

Programming a Virtual Automated Teller Machine (ATM)

An Introduction to Computer Programming with JavaScript

By
Omole Omoteniola
Linkedlist Technologies



Preface

This course is self-contained. All resources required to practice and grasp the concepts taught have been made available online freely.

A complete code editor and interactive console (required to follow through this course) have been provided on the following website:

<http://linkedlist.us/gtbank/sks>

All that is required is a web browser : Google's Chrome Browser is recommended.

Table of Contents

1.0	What is a Computer Program?
1.1	Human languages and programming languages
1.2	Getting used to the coding environment
1.3	Variables and functions
2.0	JavaScript Programming Language
2.1	Operators
2.2	Variable revisited : Boolean Variables
2.3	The “if block” and “if statements”
3.0	A few more things you should know
3.1	Interacting with our ATM
4.0	Conclusion
5.0	Practice Questions

1.0 What is a Computer Program?

It is **series** of computer **instructions** required to achieve a **goal**.

The goal may be:

- ✓ to put an astronaut in space
- ✓ to find a vaccine for a killer disease
- ✓ to solve a mathematical equation
- ✓ to create an animated movie like Disney's *Frozen*

We say “*a series of computer instructions*” because one instruction follows the next and the next until the goal is reached.

What is an instruction? It is simply something one is told to do.

For example:

“Go home now”

Instructions can be in any language provided the instructor and the instructed understand each other.

1.1 Human languages and programming languages

Did you know that there are over 6500 human languages on earth? Did you also know that according to a report by Forrester Research, there were over one billion PCs in use worldwide by the end of 2008.



Human languages allow us communicate with one another. Programing languages allow us communicate with computers to give them instructions so that we can achieve our goals.

Programing languages allow humans write computer programs to solve problems.

1.2 Getting used to the coding environment

The code editor we will be using can be found at :

<http://linkedlist.us/gtbank/sks>

Point your web browser (preferably Google Chrome) to the above address and you should see the following:

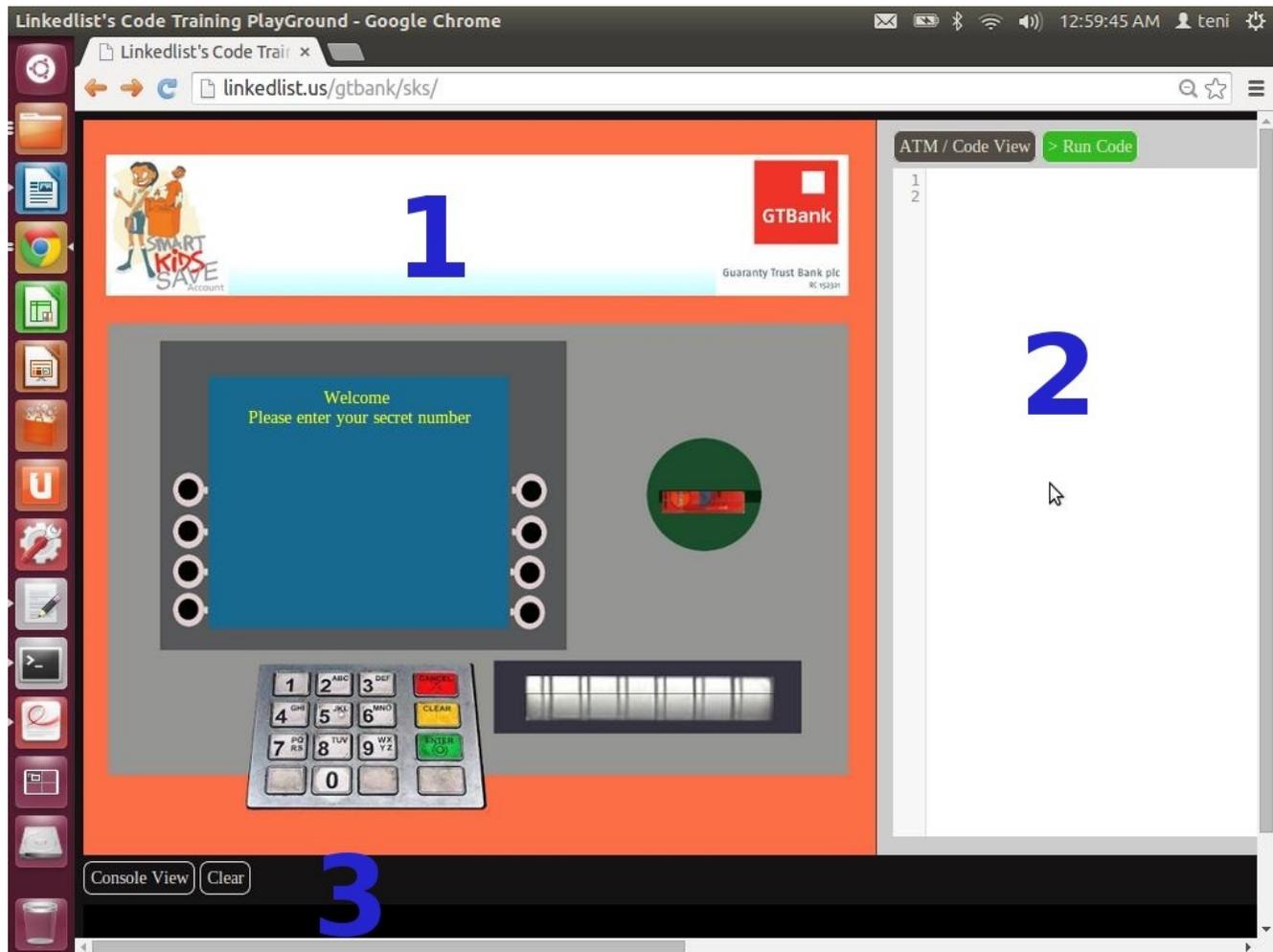


Fig 1: (The coding environment)

The coding environment is divided into three parts namely:

1. The virtual ATM (shown on the left of the screen)

This is the ATM we will be writing code to control. You will be able to write code to insert a smart-card, remove it, open the cash tray, change the welcome message, make a withdrawal from your account and more.



2. The code editor (shown on the right of the screen)

This is where most of your code will go. You will be writing JavaScript. Don't be scared, it is dead easy and will be fun.

3. The Interactive console (at the bottom)

You can write code here too. You will also be getting feedback from your programs in this console. In fact, we start using the console in the next section when we talk about variables. But just before we leave off, try something;

Click your console, and type the following:



Fig 2: (Your first program)

Now, press the Enter Key.

You should see this:

Hello World !

Congratulations, you have just written your first program.

While you are at it, try this too:

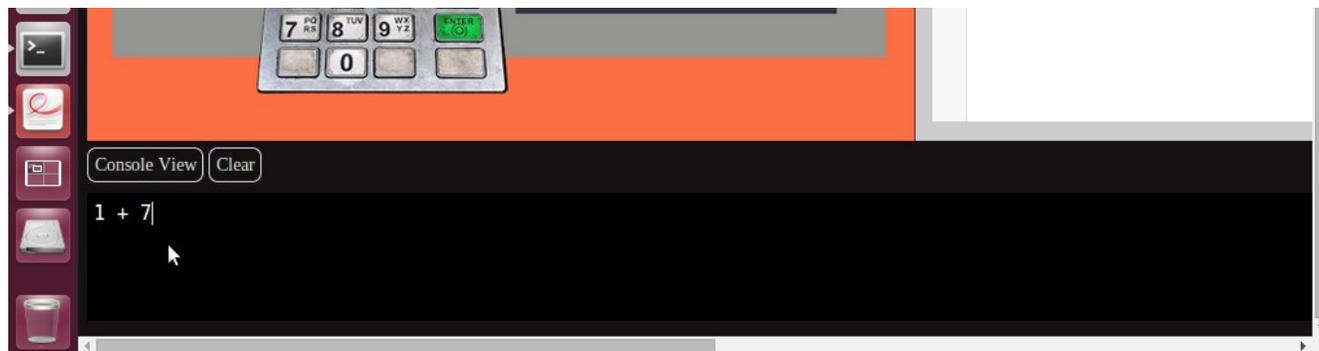


Fig 3: (Adding 1 and 7 in the console)

Hit Enter.

Of course $1 + 7 \Rightarrow 8$.

