**Lesson 1**

Here is how this code works:

In the first line, there is only one print statement that prints “*Hello My name is Shuvro Barua*”

In the second line, we printed 50 “*Yeh.........”* by using (\* 50) in the end. We also used \n to print a new line after every sentence.

In the third line, we again used a simple print statement.

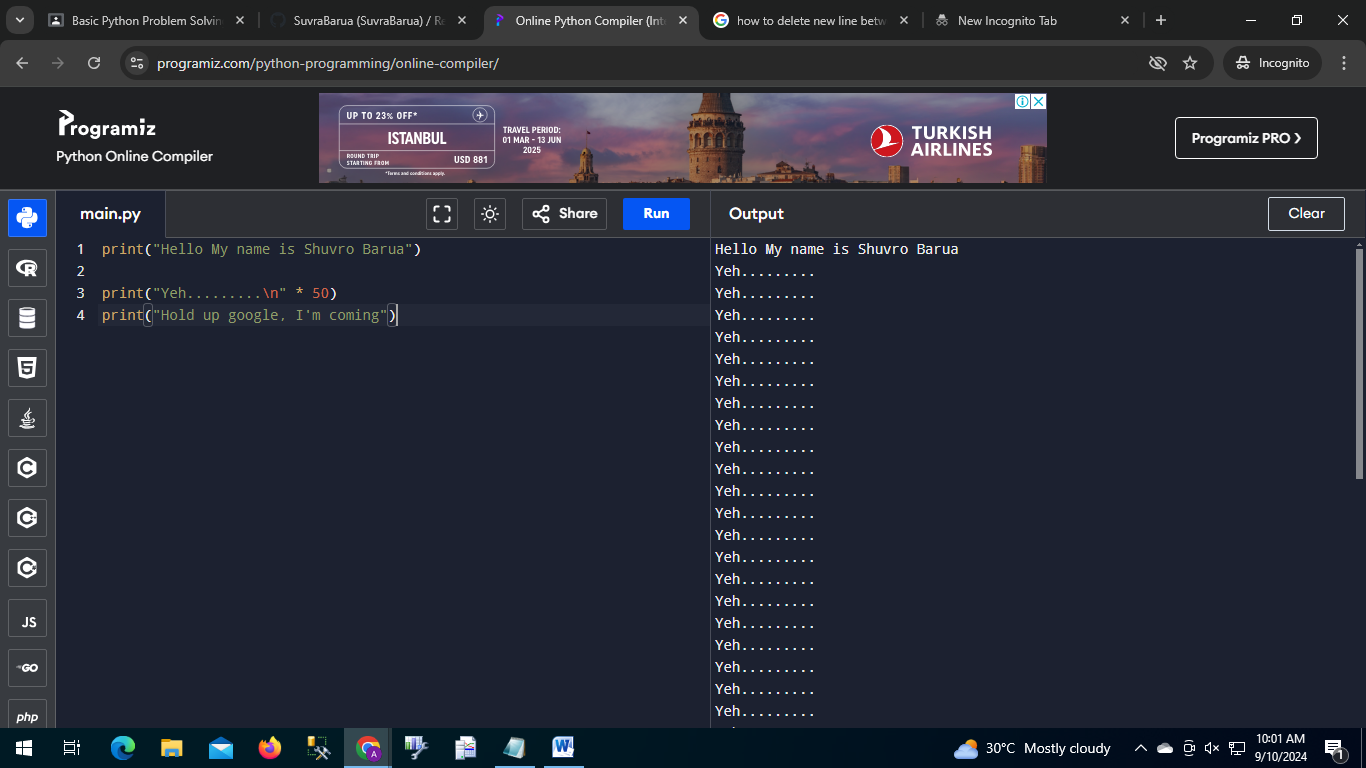


Image1: The output of lesson1.py

**Lesson 2**

This code teaches us about variables

We declared 4 variables of different types

We declared weight twice but only the second value will be counted because any new value will replace the already existing value if we use the equal “=” sign.

Then we printed the variables.

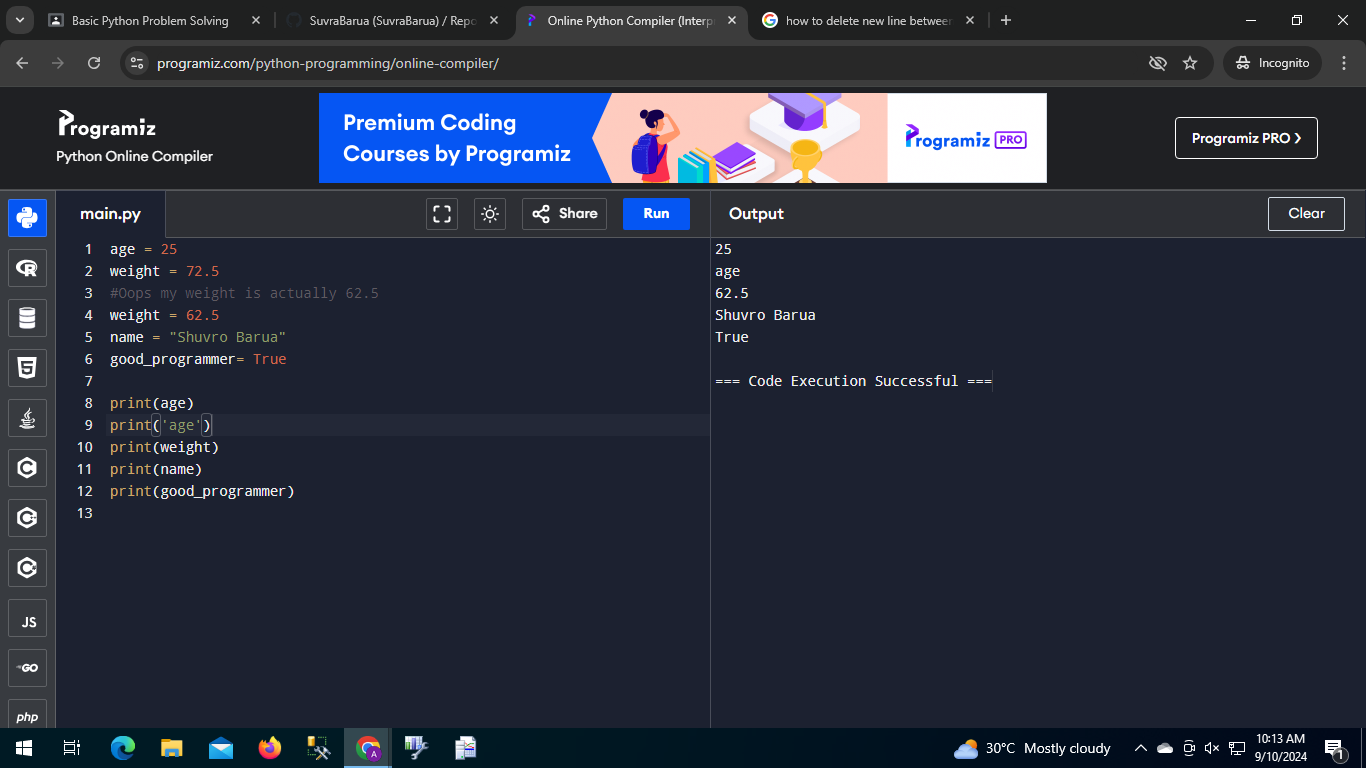


Image: The output of Lesson2.py

**Lesson 3**

In this lesson we learned about how to take user input and how to print them. Then we learned about concatenation

In the first line, we took an input and stored it in name variable. We used “What is your name” prompt to tell the users about the input. Then we took another input and stored it in address variable.

Then we printed the variable using the “+” sign to concatenate multiple variable and Strings on a single print statement.

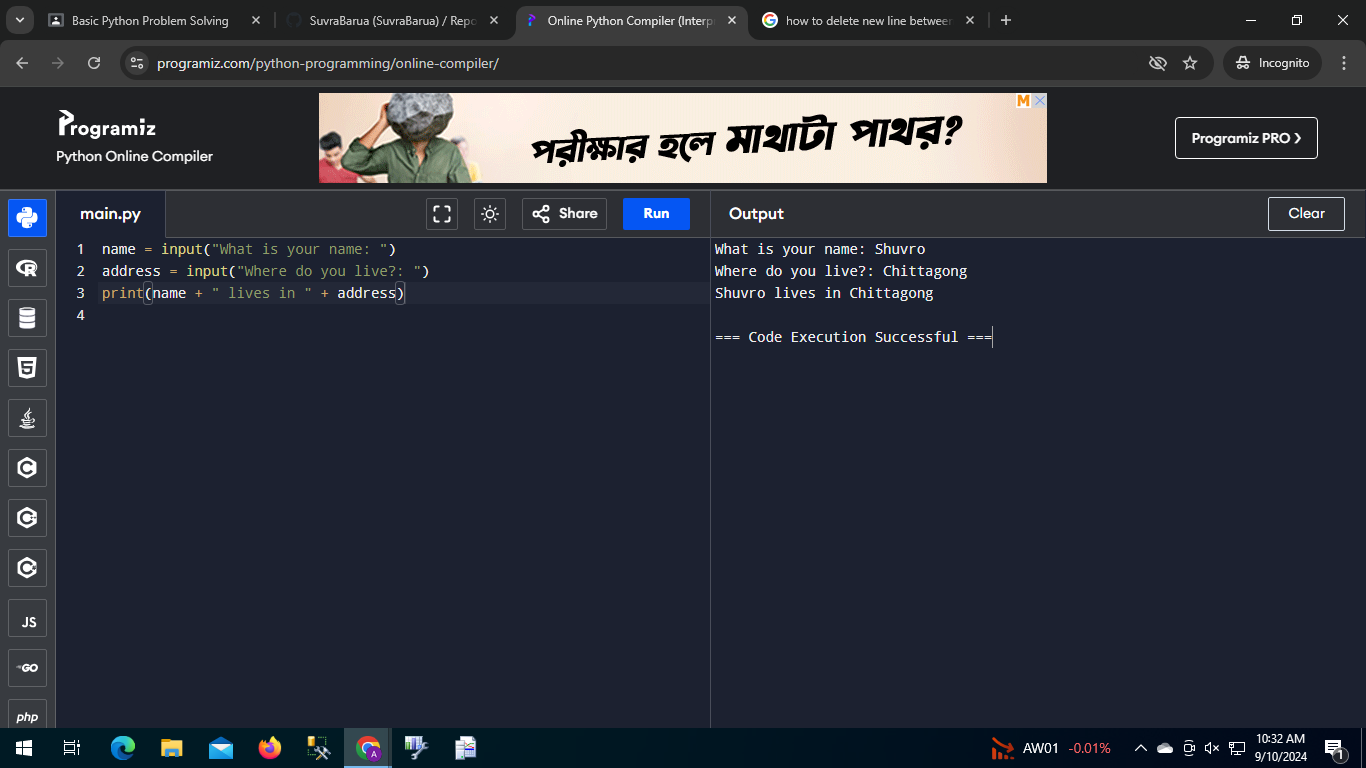


Image 3: The output of Lesson3

**Lesson 4**

Here is how this code works:

In the first line, We put a comment that shows the formula to convert Celsius to Fahrenheit.

Then We took user input of temperature in Celsius and stored it in celsius variable.

Then we used the formula to convert the Celsius to Fahrenheit and stored it in a variable. Then We printed out the value of that variable.

