the reversed number to the userecho "The reversed number is: \$rev"</div> <div>Output</div> <div>Enter a number: 1234 The reversed number is: 4321</div>		
<div>Example 31: Shell script to check whether a given number is palindrome or not</div> <div>In this article, we will learn how to write a shell script to check whether a given number is palindrome or not.</div> <div>Algorithm:</div> <div>1. Prompt the user to enter a number.</div> <div>2. Read the input from the user using the read command and store it in a variable, say 'num'.</div> <div>3. Reverse the number 'num'.</div> <div>4. Compare the reversed number with the original number 'num'. If they are equal, then the number is a palindrome, else it is not a palindrome.</div> <div>5. Display the result to the user using the echo command.</div> <div>Shell script to check whether a given number is palindrome or not</div> <div>Code</div> <div>#!/bin/bashecho "Enter a number: "read num# Store the original numberorig=\$num# Reverse the numberrev=0while [\$num -gt 0]dorem=\$((\$num % 10))rev=\$((\$rev * 10 + \$rem))num=\$((\$num / 10))done# Check whether the number is palindrome or notif [\$orig -eq \$rev]thenecho "\$orig is a palindrome number."elseecho "\$orig is not a palindrome number."fi</div> <div>Output</div> <div>Enter a number: 12321 12321 is a palindrome number.</div>		
<div>Example 32: Shell script to print the Fibonacci series</div>		

Shell Script	✕
Full Course	
Shell Script	
(/shell-script/shell-script-examples.html)	
Basics	
Operators	
Conditional Statements	
Loops	
Command Line Arguments	
Functions	
Useful Scripts	

Example 32: Shell script to print the Fibonacci series

In this article, we will learn how to write a shell script to print the Fibonacci series.

Algorithm:

- Prompt the user to enter the number of terms in the Fibonacci series.
- Read the input from the user using the read command and store it in a variable, say, 'n'.
- Initialize two variables 'a' and 'b' with 0 and 1 respectively.
- Use a for loop to iterate from 1 to 'n'.
- In each iteration, print the value of 'a'.
- Calculate the next term in the Fibonacci series by adding 'a' and 'b', and store it in a temporary variable, say, 'temp'.
- Assign the value of 'b' to 'a' and the value of 'temp' to 'b'.
- Continue the loop until 'n' terms have been printed.

Shell script to print the Fibonacci series

Code

```
#!/bin/bash
# Prompt the user to enter the number of terms in the Fibonacci series
echo "Enter the number of terms in the Fibonacci series: "
Read the input from the user and store it in a variable
read n

Initialize the first two terms in the series
a=0
b=1

Print the first term
echo "Fibonacci series: "
echo -n "$a "

Print the next n-1 terms in the series
for (( i=2; i<=n; i++ ))
do
    # Print the current term
    echo -n "$b "
    # Calculate the next term and update the variables
    temp=$b
    b=$((a+b))
    a=$temp
done

echo ""
```

Output:

Output

```
Enter the number of terms in the Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
```

Example 33: Shell script to check whether a given number is prime or not


In this article, we will learn how to check whether a given number is prime or not in a shell script.

Algorithm:

- Take the input number from the user using the read command.
- Use a for loop to iterate from 2 to the number-1.
- Check if the input number is divisible by any number in the loop.
- If the number is divisible by any number in the loop, then it is not a prime number.
- If the number is not divisible by any number in the loop, then it is a prime number.

