



python in pieces

A 2Simple Secondary Product

LESSON SOLUTIONS

Level 2

Contents

Lesson 1- while Loops.....	3
Stage 1- Set the Scene	3
Stage 2- Create variables.....	4
Stage 3- Importing the sleep command.....	5
Stage 4- Add a while loop	5
Stage 5- PMake the rocket take off.....	6
Stage 6- Debug	6
Challenge	7

Lesson 2- Numbers and Calculations.....	8
Stage 1- Set the Scene	8
Stage 2- Instructions	8
Stage 3- Variables to store the lengths	9
Stage 4- Convert the variables to integers...9	
Stage 5- Is it an equilateral triangle?	10
Stage 6- Is it an isosceles triangle?.....	10
Stage 7- It must be a scalene triangle.....	11
Stage 8- Refine the code to anticipate user errors.....	12
Stage 9- Debug	13
Stage 10- Monty munches	12
Challenge	13

Lesson 3- Loops in Action	15
Stage 1- A basic loop	15
Stage 2- Start and stop	15
Stage 3- Step size	16
Stage 4- A range inside a range	16
Stage 5- Square numbers - debug.....	17
Stage 6- Times tables	17
Stage 7- Interacting with the user.....	18
Stage 8- Formatting the times table.....	18
Challenge	19

Lesson 4- Loops and Selection.....	20
Stage 1- 'while' loops and 'break' statements	20
Stage 2- Importing libraries.....	20
Stage 3- Beginning a magic-8-ball game.....	21
Stage 4- Choose a random answer	21
Stage 5- Add elif/else statements	22
Stage 6- Debug	23
Challenge	24

Lesson 5- Functions	25
Stage 1- Customize the scene.....	25
Stage 2- Create variables and import modules.....	26
Stage 3- Ask the user to guess	26
Stage 4- Check if they got it right	27
Stage 5- Expand the 'if' statement	27
Stage 6- Finish the 'if' statement.....	28
Tasks	28
Solution	28
Stage 4- Check if they got it right	29
Stage 5- Expand the 'if' statement	30
Challenge	31

Lesson6- Using sound in a game.....	31
Stage 1- Set the scene.....	32
Stage 2- Add a character	33
Stage 3- Initialize the game.....	33
Stage 4- Add a loop.....	34
Stage 5- Handle the user choice 1.....	35
Stage 6- Handle the user choice 2.....	36
Stage 7- Add visual effects	37
Stage 8- Debug	39
Challenge	40

Lesson 1 – While Loops

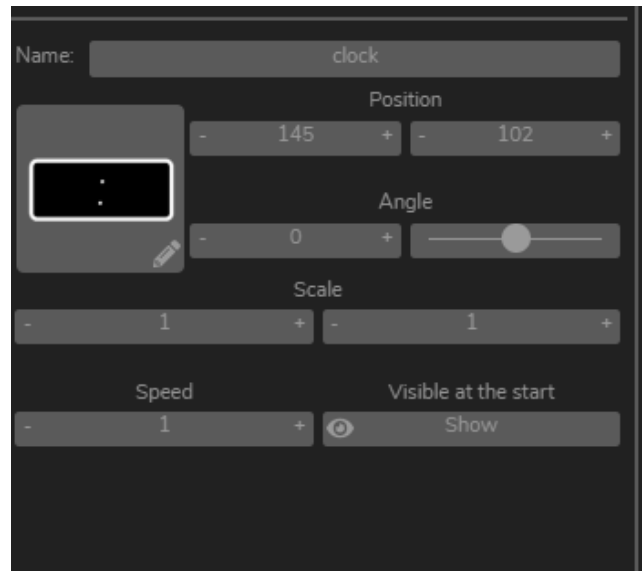
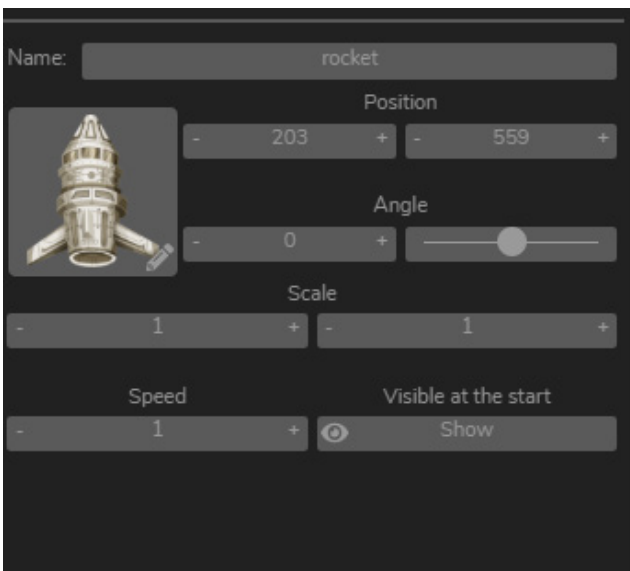
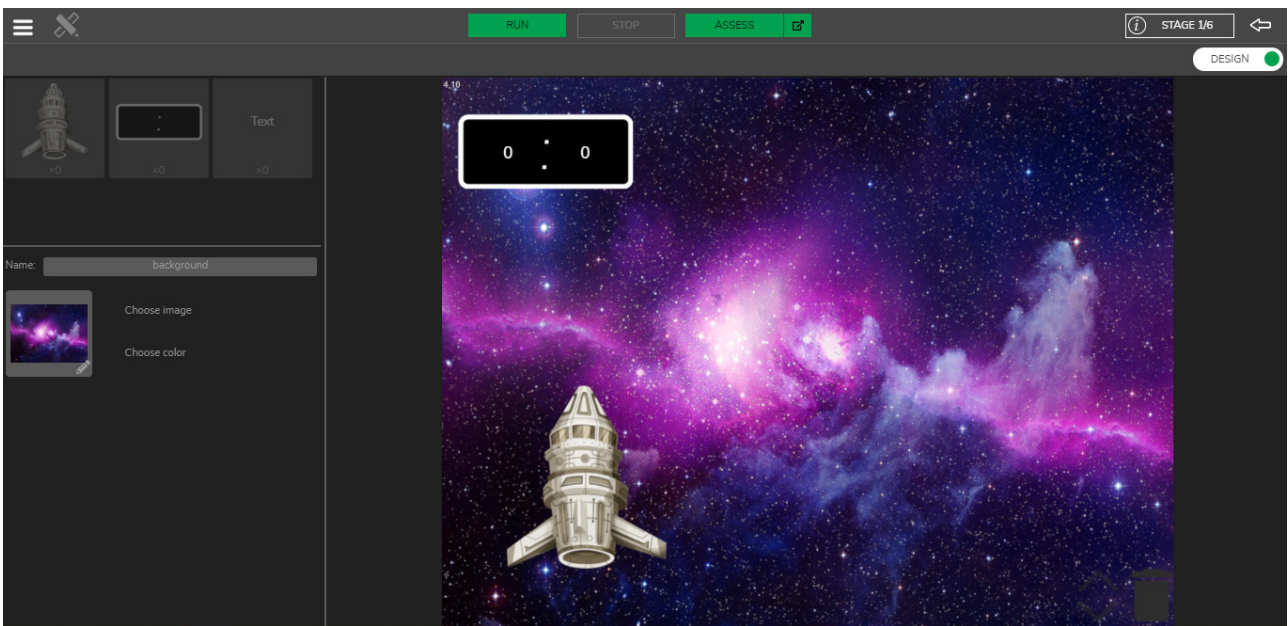
Stage 1- Set the Scene

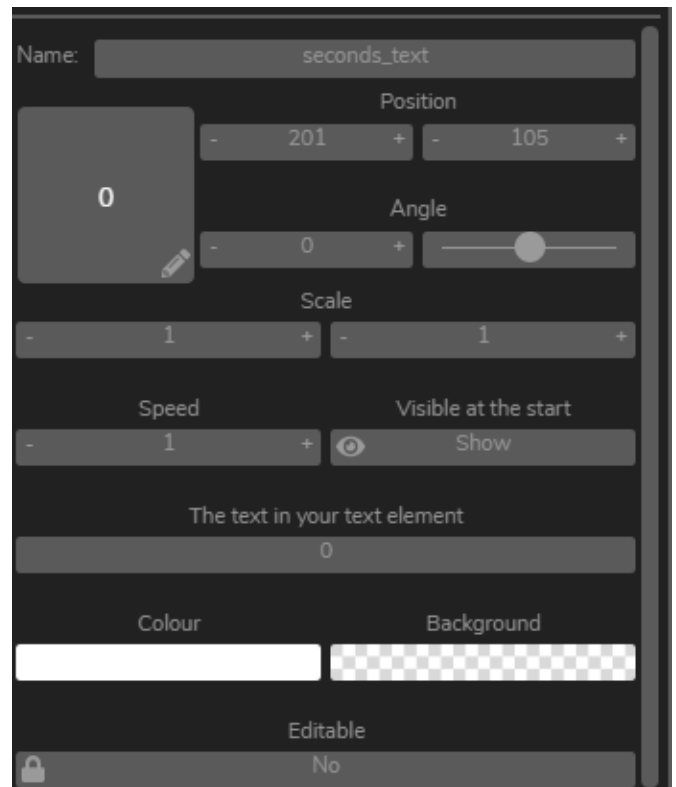
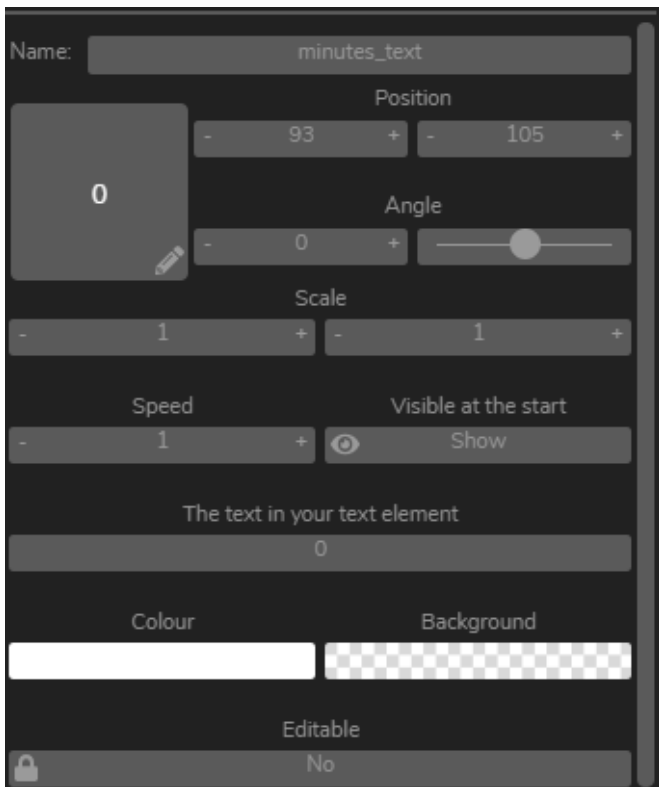
Set up a space-themed scene with a clock for the launch countdown.

Tasks

- Choose a background image.
- Add a rocket sprite and name it 'rocket'.
- Add a digital clock sprite and name it 'clock'.
- Add two text objects, called 'minutes_text' and 'seconds_text'.
- Set their text to show '0' and place these in the appropriate sections of the clock – 'minutes_text' on the left and 'seconds_text' on the right.

Solution





Stage 2- Create variables

Create variables to store the time.