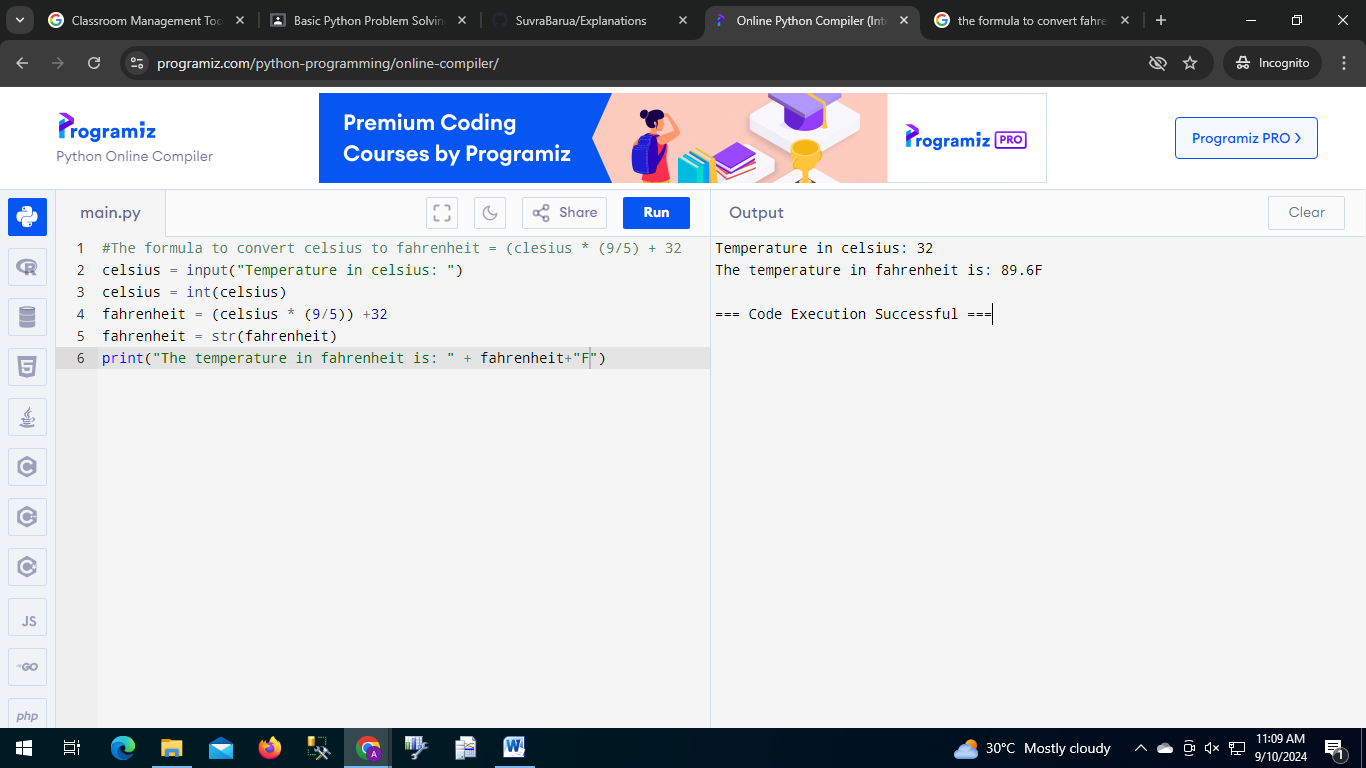


Image4: The output of lesson4.py

**Lesson 5**

Here is how this code works:

In the first line, We added a multiline comment.

Then we learned how to print ‘ and “ using a print statement.

Then we learned how to access every characters of a string. It works like an array. We used the 0 based index to access n and m of the string. Then we printed a part of the string using [starting index: ending index+1].

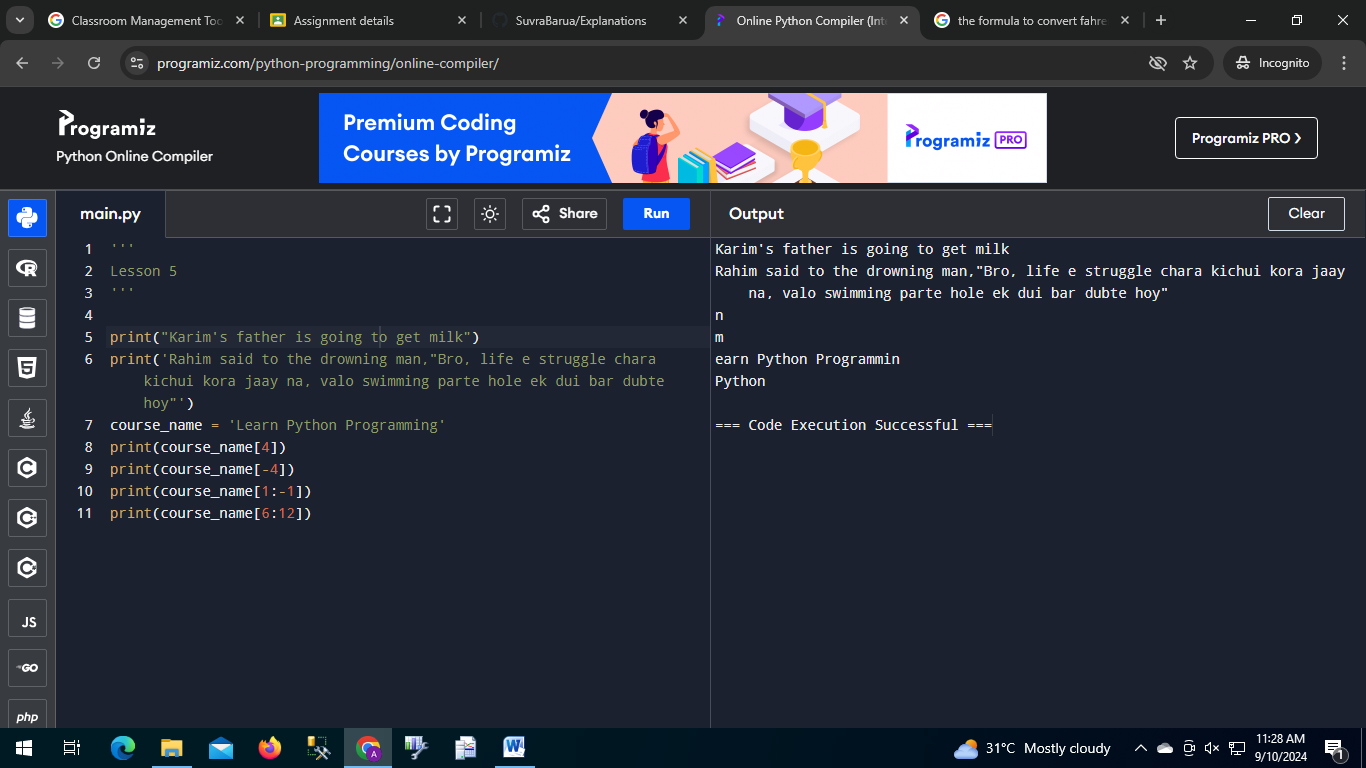


Image5: The output of lesson5.py

**Lesson 6**

Here is how this code works:

In this lesson, we learned how to print two variable and code names in brackets.

First we declared two variables named ‘name’ and ‘code\_name’. Then we printed “name (code\_name) learns AI” using two very useful ways.

In first way, we used concatenation to print. In the second line, we printed using fStrings.

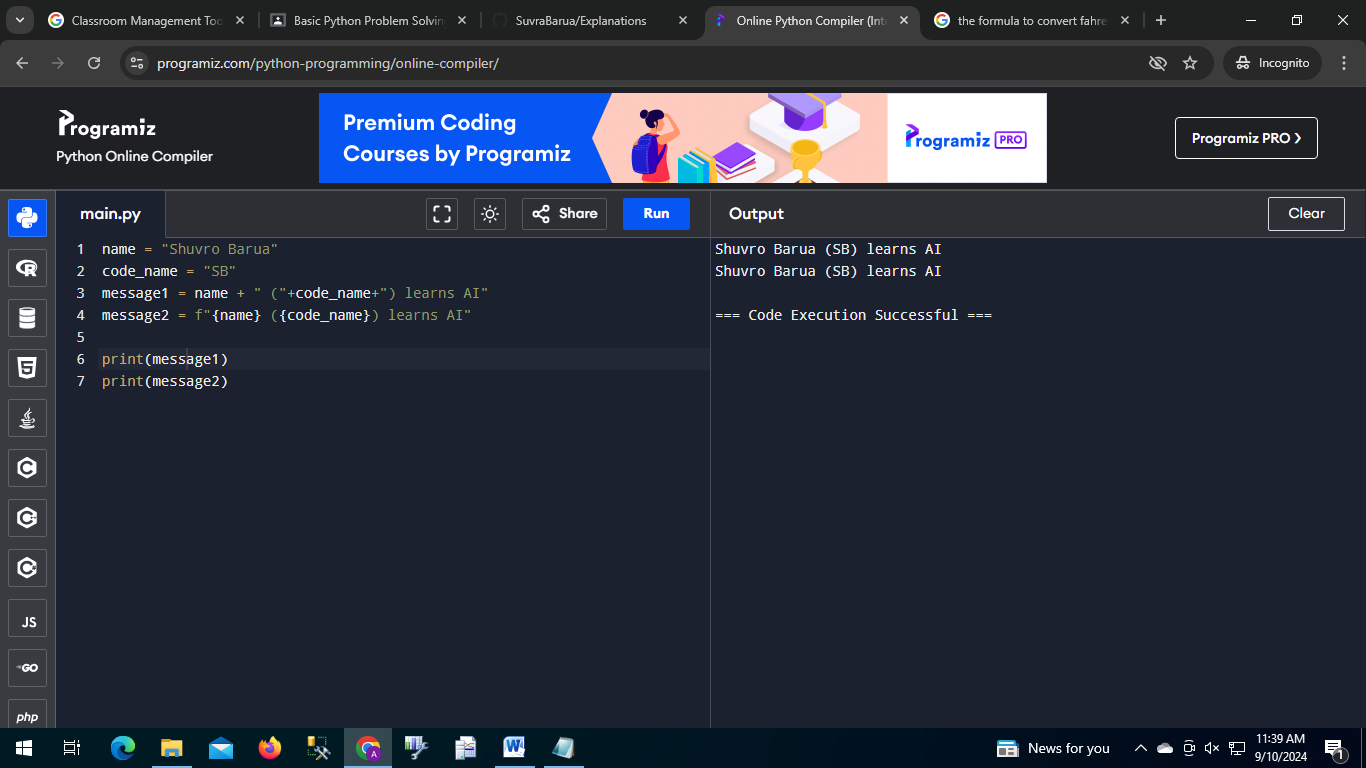


Image6: The output of lesson6.py

**Lesson 7**

Here is how this code works:

In this lesson, We learned about functions and methods.

Function: These are independent and can be used anywhere

Method: It can be used only under an object.

Then we used some methods and a function in code:

Methods:

upper() => It makes every character of a string uppercase.

lower() => It makes every character of a string lowercase.

Find() => It is used to find the starting index of any strings inside of a string.

Replace() => It is used to replace part of a string.

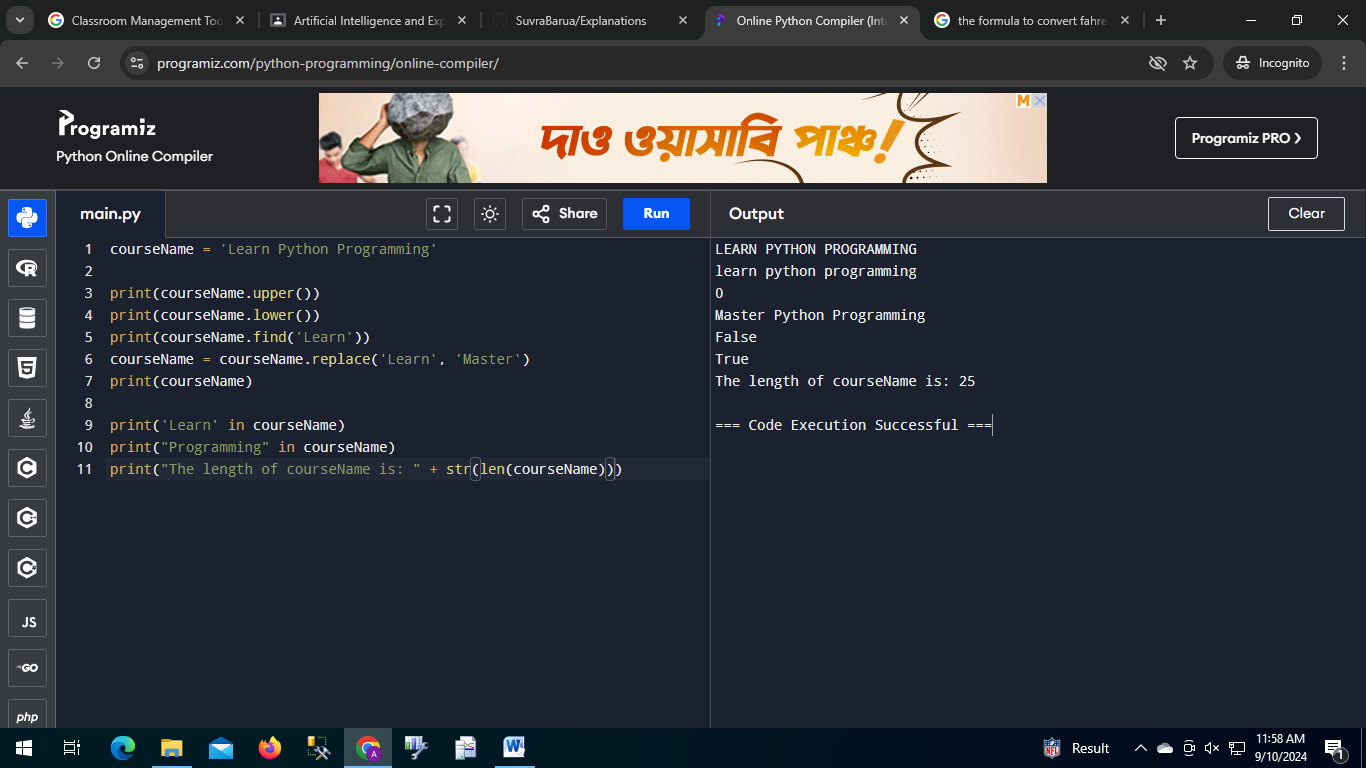


Image7: The output of lesson7.py

**Lesson 8**

(Mobile diye kora because my pc is broke)

Here is how this code works:

In this lesson we learned how math operators work in python.

