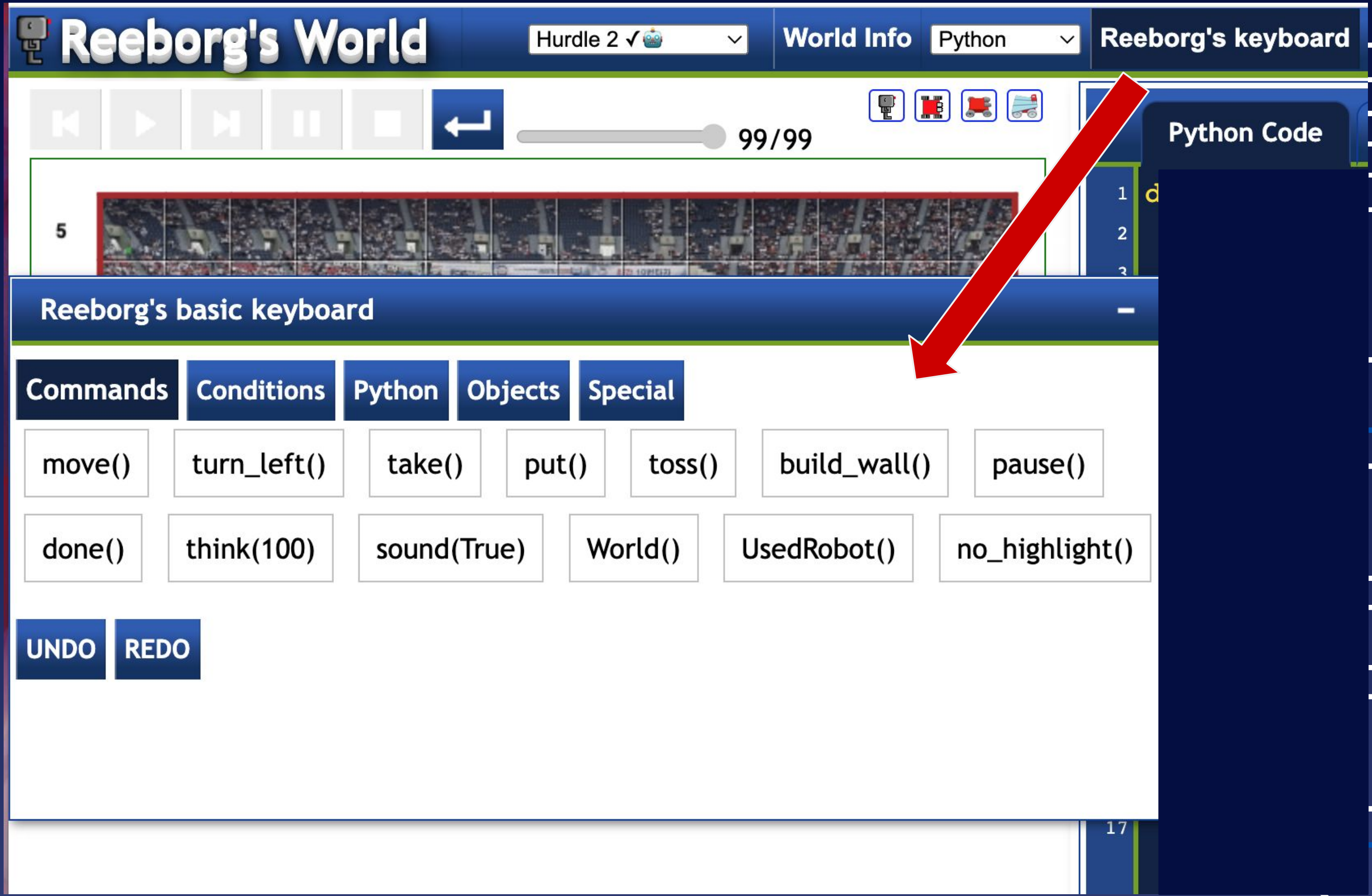
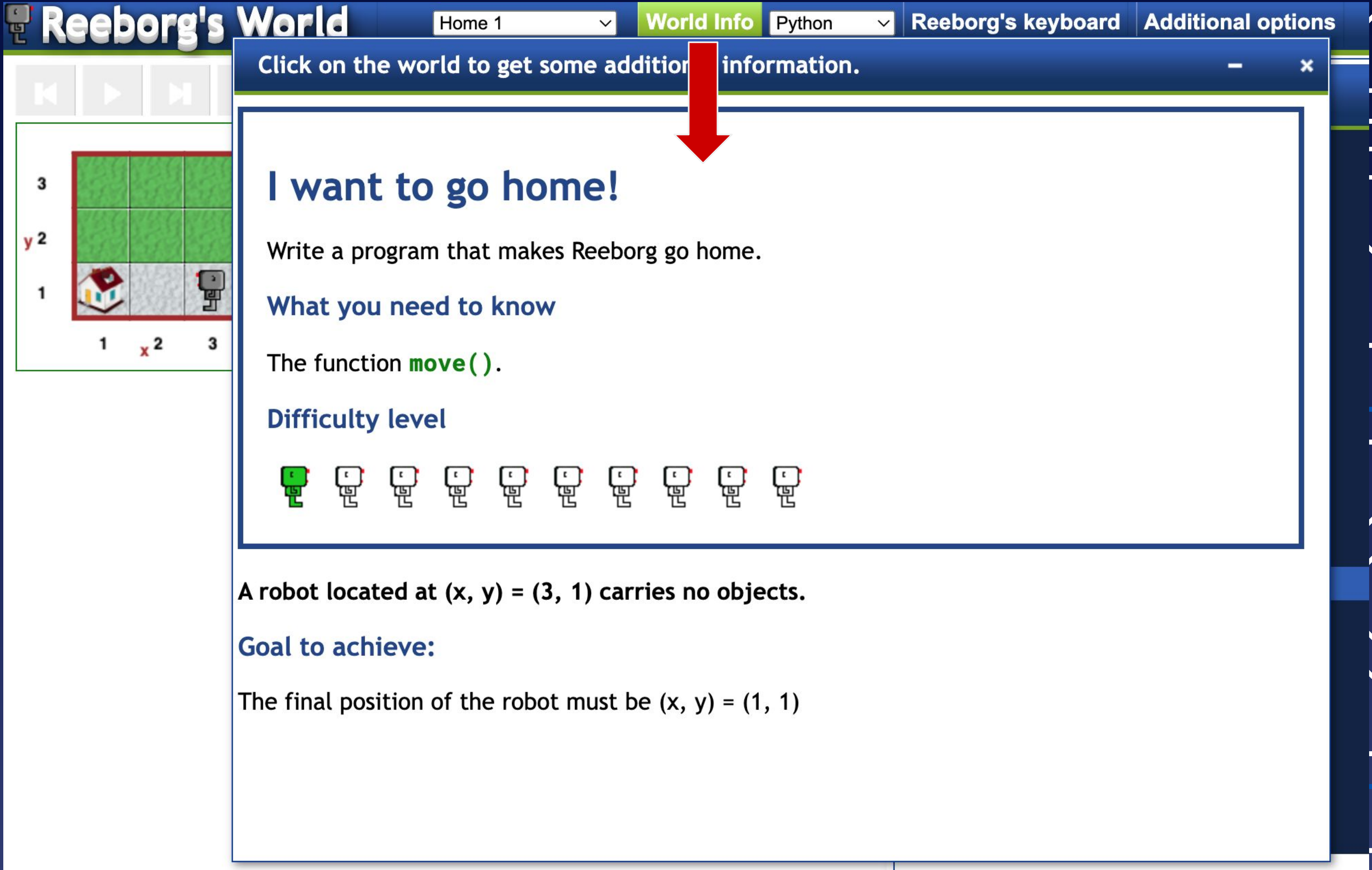
Reeborg 2