Tasks

- Create a variable called 'minutes' and initialise it to 0.
- Create a variable called 'seconds' and initialise it to 9.
- Set the text property of the element minutes_text to the value of minutes.
- Set the text property of the element seconds_text to the value of seconds.
- Run your program and check that the clock displays '0:9.'

Solution

```

set minutes to 0
set seconds to 9
minutes_text text set to minutes
seconds_text text set to seconds

```

```

1 minutes = 0
2 seconds = 9
3 minutes_text.text = minutes
4 seconds_text.text = seconds

```



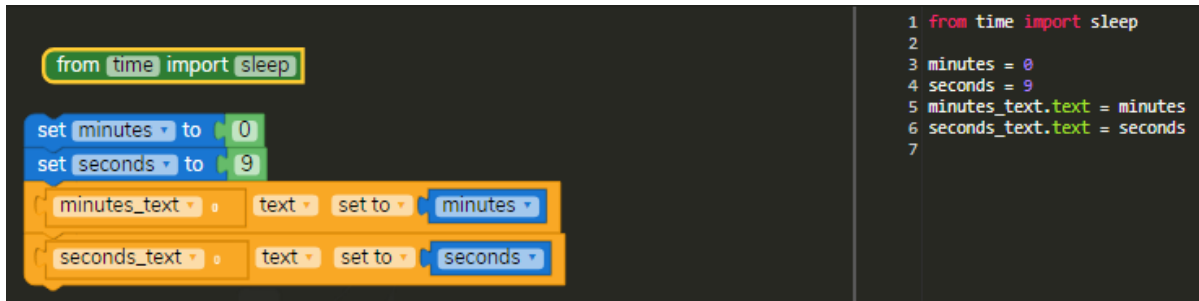
Stage 3- Importing the sleep command

To use timings in Python, we need to import the 'sleep' procedure from the 'time' module.

Tasks

- Import the 'sleep' procedure from the 'time' module. This should go at the top of your program.

Solution



The image shows the solution for Stage 3. On the left, Scratch code blocks are shown: a 'from time import sleep' block, followed by 'set minutes to 0' and 'set seconds to 9'. Below these are two 'text set to' blocks: 'minutes_text' set to 'minutes' and 'seconds_text' set to 'seconds'. On the right, the corresponding Python code is displayed:

```

1 from time import sleep
2
3 minutes = 0
4 seconds = 9
5 minutes_text.text = minutes
6 seconds_text.text = seconds
7

```



Stage 4- Add a while loop

A while loop runs over and over until a condition is met. We want the loop to run to do a countdown until 'seconds' is zero.

We'll also use a 'sleep' command to make the loop pause so that you can see the countdown happening

Tasks

- Add a while loop that runs while 'seconds' is greater than 0.
- Inside the loop, first use a 'sleep' command to make the code pause for a second each time.
- Inside the loop, reduce the value of the variable 'seconds' by one each time.
- Inside the loop, display the new value of 'seconds' in 'seconds_text'.
- Run your code and check that the clock values count down from 0:9 to 0:0.

Solution



The image shows the solution for Stage 4. On the left, Scratch code blocks are shown: 'from time import sleep', 'set minutes to 0', 'set seconds to 9', 'minutes_text' set to 'minutes', and 'seconds_text' set to 'seconds'. A 'repeat while' loop is added with the condition 'seconds > 0'. Inside the loop, there are three blocks: 'sleep for seconds: 1', 'set seconds to seconds - 1', and 'seconds_text' set to 'seconds'. On the right, the corresponding Python code is displayed:

```

1 from time import sleep
2 minutes=0
3 seconds=9
4 minutes_text.text=minutes
5 seconds_text.text=seconds
6 while seconds > 0:
7     sleep(1)
8     seconds=seconds-1
9     seconds_text.text=seconds
10

```



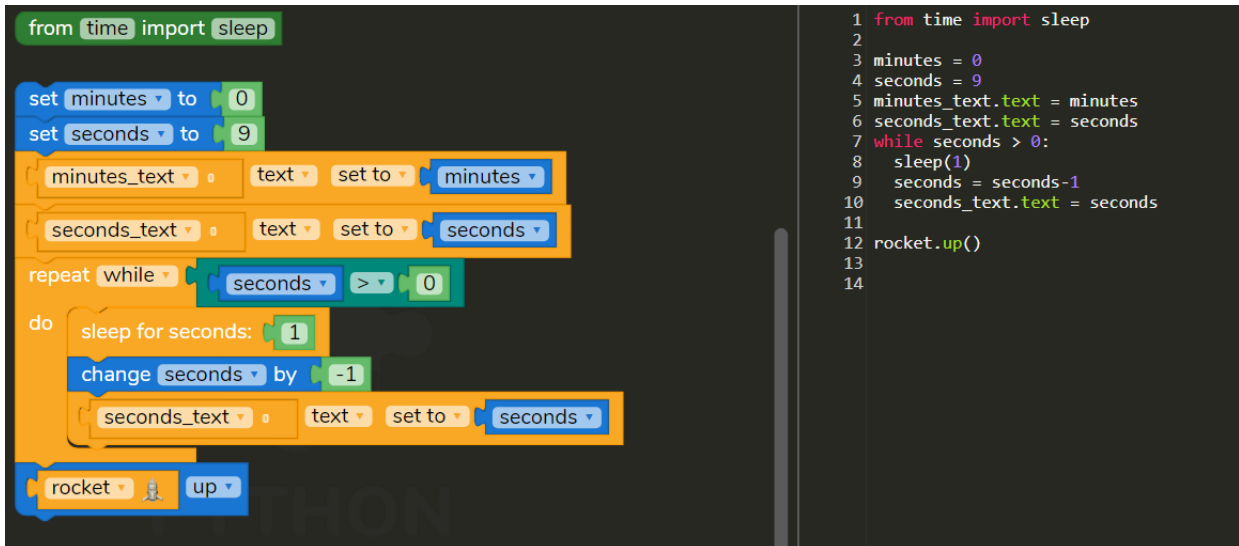
Stage 5- Make the rocket take off

Set the rocket to take off when the countdown reaches 0:0

Tasks

- When the loop has finished make the rocket move up using the up() method.
- Test your program - does the counter count down to zero and then the rocket move?

Solution



The image shows the solution for Stage 5. On the left, Scratch code blocks are arranged as follows: a 'from time import sleep' block, 'set minutes to 0' and 'set seconds to 9' blocks, two 'text set to' blocks for 'minutes_text' and 'seconds_text', a 'repeat while' loop with condition 'seconds > 0', a 'do' loop containing 'sleep for seconds: 1', 'change seconds by -1', and 'seconds_text text set to seconds', and finally a 'rocket up' block. On the right, the corresponding Python code is shown:

```

1 from time import sleep
2
3 minutes = 0
4 seconds = 9
5 minutes_text.text = minutes
6 seconds_text.text = seconds
7 while seconds > 0:
8     sleep(1)
9     seconds = seconds-1
10    seconds_text.text = seconds
11
12 rocket.up()
13
14

```



Stage 6- Debug

In this program, the countdown is not running correctly. The rocket moves up immediately.

Tasks

- Debug the code to make the countdown run properly. It should count down in seconds from 9 to 0.

Solution



The image shows the solution for Stage 6. On the left, Scratch code blocks are arranged as follows: a 'from time import sleep' block, 'set minutes to 0' and 'set seconds to 9' blocks, two 'text set to' blocks for 'seconds_text' and 'minutes_text', a 'repeat while' loop with condition 'seconds > 0', a 'do' loop containing 'sleep for seconds: 1', 'change seconds by -1', and 'seconds_text text set to seconds', and finally a 'rocket up' block. On the right, the corresponding Python code is shown:

```

1 from time import sleep
2 minutes = 0
3 seconds = 9
4 seconds_text.text = seconds
5 minutes_text.text = minutes
6 while seconds > 0:
7     sleep(1)
8     seconds = seconds - 1
9     seconds_text.text = seconds
10 rocket.up()

```

Try out the following challenges.

Suggestions

- Add an explosion image and make it invisible at the start.
- Make the rocket explode one second after take-off.
- And a sound to play along with the explosion.
- Make the clock start counting down when you click the 'Launch' button.

Possible Solution



```

1 from time import sleep
2
3 #Function to launch the rocket
4 def launch():
5     #start at 10 seconds
6     minutes = 1
7     seconds = 0
8     minutes_text.text = minutes
9     seconds_text.text = seconds
10    sleep(1)
11    #Change to 9 seconds
12    minutes = 0
13    seconds = 9
14    minutes_text.text = minutes
15    seconds_text.text = seconds
16    #Countdown to 0
17    while seconds > 0:
18        sleep(1)
19        seconds = seconds-1
20        seconds_text.text = seconds
21    #Launch and fly for 2 seconds then explode
22    rocket.up()
23    sleep(1.5)
24    pip.play_sound("2Simple Classic/bang.mp3",1)
25    rocket.stop()
26    explosion.show()
27
28    seconds = 5
29    #Make the explosion grow for 5 seconds to hide the rocket
30    while seconds > 0:
31        sleep(1)
32        pip.play_sound("2Simple Classic/bang.mp3",1)
33        explosion.scaleX = explosion.scaleX + 0.3
34        explosion.scaleY = explosion.scaleY + 0.3
35        seconds = seconds - 1
36    #Move explosion up again with rocket hiding behind it.
37    explosion.up()
38    rocket.up()
39    sleep(1)
40    #Reset the elements to re-launch
41    rocket.hide()
42    explosion.hide()
43    rocket.stop()
44    explosion.stop()
45    rocket.x = 500
46    rocket.y = 613
47    explosion.x = 500
48    explosion.y = 291
49    explosion.scaleX = 2
50    explosion.scaleY = 2
51    #Show new rocket after 1 second
52    sleep(1)
53    rocket.show()
54
55 #Call function when the launch button is pressed
56 pip.eventmanager.when_click(btnLaunch,launch)
57

```

Lesson 2 – and/or

Stage 1- Set the Scene

First let's set up the design view.

Tasks

- Choose a background
- Add a sprite of your choice to be the main talking character. Name the sprite 'character' and position it near the bottom of the screen.
- Add three triangle sprites of different types (equilateral, isosceles and scalene).
- Name your triangles 'tri_equ', 'tri_iso' and 'tri_sca'.



Stage 2- Instructions

Let's make the character say the instructions

Tasks

- Make your character show a speech bubble for 5 seconds, which says 'Tell me the three side lengths of your triangle'. Use the 'say' function to do this

Solution

```
character say: "Tell me the three side lengths of your triangle." for 5 secs  
1 character.say("Tell me the three side lengths of your triangle.",5)  
2
```




Stage 3- Variables to store the lengths

We need to ask the user to enter the lengths of the sides of their triangle. We will store these in variables to be used later.

Tasks

- Use three 'input()' commands to ask the user to input the lengths of each side (x, y and z). The first prompt should say 'Enter x: ', and so on.
- Store the values in variables called 'x', 'y' and 'z'.
- Run your program to test it. Use x = 10, y = 10, z = 10. The debug panel (bottom-right) should show that you have three variables set to "10".

Solution

```

1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')

```



Stage 4- Convert the variables to integers

In Python, the string "10" is different to the number 10.

First, we need to convert each of the variables to integers before we can use them as lengths in our code.

Tasks

- Use the 'int()' function to convert each of your variables to integers - for example 'x = int(x)'
- Run your program to test it. Use x = 10, y = 10, z = 10. Check that the debug panel (at the bottom right) shows that your variables are integers now.