+ -> the addition operator.

- -> the subtraction operator.

\* -> the multiplication operator.

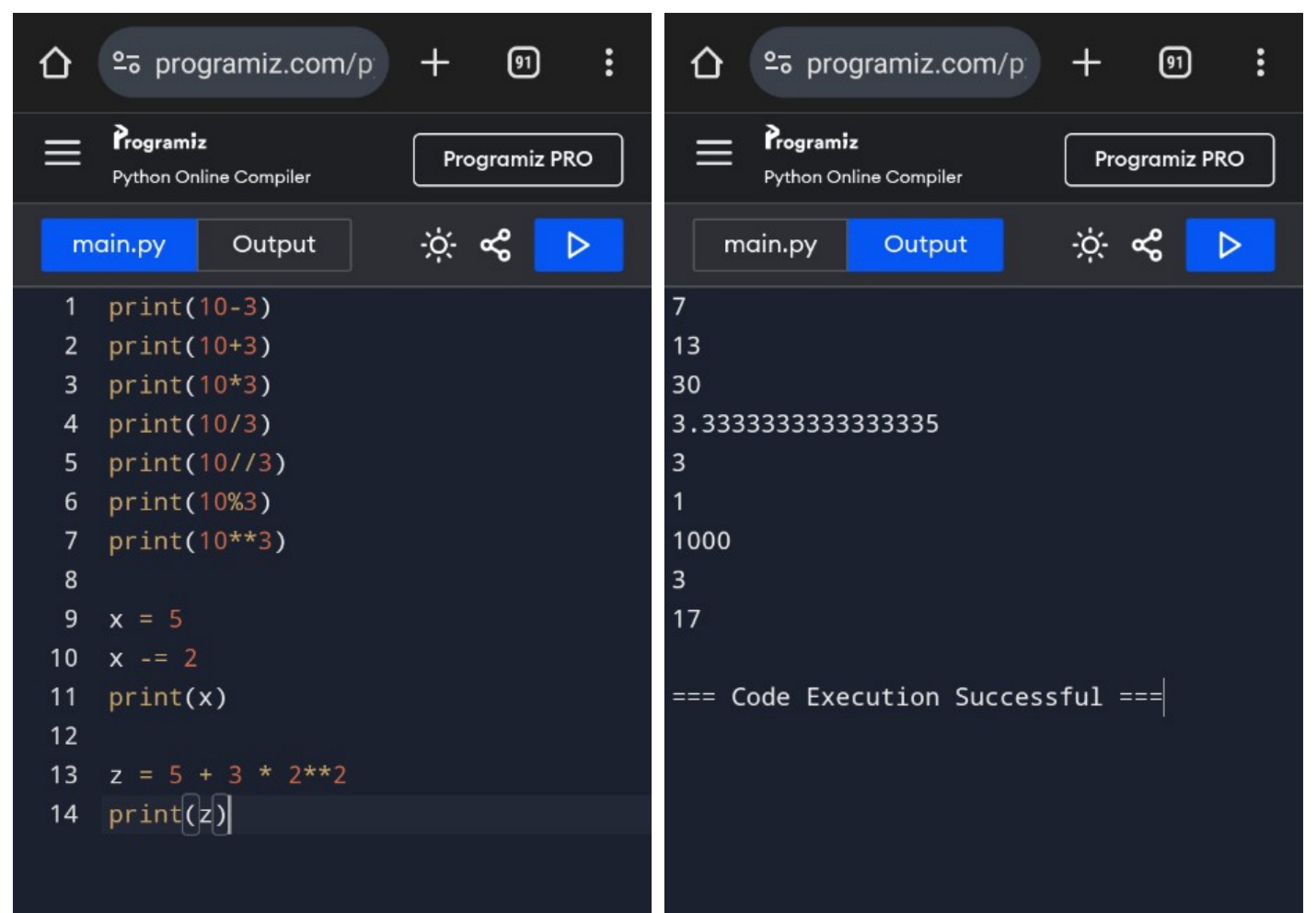
/ -> the division operator.

// -> division but gives an integer answer.

% -> the modulo operator.

\*\* -> the power operator (a\*\*b) = ab.

+= -> its like a = a+b

Image8: The output of lesson8lesson8.py

**Lesson 9**

Here is how this code works:

In this lesson, We learned about math module. It has some very important method for mathematics.

To use this, we need to import the module to our code by using import statement.

It has some useful method and functions like:

Methods:

ceil() -> returns the ceiling of a float.

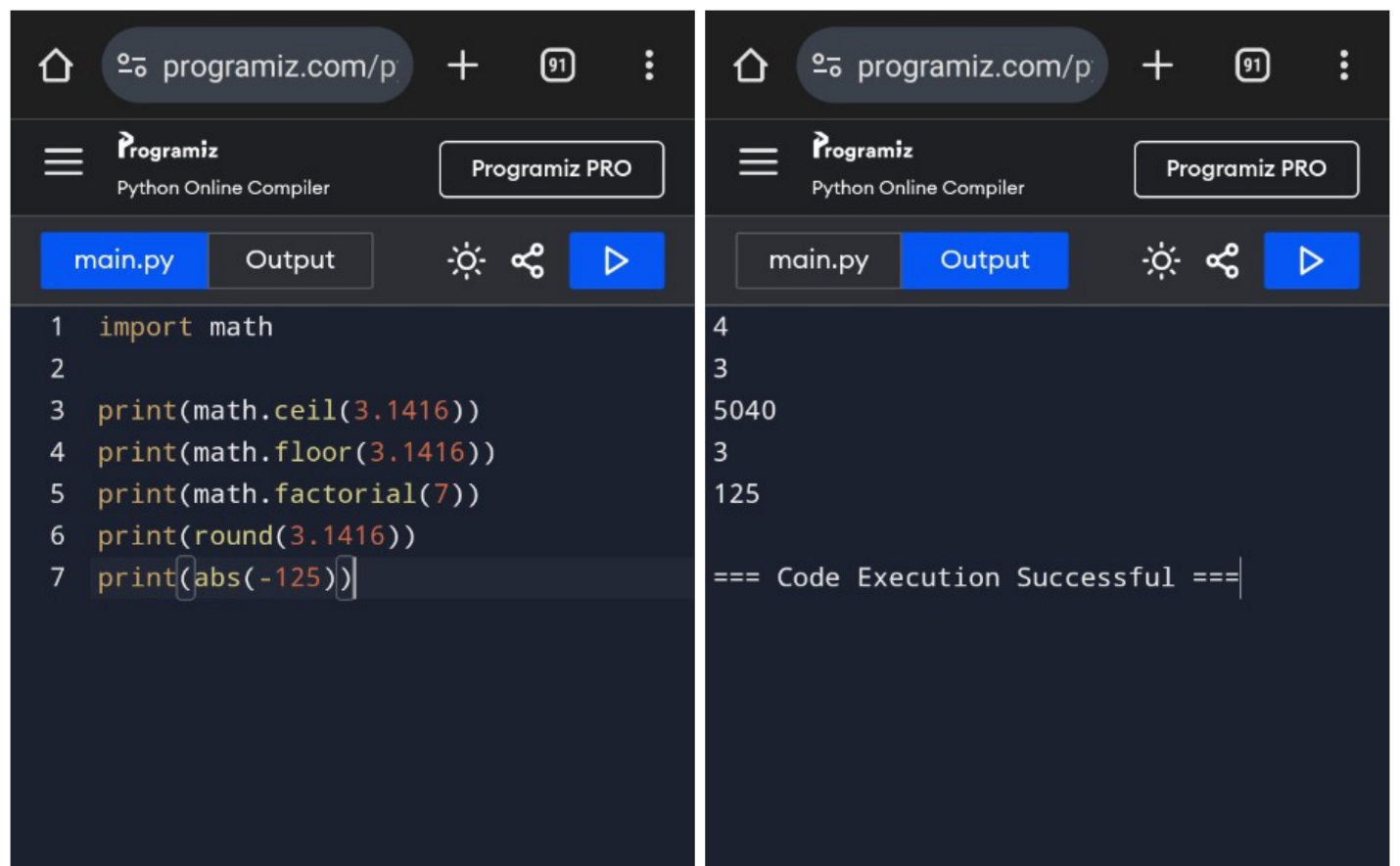
floor() -> returns the floor of a float.

factorial() -> returns the factorial of a number.

Functions:

round() -> It rounds a floating point number.

abs() -> Gives the absolute value of a number.

Image9: The output of lesson9.py

**Lesson 10**

Here is how this code works:

This code shows how a if statement works:

First we created two boolean variable named is\_rainy and is\_sunny.

Then we used if statement to print following sentences:

If is\_rainy is true then it should print You should carry an umbrella.

If is\_rainy is false but is\_sunny is true the elif statement will be executed and it should print No need to carry an umbrella

But if both is false then the else should be executed and it should print please check the weather.

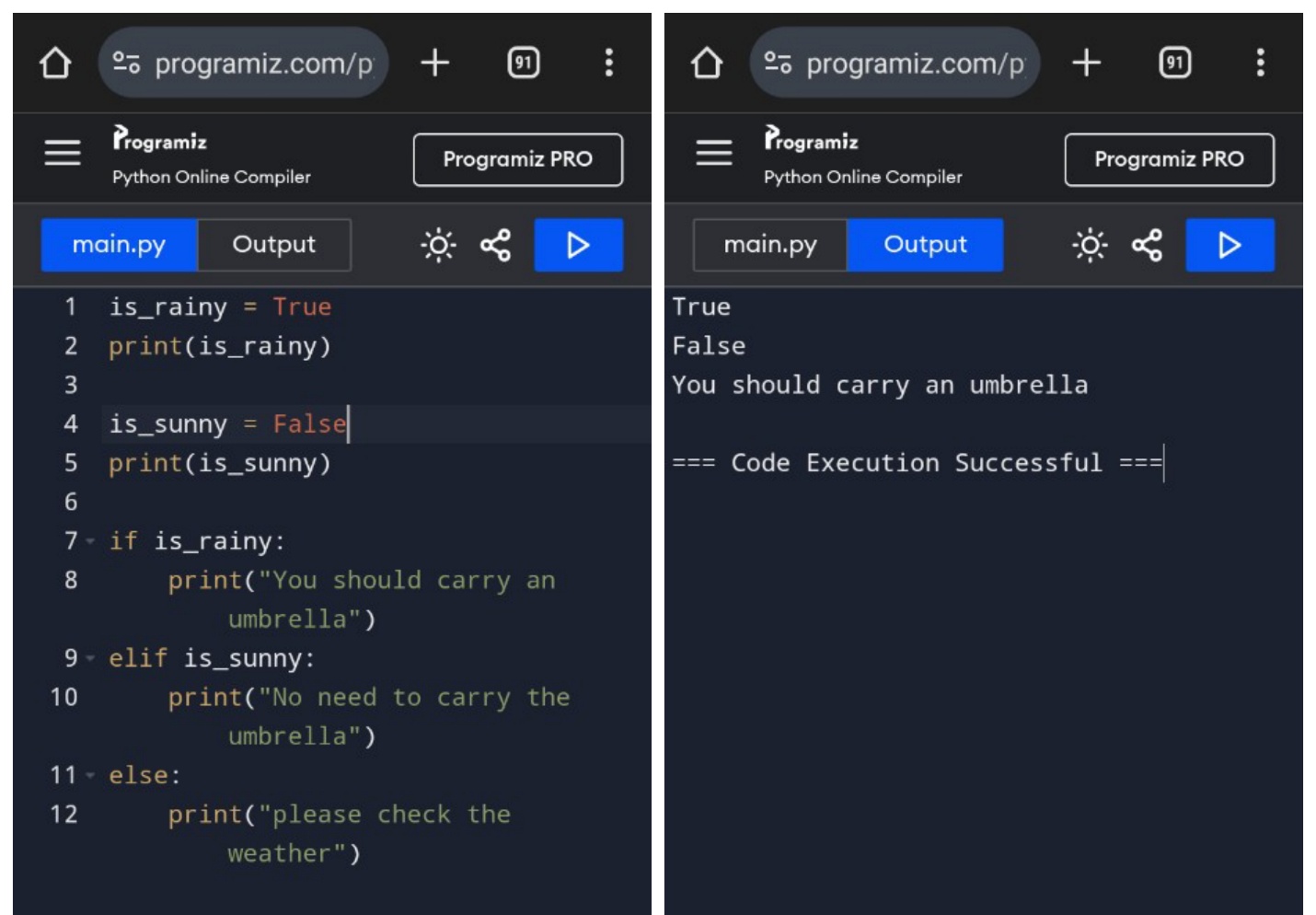


Image10: The output of lesson10.py

**Lesson 11**

Here is how this code works:

This lesson shows the uses of logical operator in python.