Everything you have typed in the console could have also been typed into the code editor like so:

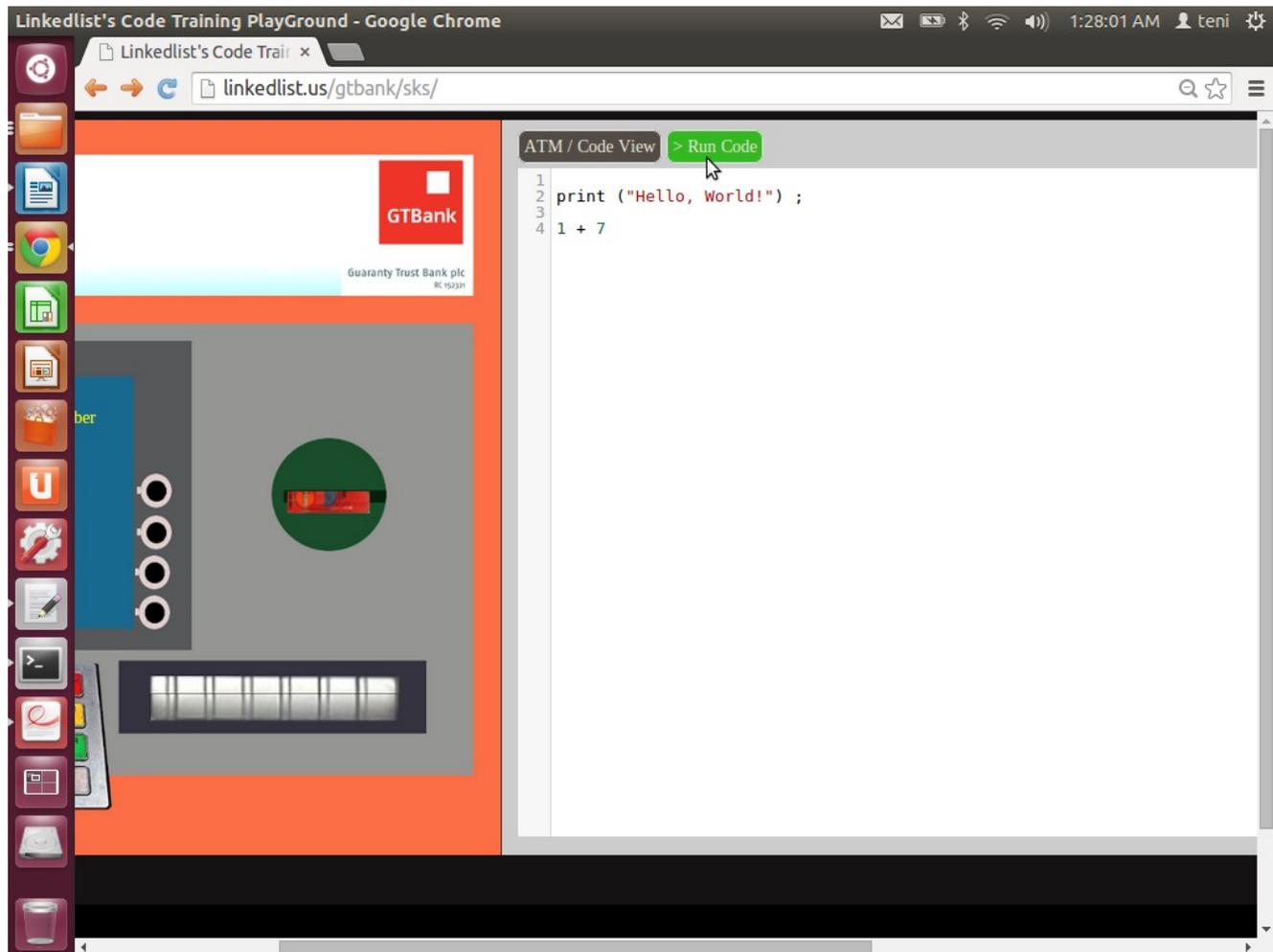


Fig 4: (Running code in the code editor instead)

To run code in the code editor, click the green button above that reads “ Run code”, now scroll down to the console to see your output there, same as before.

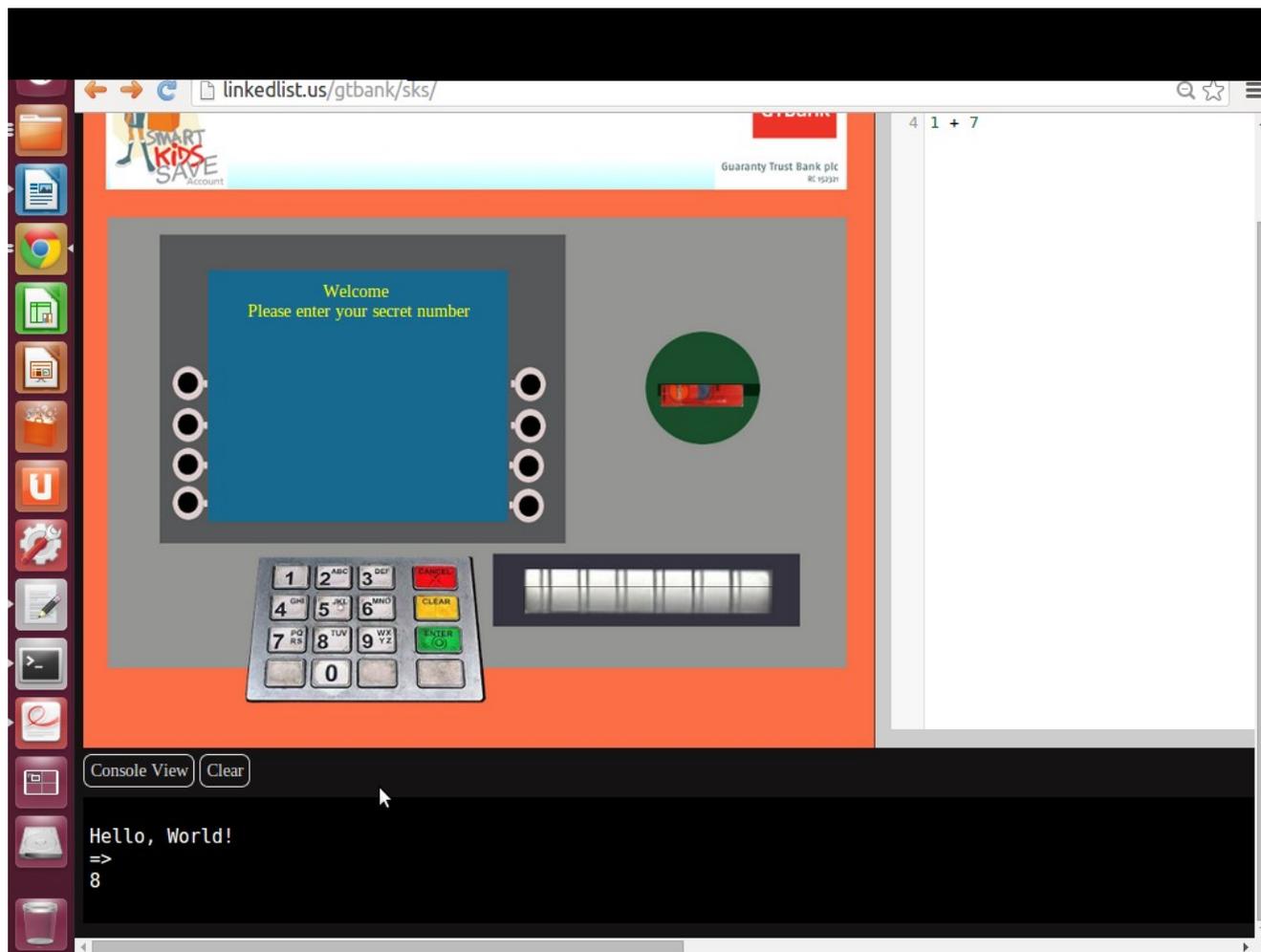


Fig 5: (Console output for the code editor after “run code button” is clicked)

1.3 Variables and functions

A variable

Imagine you wanted to tell your mom that you were going to the GT Bank across



the street with Oyinda your friend.

You could say this,

sentence 1 → “Mom, I am going to ***the bank*** with ***a friend***”

If your mom is anything like mine, she would ask you,

“Which bank are you going to and with whom?”

To which you have to answer,

sentence 2 → “Mom, I am going to **GT Bank** with **Oyinda**”

Notice how **GT Bank** takes the place of ***the bank*** and **Oyinda** takes the place of ***a friend*** in sentence 2.

Also, realize that you mom might have been asking which bank you intended to go to because there might be more than one bank across the street, and which friend because you most definitely don't have just one friend.

If you understand that, you know understand **variables**.

A variable is something that can be any value.

If I said to you over the phone, “Hey, I just got a new smartphone”.

Smartphone can mean any of the following:

- Iphone
- Android HTC

- Nokia X
- Blackberry Z 10
- and more

In this case smartphone is a variable that can take any of those values.

Let us head to the console again.

Type in the following:

```
a = 100;
```

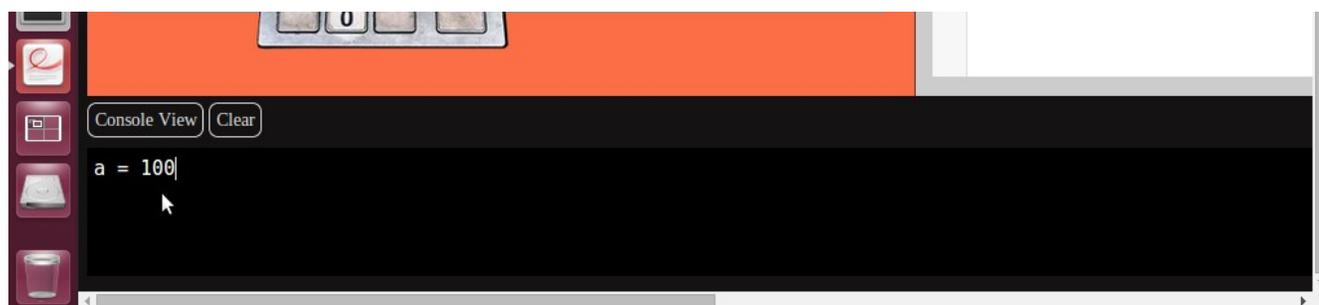


Fig 6: (Defining a variable)

Now enter.

