Sort Planets

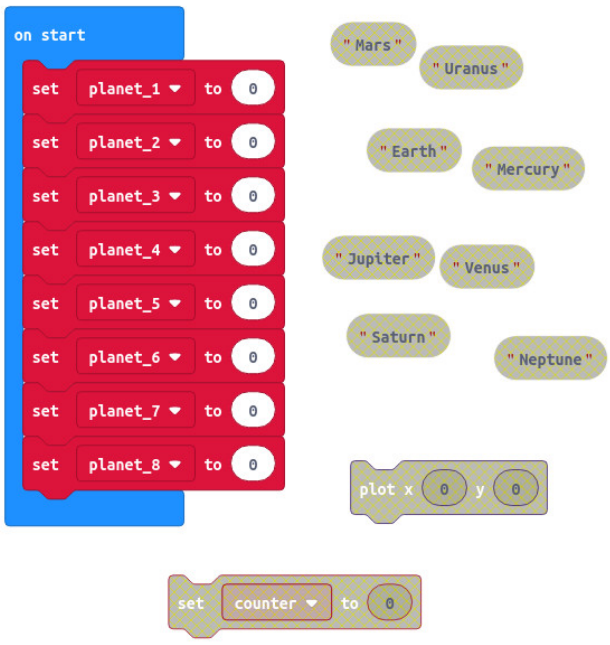
A micro:bit makecode project (workshop challenges)

CHALLENGE 1

THINGS TO MAKE YOUR PROGRAM DO AS SOON AS IT STARTS RUNNING (BEFORE YOU PRESS ANY BUTTONS OR ANYTHING ELSE)

Declaring Variables

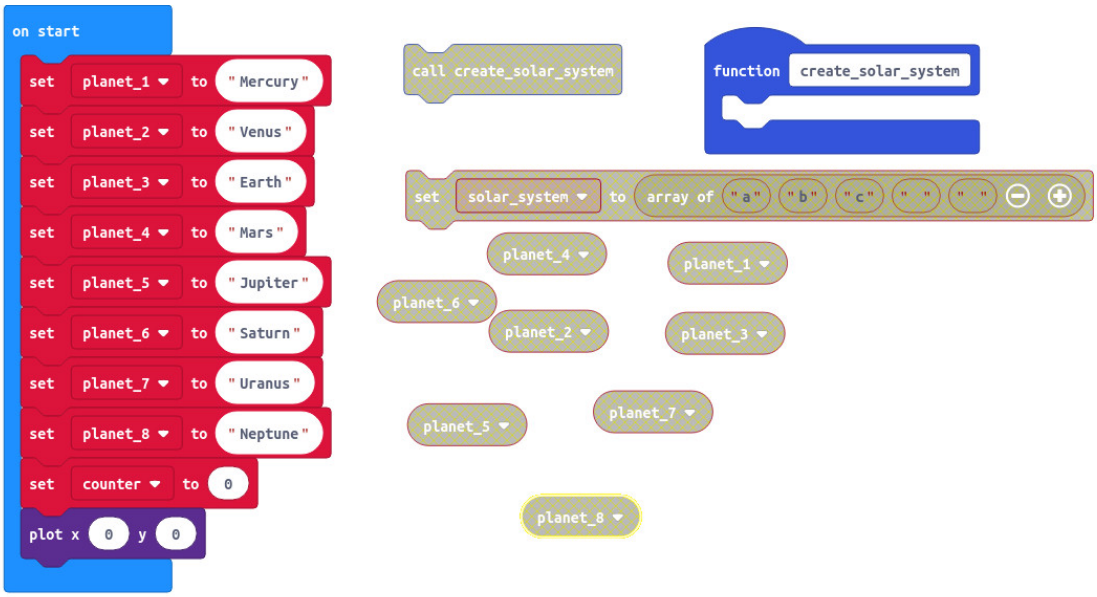
1. Make 8 variables named planet_1 to planet_8.
2. Assign a planet name (text string) to each of these 8 variables in the correct order, with planet_1 being the closest and planet_8 the furthest away from Sol (the name of our sun).
3. Add these blocks to an *on start* block.



CHALLENGE 1A

Make a function to create a solar system (called **create_solar_system**) that gets called when the program starts running.

Make an Array variable called **solar_system** and add the 8 planet variables (text strings remember!) to it. Make sure they're in the correct order.



Nothing to see so far, unless you made an LED come on as a 'power on indicator'.