First we declared 3 variables, good\_condition, reasonable\_price and poor\_foundation.

We want to print We will buy the house if good\_condition and reasonable\_price both are true. In that case we can use the logical and operator.

If we want to print when one of them is then we can use the or operator.

But if we want to print something when a variable is false then we can use the logical not operator.

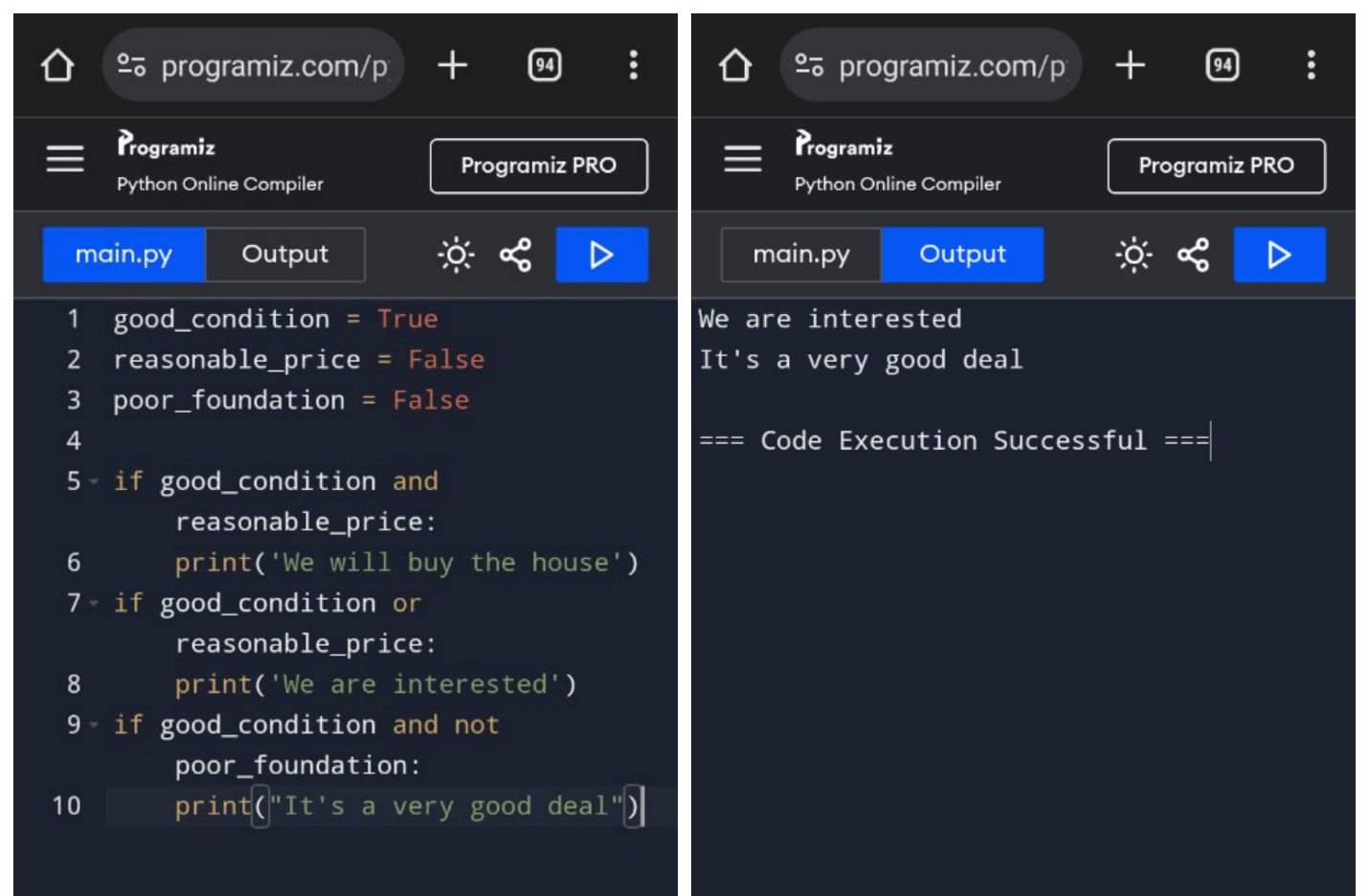


Image11: The output of lesson11.py

**Lesson 12**

Here is how this code works:

We learned about the comparison operators in this lesson:

== -> it is used to check if two variable is equal or not.

!= -> Not equal operator, will give true if two numbers are not equal.

> -> gives true if first element is greater.

< -> gives true if the second element is greater.

In this code we used these operator to print some sentences.

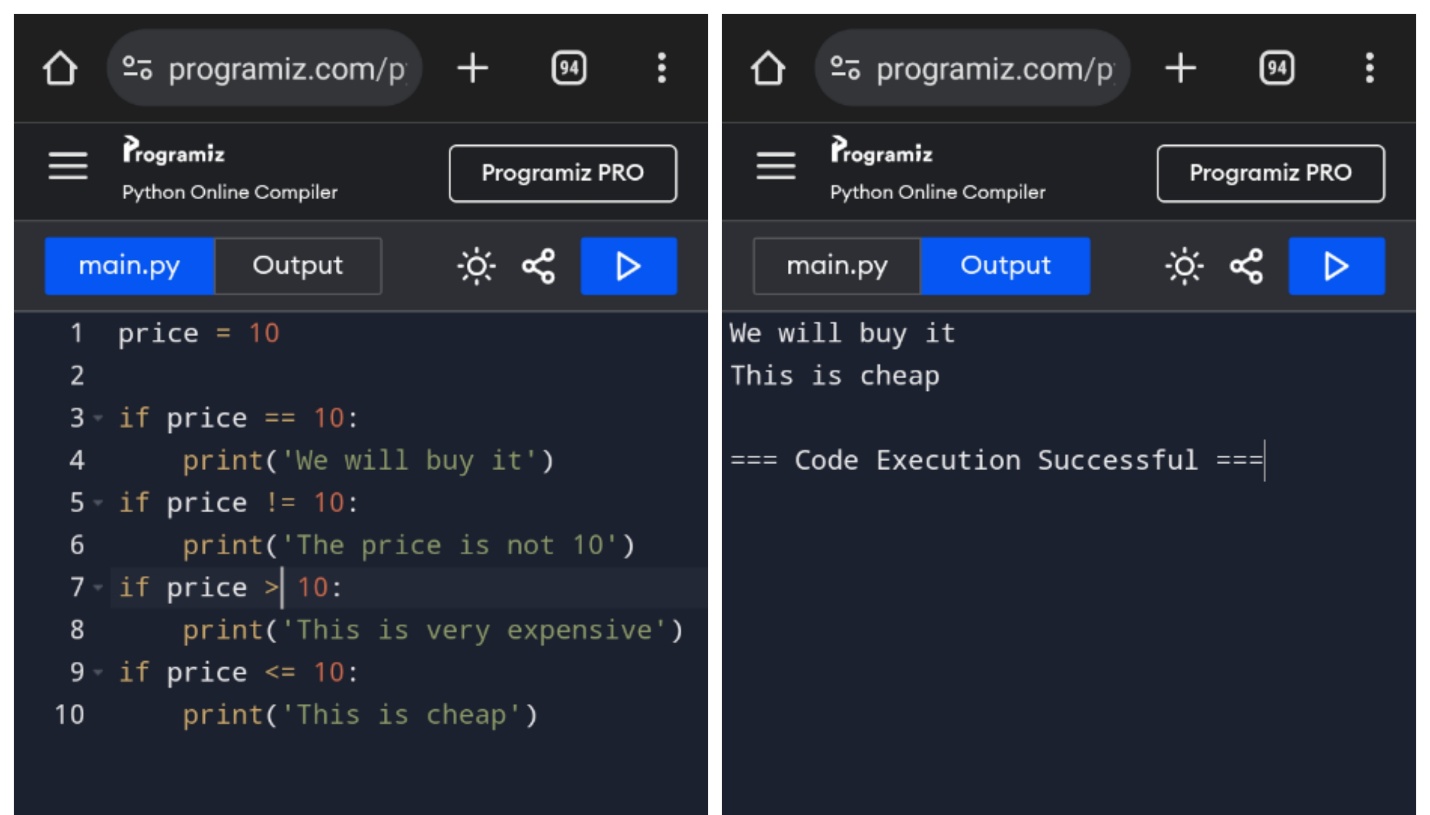


Image12: The output of lesson12.py

**Lesson 13**

Here is how this code works:

In this lesson, we learnt about while loop.

First we declared a variable to be used in the loop. The while loop works until certain condition is true. In this case, the i variable was 0 at the beginning. So the condition i<5 will be true. So the code inside the while loop gets executed and it prints i and increments i by one. It does that until i becomes equal to 5 because then i<5 evaluates to false.

Then it prints The while loop ended.

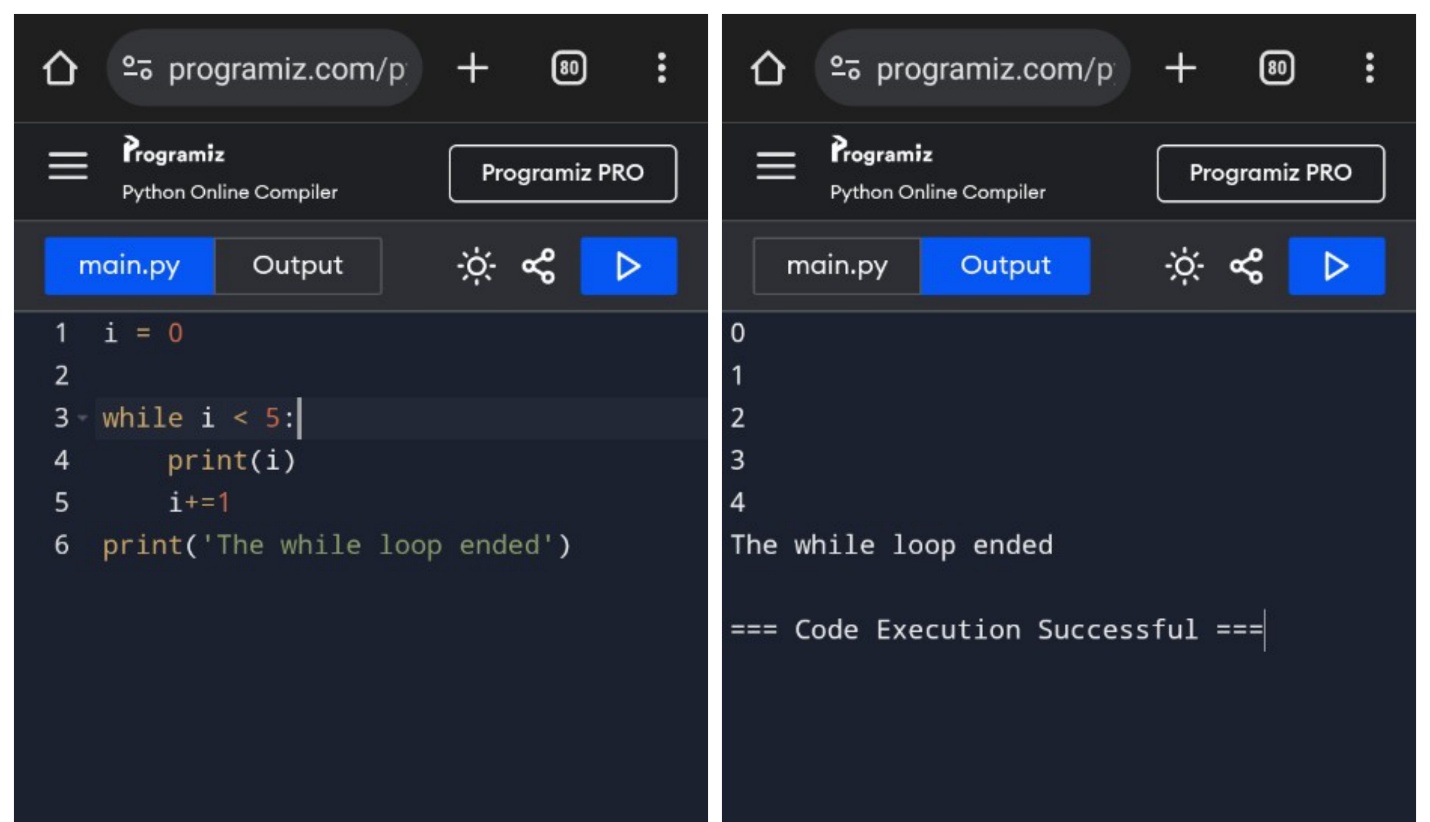


Image13: The output of lesson13.py

**Lesson 14**

Here is how this code works:

In this lesson, We created a guessing game using knowledge we gathered so far. First we created 3 variables actual\_price, guess\_count and guess\_limit.