Solution

```

1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3
4 y = input('Enter y:')
5
6 z = input('Enter z:')
7
8 x = int(x)
9 y = int(y)
10 z = int(z)

```



Stage 5- Is it an equilateral triangle?

We can use 'if' and 'and' to check if the user has described an equilateral triangle (one with all three sides equal)

We'll check if 'x' is equal to 'y' and 'x' is also equal to 'z'.

Tasks

- Use one 'if' statement that tests whether 'x' is equal to 'y' and 'x' is also equal to 'z'
- If this is true your character should say 'Equilateral triangle' for 5 seconds.
- Run your program to test it. Use x = 10, y = 10 and z = 10. Check that your character says 'Equilateral triangle'.

Solution

```

1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')
5 x = int(x)
6 y = int(y)
7 z = int(z)
8 if x == y and x == z:
9   character.say('Equilateral triangle.',5)
10

```



Stage 6- Is it an isosceles triangle?

We can use further statements to check if the triangle is isosceles.

An isosceles triangle has two sides that are equal. It could be x and y, or y and z, or z and x

Tasks

- Add one 'elif' statement which checks if x and y are equal, or if y and z are equal, or if z and x are equal.
- If this is true the character should say 'Isosceles triangle' for 5 seconds.
- Run your program to test it. Use x = 10, y = 10 and z = 15. Check that your character says 'Isosceles triangle'.

Solution

```

1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')
5 x = int(x)
6 y = int(y)
7 z = int(z)
8 if x == y and x == z:
9   character.say('Equilateral triangle.',5)
10 elif x==y or y==z or z==x:
11   character.say('Isosceles triangle.',5)

```



If it's not equilateral or isosceles it must be a scalene triangle.

Tasks

- Add one 'else' statement at the end of your program. Inside, the sprite should say 'Scalene triangle' for 5 seconds. Your program should now contain one 'if' statement, one 'elif' statement and one 'else' statement."
- Run your program and enter $x = 10$, $y = 12$ and $z = 15$. Check that your character says 'Scalene triangle'.

Solution



The image shows a Scratch script on the left and its corresponding Python code on the right. The Scratch script starts with a character saying "Tell me the three side lengths of your triangle." for 5 seconds. It then asks for three side lengths (x, y, z) and converts them to integers. An if-elif-else structure checks for equilateral (x==y and y==z), isosceles (x==y or y==z or z==x), and scalene (otherwise) triangles, each saying the result for 5 seconds.

```
1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')
5 x = int(x)
6 y = int(y)
7 z = int(z)
8 if x == y and x == z:
9     character.say('Equilateral triangle.',5)
10 elif x==y or y==z or z==x:
11     character.say('Isosceles triangle.',5)
12 else:
13     character.say('Scalene triangle.',5)
```



Run the code and enter some letters instead of a number. You will get an error message and the program will fail.

We need to check if the user has entered only digits (0-9) using the 'isdigit' function.

Tasks

- Before converting each variable to an integer, write an 'if' statement and use the 'x.isdigit()' function to check that 'x' and 'y' and 'z' contain only digits.
- Move the rest of your code that identifies the triangle inside the 'if' statement. Use an 'else' statement to make the character say 'Please enter only numbers' if this check fails.

Solution

```

1 character.say('Tell me the three side lengths of your triangle.',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')
5 if x.isdigit() and y.isdigit() and z.isdigit():
6     x = int(x)
7     y = int(y)
8     z = int(z)
9     if x == y and x == z:
10        character.say('Equilateral triangle.',5)
11    elif x == y or y == z:
12        character.say('Isosceles triangle.',5)
13    else:
14        character.say('Scalene triangle.',5)
15 else:
16    character.say('Please enter only numbers.',5)
17

```



There is a logical error in this code - for some triangles it gives the wrong answer. What is wrong with it?

Tasks

- Test this code using measurements for the three types of triangles. What is wrong? Fix the error!
- Check that your code works. Use $x = 10, y = 20, z = 10$. Check that your character says, 'Isosceles triangle'.

Solution

The image shows a Scratch script on the left and its corresponding Python code on the right. The Scratch code asks for three side lengths (x, y, z) and then uses an if-else structure to identify the triangle type based on equality conditions. The Python code replicates this logic.

```

1 character.say('Input the lengths of the triangle sides: ',5)
2 x = int(input('x: '))
3 y = int(input('y: '))
4 z = int(input('z: '))
5 if x == y and x == z:
6     character.say('Equilateral triangle',5)
7 elif x == y or y == z or x == z:
8     character.say('Isosceles triangle',5)
9 else:
10    character.say('Scalene triangle',5)
  
```



Challenge

Add in code to show the correct triangle and hide the others.

Suggestions

- Set all the triangle images to hide at the start of the program.
- Add code to show the relevant type of triangle when it has been identified.
- Some input is invalid - for example $x = 1, y = 1, z = 100$. There is simply no way to draw a triangle like that. Include a way of checking this in your program.

The Scratch code blocks are as follows:

- Character says: "Tell me the three side lengths of your triangle" for 5 secs
- Set X to Input with message "Enter x:"
- Set Y to Input with message "Enter y:"
- Set Z to Input with message "Enter z:"
- If is digit? X and is digit? Y
 - Do
 - Set X to int X
 - Set Y to int Y
 - Set Z to int Z
 - If $x + y > z$ and $x + z > y$
 - Do
 - If $x == y$ and $x == z$
 - Character says: "Equilateral triangle" for 5 secs
 - tri_equ.show()
 - Else if $x == y$ or $y == z$
 - Character says: "Isosceles triangle" for 5 secs
 - tri_iso.show()
 - Else
 - Character says: "Scalene triangle" for 5 secs
 - tri_sca.show()
 - Else
 - Character says: "Impossible triangle." for 5 secs
 - Else
 - Character says: "Please enter only numbers." for 5 secs

```

1 character.say('Tell me the three side lengths of your triangle',5)
2 x = input('Enter x:')
3 y = input('Enter y:')
4 z = input('Enter z:')
5 if (x.isdigit()) and (y.isdigit()) and (z.isdigit()):
6     x = int(x)
7     y = int(y)
8     z = int(z)
9     if x+y>z and x+z>y and y+z>x:
10
11         if x == y and x == z:
12             character.say('Equilateral triangle',5)
13             tri_equ.show()
14         elif x == y or y == z or x==z:
15             character.say('Isosceles triangle',5)
16             tri_iso.show()
17         else:
18             character.say('Scalene triangle',5)
19             tri_sca.show()
20     else:
21         character.say('Impossible triangle.',5)
22 else:
23     character.say('Please enter only numbers.',5)
24

```

Lesson 3- Loops in Action



Stage 1 – A basic loop

The 'range' keyword returns a special Python object that represents a sequence of numbers. 'range(N)' gives you a sequence of 'N' numbers starting from zero:

The number 'N' is not included

Tasks

- Create a 'for i in range(N)' loop for which 'i' takes the values 0, 1, 2, 3..., 100. What is the correct value of 'N'? Print the value of 'i' each time.

Solution



```
1 for i in range(101):
2     print(i)
3
```



Stage 2- Start and stop

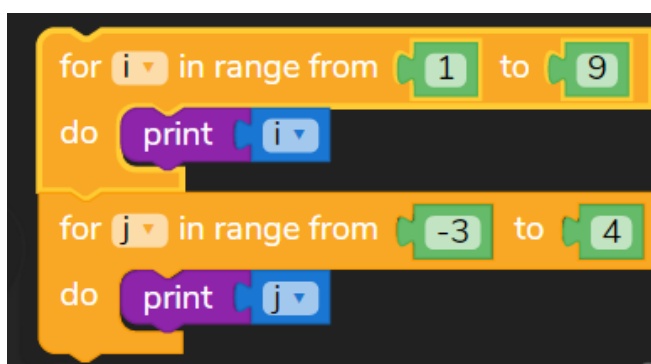
You don't have to start at zero, you can specify the 'START' and 'STOP' values for a range. Range(START, STOP) will start from 'START', and go up to 'STOP'.

The number STOP is not included - it stops before the number STOP is reached.

Tasks

- Create a 'for i in range(START, STOP)' loop for which 'i' takes the values 1 up to 8. What are the correct values for 'START' and 'STOP'? Print the value of 'i' each time.
- Create a 'for j in range(START, STOP)' loop for which 'j' takes the values -3 up to 3. Print the value of 'j'.

Solution



```
1 for i in range(1, 9):
2     print(i)
3 for j in range(-3, 4):
4     print(j)
5
```



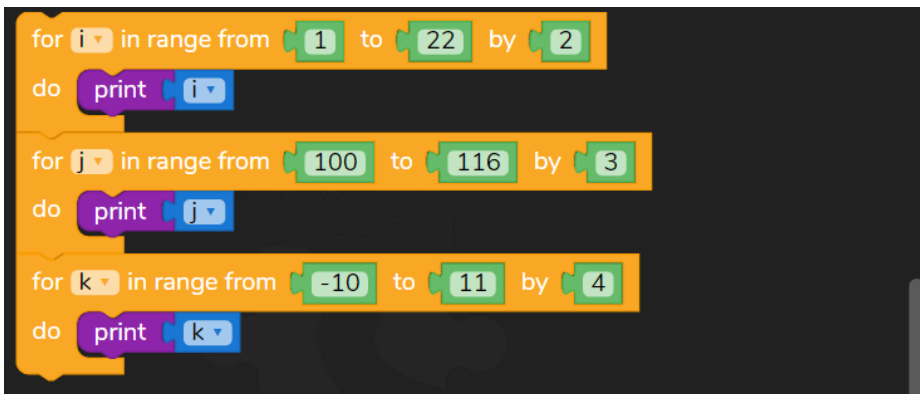
Stage 3- Step size

You don't have to go in steps of 1 each time, you can step by any amount.

Tasks

- Write a 'for i in range' loop that prints the numbers: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
- Write a 'for j in range' loop that prints the numbers: 100, 103, 106, 109, 112, 115.
- Write a 'for k in range' loop that prints the numbers: -10, -6, -2, 2, 6, 10.

Solution



```

1 for i in range(1, 22,2):
2   print(i)
3 for j in range(100, 116,3):
4   print(j)
5 for k in range(-10, 11,4):
6   print(k)

```



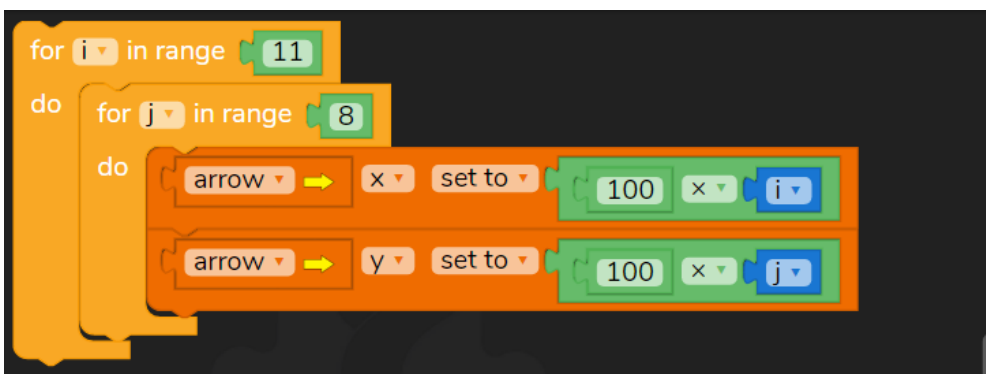
Stage 4- – A range inside a range

We don't have to just print things inside a range, we can use other code too. We can even place a range inside another range.

