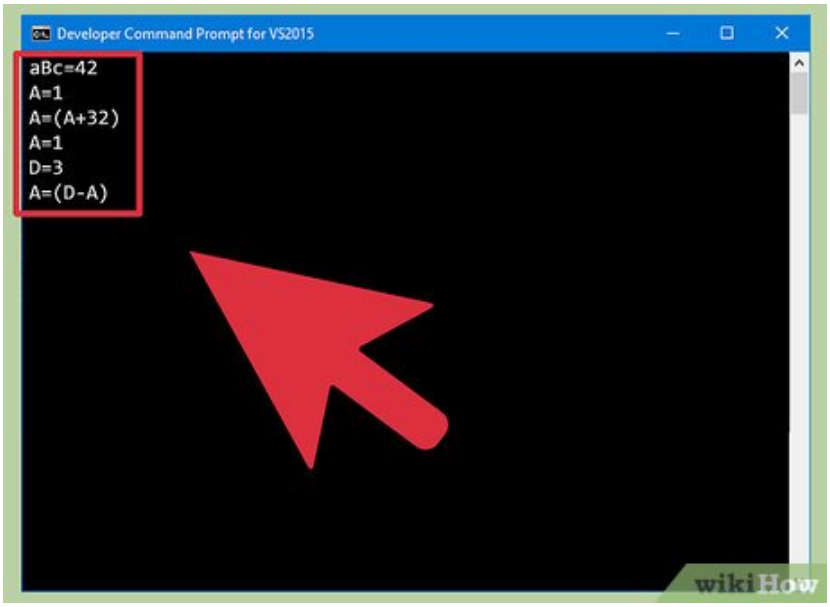
How to Get Started with Lua

Lua is a text-based programming language, which is a lot of fun. The possibilities of this program are virtually endless. It takes a bit of time to get the hang of, but with a bit of practice making and running programs becomes addictive!...

Method 1 of 2:

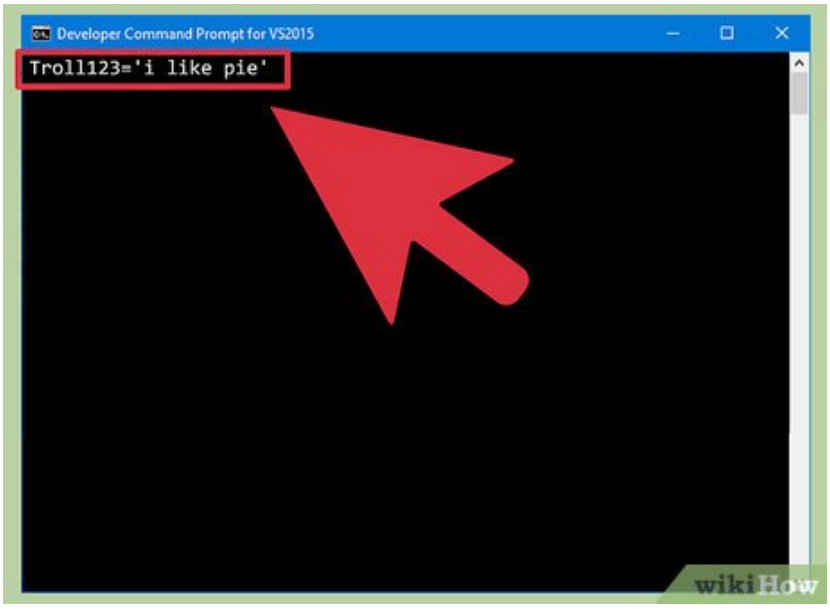
Example

```
aBc=42 (This will make a variable called aBc and will store the number 42 i it. 
```

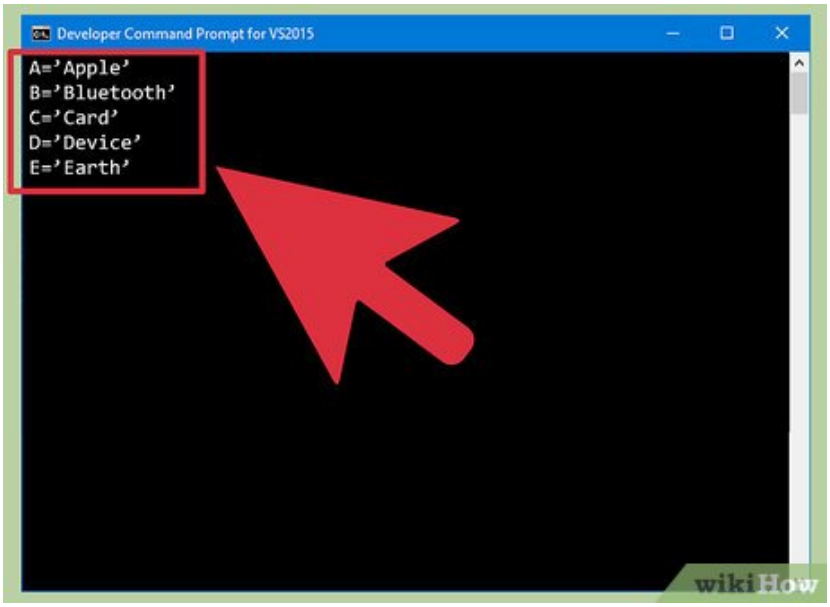
1. 

