



python in pieces

A 2Simple Secondary Product

LESSON SOLUTIONS

Level 1

Contents

Lesson 1- Text Output.....	3	Lesson 4- Moving Sprites	19
Stage 1- Hello World.....	3	Stage 1- What is a sprite	19
Stage 2- Tell me about yourself	3	Stage 2- Positioning the jellyfish	20
Stage 3- Debug	4	Stage 3- Animate the jellyfish	21
Stage 4- Print all	4	Stage 4- Moving up and down	21
Stage 5- Printing a sentence.....	5	Stage 5- Moving left and right.....	22
Stage 6- Joining strings	5	Stage 6- Flipping a sprite.....	23
Stage 7- Multiplying a string by a number ...	6	Stage 7- Shark-attack!	24
Stage 8- Line breaks.....	6	Stage 8- Debug	25
Stage 9- Debug	7	Challenge	25
Challenge	7		
 		Lesson 5- Variables and user input	26
Lesson 2- Numbers and Calculations.....	8	Stage 1- Creating a variable.....	26
Stage 1- The number block	8	Stage 2- Creating more variables.....	26
Stage 2- Debug	8	Stage 3- Different naming systems	27
Stage 3- More complex calculations.....	9	Stage 4- Re-assigning variables.....	27
Stage 4- More complex calculations.....	9	Stage 5- Creating strings	28
Stage 5- Meet Monty.....	10	Stage 6- Getting input from the user	29
Stage 6- The power operator (**).....	10	Stage 7- Debug	29
Stage 7- Division and integer division	11	Stage 8- Textfields	30
Stage 8- The modulo operator(%)	11	Challenge	30
Stage 9- Conversion - Debug	12		
Stage 10- Monty munches	12	Lesson 6 – Repeat while.....	31
Challenge	13	Stage 1- while loop	31
 		Stage 2- Changing a variable	31
Lesson 3- Repeat Loops.....	14	Stage 3- Using an 'if' statement	32
Stage 1- Introducing a for loop	14	Stage 4- Using an 'elif' statement	32
Stage 2- Debug	14	Stage 5- Debug	33
Stage 3- Tick tock.....	15	Stage 6- Taking off.....	34
Stage 4- Playing a sound.....	15	Stage 7- Make the car move	35
Stage 5- Adding a clock	16	Stage 8- Debug	36
Stage 6- What is a range?.....	16	Stage 9- Get the potion	37
Stage 7- Using the counter variable	17	Challenge	38
Stage 8- Movement	17		
Challenge	18		

Lesson 1: Text Output

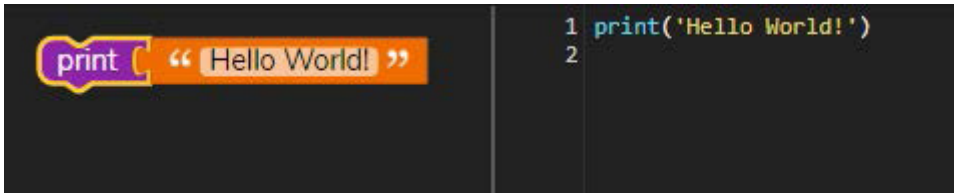
Stage 1- Hello World

In Python, a 'string' is a sequence of letters which we use to represent text. We can use the 'print' function to print strings to the screen.

Tasks

- Change the print block to output 'Hello World' instead of 'abc'. Your text needs to be surrounded by quotation marks (" or ')
- Check that you can see the output in the output panel (bottom-left). Tick this box when you're ready to proceed.

Solution



The image shows a Scratch script editor on the left and a Python code editor on the right. In Scratch, a purple 'print' block is connected to an orange text block containing the text "Hello World!". In the Python code editor, the first line of code is `1 print('Hello World!')` and the second line is empty.

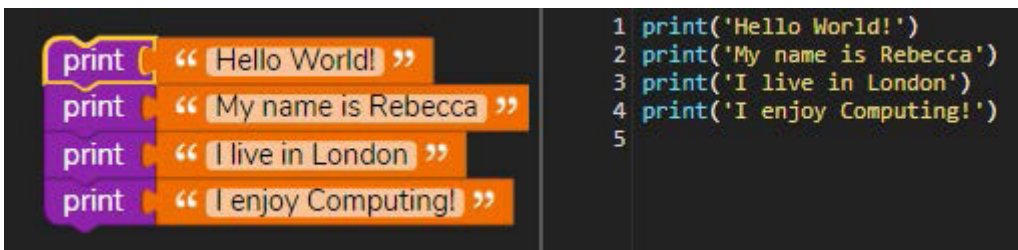
Stage 2- Tell me about yourself

Practise by printing some more text about yourself. Remember that strings need to be surrounded by quotation marks (" or ')

Tasks

- Add a print block to output "My name is".
- Add a print block to output "I live in".
- Add a print block to output "I enjoy".

Solution



The image shows a Scratch script editor on the left and a Python code editor on the right. In Scratch, four purple 'print' blocks are stacked vertically, each connected to an orange text block. The text blocks contain: "Hello World!", "My name is Rebecca", "I live in London", and "I enjoy Computing!". In the Python code editor, the first four lines of code are: `1 print('Hello World!')`, `2 print('My name is Rebecca')`, `3 print('I live in London')`, and `4 print('I enjoy Computing!')`. The fifth line is empty.

Stage 3- Debug

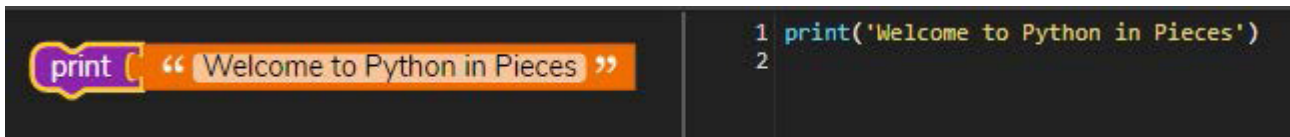
This code has a mistake in. If you run it, you will get an error message saying that there is 'bad input'.

Fix the code to make it work properly.

