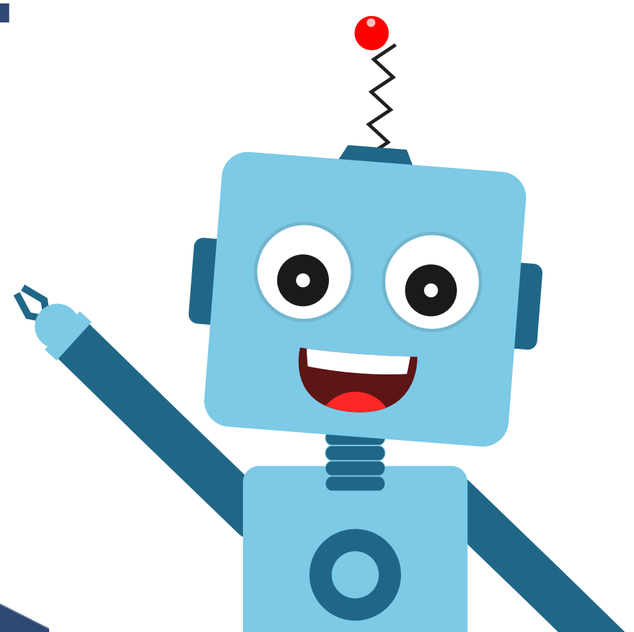
POWERED BY STEMpedia

Loops in Python – While Loop

Session 7



Topics covered

POWERED BY STEMpedia

1. Introduction to loops

2. While Loop

3. Activity:
    1. Number counter using Quarky

Introduction to loops

In this activity, you will be introduced to the first of the two types of loops,

Introduction to Loops

POWERED BY STEMpedia

- In programming, repetition of a line or a block of code is also known as iteration.
- A loop is an algorithm that executes a block of code multiple times till the time a specified condition is met.

Increment Loops :

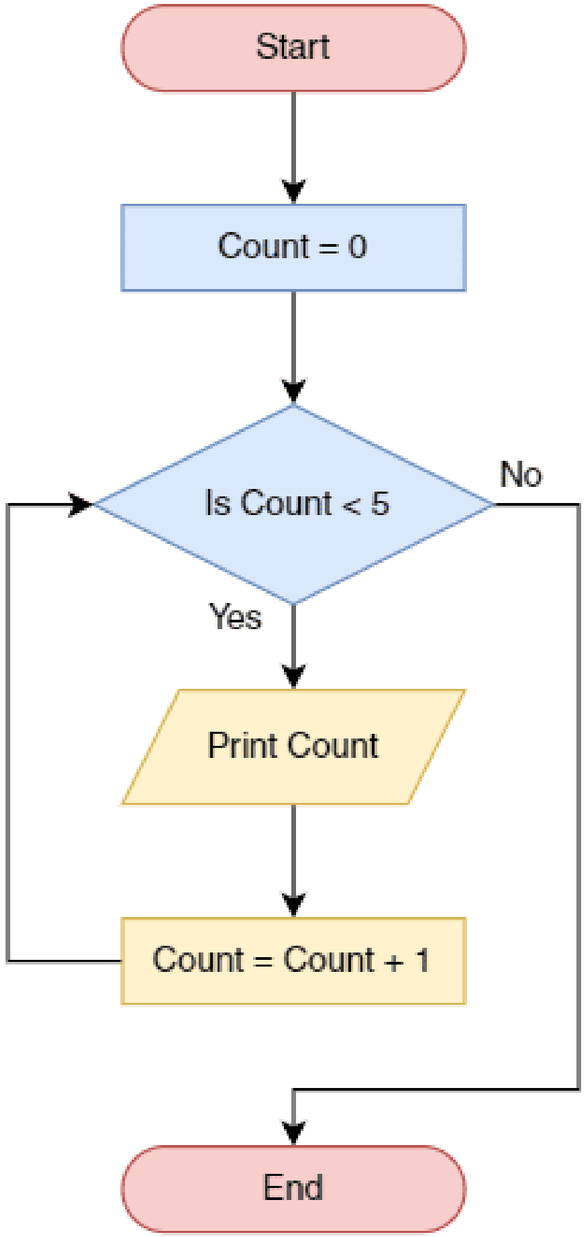
- Loops provide the facility to execute a block of code repetitively, based on a condition.
- To run a block of code in a loop, one needs to set a condition and set its number of iterations.
- Each time the condition is true, and the block of code executes once, it is counted to be one iteration.
- Before moving to the next iteration, one needs to increase the count of iterations to two. This is called incrementing a loop.