Tasks

- Create a 'for i in range' loop where 'i' takes the values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- Inside the loop, create a 'for j in range' loop where 'j' takes the values 0, 1, 2, 3, 4, 5, 6, 7.
- In design mode there is a sprite called 'arrow'. Inside your loop, set the 'x' position of 'arrow' to '100 × i' and set its 'y' position to '100 × j'.
- Run your code. Check that the arrow moves across and down the screen.

Solution



```

1 for i in range (11):
2   for j in range(8):
3     arrow.x = 100 * i
4     arrow.y = 100 * j

```




Stage 5- Square numbers - debug

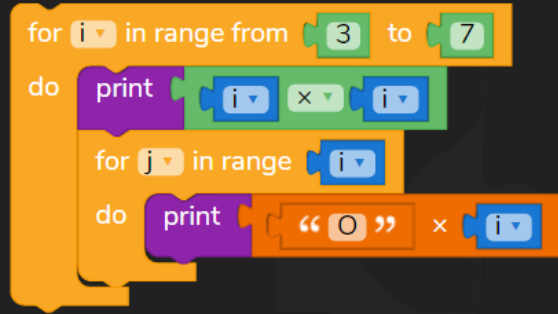
Lesson 3- Loops in Action

Someone has written some code to print the square numbers from 3 squared (9) up to 6 squared (36), and to draw pictures of each one. It's not working at the moment. See if you can fix it.

Tasks

- Fix the code. When you've finished it should print the square numbers from 3 squared (9) up to 6 squared (36), and each one should have a picture made up of O's to accompany it.

Solution



```
for i in range from 3 to 7
do
  print i x i
  for j in range i
  do
    print "O" x i
```

```
1 for i in range(3, 7):
2   print(i * i)
3   for j in range(i):
4     print("O"*i)
5
```



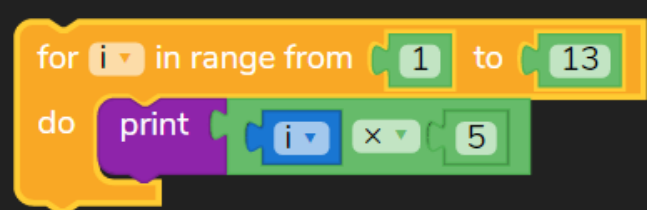
Stage 6- Times tables

In the next stages we will create an activity based around times tables. Using a for loop and a range we can print out a whole set of times-tables quite easily.

Tasks

- Create a 'for i in range' loop where 'i' takes the values: 1, 2, 3, ... , 12.
- Inside, print the value of 'i' x 5.
- Check that it prints the five times table, from 5 up to 60.

Solution



```
for i in range from 1 to 13
do
  print i x 5
```

```
1 for i in range(1, 13):
2   print(i * 5)
3
```



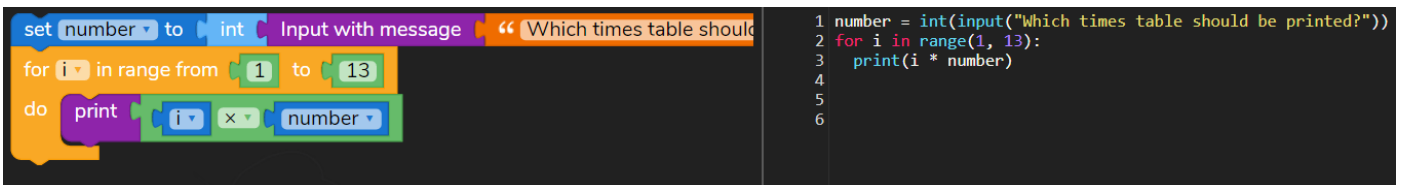
Stage 7- Interacting with the user

To improve our program we can ask the user what times table they would like to print. Remember that when you get numerical input you need to convert it to an integer first.

Tasks

- At the top of your program, use an 'input' command to ask the user 'Which times table should be printed?' Create a variable called 'number' and store the user's response in it.
- Convert 'number' to an integer. Use the 'int' function to do this.
- Edit your for loop it so that it prints the first 12 entries of the selected times table, starting with $1 \times \text{number}$.
- Test your code by entering '7'. Your code should print the first 12 members of the 7 times-table: 7, 14, 21, ... 84

Solution



The image shows the Scratch solution for Stage 7. On the left, there are code blocks: 'set number to int', 'Input with message' (with the text 'Which times table should be printed?'), a 'for i in range from 1 to 13' loop, and a 'do' block containing 'print' with 'i x number'.

```

1 number = int(input("Which times table should be printed?"))
2 for i in range(1, 13):
3     print(i * number)
4
5
6

```



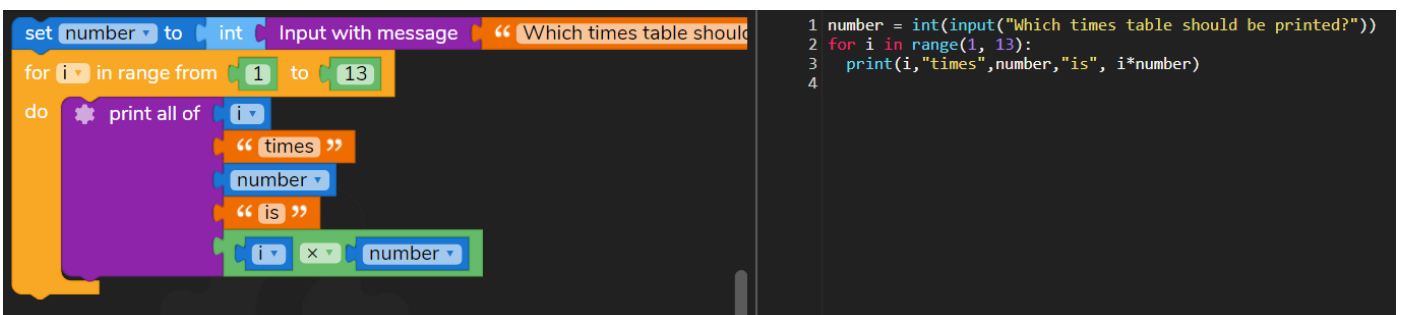
Stage 8- Formatting the times table

It will be more user-friendly to print the times-tables as sentences. For example "5 times 7 is 35, 6 times 7 is 42 etc..."

Tasks

- Edit your code so that the print out says (for example) '6 times 7 is 42', each on a new line. The easiest way to do this is to write: 'print(i, "times", number, "is", ...)
- Test your code by printing the first 12 entries from the 7 times table again, from '1 times 7 is 7' up to and including '12 times 7 is 84'.

Solution



The image shows the Scratch solution for Stage 8. On the left, there are code blocks: 'set number to int', 'Input with message' (with the text 'Which times table should be printed?'), a 'for i in range from 1 to 13' loop, and a 'do' block containing 'print all of' with 'i', 'times', 'number', 'is', and 'i x number'.

```

1 number = int(input("Which times table should be printed?"))
2 for i in range(1, 13):
3     print(i,"times",number,"is", i*number)
4

```



Have a go at these challenges to improve your magic 8 ball game even more!

Suggestions

- Use the 'isdigit' method to check if 'number' consists of only digits before trying to convert it to an integer. If not, provide the user with an error message.
- Choose a background and add a sprite to the screen called 'character1'. Make 'character1' show a speech bubble saying "Which times-table do you want to know?".
- After a few seconds ask the user to input the value of 'number'.
- Add another sprite called 'character2' and make them recite the times table in speech bubbles, with a gap of 1 second between each one.
- Try to add more user input - so that the user can say which times table, where to start and where to end for example.

Lesson 4- Loops and Selection

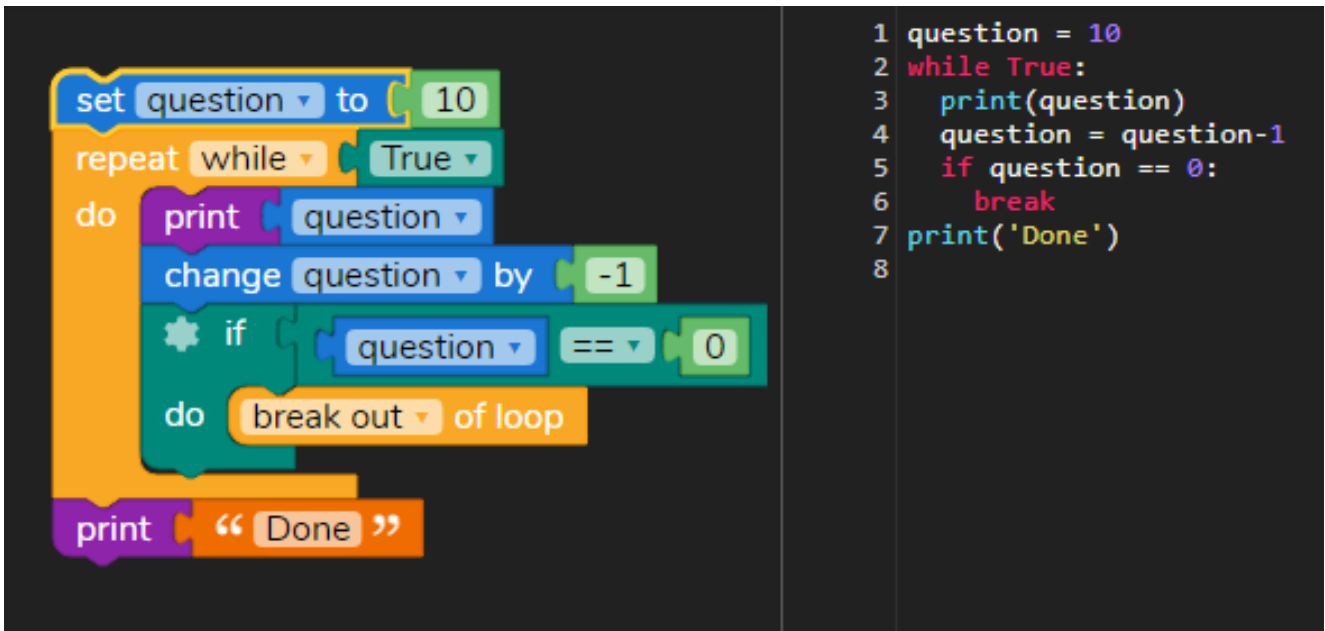
Stage 1- ,while' loops and ,break' statements

In Python, 'while' loops continue repeating the code inside them while a test is true. The 'break' statement in Python can be used to break out of a loop.

Tasks

- Create a variable called 'question' and set it equal to 10.
- Create a while loop which runs forever. In Python you can write 'while True:' to achieve this.
- Inside the loop, print the value of 'question' and then reduce 'question' by one.
- Write an 'if' statement that checks if 'question' is zero. If so, use 'break' to quit the loop.
- At the end of your program, print 'Done' so that you can check the code doesn't run forever.

