

**Largest Number Game**

**Getting started**

A circuit board

Description automatically generated Before turning on the Raspberry Pi wire up  
the board using the details below.

|  |  |
| --- | --- |
| **LED&KEY** | **Raspberry Pi** |
| VCC | 3.3V (Pin 1) |
| GND | GND (Pin 6) |
| STB | GPIO 22 (Pin15) |
| CLK | GPIO 21 (Pin 40) |
| DIO | GPIO 17 (Pin11) |

 Time to connect the   
power to boot up the   
Raspberry Pi.

**Find the largest number**

 Let’s start coding:

This time we’re going to create a game.  
The program will display 8 digits on the display and you have to press the button below the largest one.  
By default the game is to find 5 digits.

 Run Python 3 and open a new file [File]-[New].  
Save the file and call it **highestnumber.py**

You could use any name but it makes it easier for the workshop if everyone is the same

Enter the code below.

|  |
| --- |
| #!/usr/bin/env python3  # The goal of the game is to press the button related to the largest number in the display  # TM1638 playground  # Import all the needed libraries  import TM1638 #board  import time # sleep  import random # create random number  import math # power  # These are the pins the display is connected to.  DIO = 17  CLK = 21  STB = 22  # This sets the number of puzzles to solve  gamelength=5  # Initialize the board  display = TM1638.TM1638(DIO, CLK, STB)  display.enable(1)  # Display how many to find  display.set\_text("find "+str(gamelength))  time.sleep(3)  # Get the starting time  starttime = time.time()  # Main looo for the lenght of the game  for x in range(gamelength):  # Get something setup for the start  ········largest = 0  ········number=list("00000000")  ········biggest=random.randint(6,7)  # Pick 8 random numbers on the display  ········for x in range(8):  ············digit = random.randint(0, biggest)  ············number[x] = str(digit)  # If the number is a largest number then record the location so it can be changed later  ············if digit > largest:  ····················largest = digit  # Add 1 to largest to make it the largest number (but only one bigger than the others  ········largest+=1  ········largest=str(largest)  # Pick a random number to change  ········location = random.randint(0,7)  # Replace the existing number at location with the new largest number  ········number[location] = largest  # Convert the list to a string using the join command  ········number="".join(number)  ········print(number)  # Display the 8 digits on the 7 segment  ········display.set\_text(number)  # Check if the correct key is pressed····  ········result = pow(2,(location))  ········correctkey = False  ········keys = display.get\_buttons()  ········while correctkey==False:  ············keys = display.get\_buttons()  ············if result==keys:  ················correctkey = True  ················print("keys: "+str(keys))  ················print("result: "+str(result))  # Get the total time  totaltime=time.time()-starttime  # By converting to an integer after multiplying by 100  # And then dividing by 100 it rounds to 2 decimal places.  timing = "t "+str(int(totaltime\*100)/100)  # Print the score  print(timing)  display.set\_text(timing) |

Run your program and find the largest digit each time.

After the game is finished the time take will be displayed on the 7 segment display.

**A few notes**

The display.get\_buttons() returns a binary number for all the 8 buttons

From left to right they are 1, 2, 4, 8, 16, 32, 64, 128

These are the decimal values for binary numbers.

Also, if you have learned about powers then these are all 2 to the power of 0,1,2,3,4,5,6,7

As in 2 to the power of 0 is 1

Or 2 to the power of 7 is 128

We know the location of our biggest digit (as we set it) so we know the result we want

**Challenge**

What do you have to do to make the game 10 digits long.