In this game we want to give the user 5 tries. So, we set the guess\_limit to 5. Every time user inters a guesss we incremented the guess\_count by one and if the guess\_count becomes 5 the loop stops and you’ve failed prints in the window. But if the user guesses the actual\_price correctly then You’ve won! Prints in the window. In both way the game ends.

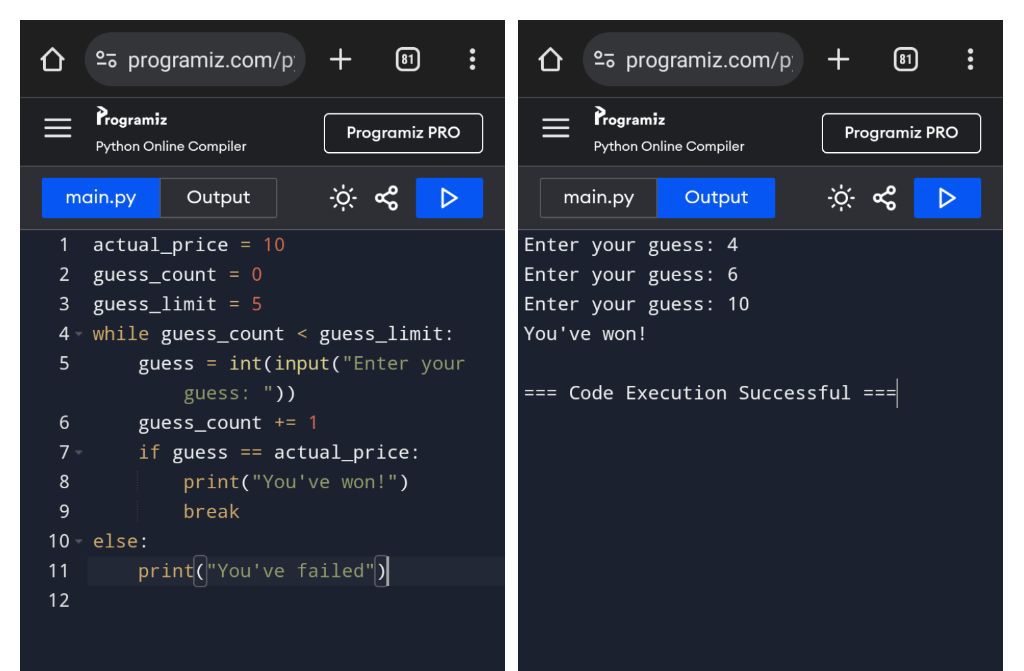


Image14: The output of lesson14.py

**Lesson 15**

Here is how this code works:

In this lesson, We learned about for loops in python.

In the first example, we learned about how to iterate through the characters in a string. We can do so by using the for keyword the a variable name that will go through all the characters of that string then in keyword followed by the name of the string variable or the string itself.

We can also iterate over the element of a list using the for loop. Example 2 shows that.

After that we printed the numbers of a number list using for loop.

We used the range() function to create a list.

Then we calculated the sum of a number array using for loop.

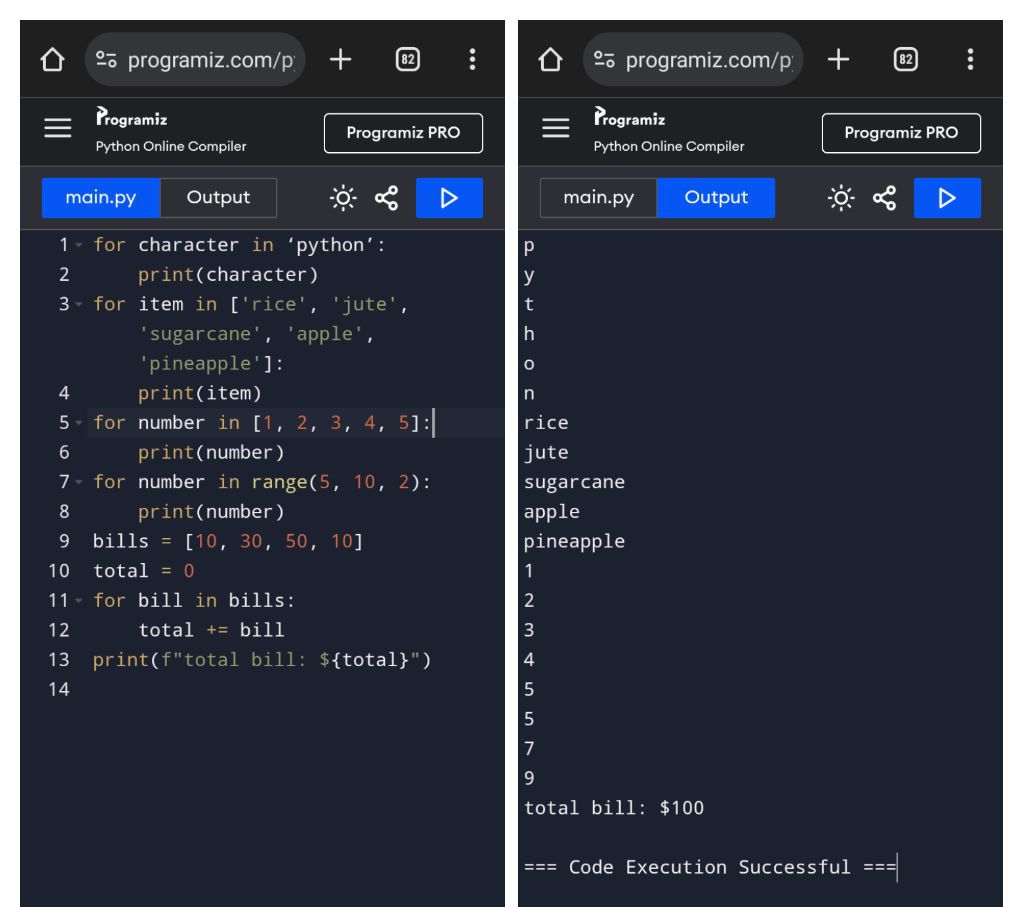


Image15: The output of lesson15.py

**Lesson 16**

Here is how this code works:

In this lesson, We used nested for loops to print all the coordinates in a 4 by 4 area (x, y):

First for loop to iterate over the value of x and the second inner for loop to iterate over the value of y.

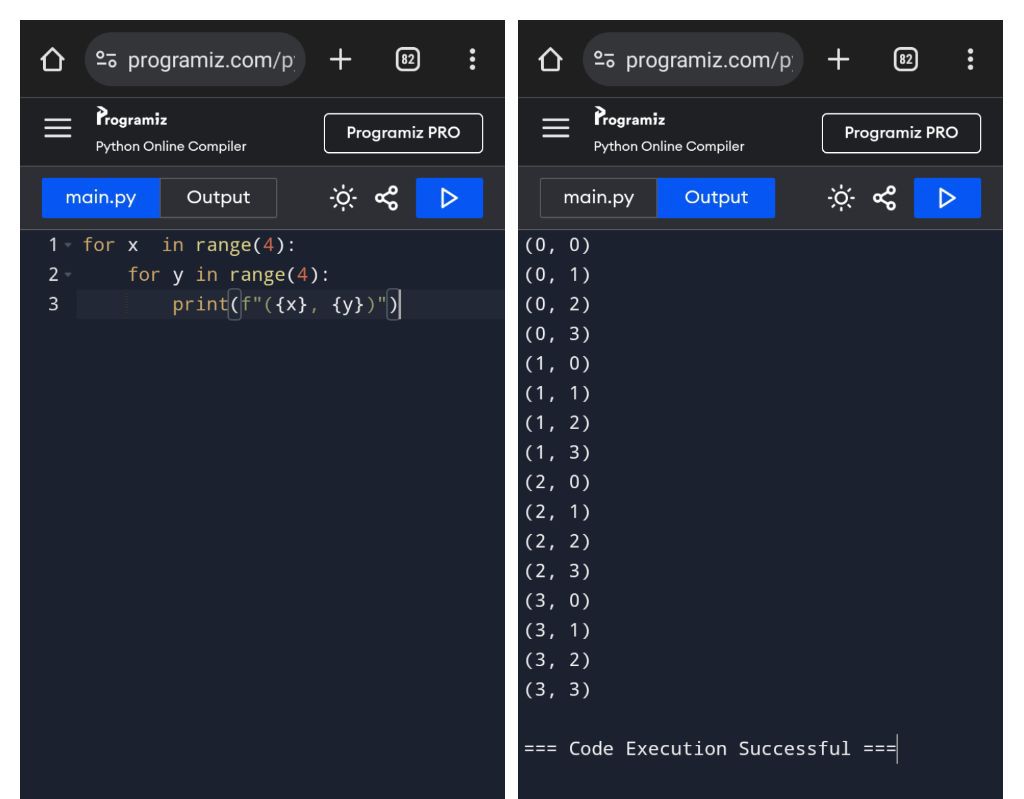


Image16: The output of lesson16.py

**Lesson 17**

Here is how this code works:

In this lesson, We learned about lists in python.

First we created a list named grocery\_list that has 4 elements.

We can print the list using print(list\_name)

We can print any element of the list using (list[index])

We can print a part of the list using (list[index1:index2]

Then we created a number list and found out the maximum value in it using for loop and if statement.

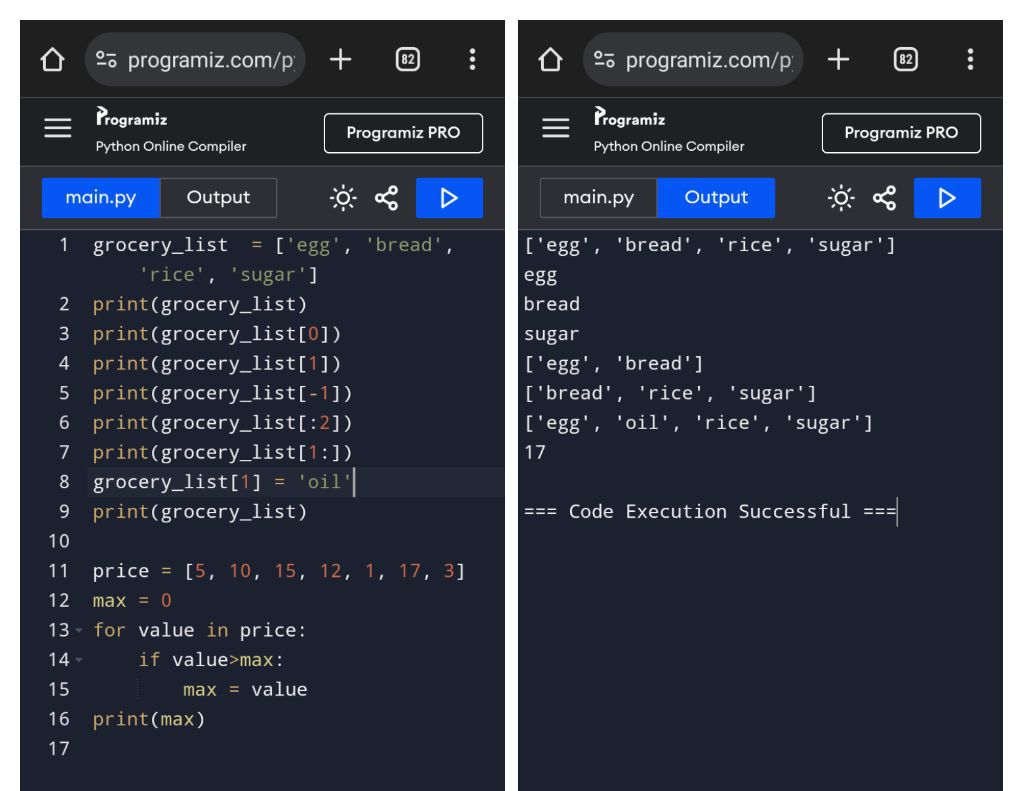


Image17: The output of lesson17.py

**Lesson 18**

Here is how this code works:

In this lesson, We learned about 2D lists.

In this example we created a matrix using the 2D list. Then we printed out the matrix simply using a print statement.

If we need to print an entire row, we can do so by using print(matrix[indexOfTheRow])

If we need to access any element of the matrix, we can do so by using matrix[rowNumber][columnNumber] and then we can print it, change it or update it.