Variables can be whatever you want, like these examples.

```
Troll123='i like pie' */This will store the string 'i like pie' in variable 'Tro
```

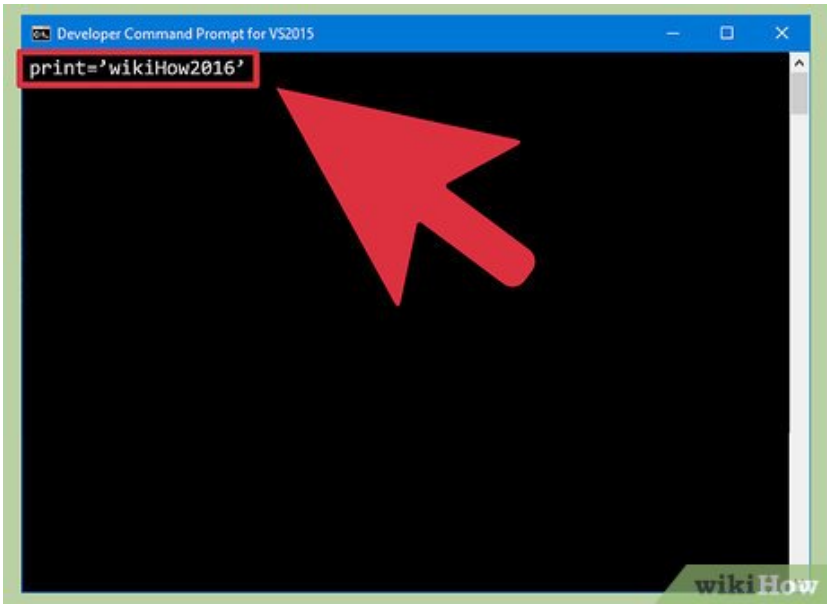
1. 

There are things called strings in lua. These can be stored in variables like numbers, but they are text. All strings must have quotes 'around them.

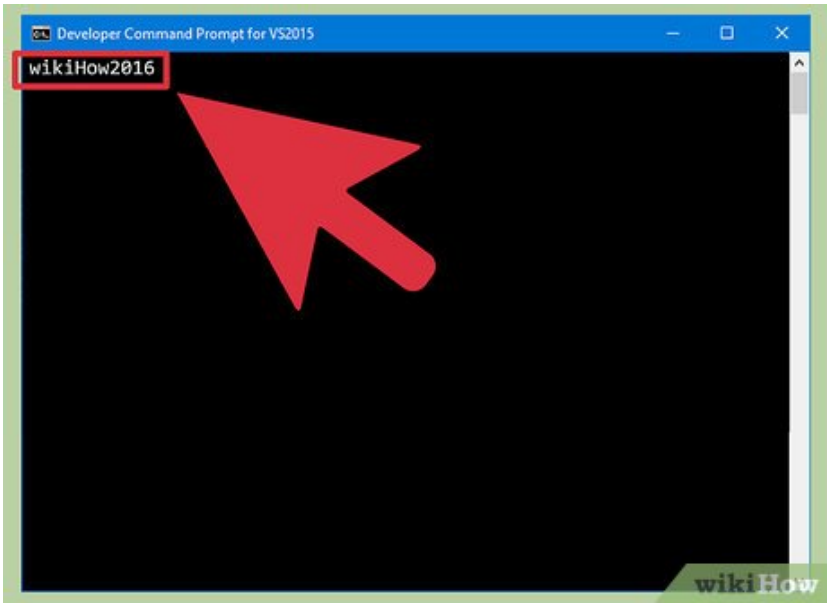
2. 

See these examples:

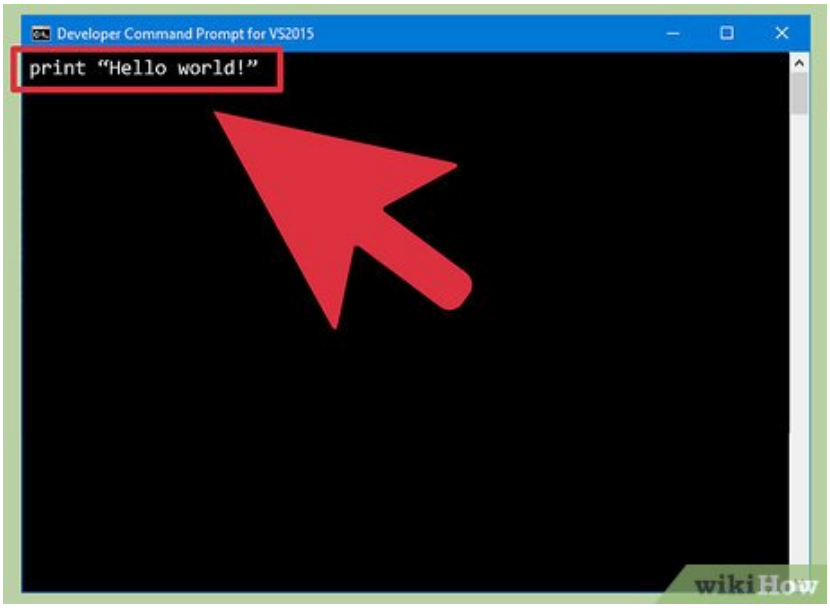
```
A='Some cheese' */This will store the string 'Some cheese' in the variable A. Th
```

1. 

This is the main command for talking to the user in the program. you can print variables or strings.

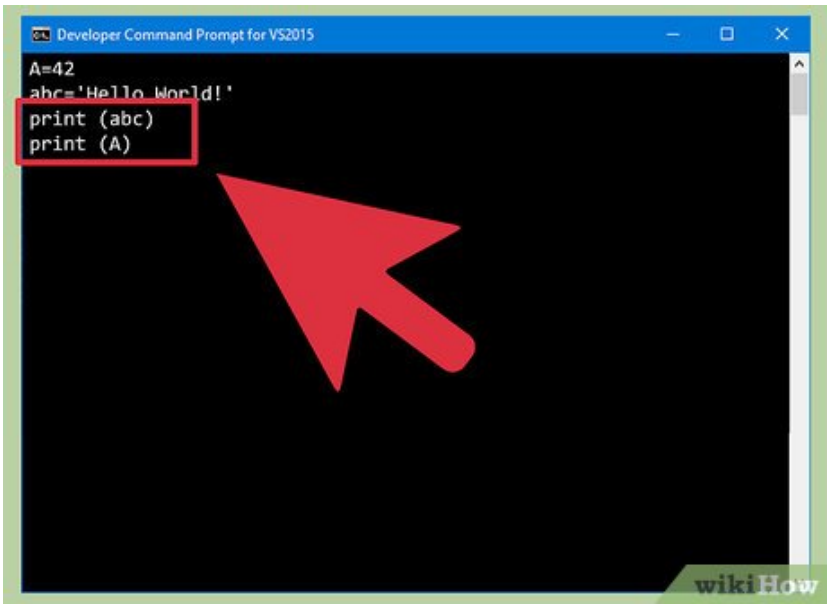
2. 

It will type the text or number you want onto the screen.

3. 

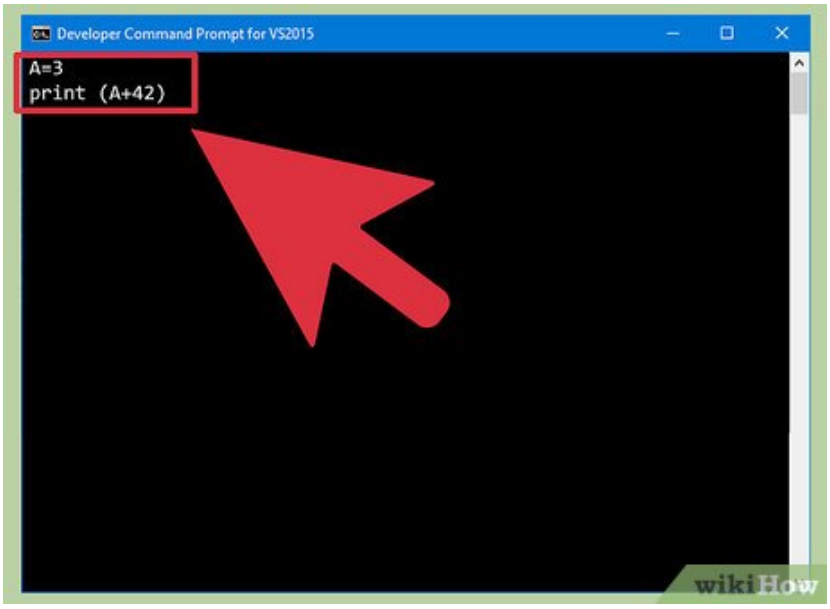
For strings, all you need to do is type 'print' then a space, then the string in quotes, as in this example.

```
print "Hello world!"
```

1. 