Intro to Python with Reeborg


Reeborg's World
Hurdle 2 ✓
World Info
Python
Reeborg's keyboard
99/99
5
Python Code
Reeborg's basic keyboard
Commands
Conditions
Python
Objects
Special
move()
turn_left()
take()
put()
toss()
build_wall()
pause()
done()
think(100)
sound(True)
World()
UsedRobot()
no_highlight()
UNDO
REDO



Click on the world to get some additional information.

I want to go home!

Write a program that makes Reeborg go home.

What you need to know

The function `move()`.

Difficulty level

A robot located at (x, y) = (3, 1) carries no objects.

Goal to achieve:

The final position of the robot must be (x, y) = (1, 1)

Challenge 1 - Around 1

Click HERE to go to Reeborg's World.

Write the following program, and replace the lines with # with the correct code to make it work:

```
put()
while front_is_clear():
    # add code here
    if wall_in_front():
        # add code here
    if object_here():
        # add code here
```

Challenge 1 - Around 1

OR simply:

1. Go to “Around 1”
2. Download the “Around 1 starting code.py” file from google classroom.
3. Load it with the button next to the save button.
4. Figure out how to solve the program by adding the correct code.

Challenge 1 - Around 1

Bonus challenge:

Can you define a function that turns corners called “corner( )” and a function that stops the program called “finished( )” ? Don’t forget to use it in your code.

Notes about Python

Spelling, punctuation, and indentation is super important in Python. It's part of what's called *syntax.*

The structure of a while loop, for loop, if statement or defining a function requires indentation (for example):

```
while object_here( ):
    take( )
    move( )
    put ( )
```

Note how everything indented belongs to the while loop. If you want to write new code that does NOT belong to the while loop, do NOT indent it.