We have just defined a variable “a” and set it's value to 100.

Next do this:

Type in `a + a` and hit enter

You should get 200.

Because, `a + a` now means `100 + 100` which is 200.

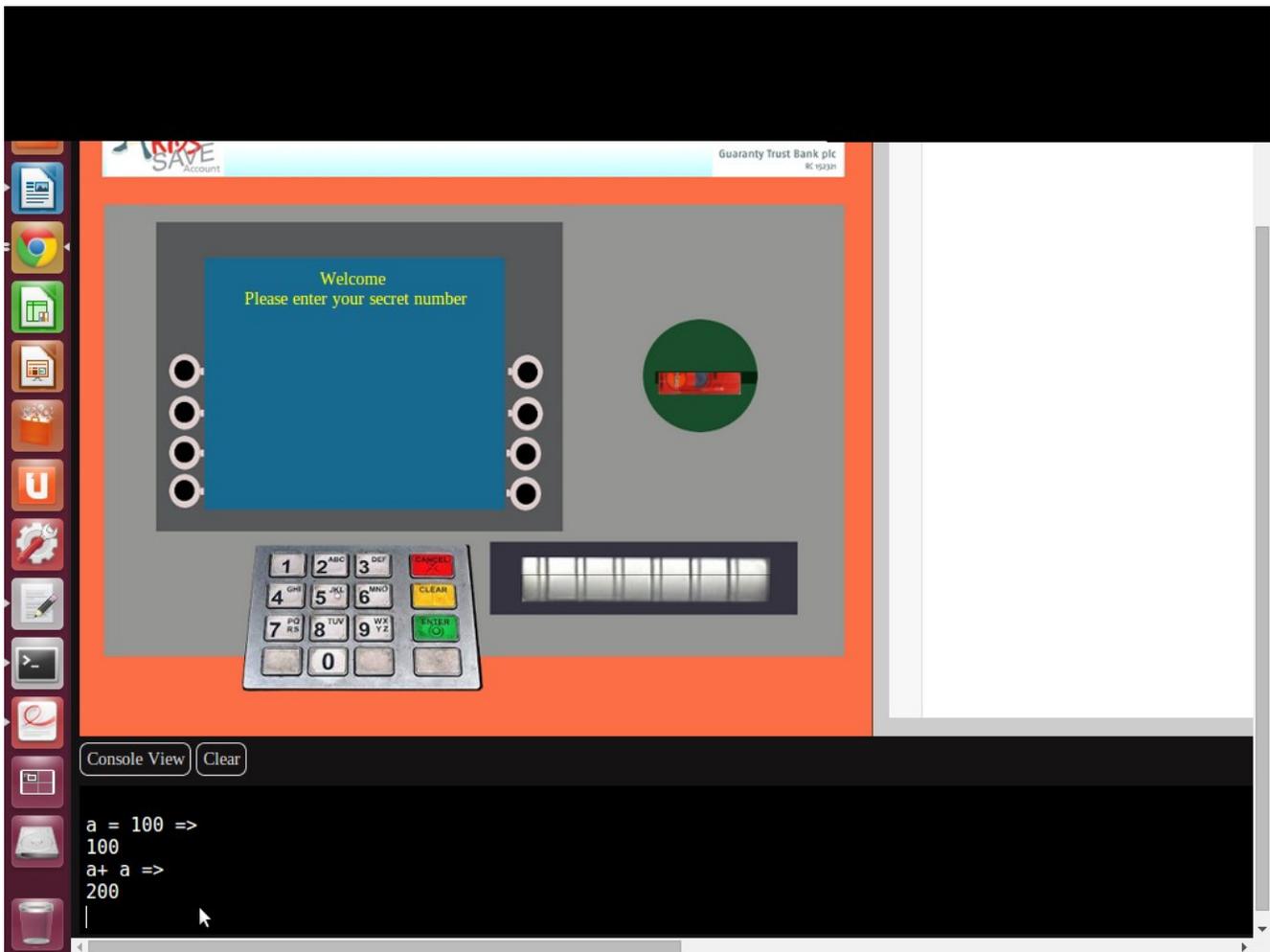


Fig 7: (Using already defined variable for further computation)

Now let us be naughty.

Type in $a + b$ and hit enter

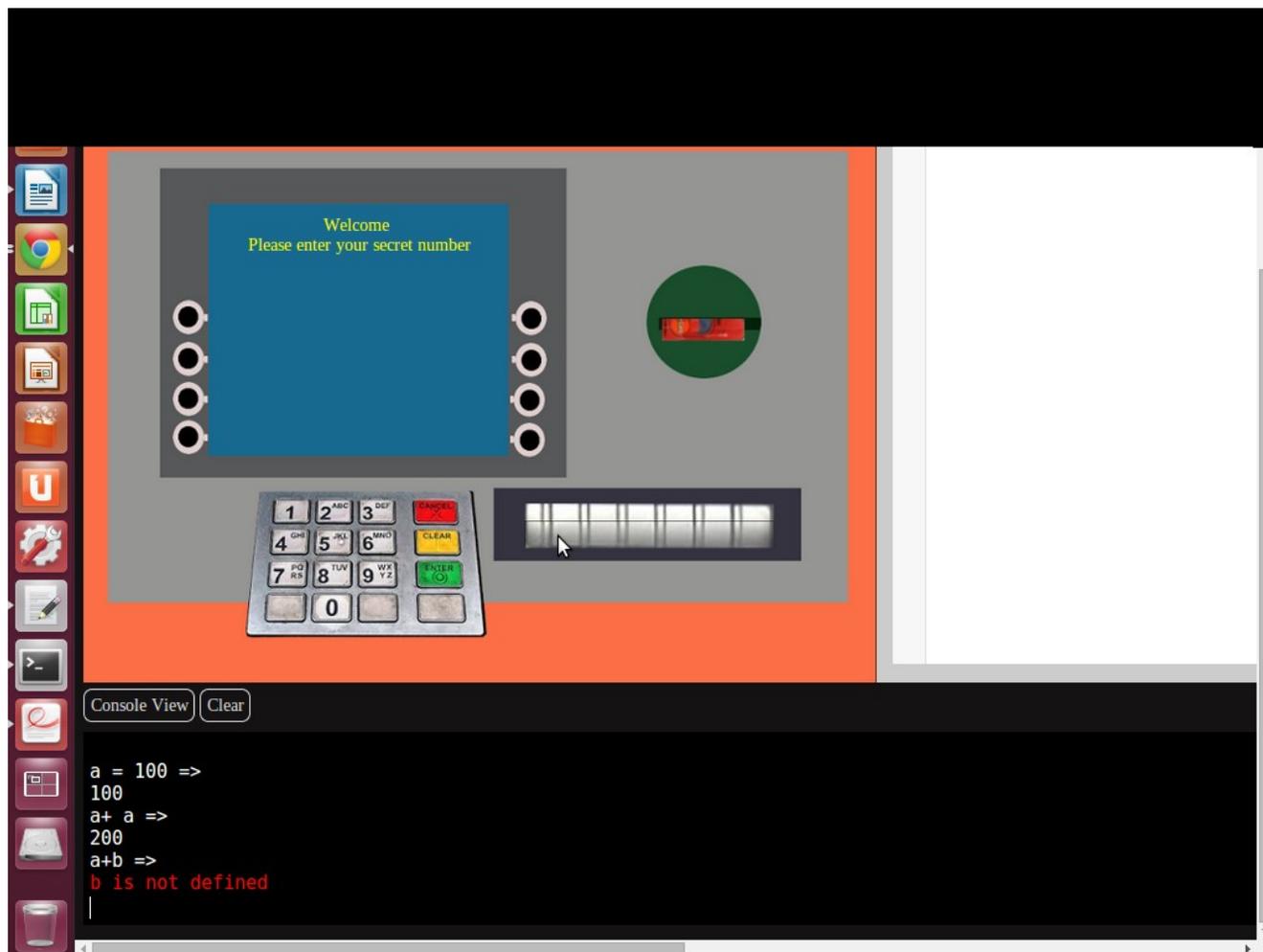


Fig 8: (Attempting to use a variable that is not defined results in an error)

The console yells back, saying “ b is not defined “.

Let us fix this:

Type : `b = 300` hit enter

`b` is now defined to be 300

We will now re-try our previous command:

a + b

You don't have to re-type it, it is in the history already, just press the up direction key to see previously entered commands.

Once you have found it, hit enter. All should work normally now. You should get 400 as output.

JavaScript variables can hold three major types of information:

- Number (e.g 1,5,6000, -56)
- String (e.g “hello”, “Oyinda”)
- Boolean (true or false)

A function

A function is a way to group program statements.