Tasks

- Fix the code so that it prints "Welcome to Python in Pieces"

Solution



The solution shows a Scratch 'print' block with the text "Welcome to Python in Pieces" and a corresponding Python code block with the following code:

```
1 print('Welcome to Python in Pieces')
2
```

Stage 4- Print all

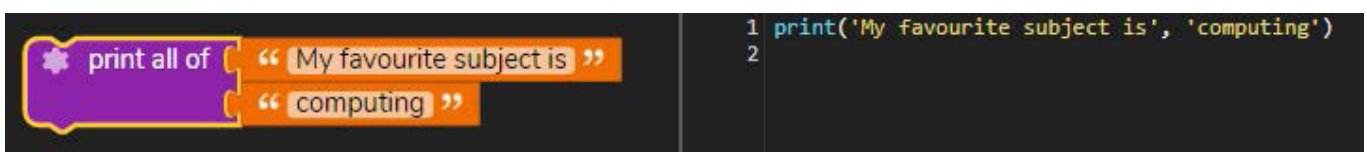
In Python you can print a mixture of several strings and numbers all in one go, using one print statement.

List the things you want to print with commas in between.

Tasks

- Print "My favourite subject is" followed by "computing". Either use the 'print all' block or use one print statement with a comma between the two strings. Notice that this function automatically inserts a space between 'is' and 'computing'.

Solution



The solution shows a Scratch 'print all of' block with two text boxes containing "My favourite subject is" and "computing" and a corresponding Python code block with the following code:

```
1 print('My favourite subject is', 'computing')
2
```

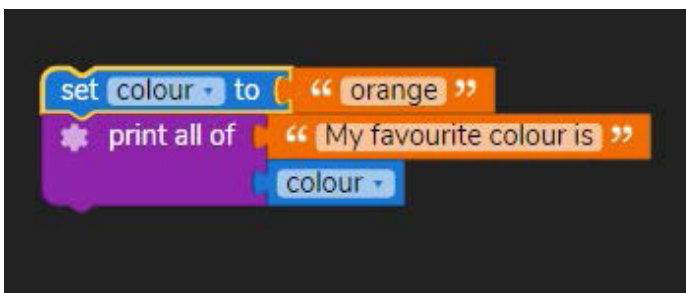


You can also create 'variables' which hold data and you can print those as well.

Tasks

- Change the 'colour' variable so that it shows your favourite colour.
- Print the sentence "My favourite colour is". Use the 'print all' block, or one print statement and a comma.

Solution



```
1 colour = 'orange'
2 print('My favourite colour is', colour)
3
```



You can add two strings together in Python using ' + '. This will create a new string, which consists of the two strings joined.

Tasks

- Edit the code so that your first name and last are correct. If you run this code, you should notice that there is a space missing in between. Change the print statement so that it prints correctly, with a space in between.

Solution



```
1 print('Rebecca '+'Summer')
2
```



Stage 7- Multiplying a string by a number

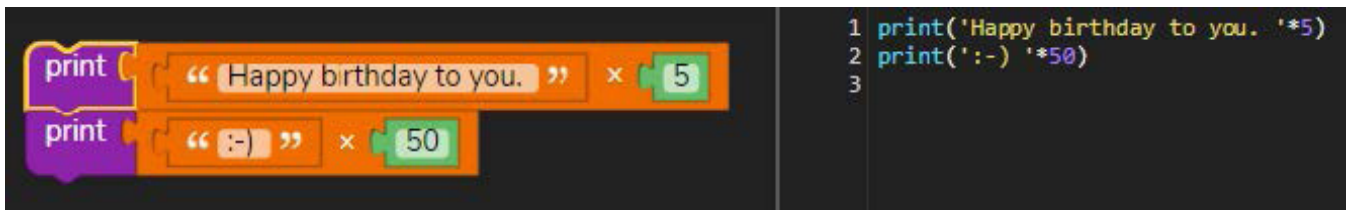
Lesson 1: Text Output

You can repeat strings using the multiplication operator (*).
For example, if you multiply the string “apple” by 3 you will get a new string, “appleappleapple”

Tasks

- Print “Happy birthday to you. “ five times using string multiplication. There should be a space after each full stop.
-
- A smiley face looks like this :-). Print a row of 50 smiley faces using string multiplication.

Solution



The image shows two parts of the solution. On the left, there are Scratch code blocks: a purple 'print' block with the text "Happy birthday to you. " followed by an orange multiplication block with the number 5, and another purple 'print' block with the text " :-)" followed by an orange multiplication block with the number 50. On the right, there is Python code:

```
1 print('Happy birthday to you. '*5)
2 print(':-)' *50)
3
```



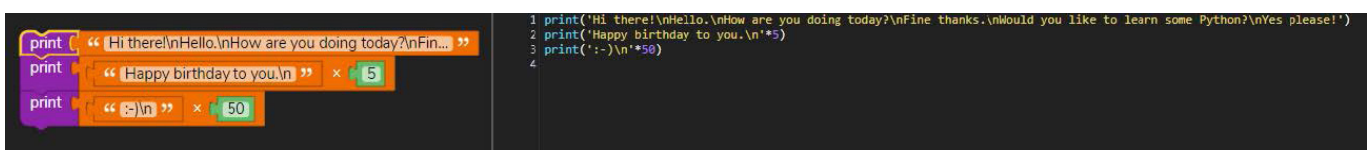
Stage 8- Line breaks

In Python you can use \n (slash followed by n) to mean ‘new line’.
This is useful when you want a string to be printed on multiple lines rather than all on one line.

Tasks

- Insert new line characters (\n) into the conversation so that each sentence is printed on a new line.
- Print “Happy birthday to you.” five times using string multiplication and the new line character. Each one should be on a separate line.
- Print fifty smiley faces, each one on a new line, using string multiplication and the new line character.

Solution



The image shows two parts of the solution. On the left, there are Scratch code blocks: a purple 'print' block with the text "Hi there!\nHello.\nHow are you doing today?\nFin...", another purple 'print' block with the text "Happy birthday to you.\n" followed by an orange multiplication block with the number 5, and a third purple 'print' block with the text " :-)\n" followed by an orange multiplication block with the number 50. On the right, there is Python code:

```
1 print('Hi there!\nHello.\nHow are you doing today?\nFine thanks.\nWould you like to learn some Python?\nYes please!')
2 print('Happy birthday to you.\n'*5)
3 print(':-)\n'*50)
4
```



Another way to output strings is to add a speech-bubble to an on-screen image (called a sprite) so that it looks like they are talking.

Someone has been making a historical scene using Python, but it isn't working yet.

Tasks

- Go to design mode and find the man called 'jester'. Fix the code so that jester says "Welcome" plus "your majesty". Use the ' + ' operator.
- Find the woman called 'poet'. Fix the code so that the poet says the poem correctly, with new line characters between each line.
- Fix the code so that the bored king says "zzzzzzzzzzzz". He should say "z" twelve times.