Increment Loops

POWERED BY STEMpedia

- For example, if you need to print numbers 0 to 4, you will execute a block of code with the Print statement in five iterations.

- With each passing iteration, you will increment the count by one.



Increment Loops

POWERED BY STEMpedia

Let us understand loops with a flowchart:

- Here every time the condition (Count < 5) is true, “Print count” gets executed. So, we do not have to write the “Print” statement multiple times. The loop takes care of that.

- What is important to note is that every loop must have an exit condition. In our example, the exit condition is (Count < 5). The loop will exit when the condition becomes false.

- Also, most loops will have a variable that is called a counter variable in programming terms. The counter variable keeps track of how many times the loop is executed. In this example, the “count” variable is our counter.

Benefits of Loops

POWERED BY STEMpedia

Below are the two important benefits of loops:

- Reduces lines of code
- Code becomes easier to understand

Different types of loops

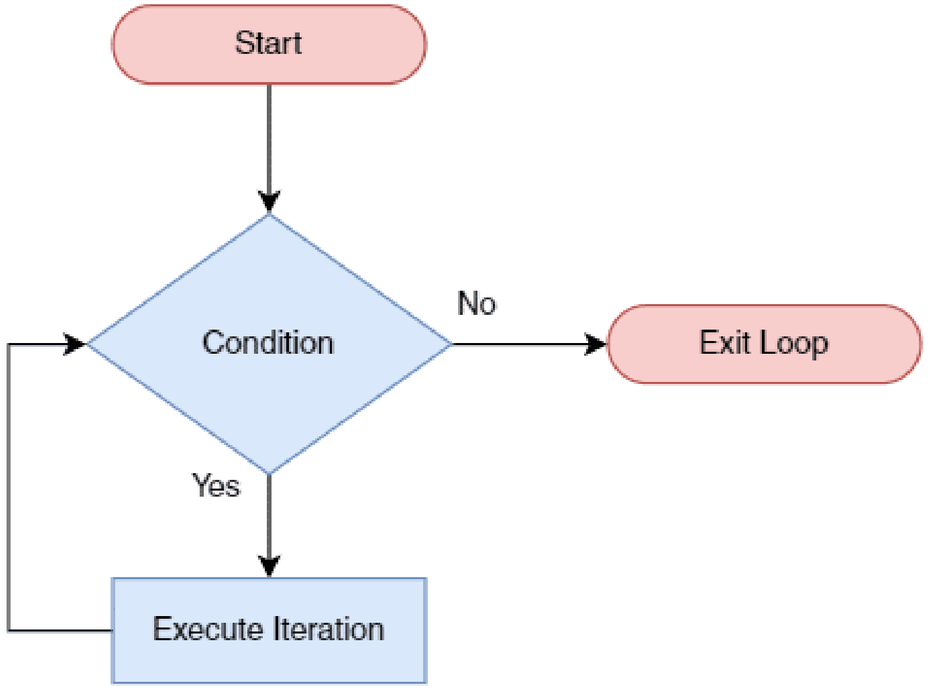
Loops make our code more manageable and organized. Let us now see what the different types of loops are:

- While Loop
- For Loop
- Nested Loop

While Loop

POWERED BY STEMpedia

- The While loop can execute a set of commands till the condition is true.
- While Loops are also called conditional loops.
- Once the condition is met then the loop is finished.