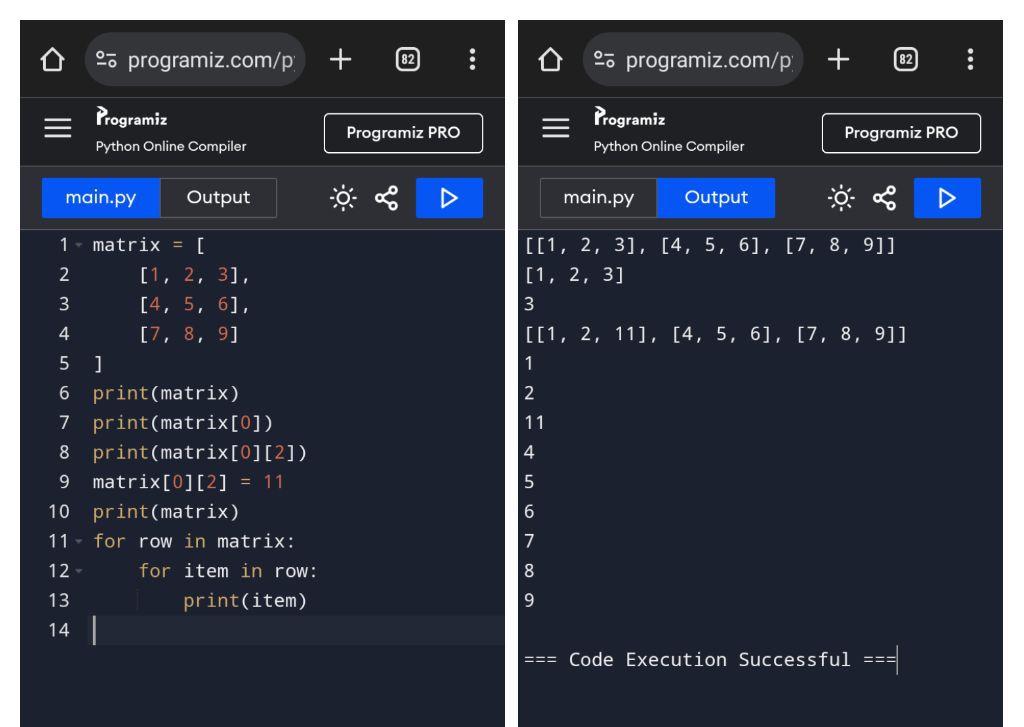


Image18: The output of lesson18.py

**Lesson 19**

Here is how this code works:

In this lesson, We learned about some useful methods of the list.

append() -> To add number at the end of the list.

insert() -> To add a number at a specific index.

remove() -> To remove a number.

sort() -> To sort the list.

reverse() -> To reverse the list.

count() -> To count the occurrence of a number in the list.

index() -> To get the index of a number of the list.

pop() -> To remove the last element of the list.

copy() -> To copy the entire list to another list.

clear() -> To empty the list.

We can also check to see if a number exists in a list by using the in keyword.

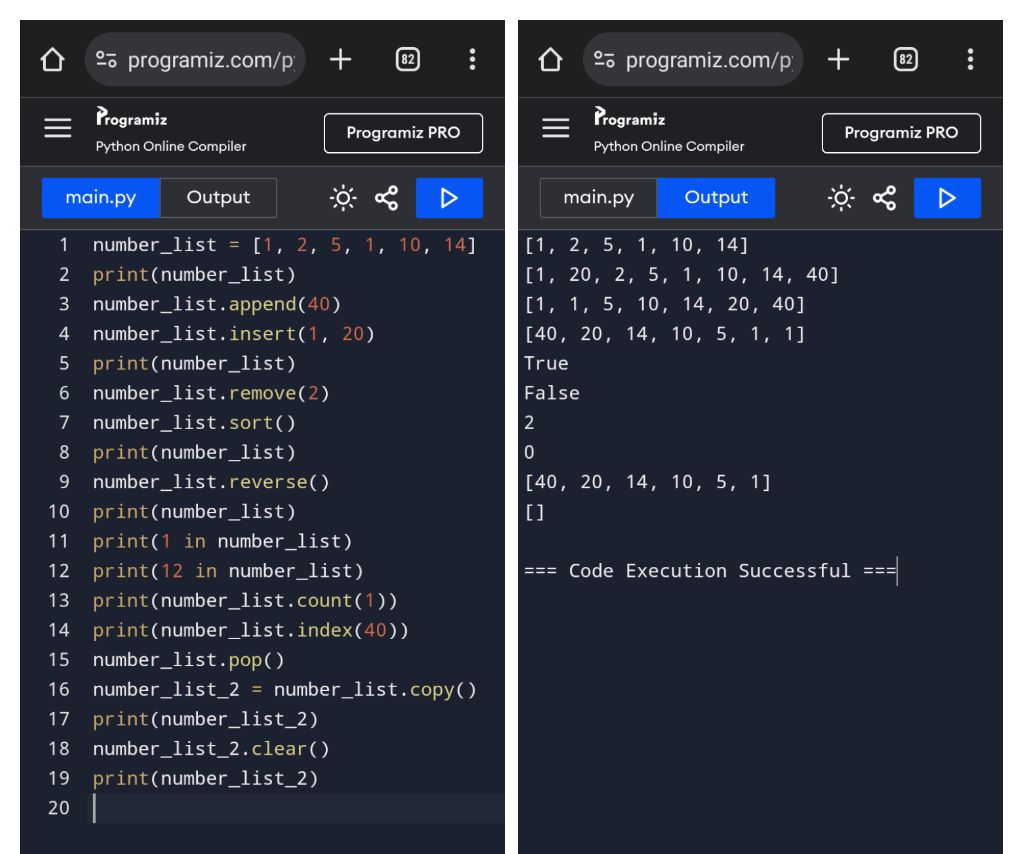


Image19: The output of lesson19.py

**Lesson 20**

Here is how this code works:

In this lesson, We learned about how to create an unique list from a list that consists duplicate items.

We did it by using a for loop and an if statement.

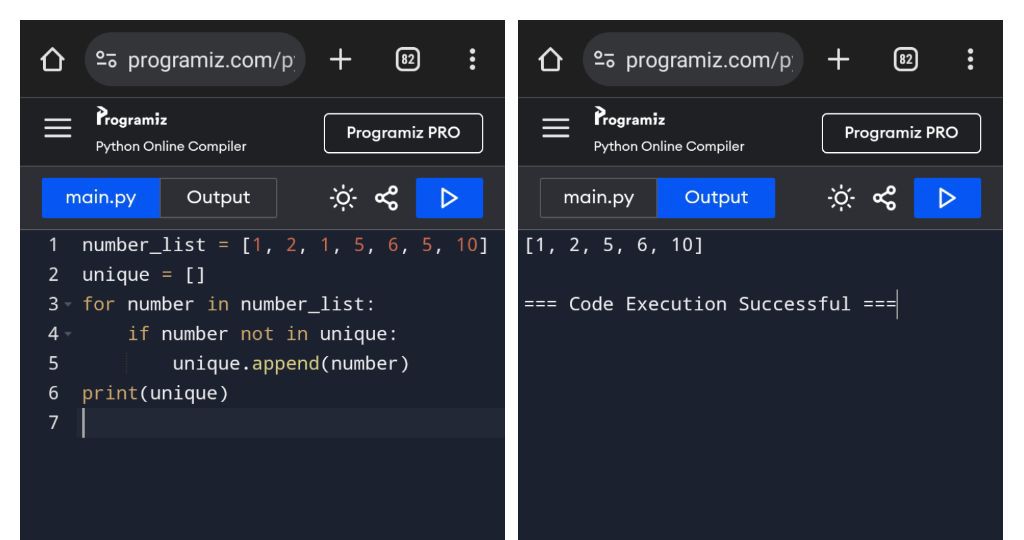


Image20: The output of lesson20.py

**Lesson 21**

Here is how this code works:

In this lesson, We learned about tuples and how to create them.

The main difference between tuple and list is that list is mutable and tuple is not but tuple is faster then list.

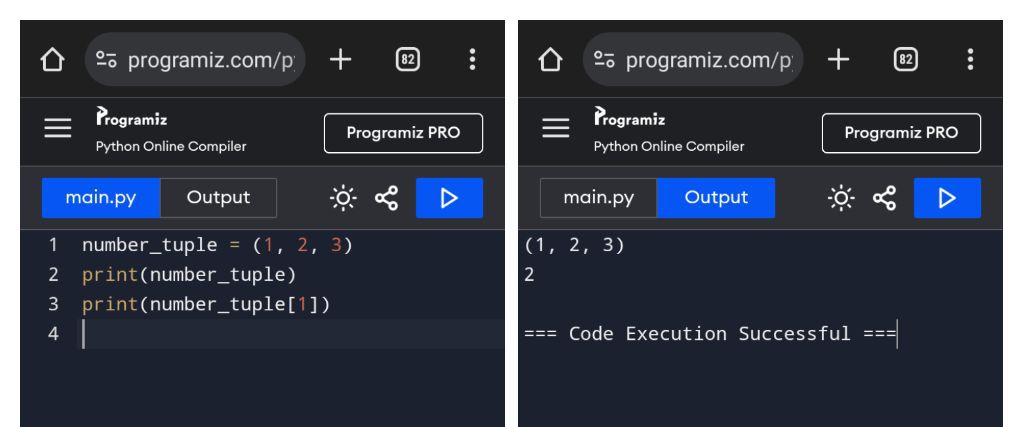


Image21: The output of lesson21.py

**Lesson 22**

Here is how this code works:

In this lesson, We learned about how can we efficiently assign some value from a list to a variable. In other languages, we need to do it this way,

x = list[0]

y = list[1]

z = list[2]

But in python, we can do so by using,

x, y, z = list.

So very simple.

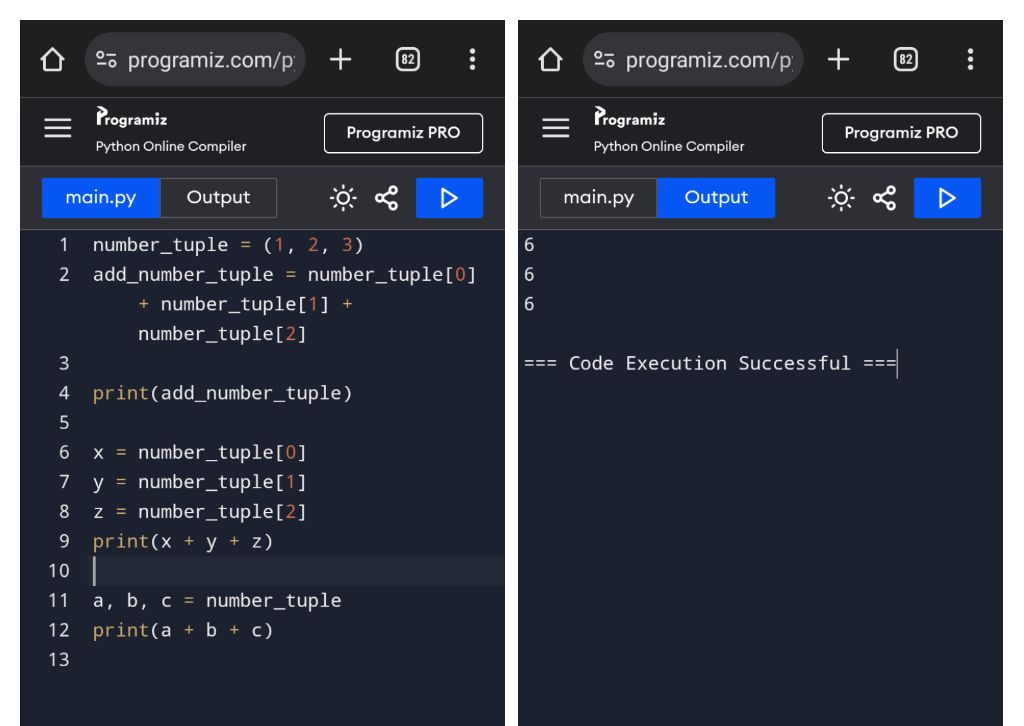


Image22: The output of lesson22.py

**Lesson 23**

Here is how this code works:

In this lesson, We learned about dictionary. It is a key-value pair. It can be declared by using name = { }.

And inside the curly braces, we can use “name” : “value” to add any key-value pair.

In our example, we created dictionary named user and added 4 key-value pairs. We then printed it using print statement. And like list, we can add or change the values or add new key-value pairs. The keys are like the indexes in list.