Solution

```
1 jester.say("Welcome" + "your majesty", 10)
2 poet.say("A Python ate my homework\nI promise you it's true\nit ate all my code and all my blocks\nit didn't even chew.", 10)
3 king.say("z" * 12, 10)
```



Here are some ideas for how you can improve your program

Suggestions

- Add some more characters and give them names.
- Make your characters say different things to each other. Try to use the ' + ' and ' * ' operators and the new line character (\n).
- Find the sleep block - this makes the code sleep for a given number of seconds. Use 'sleep' commands to simulate a conversation with pauses in.

Lesson 2- Numbers and Calculations

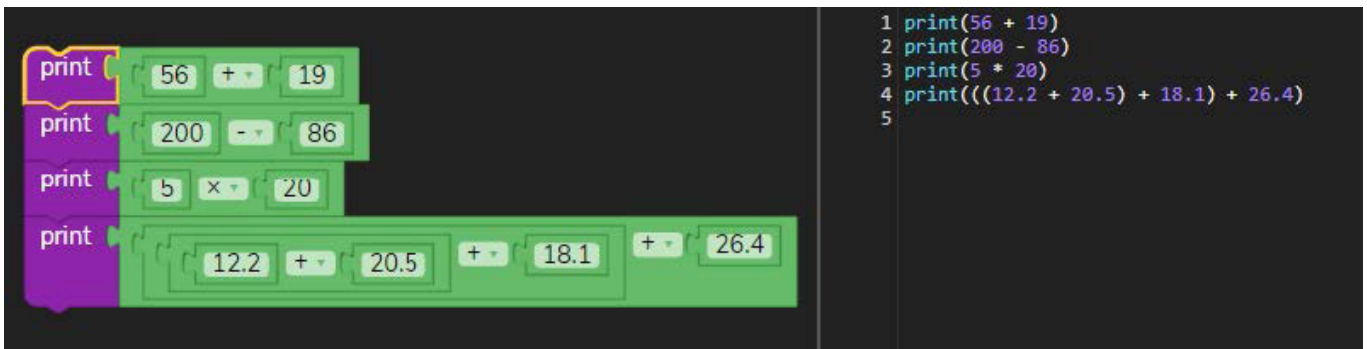
Stage 1- The number block

In Python you can use many different types of data. One of the most useful types is a 'Number'. Let's try some simple calculations using numbers. We'll print the results using the 'print' function.

Tasks

- Find the correct blocks to print $56 + 19$.
- Find the correct blocks to print $200 - 86$.
- Print 5×20 . Notice that in Python you do multiplication using an asterisk ($*$) instead of \times .
- You can position blocks inside other blocks to do longer calculations. Print the sum of 12.2, 20.5, 18.1 and 26.4. Use three $+$ operations.

Solution



The solution for Stage 1 is shown in two parts. On the left, Scratch blocks are used to print the results of four calculations: $56 + 19$, $200 - 86$, 5×20 , and $12.2 + 20.5 + 18.1 + 26.4$. On the right, the equivalent Python code is shown:

```
1 print(56 + 19)
2 print(200 - 86)
3 print(5 * 20)
4 print(((12.2 + 20.5) + 18.1) + 26.4)
5
```

Stage 2- Debug

You went shopping with £30. You spent £5.60 on food and £11.05 on a book. How much is left? Someone has written this code to find out how much is left, but it isn't correct now. Fix the bugs to find out how much money is left.

Tasks

- You started with £30. You spent £5.60 on food and £11.05 on a book. Print how much is left by fixing the code.

Solution



The solution for Stage 2 is shown in two parts. On the left, a Scratch block is used to print the result of $30 - 5.6 - 11.05$. On the right, the equivalent Python code is shown:

```
1 print((30 - 5.6) - 11.05)
2
```




Stage 3- More complex calculations

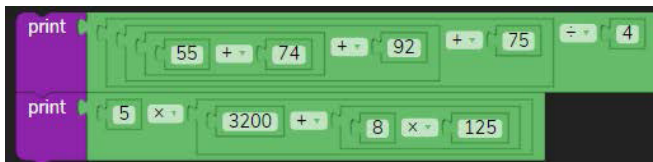
In Python we often must be careful about how we use brackets within a calculation in order to do the operations in the right order.

Have a go at these examples which use brackets.

Tasks

- Four students took an exam. Their scores were 55, 74, 92 and 75. Print the average score. Hint - to find the average you need to add all the numbers together and then divide the total by 4.
- Five trucks are to be transported on a ship. Each one weighs 3200kg and comes with 8 tyres which weigh 125kg each. Print the total weight.

Solution



```

print 55 + 74 + 92 + 75 ÷ 4
print 5 * 3200 + 8 * 125

```

```

1 print((55 + 74 + 92 + 75) / 4)
2 print(5 * (3200 + (8 * 125)))
3

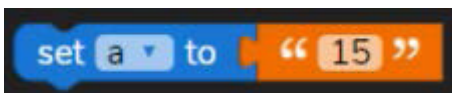
```



Stage 4- More complex calculations

You can convert between strings and numbers in Python.

For example, in this code 'a' is a string. You can tell it is a string because it is surrounded by quotation marks.

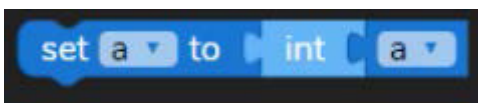


```

set a to "15"

```

You can convert it to the number 2 by using the 'int' function.



```

set a to int a

```

Tasks

- Use the 'int' block to convert 'a' to an integer. You can do this using the blocks 'set a to int a' or you can write the code 'a = int(a)'. Check that 'a' appears in the debug panel (bottom-right) without quotation marks.