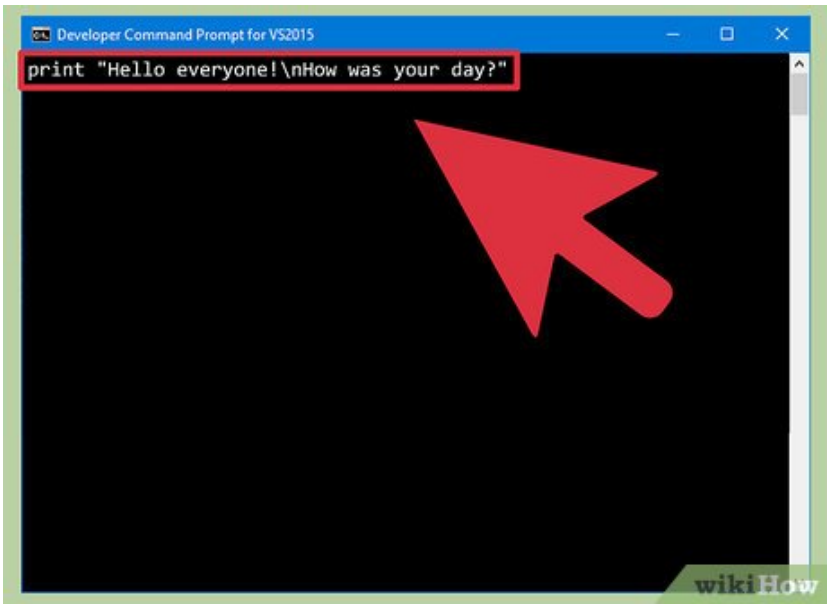
For variables, you need to type 'print' and then the name of the variable in brackets, as shown.

```
A=42 abc='Hello World!' print (abc) print (A)
```

1. 

This will print whatever is in the variable. You don't have to have a set variable, either. You can print an equation, as shown.

```
A=3 print (A+42) */This will print the number 42+3, which is 45./*
```

1. 

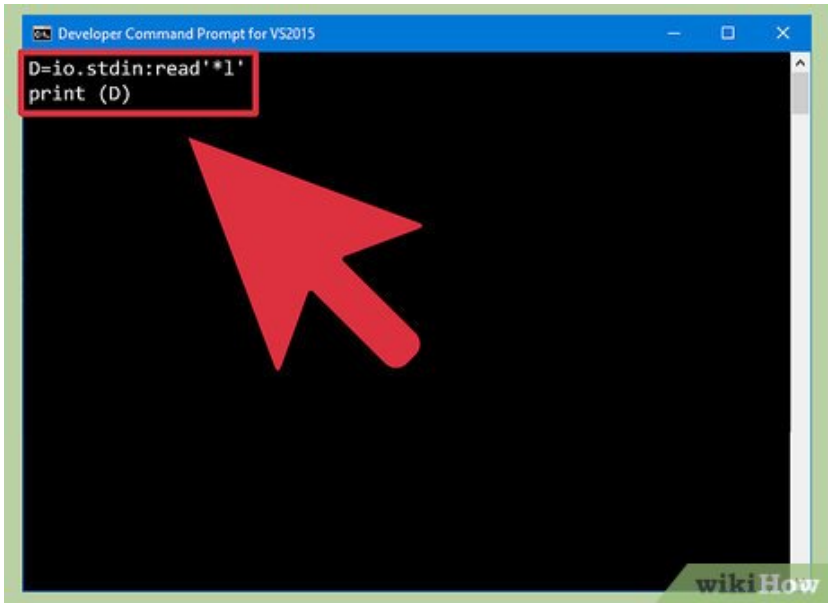
Halfway through a string you want to print, if you type n then it will print the rest of the text on a new line. For example:

```
print "Hello everyone!nHow was your day?" */will print 'Hello everyone!' On one
```

1. 

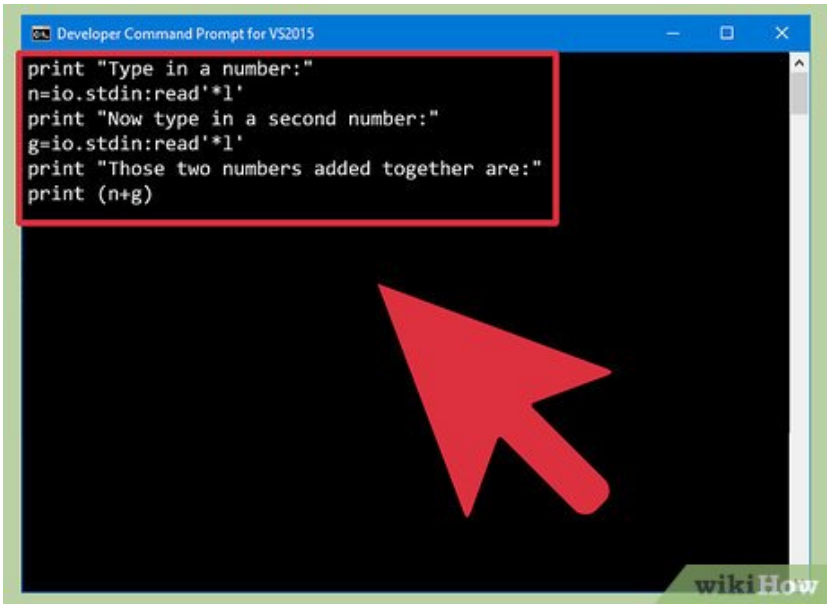
You need a way of letting the user enter information for the program to use. When this is in the program and the computer gets up to it, the computer will stop and not go onto anything else in the code until the user has entered in some information. This is where the user input command comes in. Here is what it looks like:

{{{1}}}

1. 