Shell script to check whether a given number is prime or not

Code

Codefundo (/)		
Shell Script		
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
<pre>#!/bin/bash # Take the input number from the user read -p "Enter a number: " num # Initialize the flag to 1 flag=1 # Use a for loop to iterate from 2 to the number-1 for ((i=2; i<\$num; i++)) do # Check if the input number is divisible by any number in the loop if [\$((num%i)) -eq 0] then # If the number is divisible by any number in the loop, then it is not a prime number flag=0 break fi done # Check the flag value to determine whether the number is prime or not if [\$flag -eq 1] then echo "\$num is a prime number" else echo "\$num is not a prime number" fi</pre>		
Output		
Enter a number: 13 13 is a prime number		
<div>Example 34: Shell script to check if two given numbers are amicable</div> <p>In this article, we will learn how to check if two given numbers are amicable in a shell script.</p> <p>Algorithm:</p> <ol style="list-style-type: none">Take two numbers as input from the user.Define a function that calculates the sum of all divisors of the given number.Calculate the sum of divisors of both the given numbers using the function defined in step 2.Check if the sum of divisors of one number is equal to the other number and vice versa. If true, then the given numbers are amicable. <p>Shell script to check if two given numbers are amicable</p> <div>Code</div> <pre>#!/bin/bash # Function to calculate the sum of divisors of the given number function sum_divisors { n=\$1 sum=0 for ((i=1; i<n; i++)) do if [\$((n%i)) -eq 0] then sum=\$((sum+i)) fi done echo \$sum } # Take two numbers as input from the user echo "Enter first number:" read num1 echo "Enter second number:" read num2 # Calculate the sum of divisors of both the given numbers sum1=\$(sum_divisors \$num1) sum2=\$(sum_divisors \$num2) # Check if the sum of divisors of one number is equal to the other number and vice versa if [\$sum1 -eq \$num2] && [\$sum2 -eq \$num1] then echo "\$num1 and \$num2 are amicable numbers." else echo "\$num1 and \$num2 are not amicable numbers." fi</pre> <div>Output</div>		


```
Enter first number:
220
Enter second number:
284
220 and 284 are amicable numbers.
```

In the above example, the numbers 220 and 284 are amicable as the sum of proper divisors of 220 ($1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$) is equal to 284 and the sum of proper divisors of 284 ($1 + 2 + 4 + 71 + 142 = 220$) is equal to 220.

Shell script to check if two given numbers are amicable – without using function

Code

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

```
# Read the two numbers from the user
read -p "Enter first number: " num1
read -p "Enter second number: " num2
```

```
# Find the proper divisors of the first number and calculate their sum
sum1=0
for ((i=1; i<num1; i++))
do
    if [ $(num1 % i) -eq 0 ]
    then
        sum1=$((sum1 + i))
    fi
done
```

```
# Find the proper divisors of the second number and calculate their sum
sum2=0
for ((i=1; i<num2; i++))
do
    if [ $(num2 % i) -eq 0 ]
    then
        sum2=$((sum2 + i))
    fi
done
```

```
# Check if the two numbers are amicable or not
if [ $sum1 -eq $num2 ] && [ $sum2 -eq $num1 ]
then
    echo "The numbers $num1 and $num2 are amicable."
else
    echo "The numbers $num1 and $num2 are not amicable."
fi
```

Output

```
Enter first number:
220
Enter second number:
284
220 and 284 are amicable numbers.
```

Example 35: Shell script to add two numbers using command line arguments

In this shell script, we will add two numbers passed as command line arguments.

Algorithm:

1. Check if two arguments are passed, if not, display an error message and exit.
2. Assign the first argument to a variable, say 'num1'.
3. Assign the second argument to another variable, say 'num2'.
4. Add the two numbers using the `$(())` syntax and store the result in a variable, say 'sum'.
5. Display the result to the user using the echo command.

Add two numbers using command line arguments

Code

Codefundo (/)		
Shell Script	✕	
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
		<pre>#!/bin/bash # Check if two arguments are passed if [\$# -ne 2]; then echo "Error: Invalid number of arguments." echo "Usage: \$0 num1 num2" exit 1 fi # Assign the first argument to num1 num1=\$1 # Assign the second argument to num2 num2=\$2 # Add the two numbers sum=\$((num1 + num2)) # Display the result echo "The sum of \$num1 and \$num2 is: \$sum"</pre> <p>Output</p> <pre>sh shell_sample.sh 10 20 The sum of 10 and 20 is: 30</pre> <h3>Code Explanation</h3> <p>In this code, we first check if two arguments are passed using the special parameter <code>\$#</code>. If the number of arguments is not equal to 2, we display an error message and exit the script using the <code>exit</code> command.</p> <p>Then, we assign the first argument to a variable 'num1' and the second argument to another variable 'num2'. We add these two numbers using the <code>\$(())</code> syntax and store the result in a variable 'sum'.</p> <p>Finally, we display the result to the user using the <code>echo</code> command.</p> <h3>Usage:</h3> <p>To use this script, open a terminal and navigate to the directory where the script is located. Then, run the following command:</p> <pre>./script_name.sh num1 num2</pre> <p>Here, 'script_name.sh' is the name of the script file, 'num1' and 'num2' are the two numbers that you want to add.</p>
		<h2>Example 36: Shell script to add two numbers using function</h2> <p>In this article, we will learn how to add two numbers in a shell script using a function.</p> <h3>Algorithm:</h3> <ol style="list-style-type: none">Define a function to add two numbers, say <code>add_numbers</code>.Prompt the user to enter the first number.Read the input from the user using the <code>read</code> command and store it in a variable, say, 'a'.Prompt the user to enter the second number.Read the input from the user using the <code>read</code> command and store it in another variable, say, 'b'.Call the function and pass the two variables as arguments.Display the result to the user using the <code>echo</code> command. <h3>Add two numbers using function</h3> <p>Code</p> <pre>#!/bin/bash # Define the function to add two numbers add_numbers () { sum=\$((a + b)) echo "The sum of \$a and \$b is: \$sum" } # Prompt the user to enter the first number echo "Enter the first number: " # Read the input from the user and store it in a variable read a # Prompt the user to enter the second number echo "Enter the second number: " # Read the input from the user and store it in a variable read b # Call the add_numbers function to add the two numbers add_numbers \$a \$b</pre>