Solution



```

set a to int "15"

```

```

1 a = int("15")
2

```



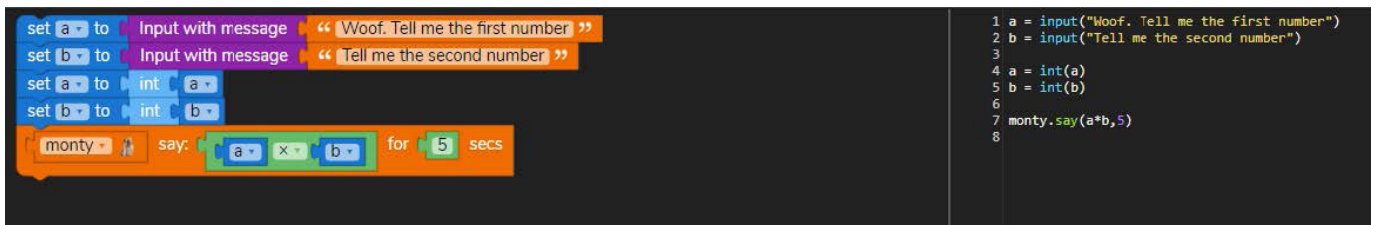
In Python you can use the 'input' function to ask the user to enter some data. You always get the result as a string, so if you want to do calculations you need to convert it to an integer.

Monty the Magnificent Multiplying Mutt will help us.

Tasks

- Use the 'int' block to convert 'a' to an integer. You can do this using the blocks 'set a to int a' or you can write Python 'a = int(a).'
- Use the 'int' block to convert 'b' to an integer.
- Go into design mode and find the dog called 'Monty'. Make Monty multiply the two numbers and say the answer for 5 seconds. Use the 'say' block or write the code 'monty.say(a * b)'. Test your code by making him multiply 15 by 13.

Solution



The solution shows a Scratch script with the following blocks: 'Input with message' (twice), 'int' blocks (twice), and a 'say' block with a multiplication block and a '5' seconds block. The corresponding Python code is:

```
1 a = input("Woof. Tell me the first number")
2 b = input("Tell me the second number")
3
4 a = int(a)
5 b = int(b)
6
7 monty.say(a*b,5)
8
```



In Python you write "5 squared" as $5^{**}2$. This means 5 to the power 2, or 5×5 .

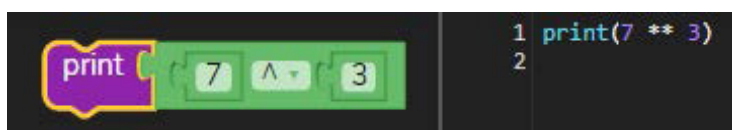
You write "5 cubed" or "5 to the power 3" as $5^{**}3$. This is $5 \times 5 \times 5$.

You write "5 to the power 4" as $5^{**}4$, which is $5 \times 5 \times 5 \times 5$.

Tasks

- Use the ** operator to print the volume of a cube which has a side length of 7cm. Hint - to find the volume you need to find 7 cubed.

Solution



The solution shows a Scratch 'print' block with a '7' block, a '^' block, and a '3' block. The corresponding Python code is:

```
1 print(7 ** 3)
2
```

In Python you write slash (/) instead of ÷ to do a division.

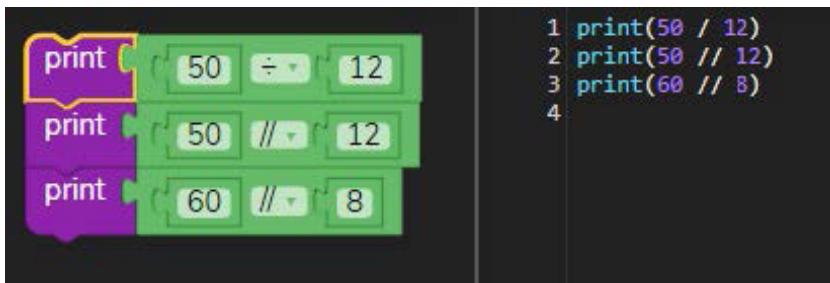
Python will always give you the result of a division as a decimal.

You can use the “integer division” operator to tell it to round down and give you the answer as a whole number.

Tasks

- Print $50 \div 12$ using the normal division operator (/).
- Print $50 \div 12$ using the integer division operator (//). This will tell you how many times 12 goes into 50.
- A truck needs 8 tyres. If you have 60 tyres how many trucks can you fully equip with tyres? Solve this problem using the integer division operator

Solution



The image shows a Scratch script on the left and Python code on the right. The Scratch script consists of three 'print' blocks: the first contains '50 ÷ 12', the second contains '50 // 12', and the third contains '60 // 8'. The Python code on the right is:

```
1 print(50 / 12)
2 print(50 // 12)
3 print(60 // 8)
4
```

Stage 8- The modulo operator (%)

The modulo operator gives you the remainder from a division calculation.

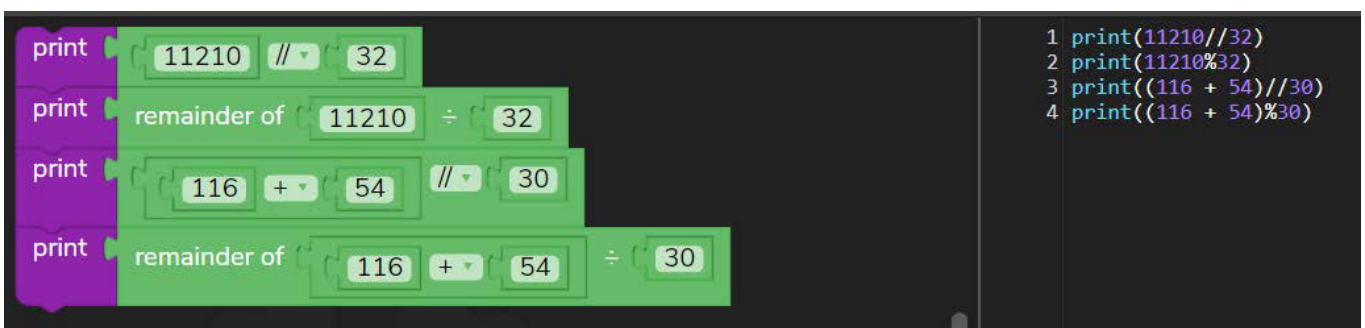
For example, in the previous stage you have 60 tyres and each truck needs 8. You can equip 7 trucks, with 4 tyres left over.

The modulo operator will give you the remainder:

Tasks

- Medication packs contain 32 tablets. A factory has made 11210 tablets. How many complete packs can it make? Print the answer using the // operator.
- How many are left over? Print the answer using the % operator.
- A trick or treat bag contains 116 sweets. Someone adds 54 more. The sweets are then shared equally between 30 children. Use + and // to print how many each child gets.
- How many are left over? Print the answer using + and %.