The syntax of the while loop is:

```
while condition: # condition is Boolean expression returning True or False
STATEMENTs BLOCK 1
```

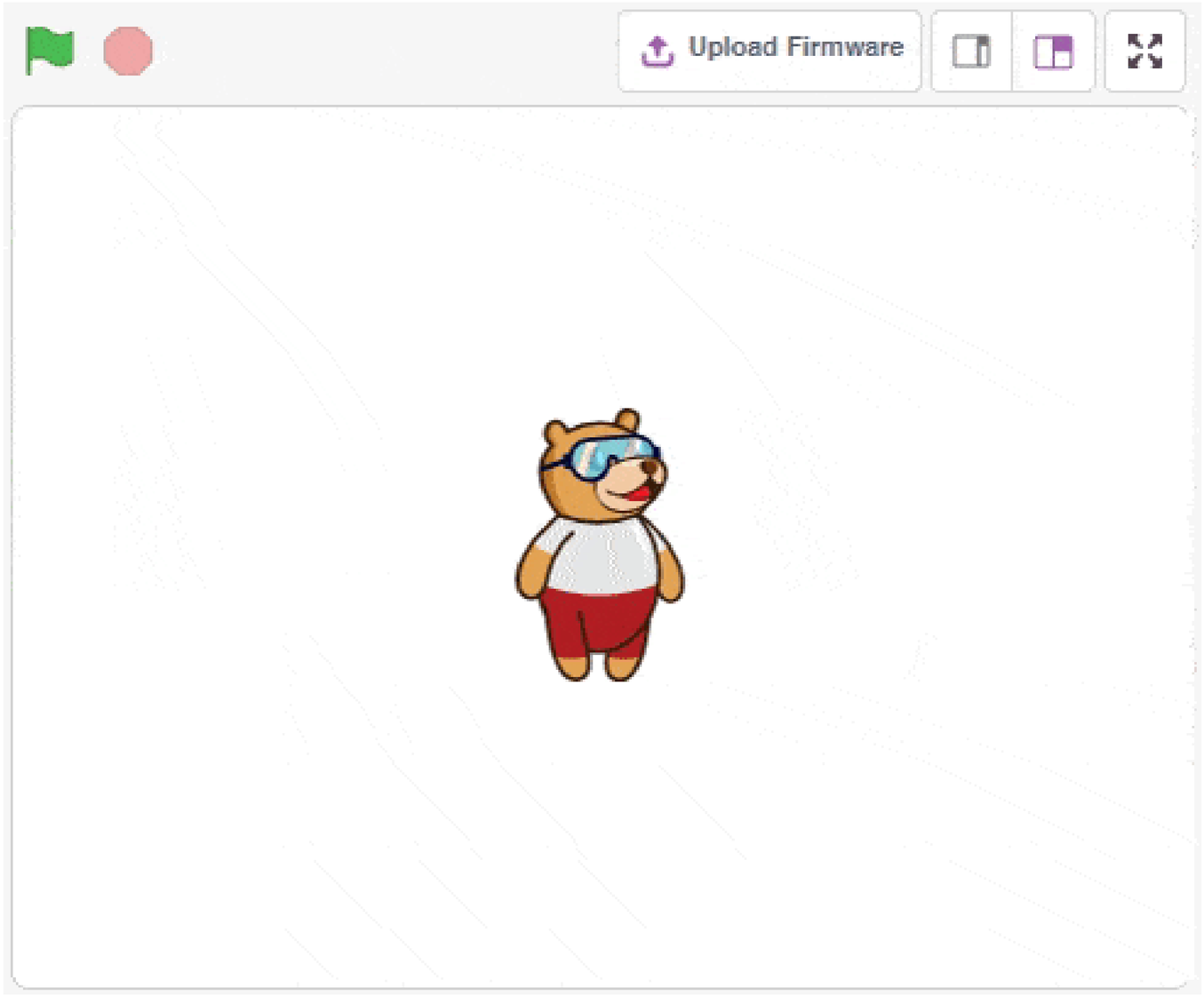
While Loop

POWERED BY STEMpedia

Example :

```
sprite = Sprite('Tobi’)

sprite.input("Enter the number")
N = int(sprite.answer())

i = 1

while (i <= 10):