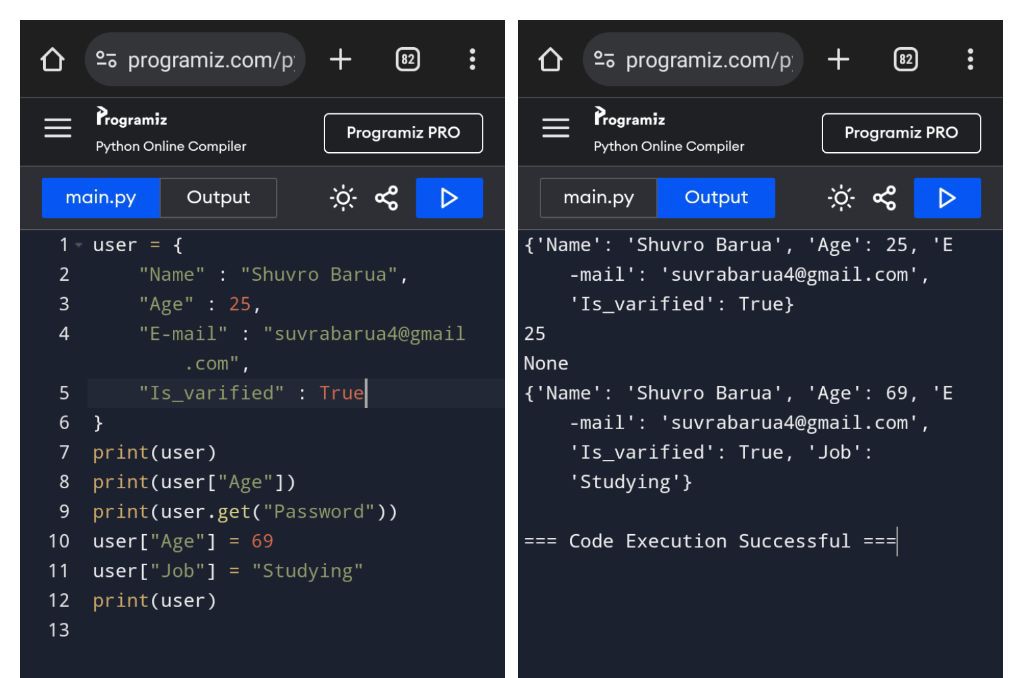


Image23: The output of lesson23.py

**Lesson 24**

Here is how this code works:

In this lesson, We created an emoji converter using the dictionary. First we took user input a sentence that consists of :( or :) Then we created a dictionary to convert the signs into imojis. The code inside the for loop:

Output += emoji.get(word, word) + “ “

This works by using the get method of the dictionary. If the word inside the first argument is available inside the dictionary then it gets replaced by the values but if it doesn’t exist then the second word will replace the first word.

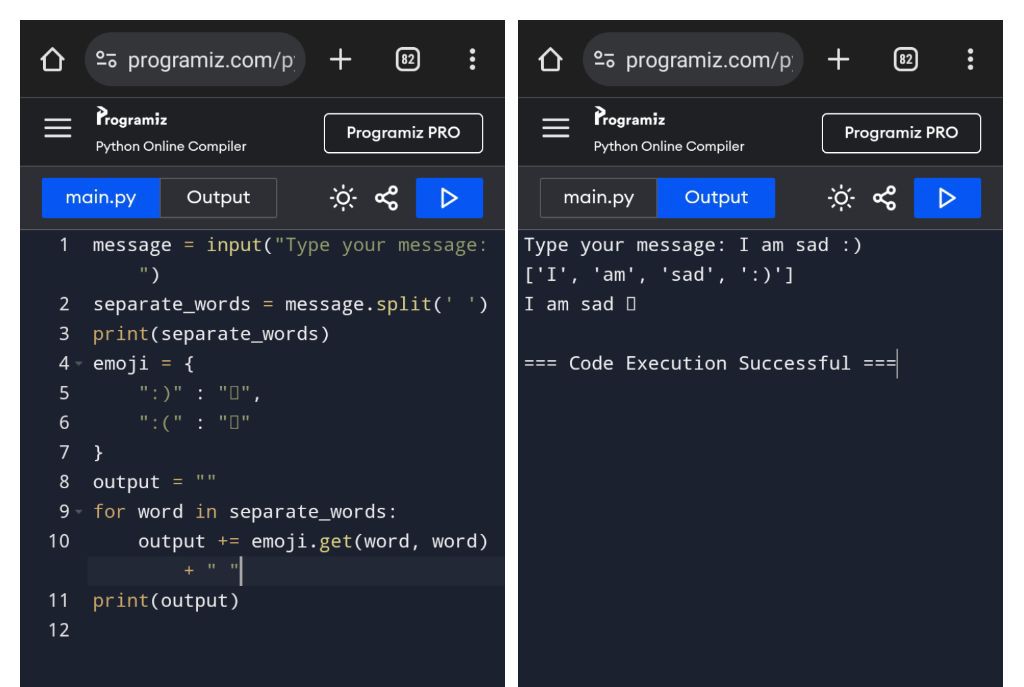


Image24: The output of lesson24.py

**Lesson 25**

Here is how this code works:

In this lesson, We learned about functions and how to create them.

We created a function called total\_cost() that has 3 variable inside, if it gets called, it calculates total\_cost considering the price, discount and shipping. But it has one flaw. If we want to change the value or other product. Then we need to change to inside of the function which is not very ideal.

So we created another function named total\_cost\_2() with 3 parameters. It does the same thing as the first function but is more flexible because everytime we need to change the value of some variable, we can do so just by changing the argument while calling. We also created another function named welcome that prints hi “name”

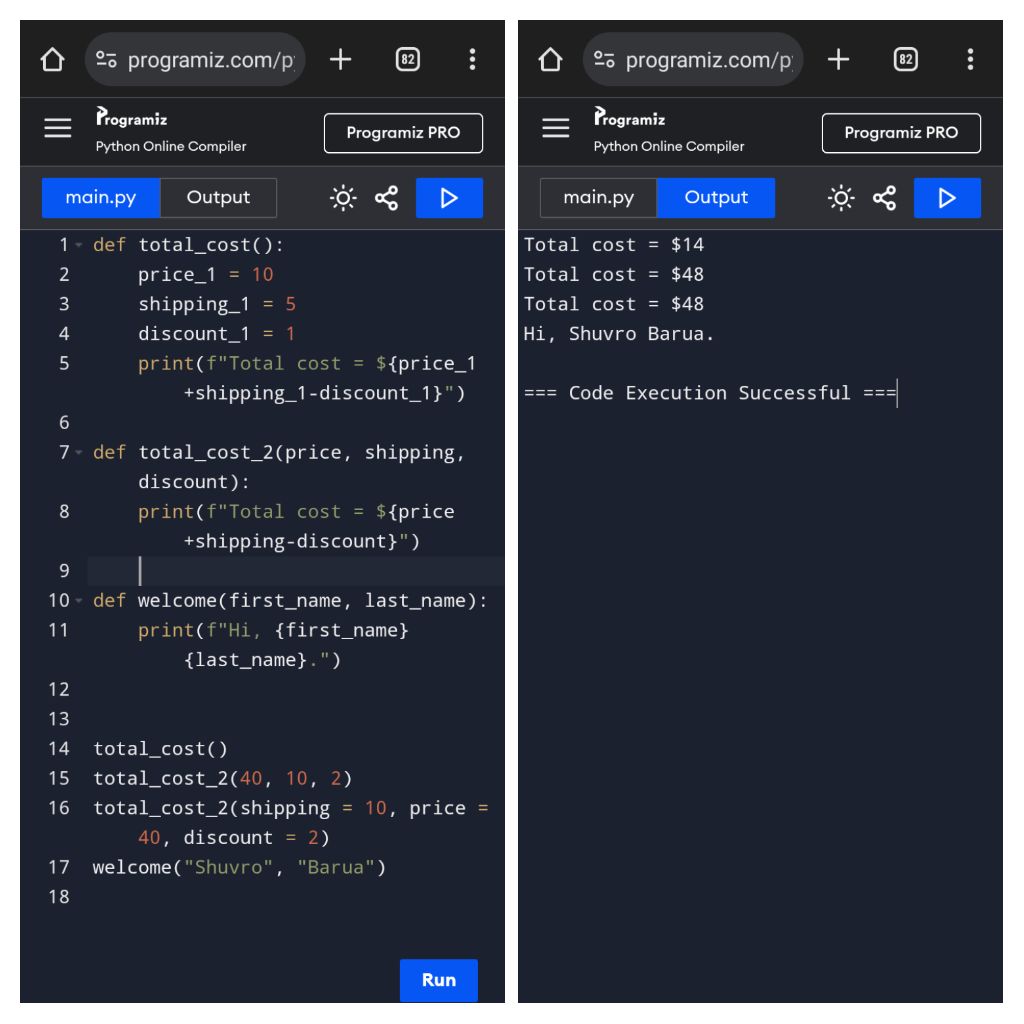


Image25: The output of lesson25.py

**Lesson 26**

Here is how this code works:

In this lesson, We created a function named add that has two argument and that returns the sum of those numbers.

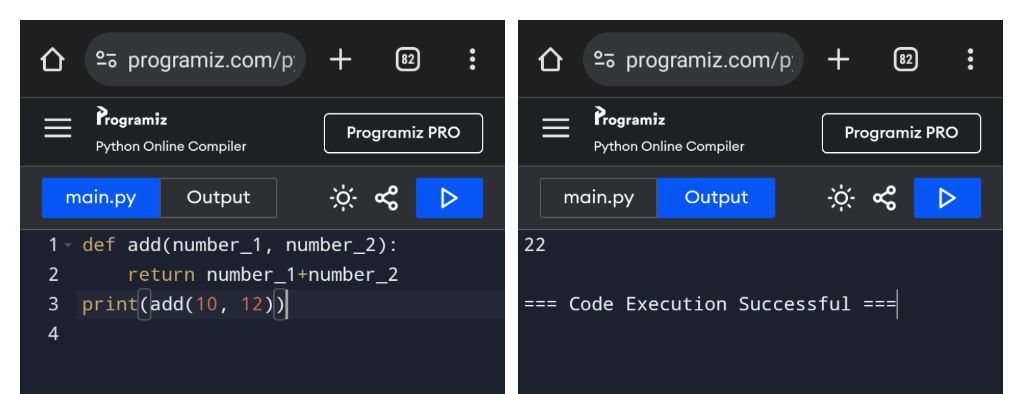


Image26: The output of lesson26.py

**Lesson 27**

Here is how this code works:

In this lesson, We created a function for the emoji converter from lesson-24 so that we don’t have to write that program every time we want to convert our emoji. We can do so by simply calling the function. To make it reusable, we separated the input segment from the function and made it as a parameter.

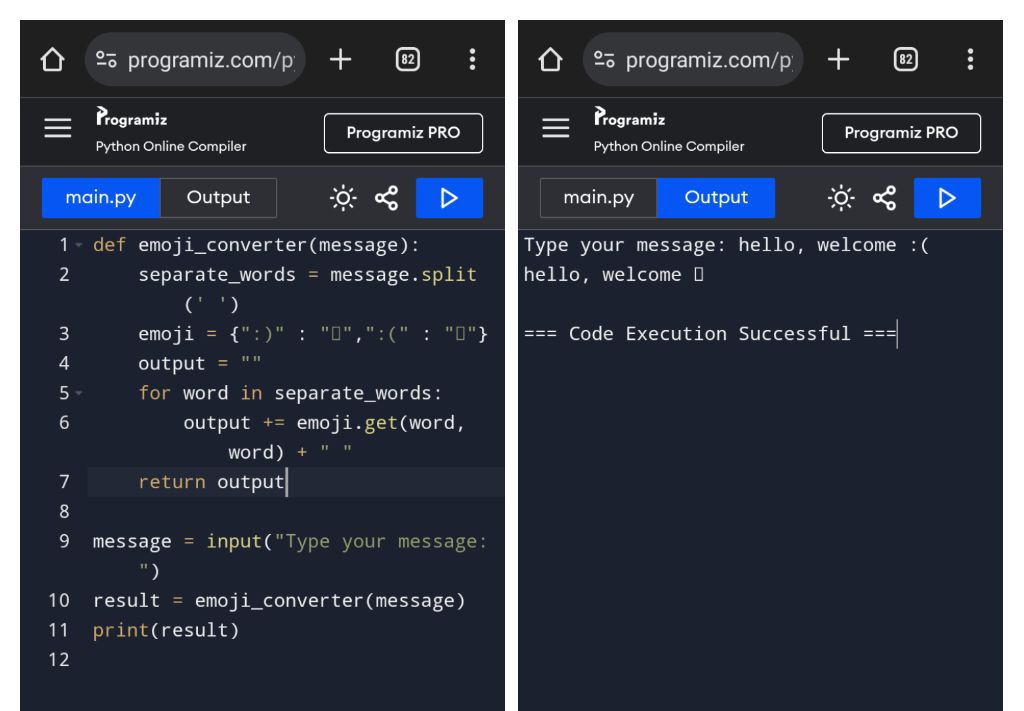


Image27: The output of lesson27.py

**Lesson 28**

Here is how this code works:

In this lesson, We learned about exception handling. While creating the temperature converter, we had came across a lot of exceptions. In this lesson, we tried to combat some of them using try and except block.