Solution



The image shows a Scratch script on the left and Python code on the right. The Scratch script consists of four 'print' blocks: the first contains '11210 // 32', the second contains 'remainder of 11210 ÷ 32', the third contains '116 + 54 // 30', and the fourth contains 'remainder of 116 + 54 ÷ 30'. The Python code on the right is:

```
1 print(11210//32)
2 print(11210%32)
3 print((116 + 54)//30)
4 print((116 + 54)%30)
```



Sometimes you will need to convert a number to a string. For example, this code will not work because Python will not let you join a string with a number using the '+' operator. You must convert 'score' to a string using the 'str' function.

Tasks

- Fix the code and make it print the correct message. Use the 'str' block to convert 'score' to a string.

Solution

The solution in Scratch consists of the following blocks:

- set `n` to `int` Input with message "Woof. How many biscuits?"
- monty say: `str` remainder of `n` ÷ `20` + " biscuits for me." for `5` secs



Monty's biscuits come in packets of 20. Complete packets are sold in shops, but any leftover go straight to Monty for immediate munching.

Tasks

- The input statement will ask the user how many biscuits there are and save their answer in a variable called 'n'. First, convert this to an integer use the 'int' function.
- Monty's biscuits come in packets of 20 and any leftover go to Monty. Write code to make Monty say "... biscuits for me". Hint - you will need to use '%' to find the remainder and the 'str' function to convert it to a string.
- Test your code by saying that there are 92 biscuits. Monty should say "12 biscuits for me"

Solution

```

1 n = int(input("Woof. How many biscuits?"))
2 monty.say(str(n % 20)+" biscuits for me.",5)
3

```

The solution in Scratch consists of the following blocks:

- set `score` to `100`
- print "Your score is: " + `str` `score`

```

1 score = 100
2 print("Your score is: " + str(score))
3

```



Here are some ideas for how you can improve your program

Suggestions

- Make Monty say “You can fill packets and there are biscuits for me”.
- Change the code so that Monty performs other calculations - for example he could ask for three numbers and find their average.
- Add some more sprites to the screen and make them perform different calculations.

Lesson 3- Repeat Loops

Stage 1- Introducing a for loop

In Python you can use many different types of data. One of the most useful types is a 'Number'. Let's try some simple calculations using numbers. We'll print the results using the 'print' function.

Tasks

- This code prints "Hello" 10 times. Change the code so that it prints "Goodbye" 6 times instead.

Solution



```
for i in range 6
do print "Goodbye"
```

```
1 for i in range(6):
2     print("Goodbye")
3
```

Stage 2- Debug

Someone is trying to print a chequerboard pattern using Python. Fix the bugs in this code so that it prints the pattern correctly.

Tasks

- Fix the bugs in this code so that it prints a chequerboard pattern.

Solution



```
for i in range 4
do print "XO" x 8
   print "OX" x 8
```

```
1 for i in range(4):
2     print ("XO"*8)
3     print ("OX"*8)
```



Stage 3- Tick tock

Lesson 3- Repeat loops

Python can execute your code so fast that you can't see what is happening. You can make it 'sleep' using a command like 'sleep(1)' - which will make it pause for 1 second.


The 'sleep' command is a special one - it is part of an extra code library called 'time' and you need to import it before you can use it.

We can use the sleep command to make our programs more realistic.

Tasks

- Import the 'sleep' function from the 'time' module. This needs to go at the top of your program.
- Create a 'for i in range' loop which will repeat 8 times.
- Inside the loop print the word "tick" and the make the code sleep for 1 second.
- Next, print the word "tock" and make the code sleep for 1 second again.
- Run your code and check that it prints tick/tock at 1 second intervals.

Solution



```

1 from time import sleep
2
3 for i in range(8):
4     print("tick")
5     sleep(1)
6     print("tock")
7     sleep(1)
8

```



Stage 4- Playing a sound

We can use the "play sound" block to play a sound effect.

Tasks

- When your program prints 'tick', play the sound effect 'sfx/tick.mp3'. You can find this in the 'sfx' folder.
- When your program prints 'tock', play the sound effect 'sfx/tock.mp3'.
- Run your code and check that it prints tick/tock at 1 second intervals and plays the sound effects.

Solution



```

1 from time import sleep
2
3 for i in range(8):
4     print("tick")
5     pip.play_sound("sfx/tick.mp3",1)
6     sleep(1)
7     print("tock")
8     pip.play_sound("sfx/tock.mp3",1)
9     sleep(1)
10

```



Stage 5- Adding a clock

Lesson 3- Repeat loops

In design mode you'll find a clock and another sprite called 'pendulum'. To make it look like it is moving we can rotate the pendulum every second.

Tasks

- Go into design mode and find the sprite called 'pendulum'. When you play the tick sound, set the pendulum's angle to 20 degrees
- When you play the tock sound, set the pendulum's angle to -20 degrees
- Run your code and check that it prints tick/tock at 1 second intervals, plays the sound effects and that the pendulum moves.

Solution

The image shows a Scratch script on the left and its corresponding Python code on the right. The Scratch script starts with a 'from time import sleep' block. It then enters a 'for i in range 8' loop. Inside the loop, it prints 'tick', plays a sound 1 time, sets the pendulum's angle to 20, sleeps for 1 second, prints 'tock', plays a sound 1 time, sets the pendulum's angle to -20, and sleeps for 1 second. The Python code on the right is a direct translation of the Scratch script.

```

1 from time import sleep
2
3 for i in range(8):
4     print("tick")
5     pip.play_sound("sfx/tick.mp3",1)
6     pendulum.angle = 20
7     sleep(1)
8     print("tock")
9     pip.play_sound("sfx/tock.mp3",1)
10    pendulum.angle = -20
11    sleep(1)
12
13
14

```



Stage 6- What is a range?

The 'range' keyword actually returns a special Python object that represents a sequence of numbers. 'range(10)' gives you a sequence of ten numbers starting from zero. Notice that the number 10 is not included. The range stops before the number 10 is reached.

Tasks

- Create a 'for i in range(10)' loop. Inside the loop, print the value of 'i' each time. Check that it prints 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Create a 'for j in range(N)' loop for which 'j' takes the values 0 to 6. What is the right value of 'N'? Inside the loop, print the value of 'j' each time. Check that it prints 0, 1, 2, 3, 4, 5, 6.