        sprite.say(str(N) + " * " + str(i) + " = " + str(N*i), 1)
        i = i + 1

sprite.say("I am out of the loop!")
```

Output

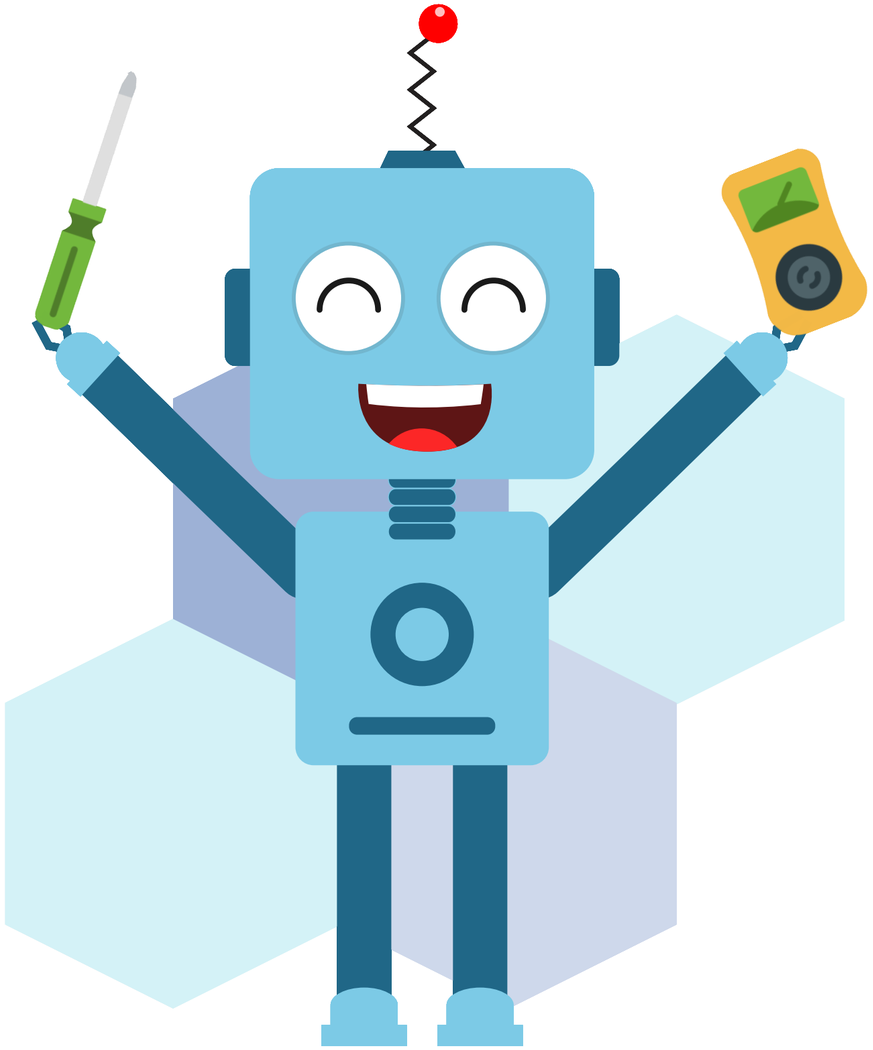


ACTIVITY

POWERED BY STEMpedia

Number Counter using Quarky

Let’s create a code to print 1-9 number on the LED of Quarky using the logic of while loop.