It looks a bit complicated, but don't worry, it's not. all you have to worry about is the 'A' at the start. This is the variable in which the information is stored. It can be anything, and once it's entered you can do whatever you want with it.

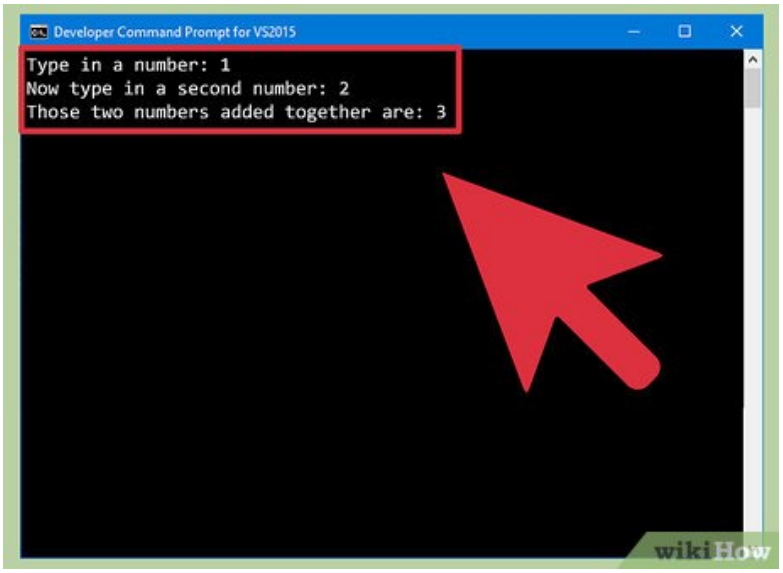
```
D=io.stdin:read'*l' print (D) */This will let you type in whatever you want, and
```

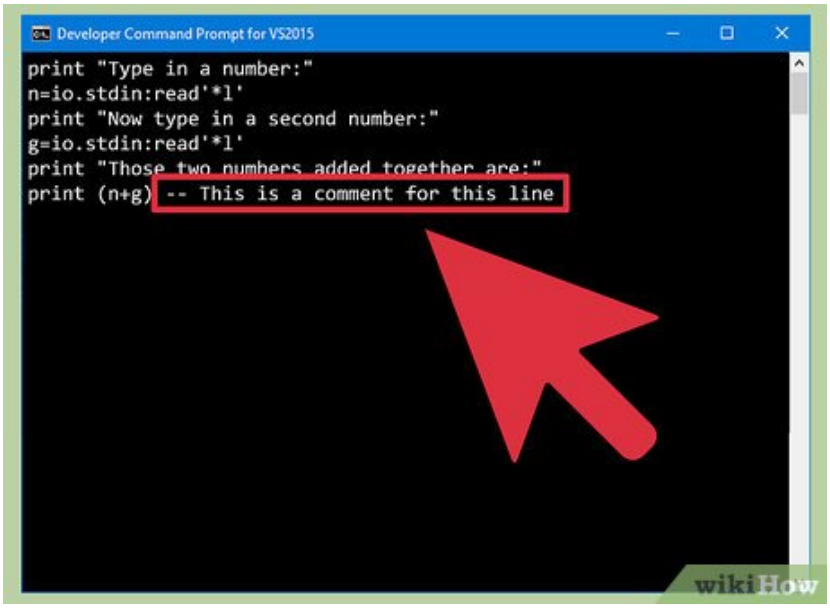
1. 

Try to find out what this program will do, just as some practice.

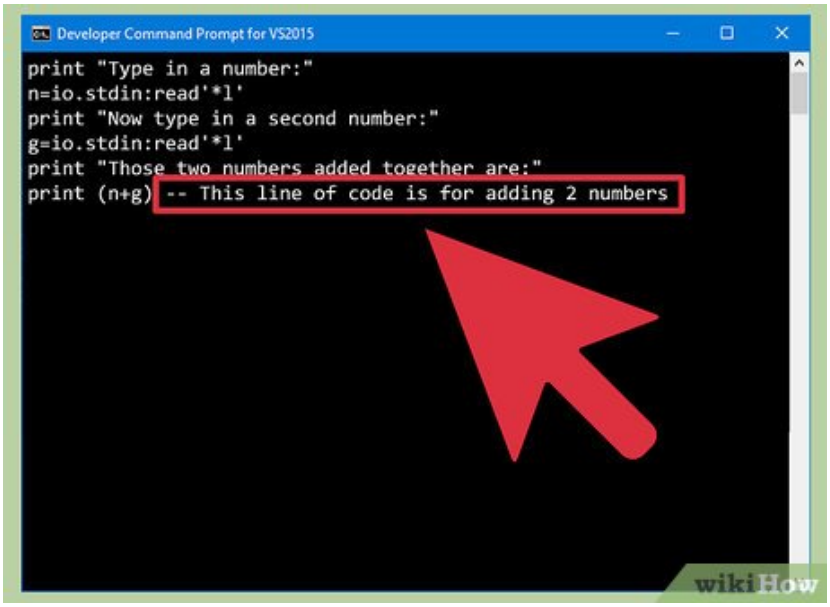
```
print "Type in a number." n=io.stdin:read'*l' print "Now type in a second number
```