Solution

The image shows a Scratch script on the left and its corresponding Python code on the right. The Scratch script has two 'for' loops. The first is 'for i in range 10' with a 'print i' block inside. The second is 'for j in range 7' with a 'print j' block inside. The Python code on the right is a direct translation of the Scratch script.

```

1 for i in range(10):
2     print(i)
3 for j in range(7):
4     print(j)
5

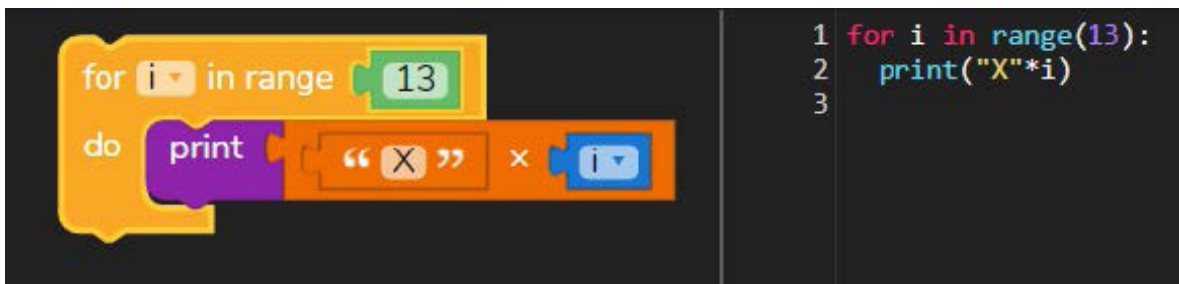
```


If we combine the loop variable it with printing, we can print some interesting patterns.

Tasks

- Create a 'for i in range()' loop where 'i' takes the values 0 up to 12. Inside, print the string "X" multiplied by 'i'.

Solution



The image shows two representations of the same code. On the left is a Scratch code block: a yellow 'for i in range' block with '13' in the field, followed by a 'do' block containing a purple 'print' block with 'X' in quotes, multiplied by 'i'. On the right is the equivalent Python code:

```
1 for i in range(13):
2     print("X"*i)
3
```

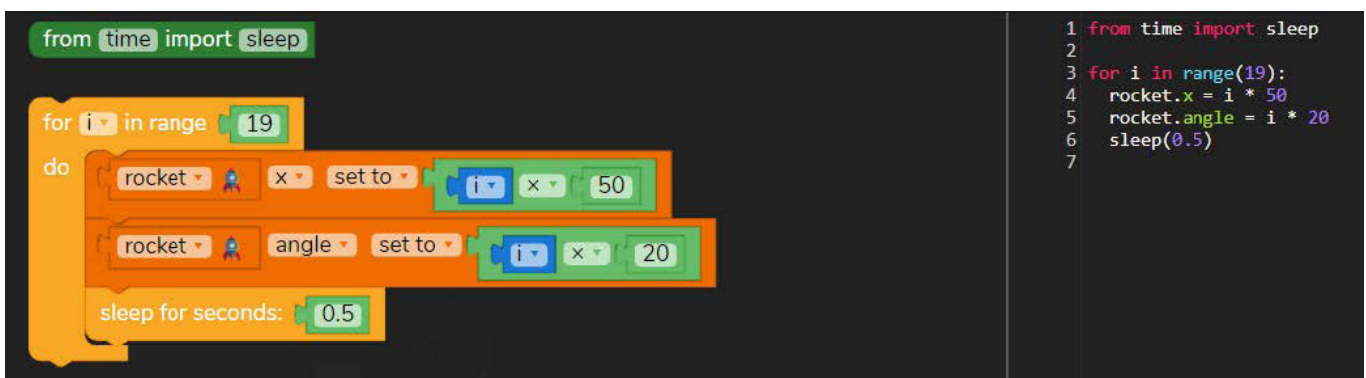
Stage 8- Movement

We can also use the variable to move objects around. Let's make the rocket move and rotate.

Tasks

- Import the 'sleep' function from the 'time' module. This needs to go at the top of your program.
- Make a 'for i in range' loop for which 'i' takes the values from 0 to 18.
- Inside the loop set the 'x' position of the rocket to $50 \times i$.
- And set the 'angle' of the rocket to $20 \times i$.
- Finally, sleep for half a second so that you can see the rocket moving.

Solution



The image shows two representations of the same code. On the left is a Scratch code block: a green 'from time import sleep' block, followed by a yellow 'for i in range' block with '19' in the field. The 'do' block contains three orange blocks: 'rocket x set to i * 50', 'rocket angle set to i * 20', and 'sleep for seconds: 0.5'. On the right is the equivalent Python code:

```
1 from time import sleep
2
3 for i in range(19):
4     rocket.x = i * 50
5     rocket.angle = i * 20
6     sleep(0.5)
7
```



Here are some ideas for how you can improve your program.

Suggestions

- Give the rocket a speech bubble which says the current value of 'i'.
- Make the rocket move back to the left when it has moved fully to the right.
- Add some more sprites to your scene and use 'for i in range' loops to make them move in different directions.

Lesson 4- Moving Sprites



Stage1- What is a sprite

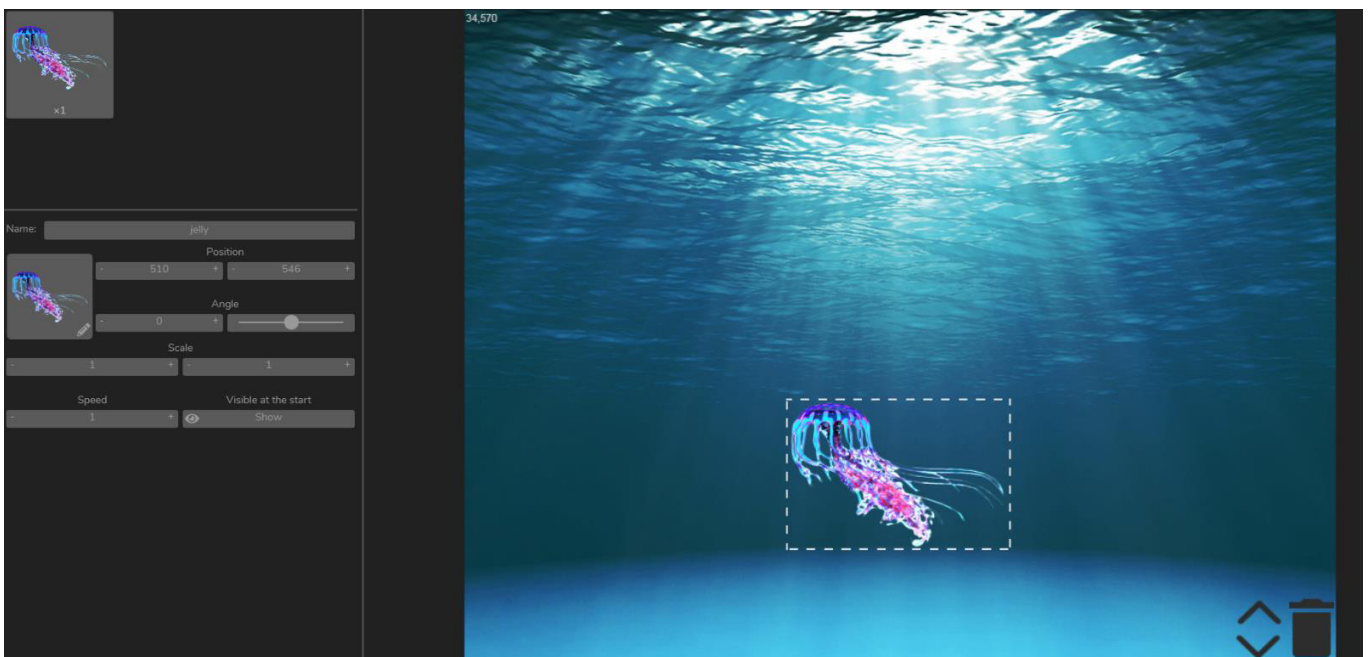
Your Python in Pieces activities can contain a background and several sprites, shapes, text fields and buttons.