It works by putting dangerous code (The code that give an exception) inside the try block. Then we put the possible exceptions inside except block to try and catch them.

In our example, we created except block for two exception

ValueError -> When the user enters the wrong type of value.

ZeroDivisionError -> When we try to divide any number by zero.

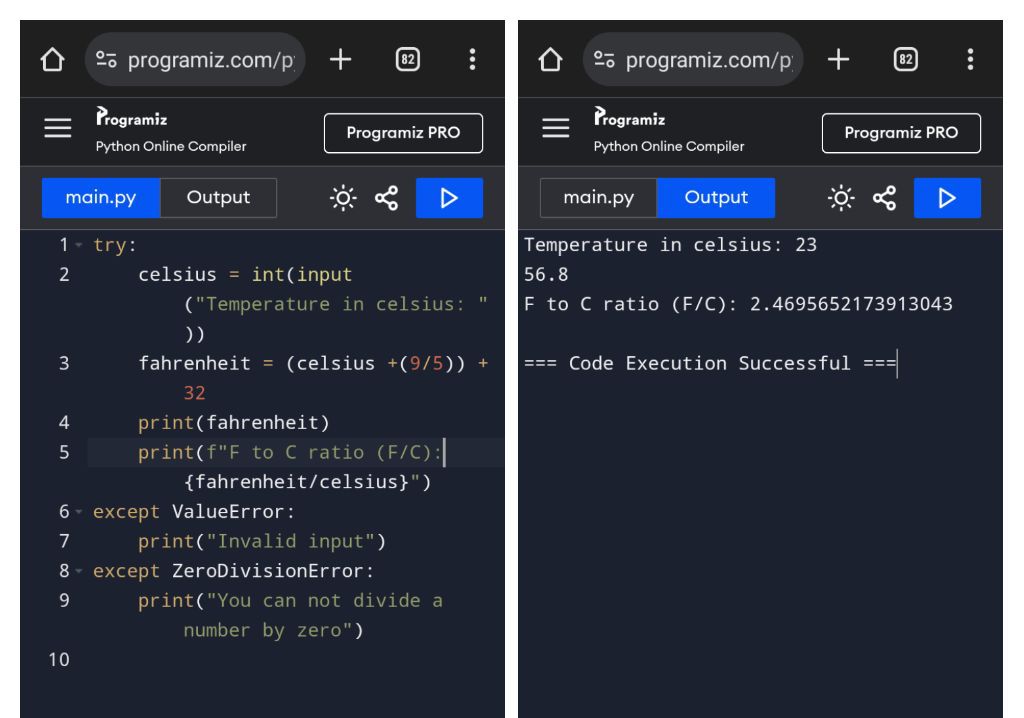


Image28: The output of lesson28.py

**Lesson 29**

Here is how this code works:

In this lesson, We learned about classes and the uses of it. We created a class named Keyboard. It has two methods named definition and number\_of\_keys.

To use any class inside our code, we need to create an object of that class. We created an object of the Keyboard class named my\_keyboard. After that we can use the dot operator to access the method and attributes of that class.

We can also define any attribute of the object.

We created another object named new\_keyboard.

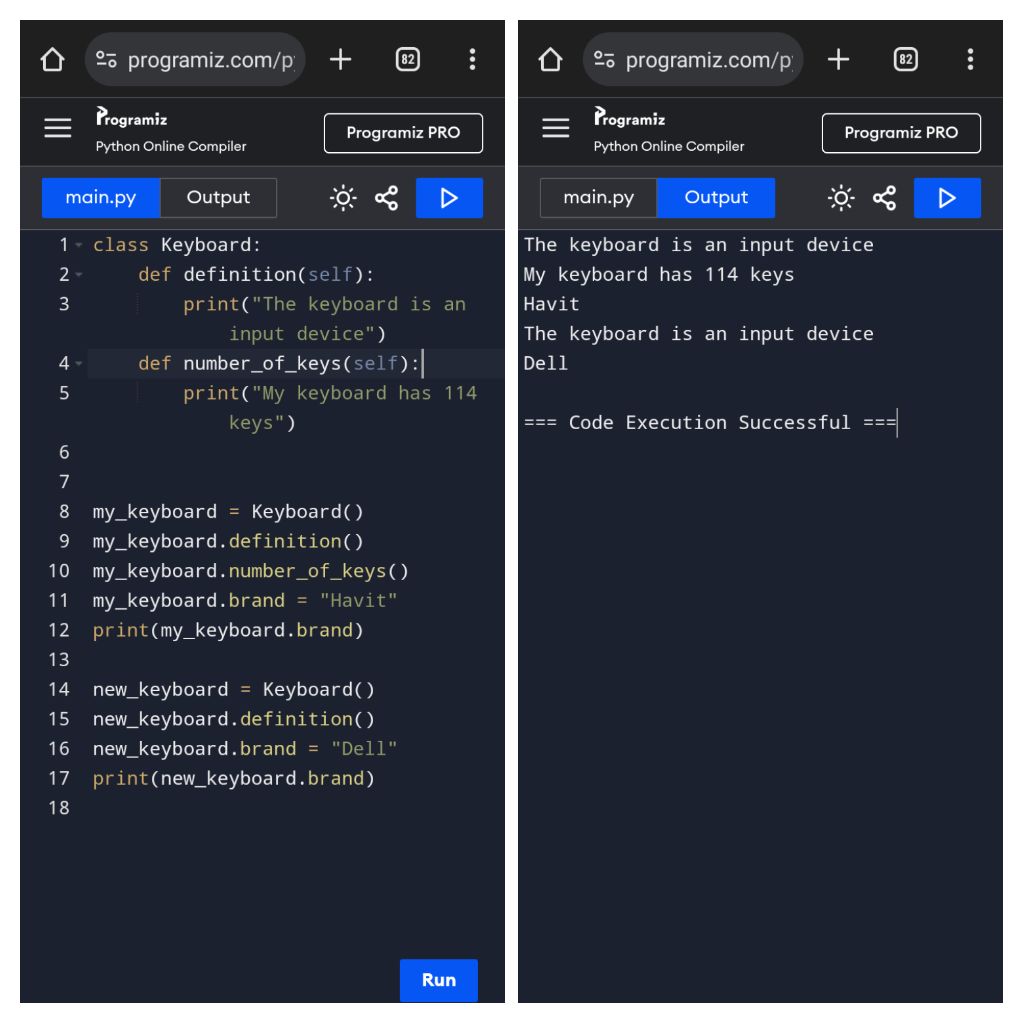


Image29: The output of lesson29.py

**Lesson 30**

Here is how this code works:

In this lesson, We learned constructor of a class. A constructor is what gets called when create an object of a class. It has the same name as the class but has parenthesis at the end. We can use \_\_init\_\_() to initialize the constructor of a class. If we give some parameter to \_\_init\_\_ then we must have to give some value to it while creating an object.

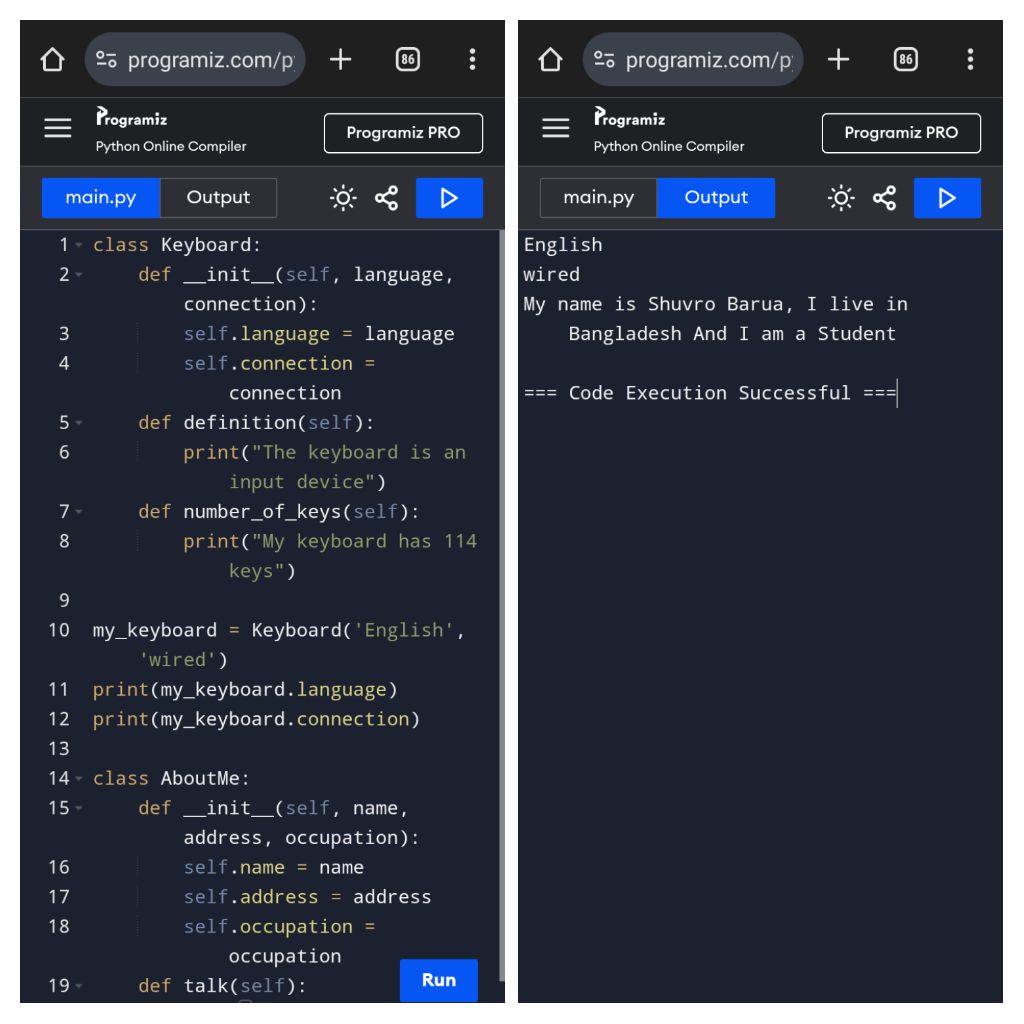


Image30: The output of lesson30.py

**Lesson 31**

Here is how this code works:

In this lesson, We learned about inheritance. We created a child class of the class Laptop named Desktop. We declare inheritance in python by using the parent class name inside of a parentheses while declaring the class name. The child class will have access to every method and attribute of the parent class. That why we can call the parts method from the Desktop even though it does not have any parts method.

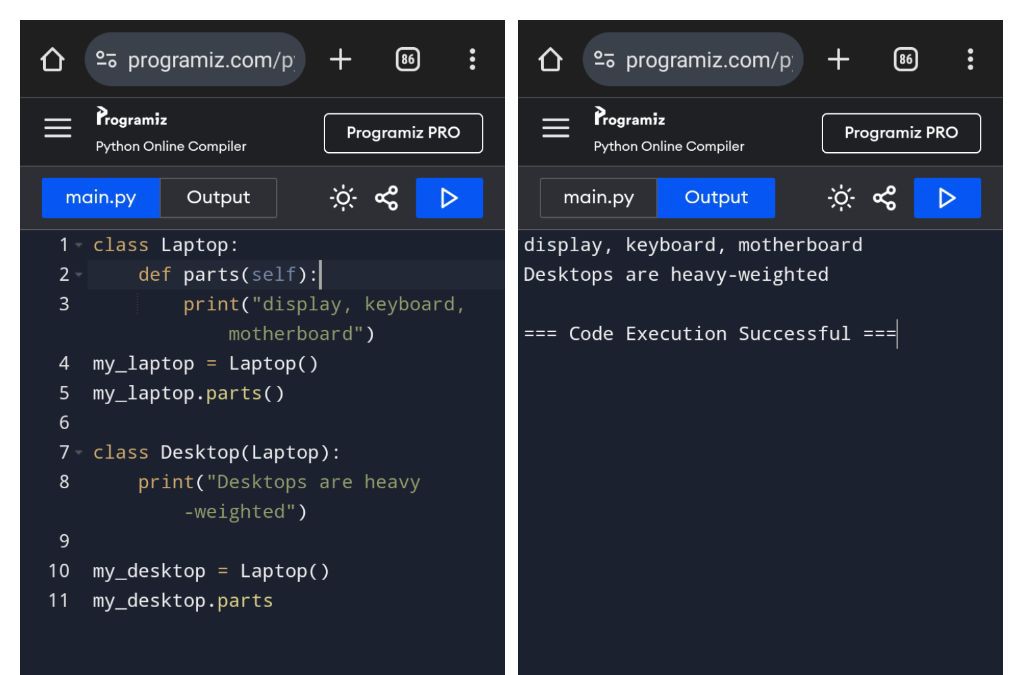


Image31: The output of lesson31.py