Solution



The image shows two representations of the same code: a Scratch block-based version on the left and a Python text-based version on the right. The Scratch code starts with a 'set question to 10' block, followed by a 'repeat while True' loop. Inside the loop, there are three blocks: 'print question', 'change question by -1', and an 'if question == 0' block with a 'break out of loop' block attached. After the loop, there is a 'print "Done"' block. The Python code is a direct translation of this logic.

```
1 question = 10
2 while True:
3     print(question)
4     question = question-1
5     if question == 0:
6         break
7 print('Done')
```

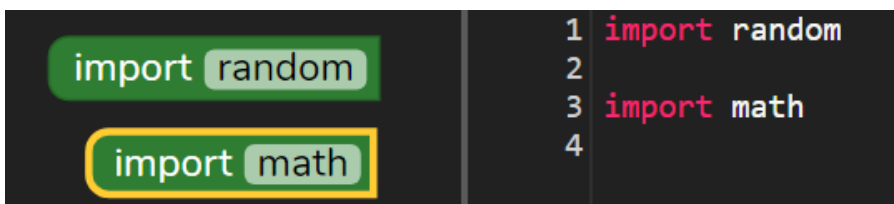
Stage 2- Importing libraries

Let's start making the game itself, using a while loop.

First, Python has built-in libraries with procedures that we can use and we need to 'import' them.

Tasks

- Import the 'random' library. This should go at the top of your program.
- Run your program and check it still works.



The image shows two representations of the same code: a Scratch block-based version on the left and a Python text-based version on the right. The Scratch code has two 'import' blocks: 'import random' and 'import math'. The Python code is a direct translation of this logic.

```
1 import random
2
3 import math
4
```



The magic-8-ball will ask the user to think of a question and will answer using a random response, until they quit.

Tasks

- Keep your 'while' loop but remove the code where 'question' is set to 10, the code where 'question' is printed, and the code where it is reduced by 1.
- Inside the 'while' loop, use the 'input' function to tell the user 'Ask the magic 8 ball a question: (leave blank to quit)' and assign that to the 'question' variable. Each time the loop runs the user can ask a new question.
- Next, use an 'if' statement to check if 'question' is equal to an empty string (" or ""). If so, 'break' out of the loop.
- Test the code, does it ask you for a question and then keep doing so until you leave the response empty?

Solution

```

import random

repeat while True:
do set question to Input with message " Ask the magic 8 ball a question: (leave blank to..."
  * if question == ""
do break out of loop
print " Done "
  
```

```

1 import random
2
3
4 while True:
5     question = input('Ask the magic 8 ball a question: (leave blank to quit)')
6
7     if question == "":
8         break
9     print('Done')
10
  
```



If 'question' is not empty, we can use an 'else' statement to continue the game. We will use random numbers between 1 and 8 to generate an answer from the magic 8 ball.

Tasks

- Write an else statement after your 'if' statement. The code inside will execute if 'question' is not empty.
- Inside, make a variable called 'answer' and use random.randint to assign it a random number between 1 and 8 inclusive.
- Test your code by asking the question "Will it rain tomorrow?" a few times. Check that you can see the random values of 'answer' in the debug panel (bottom right). When you're ready leave the question blank to exit the loop.

Solution

```

import random

repeat while True:
do set question to Input with message " Ask the magic 8 ball a question: (leave blank to..."
  * if question == ""
do break out of loop
else set answer to random integer from 1 to 8
print " Done "
  
```

```

1 import random
2
3
4 while True:
5     question = input('Ask the magic 8 ball a question: (leave blank to quit)')
6
7     if question == "":
8         break
9     else:
10        answer = random.randint(1,8)
11    print('Done')
  
```

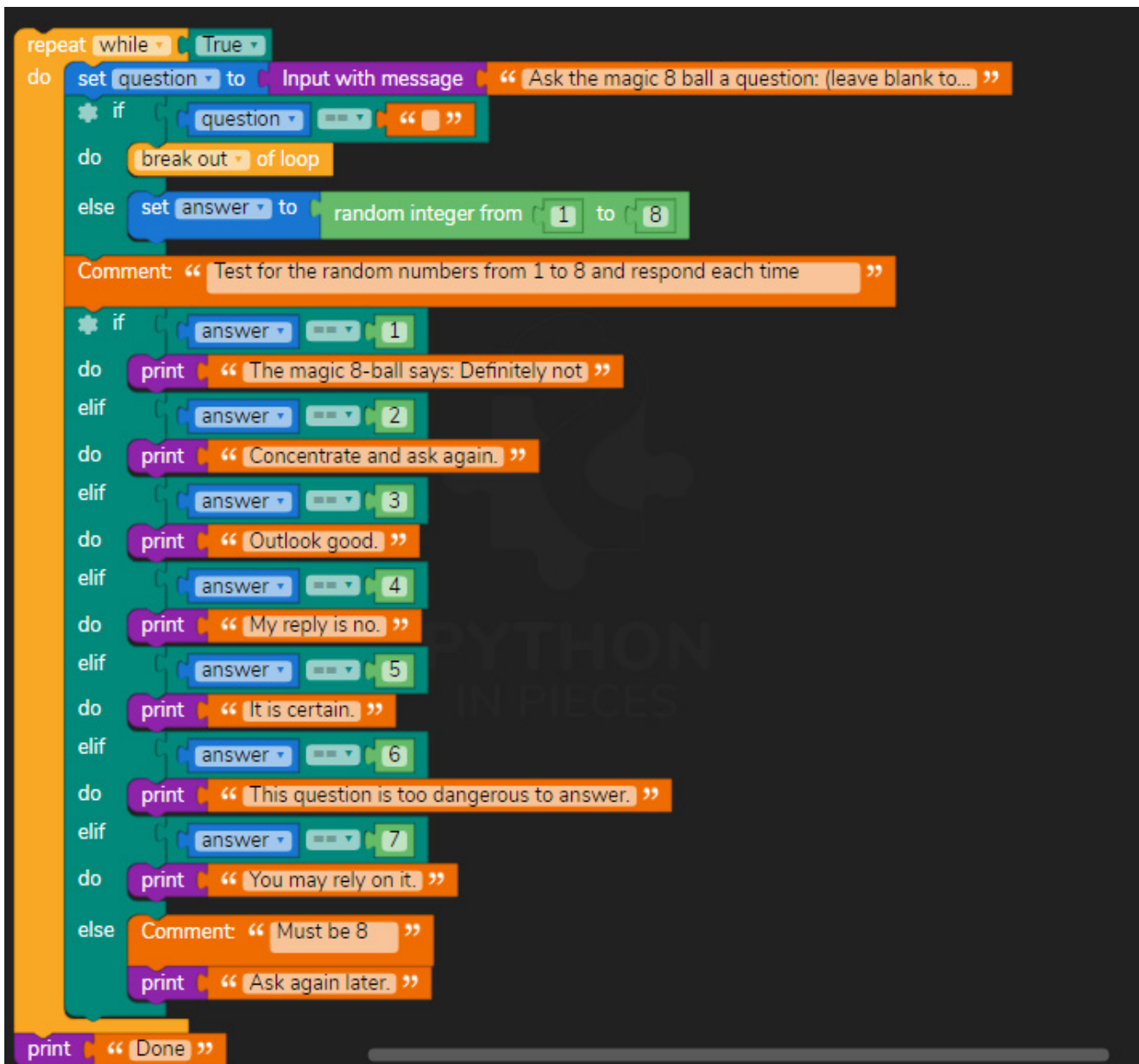


We can use if/elif statements to print the eight possible answers that the 8 ball might give.

Tasks

- Use an 'if' statement to print "The magic 8-ball says: Definitely not" if 'answer' equals 1.
- Use six 'elif' statements to check if 'answer' is equal to 2, 3, 4, 5, 6 or 7, and print appropriate responses of the form "The magic 8-ball says: ..."
- Use a final 'else' statement at the end to print an appropriate response if 'answer' equals 8.
- Test your code by asking the 8 ball "Will it rain tomorrow?". When you're ready, leave the question blank to exit the loop.

Solution



```
repeat while True
do
  set question to Input with message "Ask the magic 8 ball a question: (leave blank to..."
  if question == ""
  do
    break out of loop
  else
    set answer to random integer from 1 to 8
    Comment: "Test for the random numbers from 1 to 8 and respond each time"
    if answer == 1
    do
      print "The magic 8-ball says: Definitely not"
    elif answer == 2
    do
      print "Concentrate and ask again."
    elif answer == 3
    do
      print "Outlook good."
    elif answer == 4
    do
      print "My reply is no."
    elif answer == 5
    do
      print "It is certain."
    elif answer == 6
    do
      print "This question is too dangerous to answer."
    elif answer == 7
    do
      print "You may rely on it."
    else
      Comment: "Must be 8"
      print "Ask again later."
  print "Done"
```

```

1 import random
2
3 while True:
4     question = input('Ask the magic 8 ball a question: (leave blank to quit)')
5     if question == '':
6         break
7     else:
8         answer = random.randint(1, 8)
9
10    #Test for the random numbers from 1 to 8 and respond each time
11    if answer == 1:
12        print('The magic 8-ball says: Definitely not')
13    elif answer == 2:
14        print('Concentrate and ask again.')
15    elif answer == 3:
16        print('Outlook good.')
17    elif answer == 4:
18        print('My reply is no.')
19    elif answer == 5:
20        print('It is certain.')
21    elif answer == 6:
22        print('This question is too dangerous to answer.')
23    elif answer == 7:
24        print('You may rely on it.')
25    else:
26        #Must be 8
27        print('Ask again later.')
28
29 print('Done')
30

```

Stage 6- Debug

There is a bug with this code. The answers are meant to be random but a lot of the time it says 'Ask me again later'. Can you see why?

Tasks

- Fix the bug - make sure that it always gives a different random response for every question.
- Test your code a few times. When you're ready, leave the question blank to exit the loop.

Solution

```

import random

while True:
    question = input('Ask the magic 8 ball a question: (press enter to quit)')
    if question == '':
        break
    else:
        answer = random.randint(1, 8)

        if answer == 1:
            print('The magic 8-ball says: It is certain.')
        elif answer == 2:
            print('The magic 8-ball says: Concentrate and ask again.')
        elif answer == 3:
            print('The magic 8-ball says: The outlook is good.')
        elif answer == 4:
            print('The magic 8-ball says: My reply is no.')
        elif answer == 5:
            print('The magic 8-ball says: My sources say definitely not.')
        elif answer == 6:
            print('The magic 8-ball says: My sources say definitely not.')
        elif answer == 7:
            print('The magic 8-ball says: My sources say definitely not.')
        else:
            #Must be 8
            print('The magic 8-ball says: Ask me again later.')

```



Have a go at these challenges to improve your magic 8 ball game even more!

Suggestions

- Change the print code so that the user sees an alert box with the response and is asked if they have another question or not.
- Handle the possible responses by the user ('yes', 'y', 'no', 'n' and other possibilities). If they say 'yes' or 'y' then prompt them to ask another question.
- Go into design mode. Make the 8 ball look more circular by increasing the number of sides.
- Write code to make the 8 ball speak its answers using speech bubbles.