1. 1. What does this program do and how does it work?



2. 

Comments are lines of code put into the program to help you know what sections of the code do what. As long as one of your lines of code starts with two dashes -- then the program will ignore whatever comes after it on that line. They are purely there to help you keep track of your program.

3. 

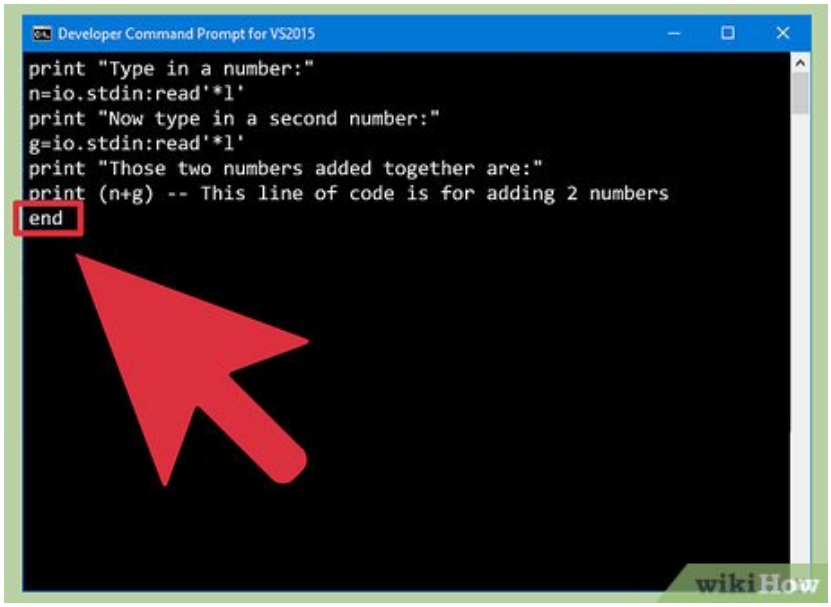
For example, if you wanted to mark a section of code which is used to add two numbers together you would add this comment.

Method 2 of 2:

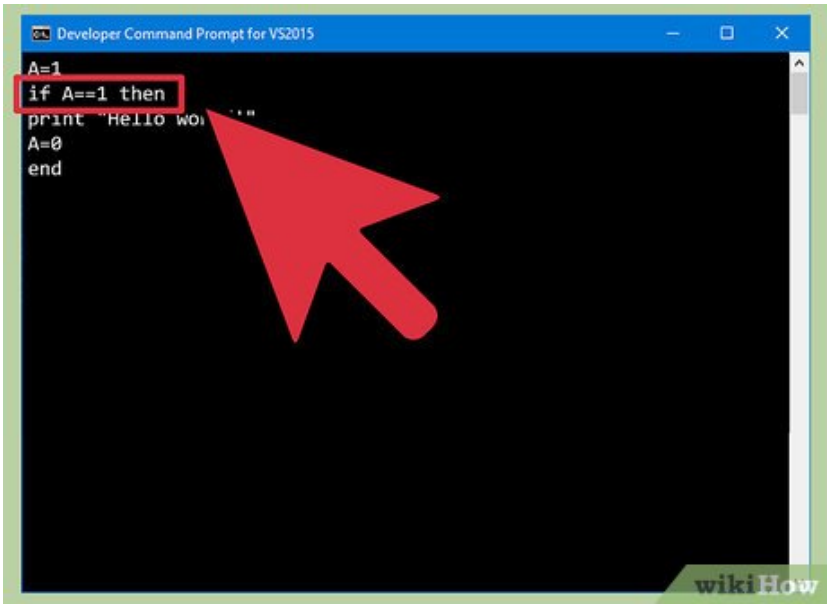
Calculating section

1. 

The computer would ignore this line, as long as it starts with the two dashes.