Let us assume that Moi Moi (an African dish made from beans) is made at your home regularly. You may go about it this way:

1. Wash the Beans
2. Go grind washed beans
3. Give it to Mom for mixing and cooking
4. Serve hot
5. Eat

These five steps can be grouped into a moi_moi function.

```
function moi_moi () {  
  
    Wash the beans ;  
    Go grind washed beans ;  
    Give it to Mom for mixing and cooking ;  
    Serve hot ;  
    Eat ;  
  
}
```

Every time you need Moi Moi at home you just call the moi_moi function like so:

```
moi_moi();
```

Please note, moi_moi() is not defined, and will yield an error if entered in the console. Now, let us look at the how a function is called:

```
the_function_name() ;
```

Pay attention to the () after the name of the function.

() look like a pair of talking lips.

We use those lips to say something to the function.

Try this in the console:

```
times2 (5) ;
```



You should get 10.

We called the function named **times2** and put **5** *on the lips*. Meaning we a *calling the times2 function and passing into it the value 5*.

The function takes 5 , multiplies it by 2 and prints 10 as expected.

Sometimes, we call a function, but put nothing on the lips.

Now we are going to get that silly ATM to do something

Try this in the code editor:

```
insert_card();  
open_tray();
```

Now click the green “run code” button

What happens? The hanging ATM card is inserted, and the cash tray is opened.
Waoh! Show your friends!

2.0 JavaScript Programming Language

The JavaScript Programming Language is a scripting language that was originally designed for the web browser by Brendan Eich one of my heroes. JavaScript is so versatile right now that it has found its way out of the web browser and now also runs on servers. JavaScript Programming is a coveted skill.



Brendan Eich (Father of JavaScript)

2.1 Operators

You should be familiar with the basic operators in mathematics. Operators also exist in programming for: addition, subtraction, division and more. Below is a table comparing the mathematic and programming operators.

Operator Name	Mathematics	Programming
Addition	+	+
Subtraction	-	-
Division	÷	/
Multiplication	X	*

2.2 Variable revisited : Boolean Variables

Remember I mentioned this earlier, that a boolean is a type of variable that is either true or false.

For example:

The light is either on or off → The light is on can either be true or false

You are either a male or female → I am female can either be true or false

5 is bigger than 8 → false

Obama is president of Ghana → false

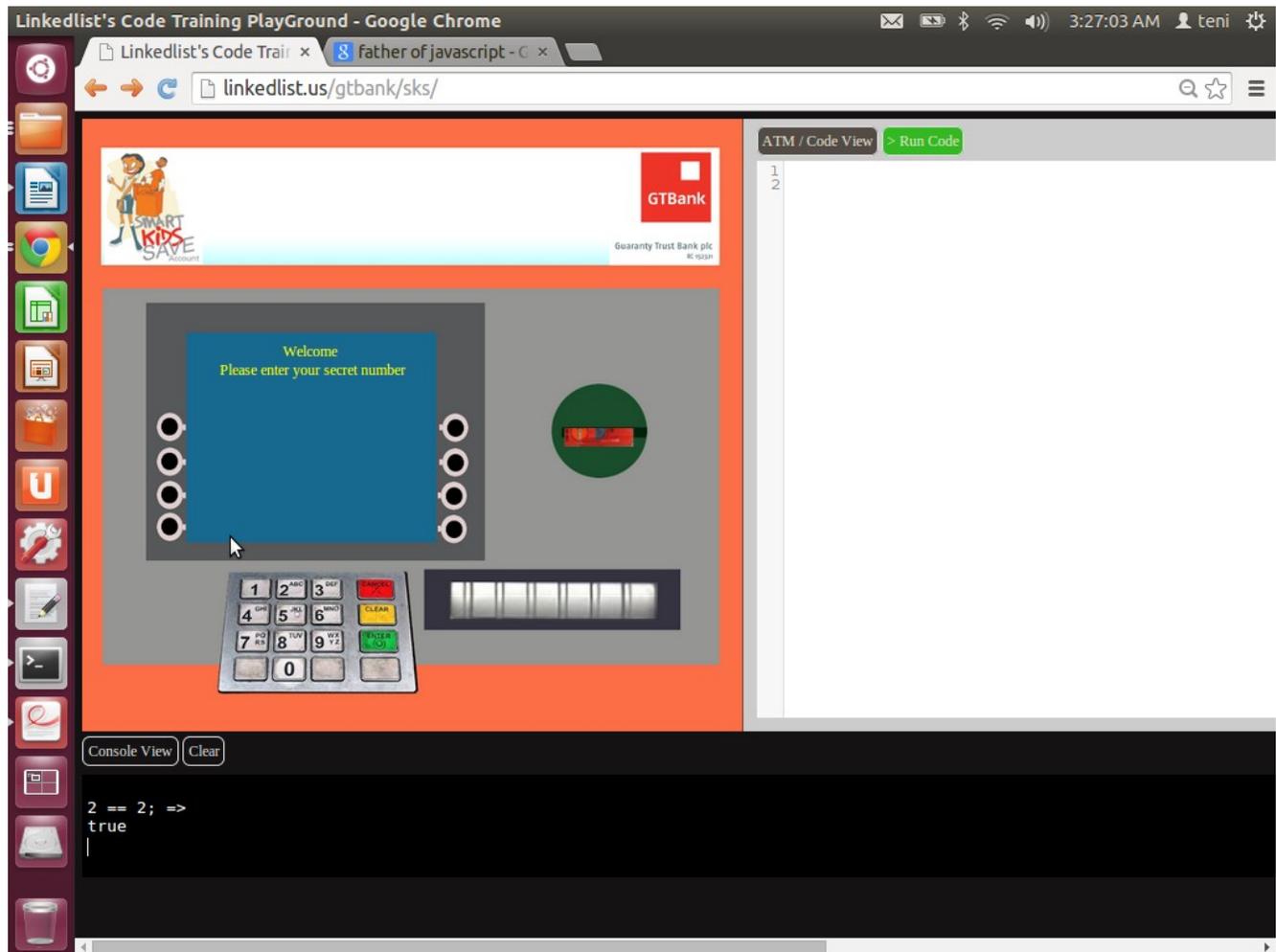
In Javascript, the way to check that two variables are equal is with the == operator.

Back to the console:

Enter in the following:

```
2 == 2
```





LinkedIn's Code Training Playground - Google Chrome

father of javascript - C x

linkedinlist.us/gtbank/sks/

ATM / Code View > Run Code

```
1  
2
```

Console View Clear

```
2 == 2; =>  
true
```

We get back true.

Now try this:

$3 > 45$;

You should get false.

20



Practice the following:

Enter each statement in the console:

$45 * 45 == 2025$

$3 + 4 == 10 - 3$

$6 == 6.000$

$45 < (3 * 57)$

$5 != 4$

2.3 The “if block” and “if statements”

From the last exercise,

$5 != 4$ means 5 **is not equal to** 4

“If “ in English language

We say “if” in English when what happens next depends on some factor:

if it rains, I will stay at home → **meaning it may rain or not**

if my dad says yes, I will spend the weekend in your house → **he may say yes or no**