Possible Solution

```
1 import random
2 from time import sleep
3
4 sleep(20)
5 while True:
6     question = input('Ask the magic 8 ball a question: (leave blank to quit)')
7     if question == '':
8         break
9     else:
10        answer = random.randint(1, 8)
11
12 #Test for the random numbers from 1 to 8 and respond each time
13 if answer == 1:
14     question = (input('The magic 8-ball says: Definitely not. \n Do you want to ask another question?')).lower()
15
16 elif answer == 2:
17     ball_inner.say('Concentrate and ask again.',5)
18     sleep(4)
19     question = (input('Do you want to ask another question?')).lower()
20     #question = (input('Concentrate and ask again. \n Do you want to ask another question?')).lower()
21
22 elif answer == 3:
23     label.text = 'Concentrate \n and ask \n again.'
24     label.scaleX = 0.7
25     label.scaleY = 0.7
26     sleep(4)
27     label.text = '8 ball'
28     label.scaleX = 1
29     label.scaleY = 1
30     question = (input('Do you want to ask another question?')).lower()
31     #question = (input('Outlook good. \n Do you want to ask another question?')).lower()
32
33 elif answer == 4:
34     question = (input('My reply is no. \n Do you want to ask another question?')).lower()
35 elif answer == 5:
36     question = (input('It is certain. \n Do you want to ask another question?')).lower()
37 elif answer == 6:
38     question = (input('This question is too dangerous to answer. \n Do you want to ask another question?')).lower()
39 elif answer == 7:
40     question = (input('You may rely on it. \n Do you want to ask another question?')).lower()
41 else:
42     #Must be 8
43     question = (input('Ask again later. \n Do you want to ask another question?')).lower()
44 #get the first character of the user's answer
45 question = question[0]
46 if question == 'n':
47     break
48 elif question != 'y':
49     pip.alert('Sorry I didn't understand.')
50
51
52 pip.alert('Thank you for playing')
53
```


Lesson 5- Functions

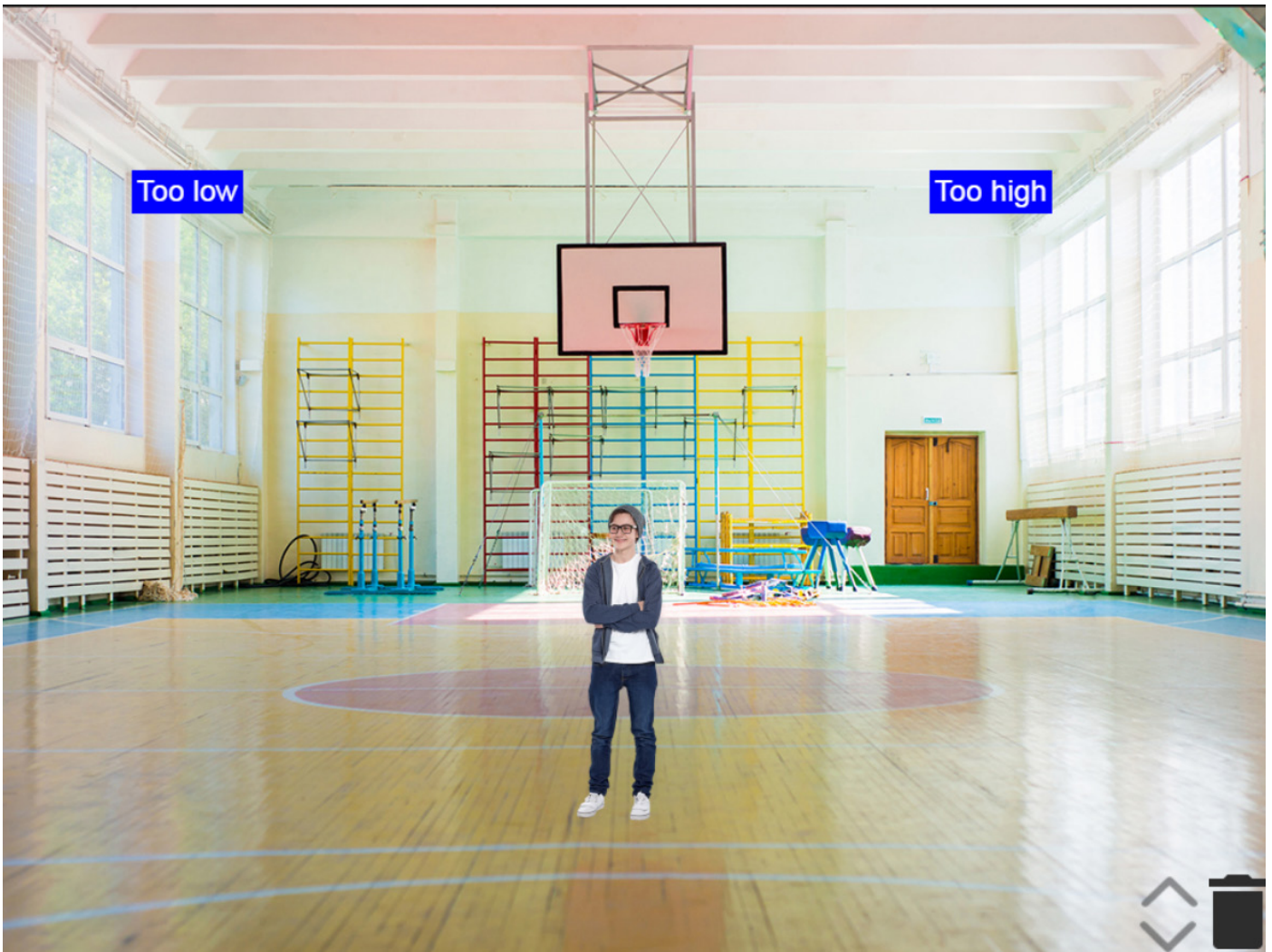
Stage 1- Customize the scene

In this activity you will be making a number guessing game. Customize the design view but keep the same object types.

Tasks

- Customize the design view but keep the same object types - one sprite and two text areas with blue backgrounds.
- Ensure your sprite is called 'character' and that the text areas are called 'too_low' and 'too_high' and that they contain the correct text.

Solution





Initialize the variables that will store information for our game.

Tasks

- Import the 'random' module and the 'sleep' module from the 'time' package.
- Create a variable called 'guess', set to 0 to start with
- Create a variable called 'answer' and initialize it to a random number between 1 and 100 using 'random.randint'.
- Run your program. Check that you can see the value of the variable 'answer' in the debugging panel at the bottom-right of the screen. It will change each time you run the program.

Solution

The image shows two parts of the solution. On the left, a block of code blocks: 'from time import sleep', 'import random', 'set guess to 0', and 'set answer to random integer from 1 to 100'. On the right, the corresponding Python code is shown: '1 import random', '2 from time import sleep', '3', '4 guess=0', '5 answer=random.randint(1,100)'.



Interact with the user.

Tasks

- Add code so that your character says: 'I am thinking of a number between 1 and 100. Try and guess it'.
- Use the 'sleep' command to make the program wait 6 seconds so that the player has time to read the instructions.
- Next, use an 'input' box that says 'Guess:' and set the value of the variable 'guess' to the user response.
- Convert 'guess' to an integer using the 'int' function.
- Test your program by entering the guess 50. Check that the variable 'guess' has the value 50 in the debug panel (bottom-right)

Solution

The image shows two parts of the solution. On the left, a block of code blocks: 'import random', 'from time import sleep', 'set guess to 0', 'set answer to random integer from 1 to 100', 'character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs', 'sleep for seconds: 6', and 'set guess to int Input with message "Guess:"'. On the right, the corresponding Python code is shown: '1 from time import sleep', '2', '3 import random', '4', '5 guess = 0', '6 answer = random.randint(1, 100)', '7 character.say("I am thinking of a number between 1 and 100. Try and guess it.",5)', '8 sleep(6)', '9 guess = int(input("Guess:"))', '10'.



Stage 4- Check if they got it right

We will give feedback to the user if their guess was too low or too high. First, let's check if they got it right.

Tasks

- Add an 'if' statement that makes the character say, 'Well done!' if the guess is correct (if guess equals answer).
- Test that it is working. You can use the debugging panel to see what the correct answer is!

Solution

```

import random

from time import sleep

set guess to 0
set answer to random integer from 1 to 100
character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs
sleep for seconds: 6
set guess to int Input with message "Guess:"
if guess == answer
do character say: "Well done!" for 5 secs

```

```

1 from time import sleep
2
3 import random
4
5 guess = 0
6 answer = random.randint(1, 100)
7 character.say("I am thinking of a number between 1 and 100. Try and guess it.",5)
8 sleep(6)
9 guess = int(input('Guess:'))
10 if guess == answer:
11     character.say("Well done!",5)
12

```



Stage 5- Expand the ,if' statement

If you guessed too low, we need to tell the user to try again.

Tasks

- Next, add an 'elif' statement to test if the guess is too low.
- If it is too low, your character should say 'Too low!'
- Also, the background colour of the 'too_low' text object should be set to 'red'
- Test your program by typing in a number less than the value of 'answer' (shown in the debugging panel)

Solution

```

from time import sleep

import random

set guess to 0
set answer to random integer from 1 to 100
character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs
sleep for seconds: 6
set guess to int Input with message "Guess:"
if guess == answer
do character say: "Well done!" for 5 secs
elif guess < answer
do character say: "Too low!" for 5 secs
too_low set background color to red

```

```

1 import random
2 from time import sleep
3
4
5 guess = 0
6 answer = random.randint(1, 100)
7 character.say("I am thinking of a number between 1 and 100. Try and guess it.",5)
8 sleep(6)
9 guess = int(input('Guess:'))
10
11 if guess==answer:
12     character.say("Well done!",5)
13 elif guess < answer:
14     character.say("Well done!",5)
15     too_low.set_bg_color("red")
16
17
18

```



Stage 6- Finish the ,if' statement

If the guess wasn't correct and it wasn't too low, then it must be too high. Let's tell the user if their answer is too high.

Tasks

- Add an 'else' statement at the end of your program. Your program should now contain one 'if' statement, one 'elif' statement and one 'else' statement.
- The code inside the 'else' statement will execute if the guess wasn't correct and wasn't too low, so it must be too high. Inside the 'else' statement make the character say, 'Too high!'
- The background colour of the 'Too high' text object should change to red.
- Test your program by typing in a number more than the value of 'answer' (shown in the debugging panel).