A sprite is a two-dimensional picture that can be controlled by code. You can move it around and change what it looks like.

Tasks

- Go into design mode, click on the background and then on 'Choose image'. Change the background image to the image of underneath the sea.
- Drag a jellyfish onto the screen.
- We need to give it a name so that we can control it using code. Call your jellyfish 'jelly'. Position it in the bottom half of the screen, in the middle.

Solution





Stage 2- Positioning the jellyfish

You can set the position of a sprite exactly by defining its 'x' and 'y' properties. You can use blocks or Python in order to do this.

Tasks

- Set the 'x' position of the jellyfish to 540
- Set the 'y' position to 670. When you run your code, the jellyfish should jump to exactly the correct position.

Solution



The image shows two ways to set the position of a sprite named 'jelly'. On the left, there are two Scratch 'set to' blocks: one for 'x' set to 540 and one for 'y' set to 670. On the right, the equivalent Python code is shown:

```
1 jelly.x = 540
2 jelly.y = 670
3
```



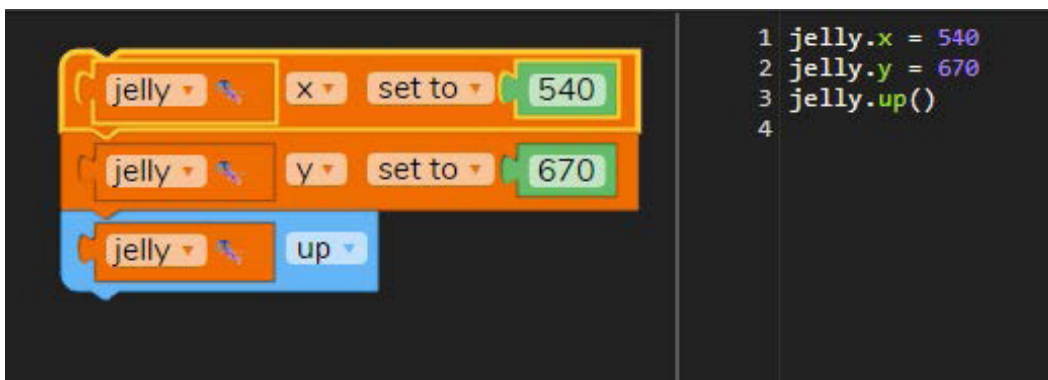
Stage 3- Animate the jellyfish

Let's make the jellyfish move upwards. You can use blocks or Python in order to do this.

Tasks

- Make the jellyfish move upwards

Solution



The image shows how to animate the jellyfish. On the left, there are three Scratch blocks: two 'set to' blocks for 'x' (540) and 'y' (670), and a blue 'up' block. On the right, the equivalent Python code is shown:

```
1 jelly.x = 540
2 jelly.y = 670
3 jelly.up()
4
```



Let's make the jellyfish move up and down.

Tasks

- First, import the 'sleep' module from the 'time' package. This needs to go at the top of your program.
- Use a 'sleep' command to make the program sleep for four seconds while the jellyfish moves upwards.
- After four seconds, make the jellyfish move downwards.

Solution



```
from time import sleep
jelly x set to 540
jelly y set to 670
jelly up
sleep for seconds: 4
jelly down
```

```
1 from time import sleep
2
3 jelly.x = 540
4 jelly.y = 670
5 jelly.up()
6 sleep(4)
7 jelly.down()
8
```



Stage 5- Moving left and right

Let's make our ocean scene more interesting.

Tasks

- Drag another sprite into the scene. Click on its thumbnail and change it to a shark swimming towards the right.
- Position it on the far left of the scene and name it 'shark'. Change its speed to '10' so that it moves very fast.
- At the end of your program, after the code that moves the jellyfish, add code to make the shark move right and left 3 times (right, left, right, left, right, left) with two seconds between each movement.

Solution

The screenshot shows the Scratch code editor with a sequence of blocks for moving a jellyfish and a shark. The code is as follows:

```
from time import sleep

jelly.x set to 540
jelly.y set to 670
jelly up
sleep for seconds: 4
jelly down
shark right
sleep for seconds: 2
shark left
sleep for seconds: 2
shark right
sleep for seconds: 2
shark left
sleep for seconds: 2
shark right
sleep for seconds: 2
shark left
```



Sharks don't swim backwards. Let's make our shark swim forwards.

Tasks

- Before the shark moves left, set its 'scaleX' property to -1. This will flip the shark image horizontally, so it faces left. You can do this using the 'property set to' block or by writing Python code 'shark.scaleX = -1'.
- When it moves right, set its 'scaleX' property back to 1. Run your program and check that the shark looks more realistic now.