“If block“ in Programing

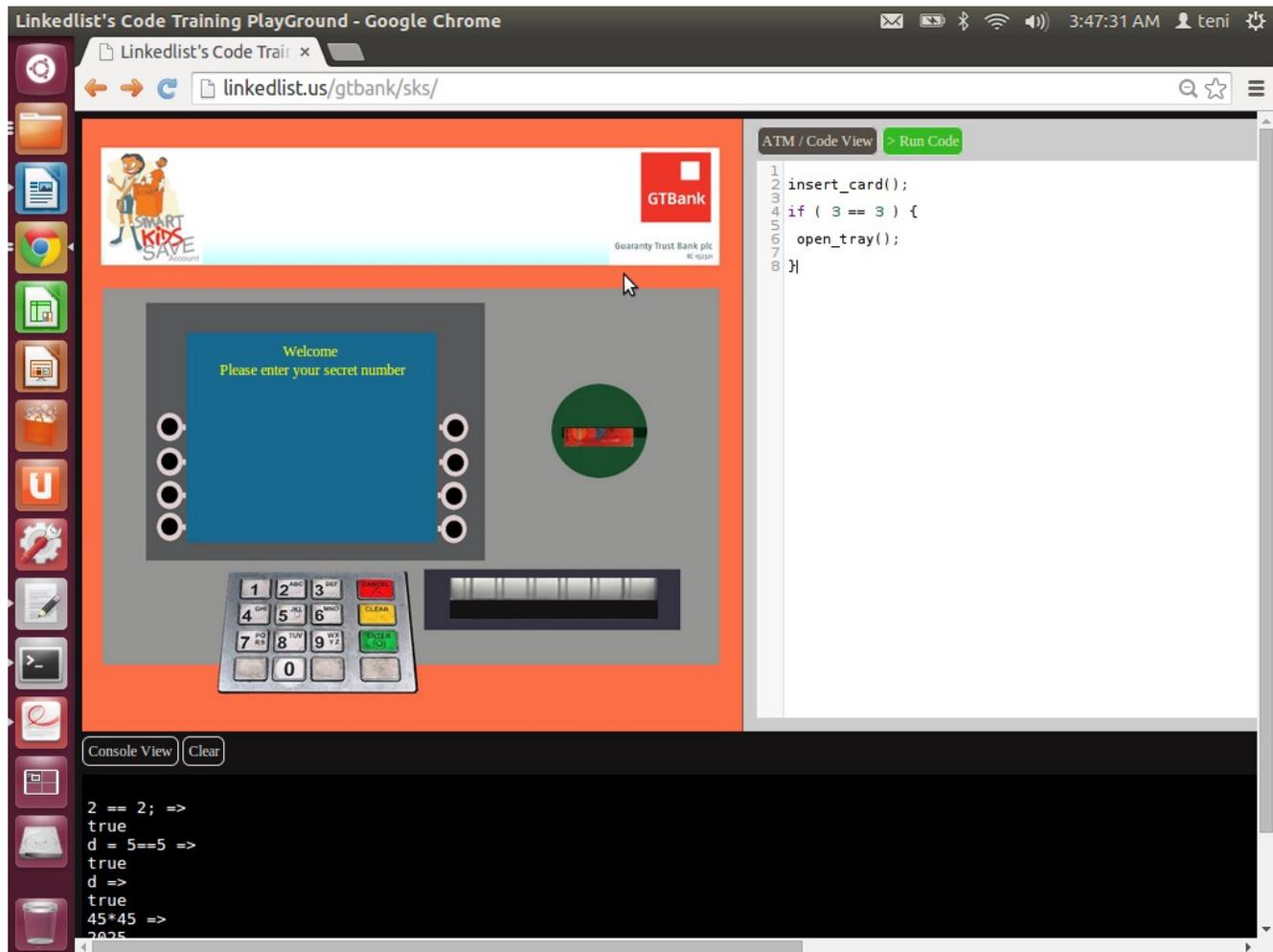
```
if ( rain ) {  
stay_home  
}
```

```
if ( dad_says_yes ) {  
  
weekend_in_your_house  
  
}
```

“If” in programming will test *what you put on the lips*

Head to the code editor and type the following in, then click the run code button

```
insert_card();  
  
if ( 3 == 3 ) {  
  
open_tray();  
  
}
```



Your ATM tray should open. This is because what “if” is testing is that `3 == 3`. Of course `3 == 3`, so the `open_tray` function is called.

Now go to the console to type the following:

```
close_tray();
```



That should close the tray.

Now edit the code in the editor to be like so:

```
insert_card();
```

```
if ( 3 == 44 ) {
```

```
    open_tray();
```

```
}
```

Because `3 == 44` returns false, the if block doesn't run any further and the tray stays closed.

3.0 A few more things you should know

In this section, we will be using all the stuff you have learned in previous sections. But just before we start interacting with our ATM, let us cover a few more things.

Comments

In section 1, we established that a programming language enables us communicate with a computer thereby getting it to do what we want.

However, there is something that happens to all programmers, we write tons and tons of code to the extent that code that we once understood perfectly becomes very confusing when we look at it on a future date.

A comment allows you write text within your program code to aid understanding the code in future by you or whoever will be reading it.

Comments are totally ignored by the computer.

Below, I show how to write comments in JavaScript:

```
// This is a single line comment
```

```
/** This is  
    a multi-line  
    comment */
```

Type the above code in your code editor and hit the run button.
Nothing is evaluated.

Reserved Words

Every programming language has reserved words.
A reserved word is a word that is meant for the computer only.

Just like the seats on the “high table” in parties are reserved for the Chairman and other *important people*.

The programming language says, “No! You can't use those”.

Examples of reserved words in JavaScript are:

if case switch for
else continue try except

return while in
.....

If you have noticed, when you type in a reserved word into the code editor, it colors that word different from non-reserved words.

So if you tried to define a variable like so:

```
if = 5; /** Results in an error **/
```



```
switch = "I hope this works"; // An error again
```

The error you get is the following:

Unexpected token =

Javascript is telling you that you can't have an equal to sign after a reserved word.

Incrementing a variable

Type the following in the console :

```
a = 1; /** Hit Enter **/
```

Now type:

```
a = a + 1; /** Hit Enter Again **/
```

Now type:

```
a /** Hit enter, you should get 2 **/
```

Looping

A computer is not as smart as a human, but a human is not as fast as a



computer.

Computers are able to perform task over and over again without getting tired.

If I asked you to say ,”Red Lorry, Yellow Lorry” over and over for the next 10 times as fast as possible: You will most likely end up saying “*Led Yorry Rellow Lorry*” .

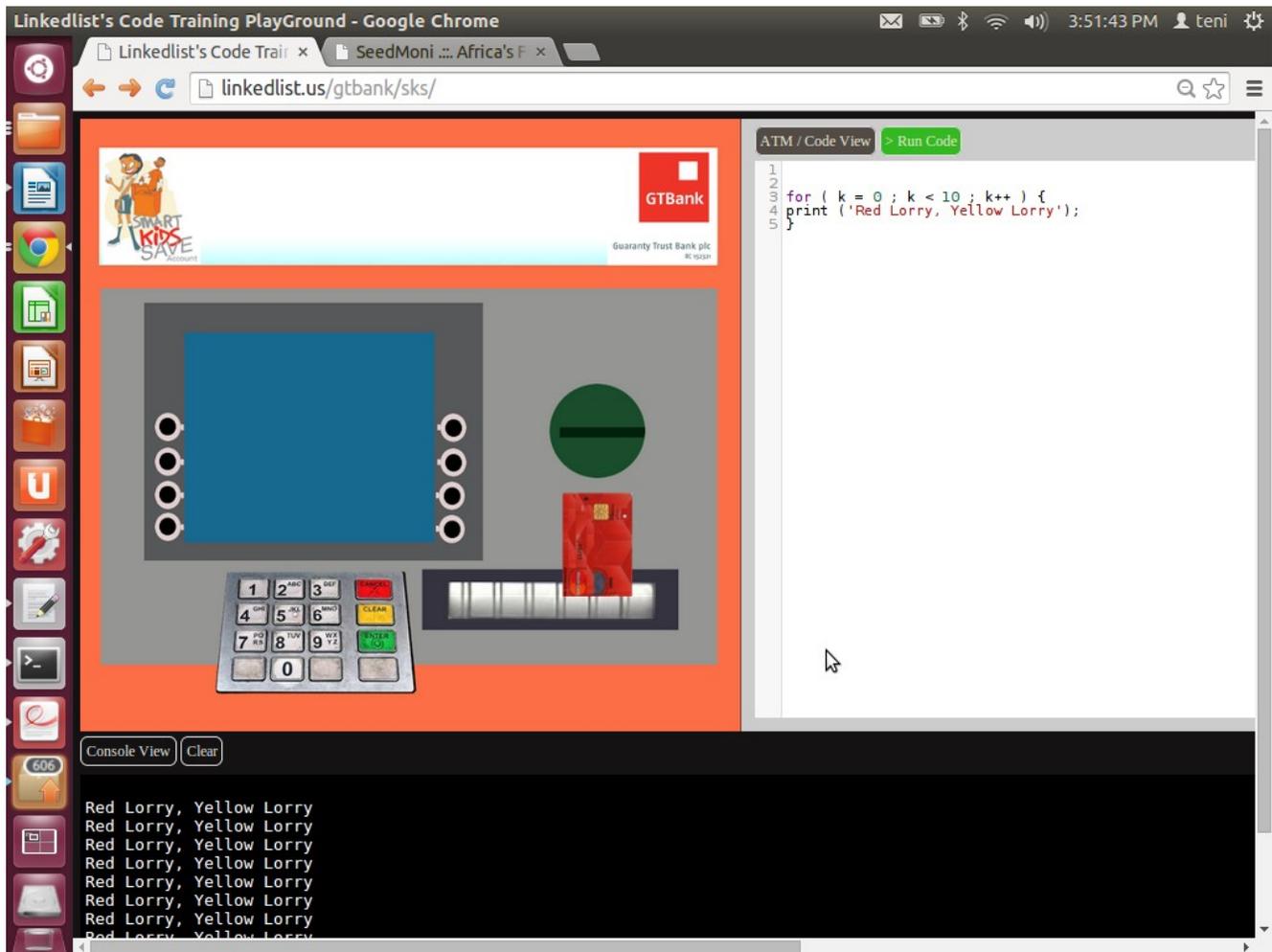
But a computer will get it right a thousand times over.

When computers need to do the same time over and over they use a LOOP.

We will cover the for loop.

Type the following code into your code editor and hit run:

```
for ( k = 0 ; k < 10 ; k=k+1 ) {  
print ('Red Lorry, Yellow Lorry');  
}
```



The output comes out in the console.
No “Led Yorry Rellow Lorry”

3.2 Interacting with our ATM

When you load the code environment, your ATM pin is:

0000

29



At any time you can see your pin by running the command:

Type:

```
pin /** Enter **/
```

To change your pin to **5577**

Type:

```
pin = "5577";
```

Then hit enter.