Solution

```

import random

from time import sleep

set guess to 0
set answer to random integer from 1 to 100

character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs

sleep for seconds: 6

set guess to int input with message "Guess"

if guess == answer
do
character say: "Well done." for 5 secs
elif guess < answer
do
character say: "Too low." for 5 secs
too_low set background color to red
else
character say: "Too high." for 5 secs
too_high set background color to red

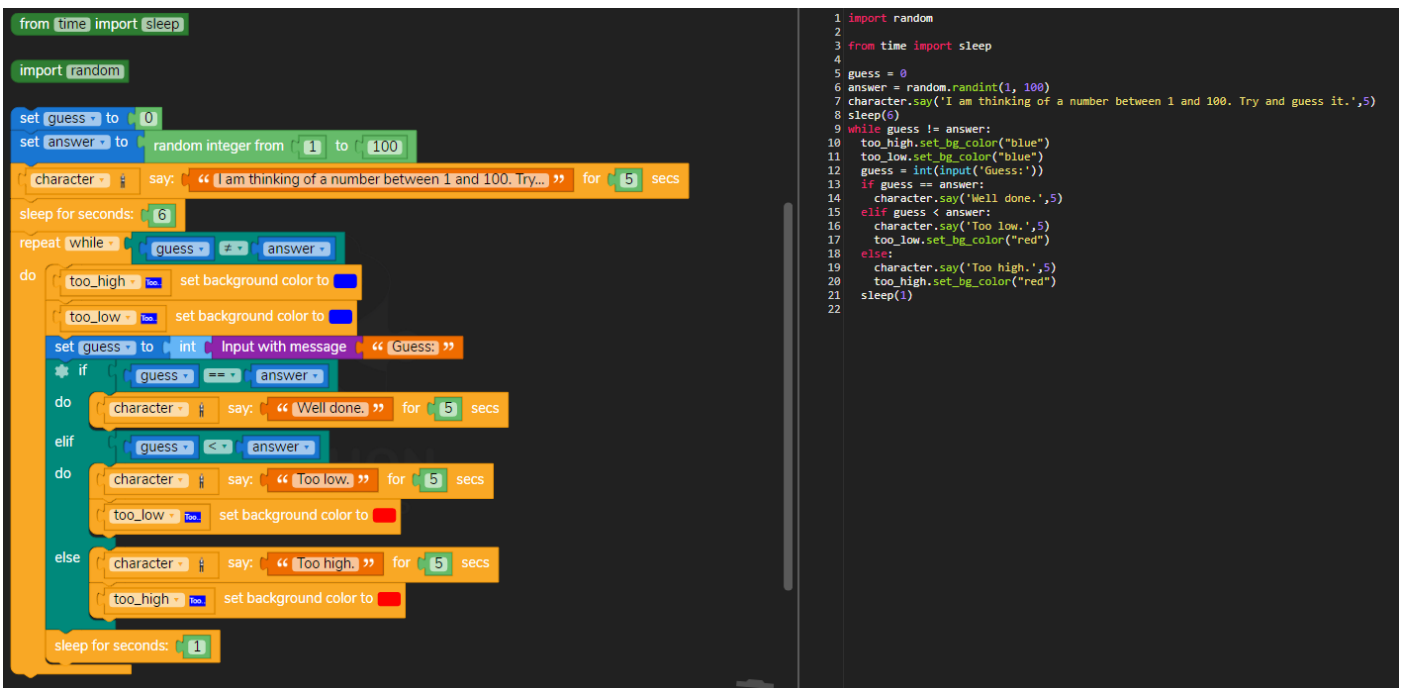
```

Let's add a 'while' loop so the game continues if you get it wrong, so that you can carry on guessing.

Tasks

- Add a while loop at the start of your program, after the 'guess' and 'answer' variables are initialized and the character has said "I am thinking of a number", and before the 'input' statement. The while loop should run while 'guess' is not equal to 'answer'.
- Inside the while loop, set the background color of both text fields to blue. This will make sure that they go back to blue before each guess.
- Indent the rest of your code so it is completely inside the while loop. This will enable guesses to be made again and again.
- At the end of your code, after the 'else' statement, use a 'sleep' command to sleep the code for 1 second, so that users can see the text fields go red before they go blue again.
- Test your program. Please get the answer too high once, then too low once and then get it correct.

Solution



```

from time import sleep

import random

set guess to 0
set answer to random integer from 1 to 100

character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs

sleep for seconds: 6

repeat while guess != answer
do
  too_high set background color to blue
  too_low set background color to blue
  set guess to int Input with message "Guess"
  if guess == answer
  do
    character say: "Well done." for 5 secs
  elif guess < answer
  do
    character say: "Too low." for 5 secs
    too_low set background color to red
  else
    character say: "Too high." for 5 secs
    too_high set background color to red
  sleep for seconds: 1
  
```

```

1 import random
2
3 from time import sleep
4
5 guess = 0
6 answer = random.randint(1, 100)
7 character.say("I am thinking of a number between 1 and 100. Try and guess it.",5)
8 sleep(6)
9 while guess != answer:
10     too_high.set_bg_color("blue")
11     too_low.set_bg_color("blue")
12     guess = int(input('Guess:'))
13     if guess == answer:
14         character.say("Well done.",5)
15     elif guess < answer:
16         character.say("Too low.",5)
17         too_low.set_bg_color("red")
18     else:
19         character.say("Too high.",5)
20         too_high.set_bg_color("red")
21     sleep(1)
22
  
```



Put the game code into a function so that you can play over again. Each time you call the function we will execute all of your code.

Tasks

- Add a 'Play' button to the design view. Call it 'play_button'.
- Define a function called 'play_game'.
- Move all your code (except the imports) inside the function.
- Write some code to call the function when the button is clicked. Use 'pip.eventmanager.when_click' to do this.
- Test your program. Click "Play" and get the answer correct. Click Play again and check that you get a brand new game. Get it correct again.

Solution

The image shows a side-by-side comparison of a game implementation. On the left is a block-based solution (Scratch-style), and on the right is the equivalent Python code.

Block-based solution (left):

- Imports: `from time import sleep`, `import random`
- Function definition: `def play_game`
- Initialization: `set guess to 0`, `set answer to random integer from 1 to 100`
- Initial message: `character say: "I am thinking of a number between 1 and 100. Try..." for 5 secs`
- Initial sleep: `sleep for seconds: 6`
- Repeat loop: `repeat while guess != answer`
- Loop body:
 - `do` block:
 - `too_high set background color to blue`
 - `too_low set background color to blue`
 - `set guess to int Input with message "Guess:"`
 - `if guess == answer` block:
 - `do` block: `character say: "Well done!" for 5 secs`
 - `elif guess < answer` block:
 - `do` block: `character say: "Too low!" for 5 secs`
 - `too_low set background color to red`
 - `else` block:
 - `do` block: `character say: "Too high!" for 5 secs`
 - `too_high set background color to red`
 - Final sleep: `sleep for seconds: 1`
- Event listener: `When click on play_button execute the function play_game`

Python code solution (right):

```
1 def play_game():
2     guess = 0
3     answer = random.randint(1, 100)
4     character.say("I am thinking of a number between 1 and 100. Try and guess it.",5)
5     sleep(6)
6     while guess != answer:
7         too_high.set_bg_color("blue")
8         too_low.set_bg_color("blue")
9         guess = int(input("Guess:"))
10        if guess == answer:
11            character.say("Well done.",5)
12        elif guess < answer:
13            character.say("Too low.",5)
14            too_low.set_bg_color("red")
15        else:
16            character.say("Too high.",5)
17            too_high.set_bg_color("red")
18        sleep(1)
19
20
21 from time import sleep
22
23 import random
24
25 pip.eventmanager.when_click(play_button,play_game)
26
```



Have a go at these challenges to improve your quiz.

Suggestions

- Add some code to keep a record of how many guesses it took the player and let them know at the end of the game. Try to use a function for this.
- Make the “Play” button hidden when it is not needed and make it visible when the player has guessed correctly.

Possible Solution

```
1 import random
2 from time import sleep
3
4 def play_game():
5     play_button.hide()
6     number_of_guesses = 0
7     number_guesses.text = ''
8     guess = 0
9     #answer = random.randint(1, 100)
10    answer = 41
11    character.say('I am thinking of a number between 1 and 100. Try and guess it.',5)
12    sleep(6)
13    while guess != answer:
14        too_high.set_bg_color("blue")
15        too_low.set_bg_color("blue")
16        guess = int(input('Guess:'))
17        number_of_guesses= number_of_guesses+1
18        number_guesses.text=str(number_of_guesses)
19        if guess == answer:
20            character.say('Well done. It took you ' + str(number_of_guesses) + ' guesses.',5)
21            play_button.show()
22        elif guess < answer:
23            character.say('Too low.',5)
24            too_low.set_bg_color("red")
25        else:
26            character.say('Too high.',5)
27            too_high.set_bg_color("red")
28        sleep(1)
29
30
31
32
33    pip.eventmanager.when_click(play_button,play_game)
34
```

Lesson 6 – Using sound in a game

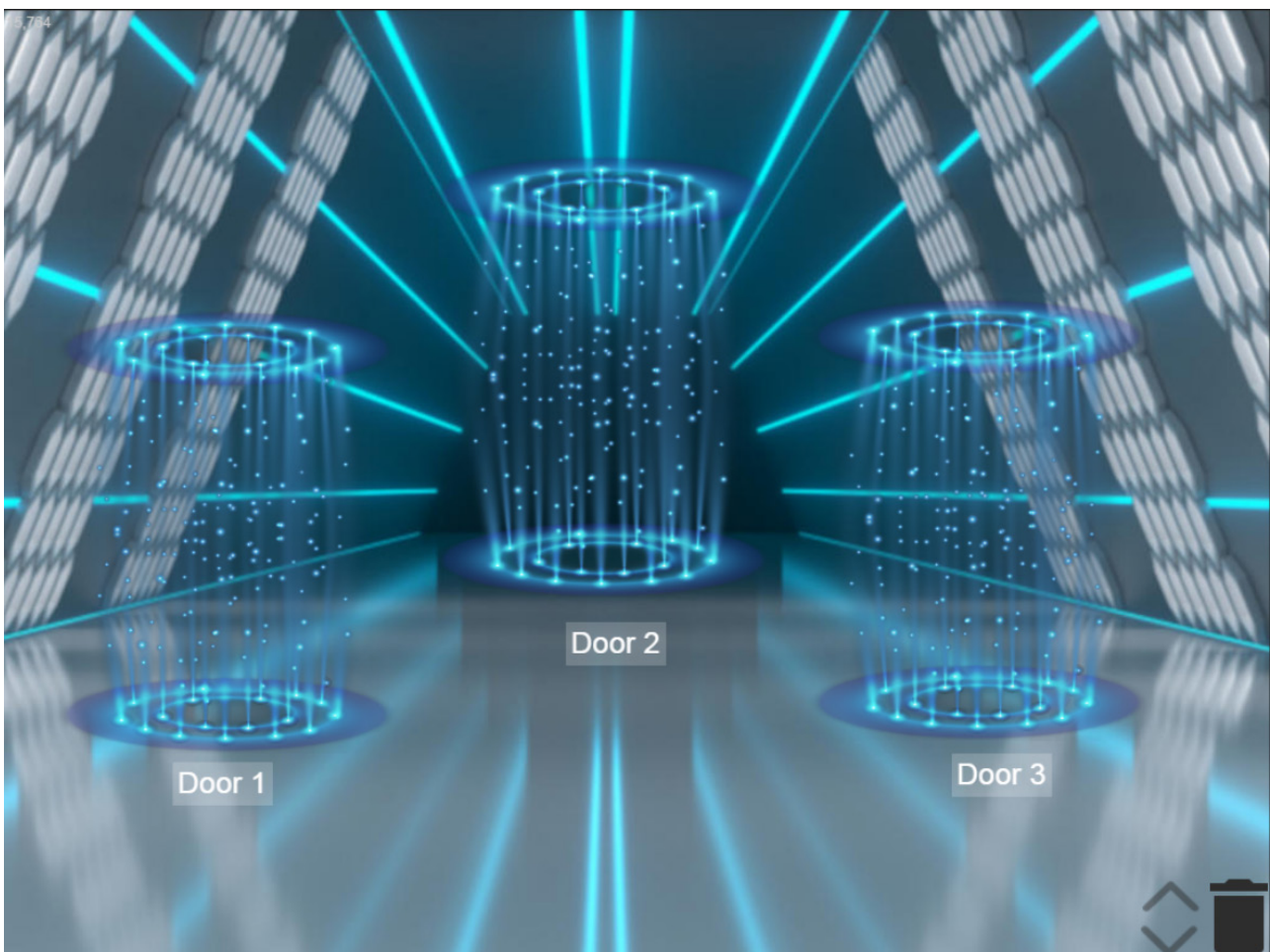
Stage 1- Set the scene

In this game, the player will be asked to choose a door to go through. One will hide a scary monster and the other two will be safe. First let's set up the scene.