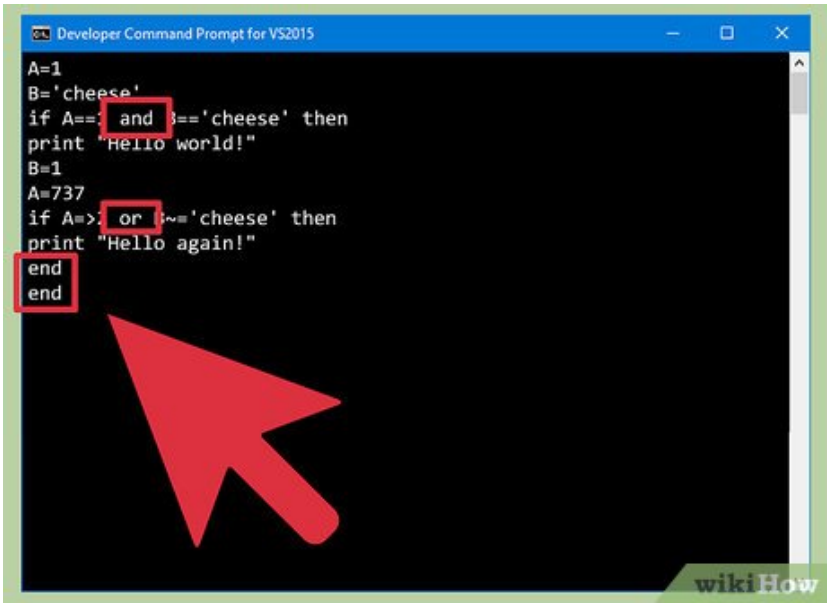
2. 

These are sections of code which allows you to only run that section of code if certain variables are certain things. Whenever you want in your code you can add one of these, and then enter some code into it. It will only do that code if the variables you say are equal to a certain thing. Once you have finished the code in the statement you must type a line of code which simply says 'end' without the quotes and it will tell the computer that that is the end if the code in the statement.

3. 

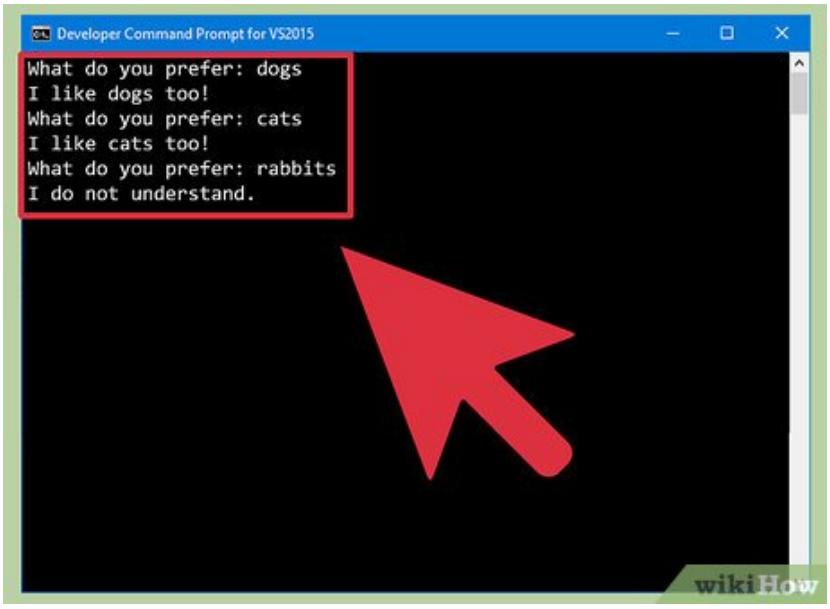
on the line of code that starts with 'if' you then have a space, then the name of a variable, then what operation (== means equal to, ~= means not equal to, => means bigger than etc) then the line must end with 'then'.

```
A=1 if A==1 then print "Hello world!" A=0 end
```

1. 

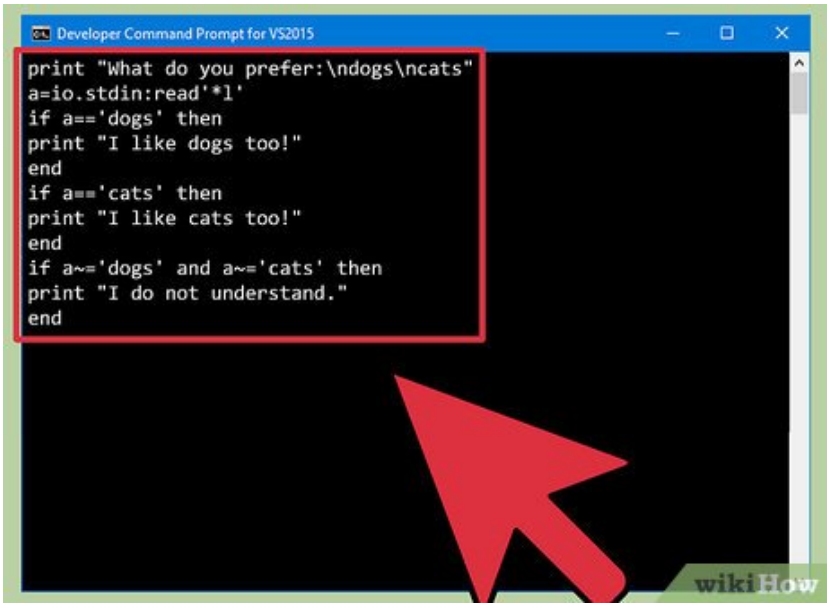
You can even have statements inside other statements, but just be sure you have the right amount of 'end's at the end. You can also have more than one variable, by saying either 'and' or 'or'.

```
A=1 B='cheese' if A==1 and B=='cheese' then print "Hello world!" B=1 A=737 if A=>
```

1. 