If you have been following this course you will realize that the pin is simply a variable that holds a string of digits.

Also, out of the generosity of my heart, your account was opened with a balance of N1000. So you've got bucks!

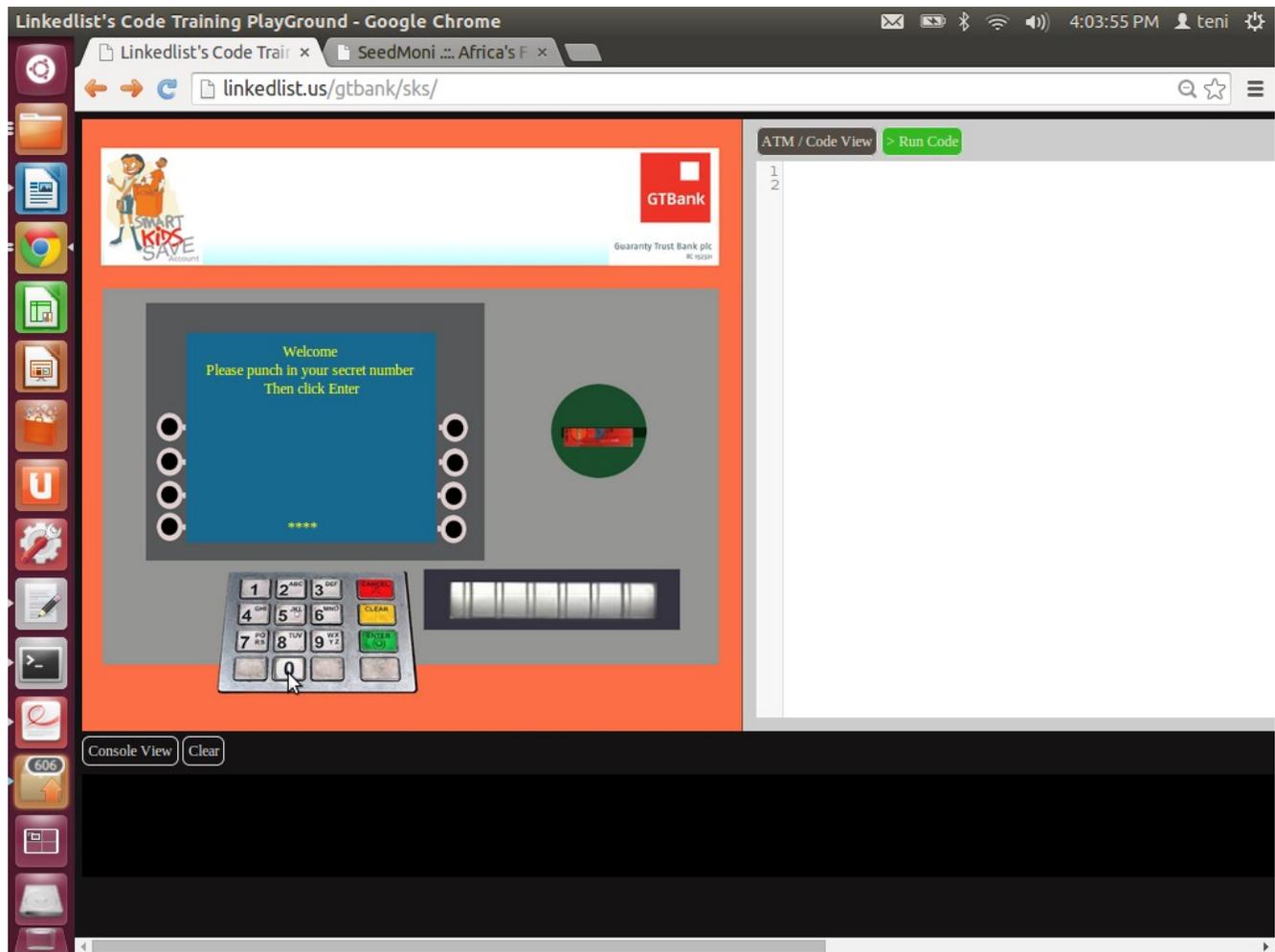
In real life, your ATM pin should be kept secret (known to you alone). And it is usually a four – digit code.

Click on the ATM card to insert it.

The display now prompts you for your pin.

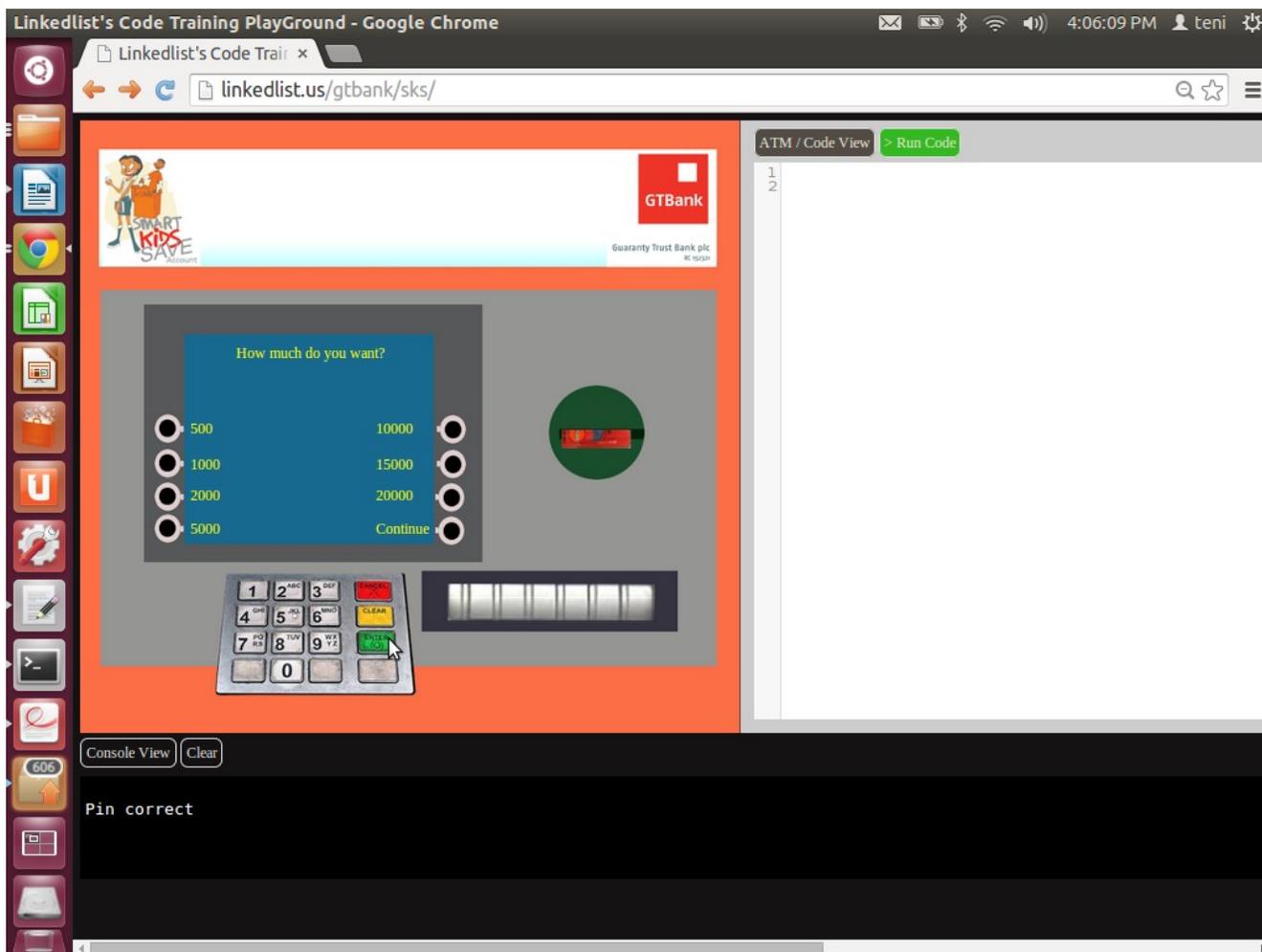
Use the number pad to enter your pin (If you have not changed your pin, enter **0000** if you have changed it, enter the new pin)

Please note, upon refreshing this webpage the pin is reset to **0000** .



Once you have entered your pin click the green button to login.