Tasks

- In design view, change the background if you want to.
- Set up a scene with three closed doors (choose what images you want to use) and three text fields.
- Name the doors 'door1', 'door2' and 'door3'.
- Name the labels 'label1', 'label2', 'label3'.
- Make the text in the labels say 'Door 1', 'Door 2', 'Door 3' and position them near the corresponding doors.

Solution



Stage 2- Add a character

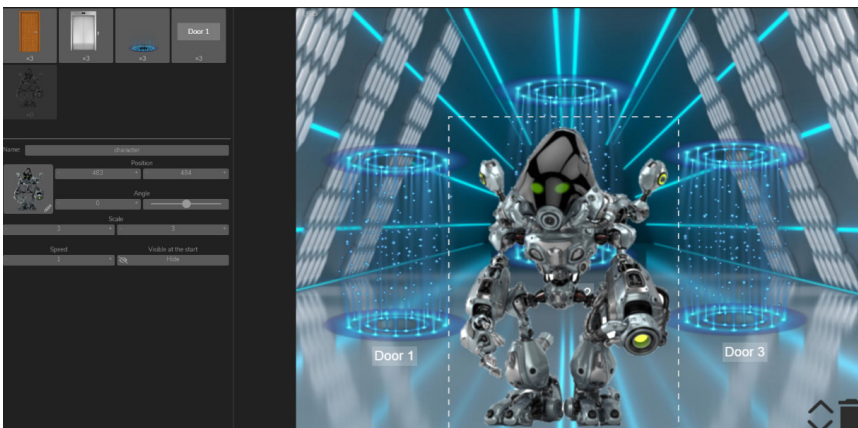
Lesson 6- Using sound in a game

When the player chooses the wrong door, a scary character will attack them. Let's add that next.

Tasks

- Add a scary character. Name it 'character'
- Set it to hide at the beginning of the game. It should be large and positioned in the centre of the screen.

Solution



Stage 3- Initialize the game.

Let's initialize the variables we need.

Tasks

- import the 'random' module and the 'sleep' module from the 'time' package. These should go at the top of your program.
- Create a variable called 'playing' and set it to True.
- Create variables called 'score', 'character_door', and 'chosen_door' and set these all to zero.

Solution

```
from time import sleep

import random

set playing to True
set score to 0
set character_door to 0
set chosen_door to 0
```

```
1 from time import sleep
2
3 import random
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9
```



We need the game to continue playing, asking the player which door they want to open. A while loop is perfect for this.

Tasks

- Add a while loop that runs while 'playing' is True. In Python you can use 'while playing:' to achieve this.
- Inside the loop, use 'random.randint' to set 'character_door' to a random number between 1 and 3 inclusive.
- Also inside the loop, ask the user to input which door they want to choose. Tell the user: 'Choose a door 1, 2 or 3 (0 to exit)'. Set 'chosen_door' to the user's response. Convert this to an integer using the 'int' function.
- Use an 'if' statement to check if the user has entered 0. If so, set 'playing' to False. This will end the game.
- Test your game a few times - check that you get different values of 'character_door' (in the debug panel at the bottom right). When you're ready, end the game by choosing door 0.

Solution

```
from time import sleep

import random

set playing to true
set score to 0
set character_door to 0
set chosen_door to 0
repeat (while) (playing)
do
  set character_door to random integer from 1 to 3
  set chosen_door to int (input with message "Choose a door 1, 2 or 3 (0 to exit)")
  if chosen_door == 0
  do
    set playing to False
```

```
1 import random
2
3 from time import sleep
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9 while playing:
10     character_door = random.randint(1,3)
11     chosen_door = int(input('Choose a door 1, 2 or 3 (0 to exit)'))
12     if chosen_door == 0:
13         playing = False
14
```



We need to make the character appear if the user chooses their door, and the game should end.

Tasks

- Use an 'else' statement to continue the game if the user doesn't enter 0.
- First, play sound of a door opening and sleep for one second so that the user can hear the whole sound.
- Next, write an 'if' statement to check if the user has selected the hidden character's door. If so, the character should be shown, a scary sound should play and 'playing' should be set to False.
- Test your game by looking in the debugger panel and choosing the hidden character's door on purpose. Check that the character appears, the game ends and your score stays as zero.

Solution

```
1 import random
2
3 from time import sleep
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9 while playing:
10     character_door = random.randint(1, 3)
11     chosen_door = int(input('Choose a door 1, 2 or 3 (0 to exit)'))
12     if chosen_door == 0:
13         playing = False
14     else:
15         pip.play_sound("Scary/Door.mp3",1)
16         sleep(1)
17         if chosen_door == character_door:
18             character_show()
19             pio.play_sound("Scary/Scream.mp3",1)
20             playing = False
21
```



If the user chose a safe door, we need to tell them and increase their score by 1.

Tasks

- Use a final 'else' statement, and inside show an alert using 'pip.alert', which says "You're safe!".
- Also increase the value of 'score' by 1.
- Test your game by looking in the debugger panel and choosing a 'safe' door (one different to character_door). When you've played a few times and you're ready, enter 0 to quit. Your score should be least 1 this time.

Solution

```
from time import sleep

import random

set playing to true
set score to 0
set character_door to 0
set chosen_door to 0
repeat (while) playing
do
  set character_door to random integer from 1 to 3
  set chosen_door to int (Input with message "Choose a door 1, 2 or 3 (0 to exit)")
  if chosen_door == 0
  do
    set playing to False
  else
    play sound 1 times
    sleep for seconds: 1
    if chosen_door == character_door
    do
      character show
      play sound 1 times
      set playing to False
    else
      change score by 1
      alert "You're safe."
```

```
1 import random
2
3 from time import sleep
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9 while playing:
10 character_door = random.randint(1, 3)
11 chosen_door = int(input('Choose a door 1, 2 or 3 (0 to exit)'))
12 if chosen_door == 0:
13 playing = False
14 else:
15 pip.play_sound("Scary/Door.mp3",1)
16 sleep(1)
17 if chosen_door == character_door:
18 character.show()
19 pip.play_sound("Scary/Scream.mp3",1)
20 playing = False
21 else:
22 score = score + 1
23 pip.alert("You're safe.")
```



Let's make the doors open and close by changing their images using the 'set_image' method.

Tasks

- Before playing the sound of the door opening, change the image of the door the user chose so it looks open. For example, if they chose door 1, use 'door1.set_image(...)', and so on for door2 and door3.
- Test your program by clicking on door1. Check that the image of door1 changes to an open door. When you're ready, enter 0 to end the game.

Solution

```
from time import sleep

import random

set playing to True
set score to 0
set character_door to 0
set chosen_door to 0
repeat while playing
do
  set character_door to random integer from 1 to 3
  set chosen_door to int Input with message "Choose a door 1, 2 or 3 (0 to exit)"
  if chosen_door == 0
  do
    set playing to False
  else
    if chosen_door == 1
    do
      door1 set image to [door1 open image]
    elif chosen_door == 2
    do
      door2 set image to [door2 open image]
    else
      door3 set image to [door3 open image]
    play sound [door open sound] 1 times
    sleep for seconds: 1
    if chosen_door == character_door
    do
      character show
      play sound [door open sound] 1 times
      set playing to False
    else
      change score by 1
      alert "You're safe"
```

```
1 import random
2
3 from time import sleep
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9 while playing:
10     character_door = random.randint(1, 3)
11     chosen_door = int(input('Choose a door 1, 2 or 3 (0 to exit)'))
12     if chosen_door == 0:
13         playing = False
14     else:
15         if chosen_door == 1:
16             door1.set_image("doors/transporter_open.png")
17         elif chosen_door == 2:
18             door2.set_image("doors/transporter_open.png")
19         else:
20             door3.set_image("doors/transporter_open.png")
21
22     pip.play_sound("Scary/Door.mp3",1)
23     sleep(1)
24     if chosen_door == character_door:
25         character.show()
26         pip.play_sound("Scary/Scream.mp3",1)
27         playing = False
28     else:
29         score = score+1
30         pip.alert("You're safe.")
31
```



To finish off our game, it would be nice to add a 'Game over' alert and tell the user their score. Someone has written a game similar to yours, but it has two bugs. First, the game never ends - it keeps on playing even after the tiger attacks you. Second, the "Game over" message doesn't work.

Tasks

- Fix the code so that it doesn't keep asking you to choose a door after you have been attacked by the tiger.
- Fix the error in the "Game over" alert and play the game again. When you're ready, choose the hidden character's door on purpose and check that the alert appears telling you your score.

Solution

```
1 from time import sleep
2
3 import random
4
5 playing = True
6 score = 0
7 character_door = 0
8 chosen_door = 0
9 while playing:
10     character_door = random.randint(1, 3)
11     chosen_door = input('Choose a door 1, 2 or 3 (0 to exit)')
12     chosen_door = int(chosen_door)
13     if chosen_door == 0:
14         playing = False
15     else:
16         pip.play_sound("Scary/Door.mp3",1)
17         sleep(1)
18         if chosen_door == character_door:
19             character.show()
20             pip.play_sound("Scary/Door.mp3",1)
21             playing = False
22         else:
23             pip.alert("You're safe")
24             score = score+1
25     pip.alert('Game over. You scored '+ str(score) +' points')
26
```



Here are some ideas for how to improve your game.

Suggestions

- Make sure that all the doors are closed before the next time it asks you to choose a door. Currently they stay open.
- Add some success sounds when the user chooses a 'safe' door.
- Add an on-screen display so the player can see their score at all times. Make it update when the score changes.
- Position the character closer to the door they are supposed to be hiding behind to make it look more realistic.

Possible Solution

```
from time import sleep
import random

set playing to true
set character_door to 0
set chosen_door to 0
set score to 0

txt_score text set to str score

repeat while playing
do
door1 set image to
door2 set image to
door3 set image to

set character_door to random integer from 1 to 3
set chosen_door to int Input with message "(Choose a door 1, 2 or 3 (0 to exit))"

if chosen_door == 0
do set playing to False
else
if chosen_door == 1
do door1 set image to
elif chosen_door == 2
do door2 set image to
else
door3 set image to

play sound 1 times
sleep for seconds: 1

if chosen_door == character_door
do character show
play sound 1 times
sleep for seconds: 1
set playing to False
else
change score by 1
txt_score text set to str score
play sound 1 times
alert "You're safe."

alert "Game Over! You scored " + str score
```



```
1 from time import sleep
2
3 playing = True
4 character_door = 0
5 chosen_door = 0
6 score = 0
7 txt_score.text = str(score)
8 while playing:
9     door1.set_image("doors/transporter_closed.png")
10    door2.set_image("doors/transporter_closed.png")
11    door3.set_image("doors/transporter_closed.png")
12    character_door = random.randint(1, 3)
13    chosen_door = int(input('Choose a door 1, 2 or 3 (0 to exit)'))
14    if chosen_door == 0:
15        playing = False
16    else:
17        if chosen_door == 1:
18            door1.set_image("doors/transporter_open.png")
19        elif chosen_door == 2:
20            door2.set_image("doors/transporter_open.png")
21        else:
22            door3.set_image("doors/transporter_open.png")
23    pip.play_sound("Scary/Door.mp3",1)
24    sleep(1)
25    if chosen_door == character_door:
26        character.show()
27        pip.play_sound("Scary/Scream.mp3",1)
28        sleep(1)
29        playing = False
30    else:
31        score = score+1
32        txt_score.text = str(score)
33        pip.play_sound("Other Sounds/Twinkle.mp3",1)
34        pip.alert("You're safe.")
35    pip.alert('Game Over! You scored '+str(score))
36
37 import random
38
```