Let's Code

POWERED BY STEMpedia

- In order to start counting from 1 to 9 and display the same on Quarky's display, we will be writing the code using while loop as follows:

```
while i <= 9:
  quarky.showtext(str(i), [0, 225, 225])
  time.sleep(1)
  i = i + 1
```

  - **showtext()** function is used to display the required text on Quarky's display in the required color.

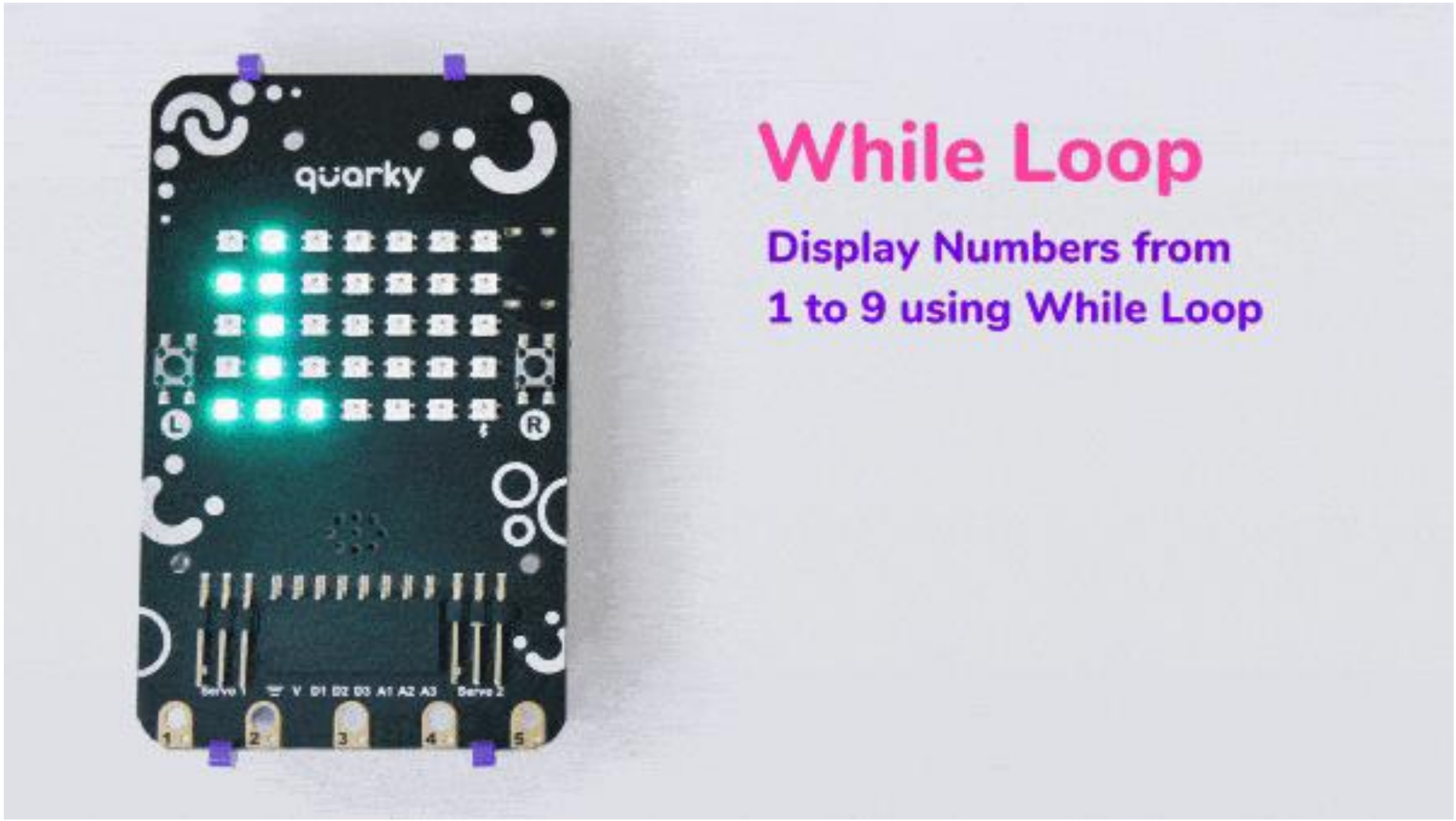
```
showtext([1],[2])
[1]:Char-TEXT="A" , [1]:((A-Z),(0-9))
[2]:Num_Array-COLOR=[R,G,B] ,[2]: (R-(0-255),G-(0-255),B-(0-255))
Number:(R,G,B)
```