If you entered the correct pin you get the following screen:



LinkedList's Code Training Playground - Google Chrome

linkedList.us/gtbank/sks/

ATM / Code View Run Code

1
2

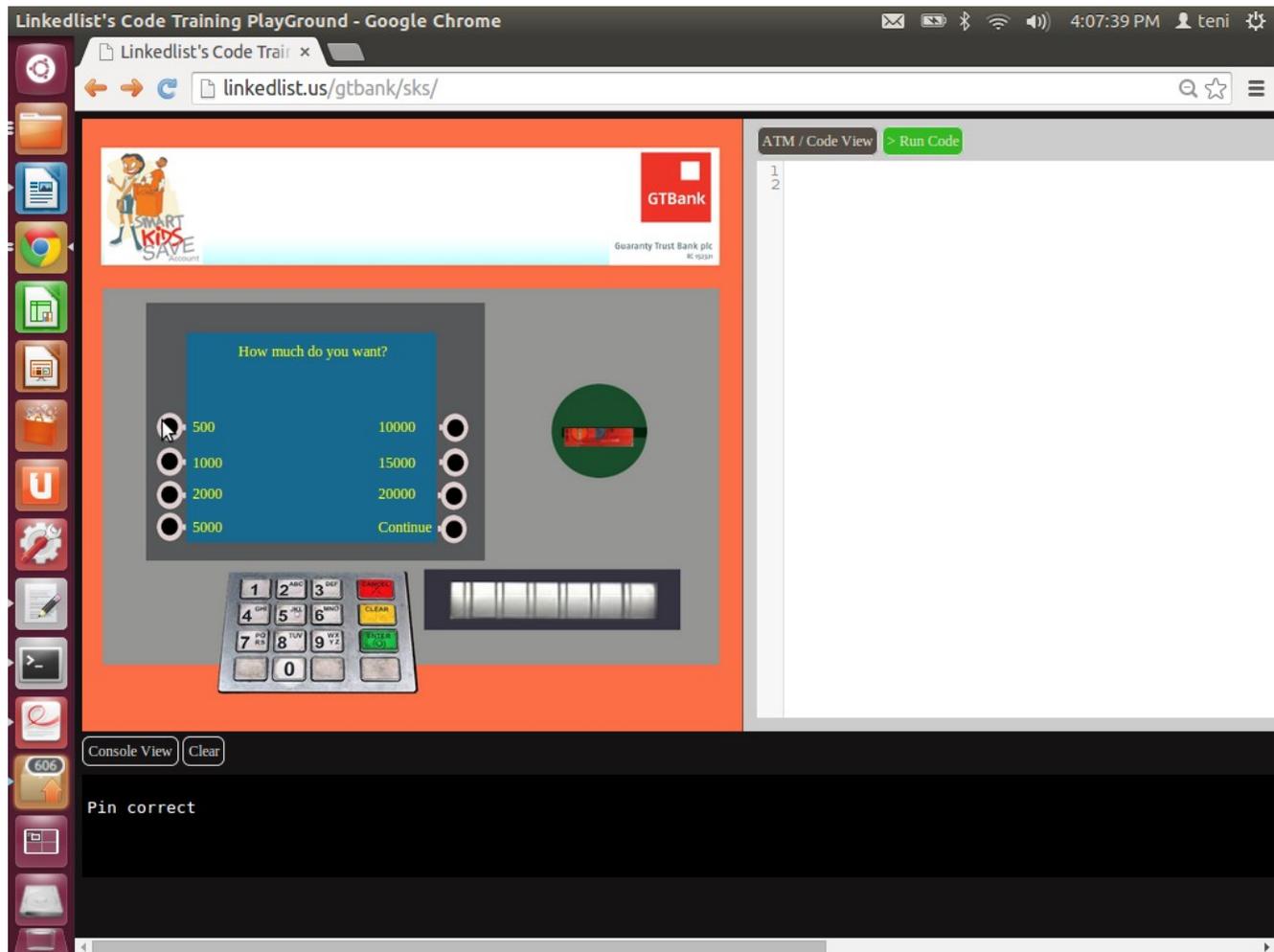
How much do you want?

500 10000
1000 15000
2000 20000
5000 Continue

Pin correct

Now let us withdraw N500.





Upon withdrawal of N500, N500 is removed from our account and our balance becomes N500

Clicking around on this ATM is cool, but programming it with code *awesometastic* of course there is no word like that.

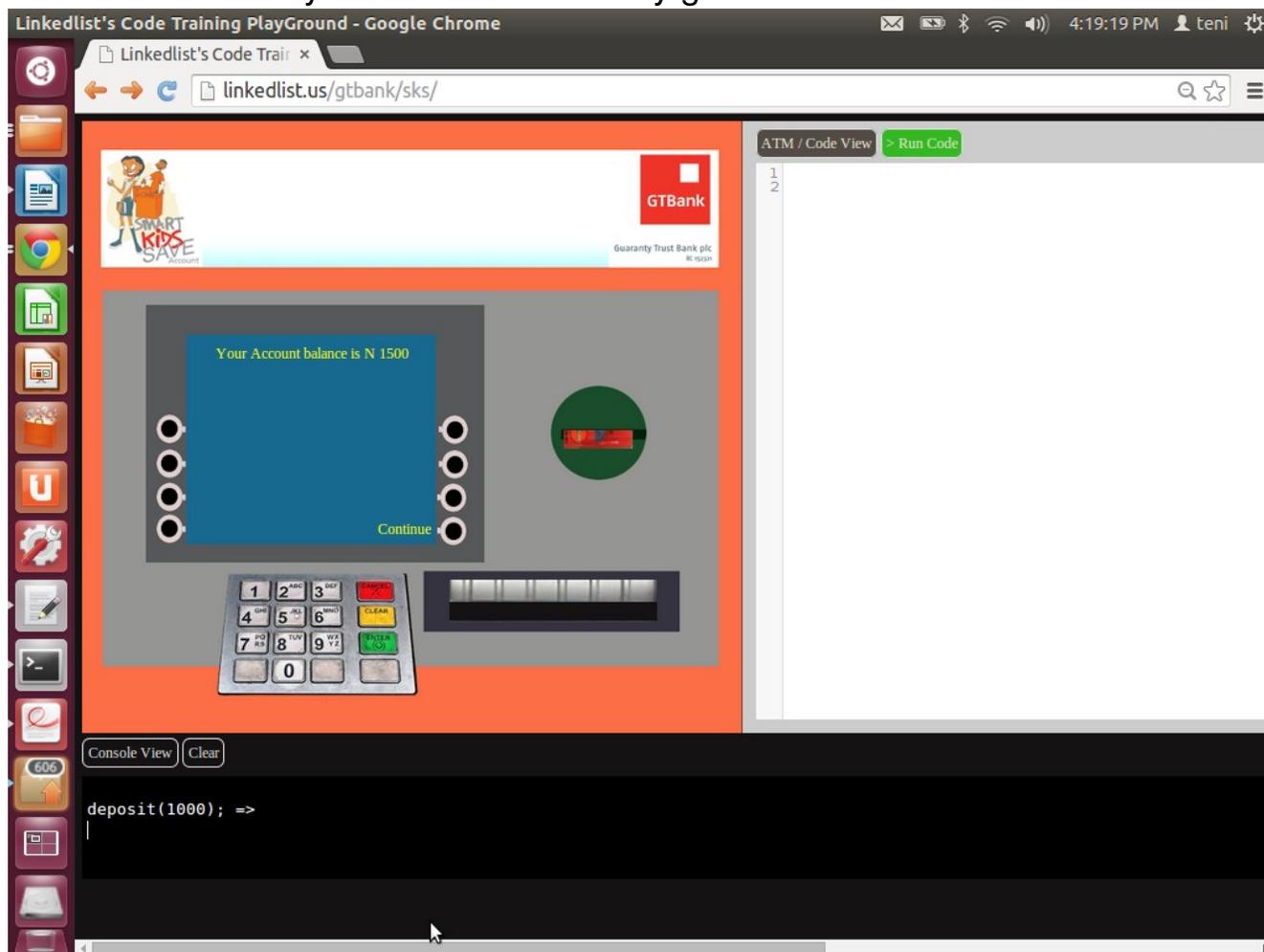
First, lets break the bank. Lets get more money.



Type this in your console:

```
deposit(1000);
```

Hit enter. What do you know? Now money grows on trees.



The screenshot shows a web browser window titled "LinkedList's Code Training Playground - Google Chrome". The address bar contains "linkedlist.us/gtbank/sks/". The main content area features a simulated ATM interface with a screen displaying "Your Account balance is N 1500" and a "Continue" button. A numeric keypad is visible below the screen. The right side of the browser shows a code editor with a "Run Code" button. The bottom of the browser shows the console with the command "deposit(1000); =>" entered.

Come on, we want more money. We are going to do that many times.