Explain to someone how the program works so far.

What's the point of the plot 0,0 block?

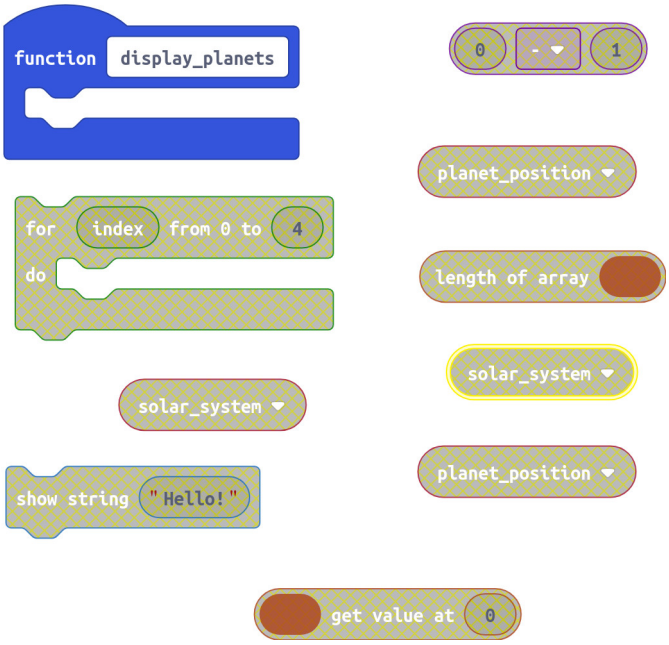
Question: I have a set of books labelled from 0 to 7. How many books are there?

Which variable type is used for planet variables?

CHALLENGE 2

THINGS TO MAKE YOUR PROGRAM DO WHEN YOU PRESS A BUTTON

1. Create a function (called **display_planets)** to loop through the array, displaying the value of a planet at each loop.
2. Make a for loop that will *do something* the same number of times as there are planets in the solar_system... BUT it has to start from zero!
3. The value of a variable called planet_position must change for each loop, going FROM (zero) TO (the length of the solar_system array minus 1). How many loops is that?
4. The *something to do bit* : Make a block to show a string at every loop. Make this string by getting the value at planet_position in the solar_system array.



5. Create the code to call the display_planets function whenever button A is pressed

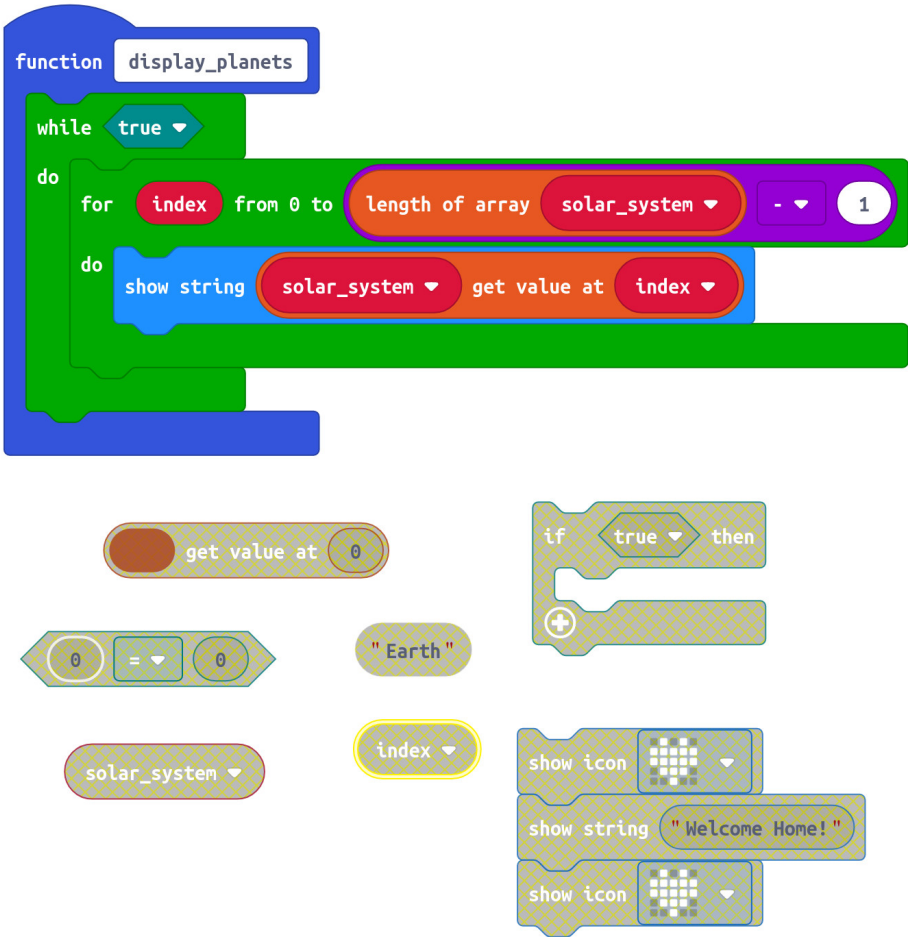
See if you can name the next planet before it scrolls onto the display!