When you return to the beginning of a line, you have left the part of the statement you are in. We do this with if/elif/else statements

```
if wall_in_front( ):
    turn_left( )
else:
    move( )
```

Note how both the if and the else are at the same indentation, and the code we want to run for each circumstance is indented.

If statements work with Booleans (True/False). It works like this:

```
   if condition == True:
       execute this code
OR elif this condition == True:
       execute this code instead
OR else:
       execute this code instead
```

A standard if/elif/else statement represents a series of ORs.

When the first condition is True:

```
        if condition == True:
            execute this code and ignore the rest
IGNORE  elif condition:
            code
IGNORE  else:
            code
```

When a condition is True,
we ignore the rest.

When the first condition is False:

```
IGNORE  if condition == False:
            ignore this code and check the elif
        elif this condition == True:
            execute this code instead
IGNORE  else:
            code
```

It will check each condition until it finds one to be True, OR it will execute the else.

```
       if condition == False:
IGNORE     ignore this code and check the elif
       elif this condition == False:
IGNORE     ignore this code and check the else
       else:
           execute this code if all else is False
```

We do not have to include an else:

```
    if condition == True:
        execute this code
OR  elif this condition == True:
        execute this code instead
OR  elif this condition == True:
        execute this code instead
```

You can use as many elifs as you want. They mean “else if” and will only be checked when the if and other elifs before it are False.

We do not have to include an else:

```
if condition == True:
    execute this code
elif this condition == True:
    execute this code instead
elif this condition == True:
    execute this code instead
```

OR

OR

In this case, since there is no else, if all conditions evaluate as False, then nothing happens.

We do not have to include an else:

```
        if condition == False:
IGNORE      code
        elif this condition == False:
IGNORE      code
        elif this condition == False:
IGNORE      code
        else:
            execute this code instead
```

When we add an else, it means: when the if and elifs all evaluate as False, execute this code.

When we write multiple if statements, they are read as separate statements:

```
if condition == True:
    execute this code
if this condition == True:
    execute this code as well
```

AND/OR

If the first is False, the second will still be checked and run if True.

When we NEST if statements, the first must be True for the second to be checked:

```
if condition == True:
    execute this code
    if this condition == True:        AND
        execute this code as well
```

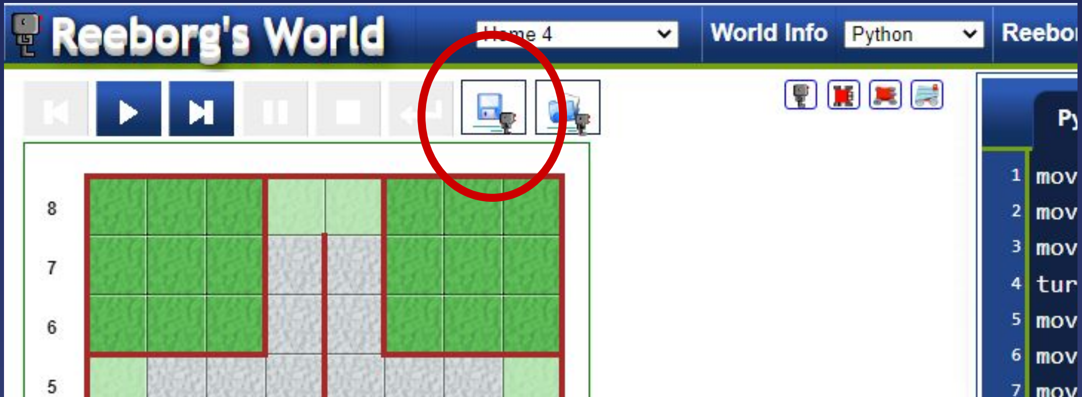
Notice that we have to indent again

This is an AND logic gate

Save your work

WHEN DONE, CLICK THE **SAVE** BUTTON, **SAVE TO DESKTOP**

You will submit 2 different files to GC at the end of class.



```
mov
mov
mov
tur
mov
mov
mov
```

Challenge 2 - Around 2

Click HERE to go to Reeborg's World.

See if you can make the code work. Note: you will need to use the wall_on_right( ) function. Remember, you can make a function to turn right by adding the correct code below the following:

```
def turn_right():
```

Challenge 2 - Around 2

IF STUCK:

1. Go to “Around 2”
2. Download the “Around 2 starting code.py” file from google classroom.
3. Load it with the button next to the save button.
4. Figure out how to solve the program by adding the correct code.

Challenge 2 - Around 2

Bonus challenge:

Can you define useful functions for this program? (remember to use names that aren't already used by other functions)