Head to your editor:

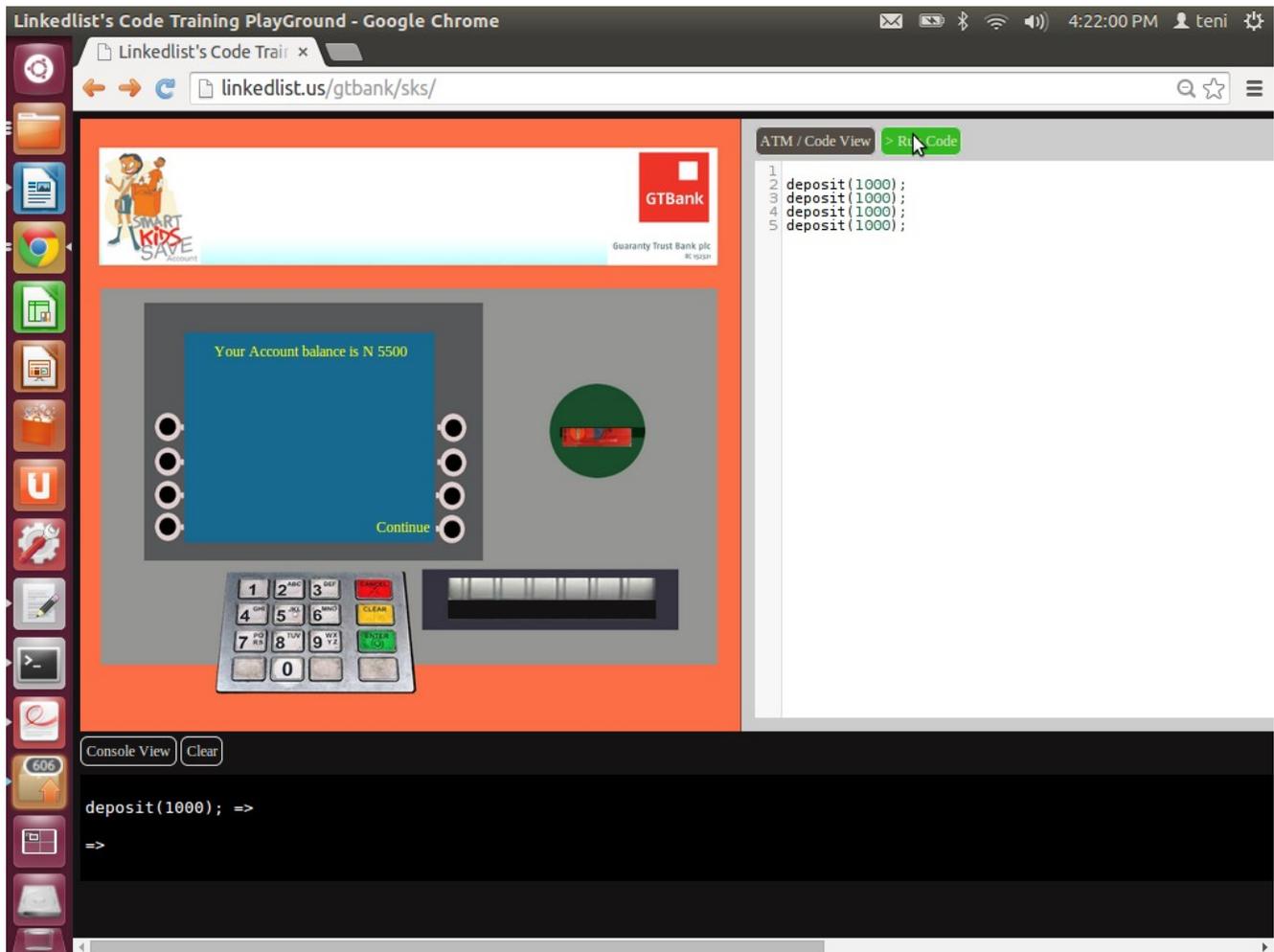
Type the following in there, and click the run code button:

```
deposit(1000);
```

```
deposit(1000);
```

```
deposit(1000);
```

```
deposit(1000);
```



LinkedIn's Code Training Playground - Google Chrome

linkedinlist.us/gtbank/sks/

ATM / Code View > Run Code

```
1 deposit(1000);
2 deposit(1000);
3 deposit(1000);
4 deposit(1000);
5 deposit(1000);
```

Console View Clear

```
deposit(1000); =>
=>
```

We are going to be reach soon. But let us say we could only deposit 1000s, and we wanted to join the SKS millionaires club. O dear, we will have to deposit 1000 times!

```
deposit(1000);
deposit(1000);
deposit(1000);
```



```
.  
.   
.   
.   
.   
deposit(1000);
```

But remember we said, computers are fast and can do the same thing over and over without getting tired. We need to do deposit(1000) over and over 1000 times.

We need a loop!

So lets go fish our *Red Lorry , Yellow Lorry* Loop.

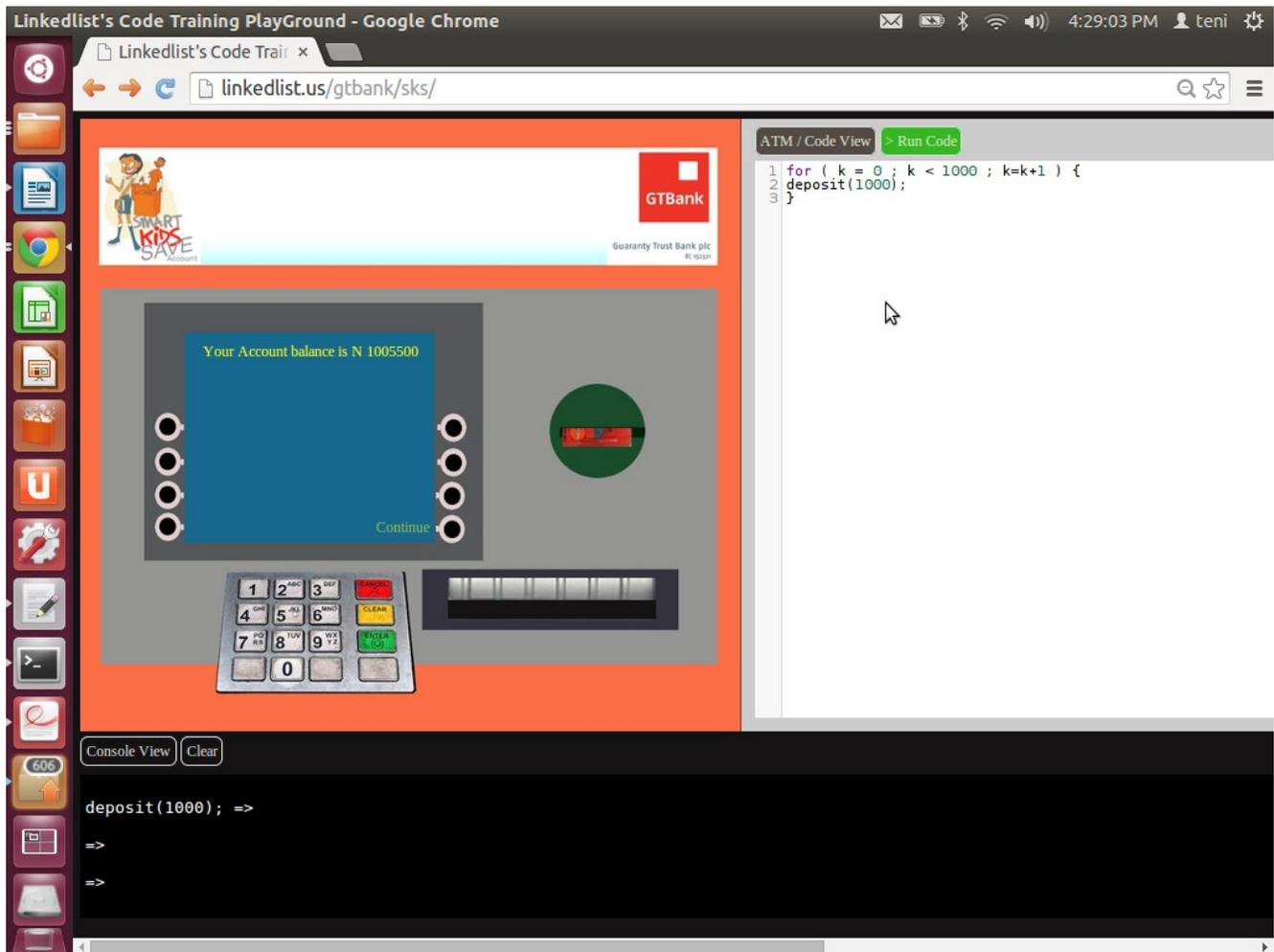
```
for ( k = 0 ; k < 10 ; k=k+1 ) {  
print ('Red Lorry, Yellow Lorry');  
}
```

we will change it to this:

```
for ( k = 0 ; k < 1000 ; k=k+1 ) {  
deposit(1000);  
}
```

Now copy and paste the above code in the code editor and hit run.





LinkedIn's Code Training Playground - Google Chrome

linkedinlist.us/gtbank/sks/

ATM / Code View > Run Code

```
1 for ( k = 0 ; k < 1000 ; k=k+1 ) {
2 deposit(1000);
3 }
```

Console View Clear

```
deposit(1000); =>
=>
=>
```

You are welcome to the SKS millionaire's club!



4.0 Conclusion

Programming a real ATM will involve a lot more than we can cover in a course. But this is a start. The world needs more programmers, people that can solve problems quickly and more efficiently with computers.

Medicine, Engineering, Government, Banking, Manufacturing all depend on computer systems.

Practice everything you have learned, seek to solve your problems with computers and the world will be a much better place.

Thank you

5.0 Practice Questions

- 1) $a = 45$, $b = 77$, $c = 789$ what is :
 $a + b + c$ hint: Use your console. Also, a,b and c are just variables
- 2) $1 + 5$
- 3) $'1' + '5'$
- 4) Enter the following in the console:
`insert_card();`
`open_tray();`
`close_tray();`
`remove_card();`

5) Type in the following:

```
display_message('I have finished this programming course');
```