Codefundo (/)	Output	
Shell Script		
	Enter the first number: 10 Enter the second number: 20 The sum of 10 and 20 is: 30	
Full Course		
Shell Script		
(/shell-script/shell-script-examples.html)		
Basics		
Operators		
Conditional Statements		
Loops		
Command Line Arguments		
Functions		
Useful Scripts		
	<div>Code Explanation</div> <p>In this code, we define a function called add_numbers that takes two parameters \$1 and \$2 representing the first and second numbers respectively. The function then adds these two numbers using the \$(()) syntax and stores the result in a variable called sum. Finally, the function prints the result using the echo command.</p> <p>In the main part of the script, we prompt the user to enter the first and second numbers using the read command, and then call the add_numbers function with these two numbers as arguments. The function then adds these two numbers and prints the result to the user.</p>	
	<div>Example 37: Shell script to display your home directory</div> <p>1. To retrieve the home directory path, we can use the \$HOME variable in a shell script.</p> <div>Code</div> <pre>#!/bin/bash echo 'Your home directory is: \$HOME'</pre> <p>2. Save the script with a .sh file extension (e.g., script.sh) and run it in the terminal by typing ./script.sh.</p> <div>Code</div> <pre>./script.sh</pre> <p>3. The script will display your home directory path in the terminal.</p> <div>Output</div> <pre>Your home directory is: /users/username</pre>	
	<div>Example 38: Shell script to print current date and time</div> <p>1. To print the current date and time, we can use the date command in a shell script.</p> <div>Code</div> <pre>#!/bin/bash echo 'The current date and time is: \$(date)'</pre> <p>2. Save the script with a .sh file extension (e.g., script.sh) and run it in the terminal by typing ./script.sh.</p> <div>Code</div> <pre>./script.sh</pre> <p>3. The script will display the current date and time in the terminal.</p> <div>Output</div> <pre>The current date and time is: Sun Feb 26 01:51:35 GMT 2023</pre>	
	<div>Example 39: Get the name of the script inside the shell script</div> <p>We can get the name of the current script in a shell script by using the special variable "\$0". This variable contains the name of the script that is currently being executed.</p> <p>Here is an example of how we can use this variable in a shell script:</p> <div>Code</div>	



```
#!/bin/bash
echo "The name of this script is: $0"
```

Full Course

Shell Script

(/shell-script/shell-script-examples.html)

Basics

Output

Operators

```
sh script.sh
The name of this script is: script.sh
```

Conditional Statements

Loops

Note that if we execute the script using a relative or absolute path, the "\$0" variable will print the script name along with the absolute or relative path.

For example, if we place the script.sh file in a folder called **'temp'** and execute it from the current directory, the output will include both the **path name and the script name**.

Output

```
sh temp/script.sh
The name of this script is: temp/script.sh
```

How to get only the script name?

If we want to get only the name of the script without the path, we can use the **"basename"** command like this:

Code

```
#!/bin/bash
script_name=$(basename "$0")
echo "The name of this script is: $script_name"
```

The "basename" command strips the path from the "\$0" variable and returns only the filename.

Output

```
sh temp/script.sh
The name of this script is: script.sh
```