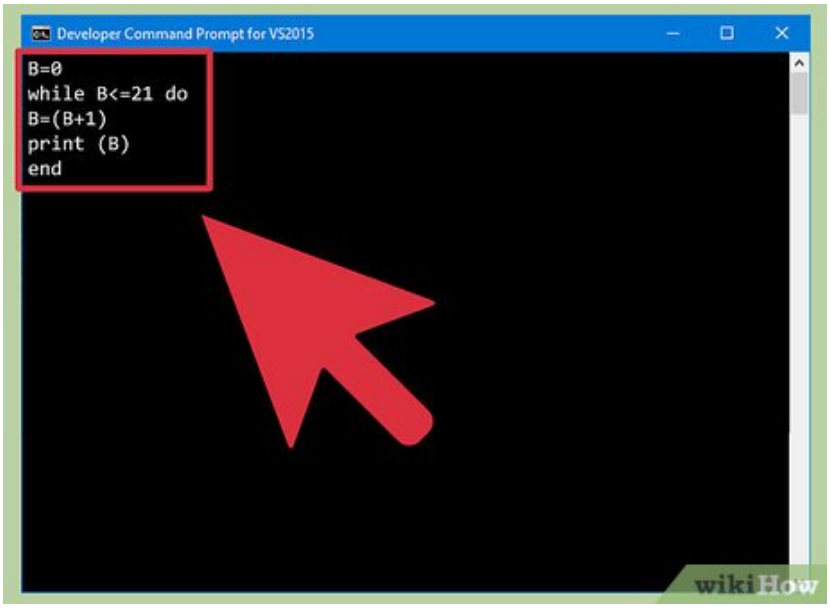
This can be useful for a variety of things, for example this. It will ask you what your favourite animal is, then says it likes that animal too. If you type in something completely different then it will say it doesn't understand you. Try to see how it works.

```
print "What do you prefer:ndogsncats" a=io.stdin:read'*l' if a=='dogs' then print
```

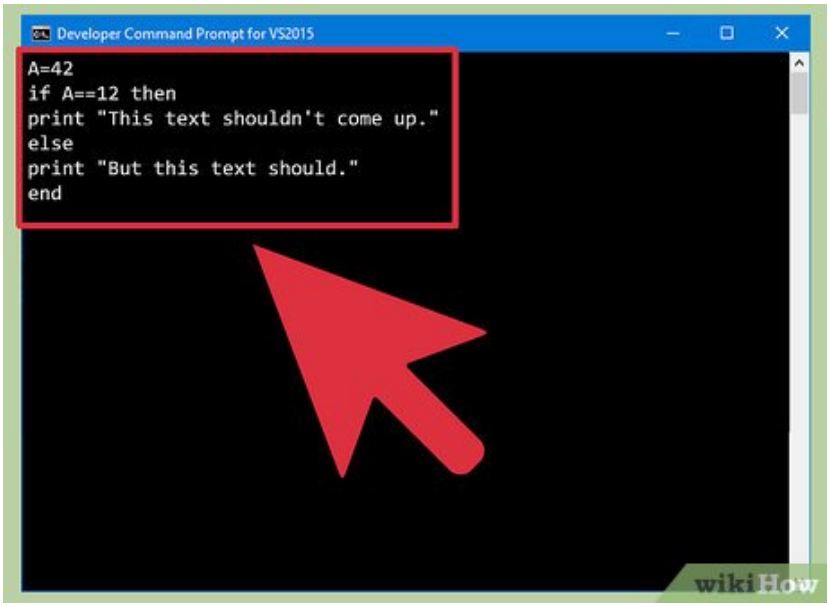
1. 

This is very similar to the 'if' statement, but instead it will keep looping through the code inside it until it is told to stop. This is how you use it.

```
B=0 while B=21 do B=(B+1) print (B) end
```

1. 

This program will print the numbers from 1 to 20 and then it will stop, because it got above 20. A lot of Lua programs are just massive 'while' loops with a whole lot of 'if' statements in them.

2. 

You use this in your 'if' statement, at the end. If the variables don't match to anything on the 'if' line, then it goes onto the 'else' section.

```
A=42 if A==12 then print "This text shouldn't come up." else print "But this tex
```