(1-9) Number Counter

POWERED BY STEMpedia

```
sprite = Sprite('Tobi')
quarky = Quarky()
import time
quarky.setbrightness(15)
i = 1
# Count from 1 to 9
while i <= 9:
  quarky.showtext(str(i), [0, 225, 225])
  time.sleep(1)
  i = i + 1
quarky.cleardisplay()
```

Output

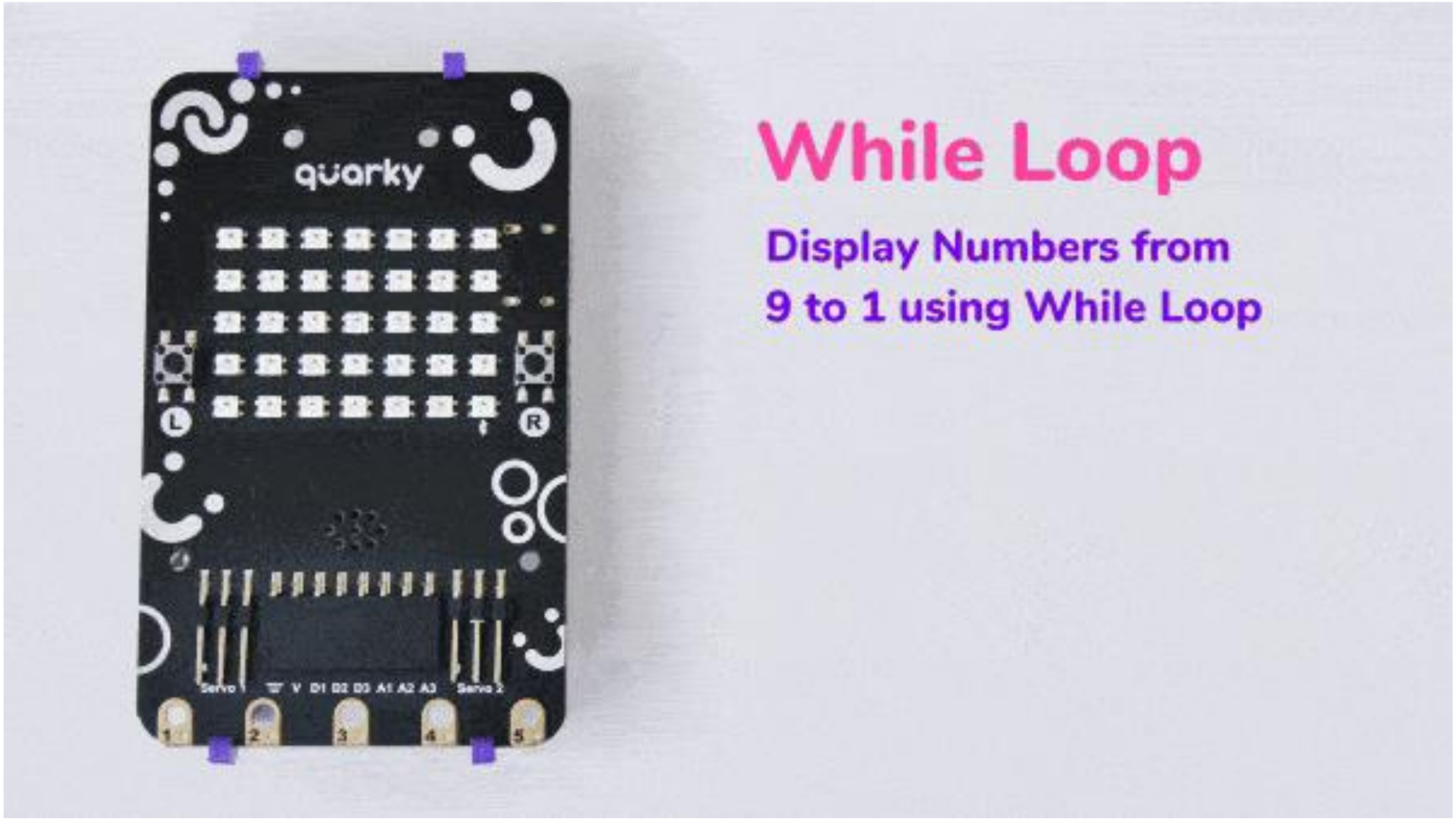


(9-1) Number Counter

POWERED BY STEMpedia

```
sprite = Sprite('Tobi’)
import time
# Instantiate the quarky object
quarky = Quarky()
i = 9
# Set the brightness level
quarky.setbrightness(15)
# Count from 9 to 1
while i >= 1:
        quarky.showtext(str(i), [0, 225, 225])
        time.sleep(1)
        i = i – 1
quarky.cleardisplay()
```

Output



Number counter using quarky(Final Code)

POWERED BY STEMpedia

```
sprite = Sprite('Tobi’)
import time
# Instantiate the quarky object
quarky = Quarky()
i = 1
# Set the brightness level
quarky.setbrightness(15)
# Count from 1 to 9
while i <= 9:
                quarky.showtext(str(i), [0, 225, 225])
                time.sleep(1)
                i = i + 1
quarky.cleardisplay()
time.sleep(4)
i = 9
# Set the brightness level
quarky.setbrightness(15)
# Count from 9 to 1
while i >= 1:
                quarky.showtext(str(i), [0, 225, 225])
                time.sleep(1)
                i = i - 1
quarky.cleardisplay()
```

AVERAGE AGE

POWERED BY STEMpedia

- First, we will create a step creates a variable called sum and sets it to 0. This variable will be used to keep track of the total age of all the students.

```
#Create a sum variable