CHALLENGE 3

Update your display_planets function to use a while loop to keep the display going forever in an ‘infinite loop’.

CHALLENGE 4

Update your display_planets function to show the text string “Home” whenever the text string used for planet3 (“Earth”) is displayed.



CHALLENGE 5

Update your makecode to display one planet at a time (still in order of distance from the sun), each time button A is pressed.

To do this you'll need to make a counter that goes up by one every time you press the button, but NEVER goes above the number of planets.

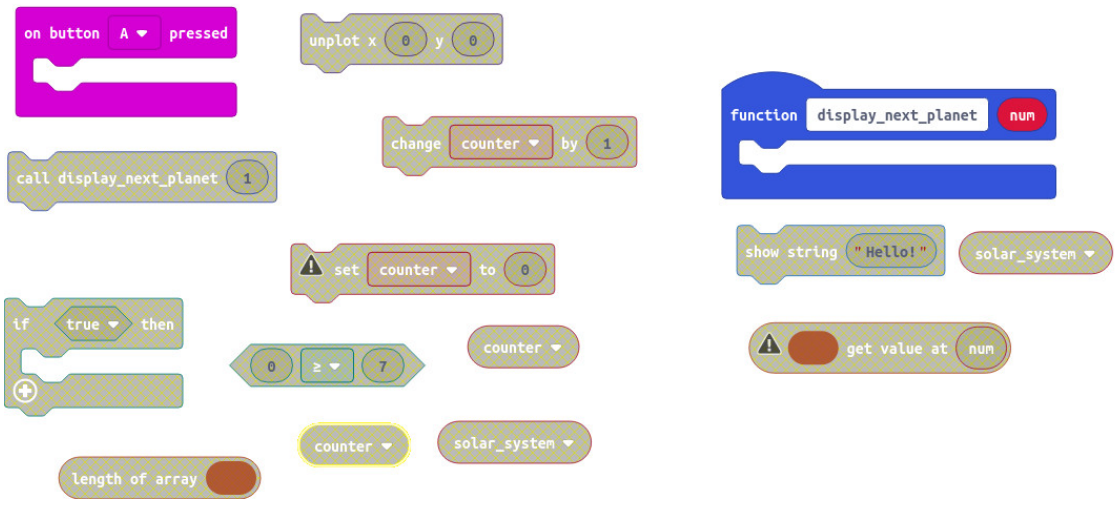
You'll also need to make a new function called **display_next_planet**.

Inside the button A block add blocks to do the following:

1. Switch off the micro:bit 'power on' indicator at 0,0 if you used one.
2. Call the display_next_planet function, passing it the number of the planet you want it to display.
3. Increment the value of the counter by 1.
4. Use an if block to decide whether the counter needs to be reset to zero.

Inside the display_next_planet function, tell it to do the following each time it's called:

1. Show the name of the planet that has the position of the number (num) it received.



What's wrong with not just saying 'if counter >= 8...' ?

CHALLENGE 6

To display a randomly chosen planet each time button A is pressed.

We need to be able to generate a randomly chosen number between 0 and 7.

CHALLENGE 7

Now completely separate from the random planet displaying you've just done, you need to program button B to increment a counter with values between 1 and 8 (or 0 to 7) . Just as before, when the counter tries to go above 8 (or 7), it gets reset to the beginning. This time though, you need to display the counter value. A new function called display_counter to

In this order, put these blocks into an on button B pressed block:

1. A block to call the display_counter function and pass it the counter variable at the same time.
2. create a block to increment the value of counter by 1.
3. create an if block that does something if the value of counter becomes equal or greater than the number of planets in the solar_system.
4. Create a single block to do that 'something'.