sum=0
```

- Then the user to enter the number of students, reads the input as a string, and converts it to an integer using the **int()** function. The resulting integer is stored in the variable num.

```
#Take input from the user
num=int(input("Enter how many students: "))
```

- Then we will simply print out a message to the user to prompt them to enter the age of each student.

AVERAGE AGE

POWERED BY STEMpedia

- Furthermore, we use main loop of the program. It initializes a counter variable i to 0 and repeatedly loops through the following steps until i reaches the value of num: Reads the user input as a string using the **input()** function.

```
i=0     #create iteration variable
while i<num:
```

- Converts the input string to a number using the eval() function (which can handle both integers and floating-point numbers).adds the resulting number to the sum variable.
- Then, Increments the i-counter variable by 1.”we calculate the average age of the students by dividing the sum variable by num, and prints out the result along with a message to the user.

```
sum+=n
 i+=1
print("The average age is:", sum/num)
```

AVERAGE AGE(Final code)

POWERED BY STEMpedia

```
sum=0   #Create a sum variable
num=int(input("Enter how many students: ")) #Take input from the user
print("enter age:")
i=0      #create iteration variable
while i<num:
  n=eval(input())
  sum+=n
  i+=1
print("The average age is:", sum/num)
```

AVERAGE AGE(Final Output)

POWERED BY
STEMpedia

www.ai.thestempedia.com