Solution



```
from time import sleep
jelly.x set to 540
jelly.y set to 670
jelly.up
sleep for seconds: 4
jelly.down
shark.right
sleep for seconds: 2
shark.scaleX set to -1
shark.left
sleep for seconds: 2
shark.scaleX set to 1
shark.right
sleep for seconds: 2
shark.scaleX set to -1
shark.left
sleep for seconds: 2
shark.scaleX set to 1
shark.right
sleep for seconds: 2
shark.scaleX set to -1
shark.left
```

```
1 from time import sleep
2
3 jelly.x = 540
4 jelly.y = 670
5 jelly.up()
6 sleep(4)
7 jelly.down()
8 shark.right()
9 sleep(2)
10 shark.scaleX = -1
11 shark.left()
12 sleep(2)
13 shark.scaleX = 1
14 shark.right()
15 sleep(2)
16 shark.scaleX = -1
17 shark.left()
18 sleep(2)
19 shark.scaleX = 1
20 shark.right()
21 sleep(2)
22 shark.scaleX = -1
23 shark.left()
```



You can change a sprite's image using the 'set image to' block, or the 'set_image' method in Python.

Tasks

- At the end of your program, make the shark stop moving. Use the 'shark.stop()' method.
- Set its 'x' position to 500 and its 'y' position to 400, now it will be roughly in the middle of the screen. Set both scaleX and scaleY to 2 to make it bigger.
- Change the image of the shark so that it looks like it is attacking. You can use the 'set image to' block, or in Python you can use the 'shark.set_image' method. Find the image 'nature/shark2.png' and use that for your new image.
- Run your program. Check that the shark attacks after swimming back and forth.

Solution



```
1 from time import sleep
2
3 jelly.x = 540
4 jelly.y = 670
5 jelly.up()
6 sleep(4)
7 jelly.down()
8 shark.right()
9 sleep(2)
10 shark.scaleX = -1
11 shark.left()
12 sleep(2)
13 shark.scaleX = 1
14 shark.right()
15 sleep(2)
16 shark.scaleX = -1
17 shark.left()
18 sleep(2)
19 shark.scaleX = 1
20 shark.right()
21 sleep(2)
22 shark.scaleX = -1
23 shark.left()
24 shark.stop()
25 shark.x = 500
26 shark.y = 400
27 shark.scaleX = 2
28 shark.scaleY = 2
29 shark.set_image("nature/shark2.png")
```




Someone has made a desert scene with two aeroplanes in, but it's not working. The green aeroplane is supposed to fly left for 3 seconds and right for 3 seconds twice (left, right, left, right) The red aeroplane is supposed to fly right for 3 seconds and left for 3 seconds twice (right, left, right, left)

Tasks

- Fix the code so that both aeroplanes fly back and forth twice, with a 3 second pause between each movement. Use 'scaleX' to make sure that they don't fly backwards.

Solution

```
1 from time import sleep
2 green.MoveLeft()
3 red.MoveRight()
4 green.MoveLeft()
5 red.MoveRight()
6 green.MoveLeft()
7 red.MoveRight()
8 green.MoveLeft()
9 red.MoveRight()
```



Challenge

Suggestions

- Add some more sea creatures to your scene.
- Make them swim up/down/left and right
- Practice using scaleX and scaleY to make them look realistic
- Make some of them swim at the same time as others

Lesson 5- Variables and user input

Stage 1- Creating a variable

In Python we use variables as storage locations for data. Two of the most common types of data we want to store are:

- Numbers (whole numbers or decimals)
- Text (called 'strings' in Python)

Let's make a variable and store some data in it. This process is called 'assigning' a value to a variable.

Tasks

- Create a variable called 'age' and assign it your age in years. For example, if you are twelve years old, write 'age = 12'. The 'type' of this variable is 'Number' because the value it stores is a number.
- Check that you can see the value of your variable in the debug panel (bottom-right)

Solution



The image shows a Scratch code block on the left that says "set age to 12". On the right, a Python code editor shows the equivalent code:

```
1 age = 12
2
```

Stage 2- Creating more variables

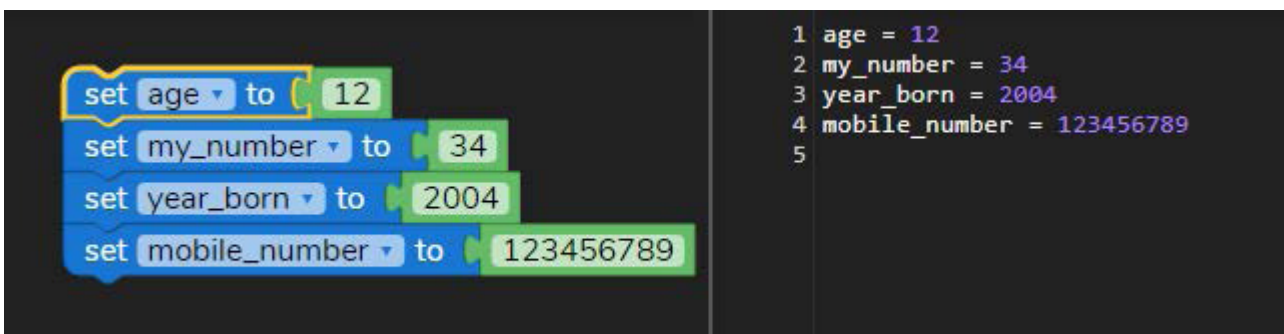
It is important to choose good names for your variables so that other people can understand your program.

Let's make some more variables and give them sensible names.

Tasks

- Think of a whole number between 1 and 50. Create another variable called 'my_number' and assign it the number you thought of.
- Using lower-case words separated by _ (underscore) is a method that is often used to make variable names that are easy to read. Make a few more variables using this system and give them numerical values. Try to give them meaningful names that are easy to read and understand.

Solution



The image shows four Scratch code blocks on the left: "set age to 12", "set my_number to 34", "set year_born to 2004", and "set mobile_number to 123456789". On the right, a Python code editor shows the equivalent code:

```
1 age = 12
2 my_number = 34
3 year_born = 2004
4 mobile_number = 123456789
5
```



Lower-case words separated by underscores is called “snake-case” in Python because your variable names can end up looking like a long snake.

Camel-case variable names start with a lower-case letter, and each subsequent word has a capital letter - for example, ‘myNumber’, ‘weightOfSugar’. They are supposed to look a bit like the humps of a camel.

Tasks

- Make a variable called ‘weightOfSugar’ and assign it any numerical value.
- Make a few more variables using the camel-case convention and assign them numerical values.
- Check that you can see the values of all your variables in the debug panel (bottom-right)

Solution

```
1 age = 12
2 year_born = 2004
3 mobile_number = 123456789
4 weightOfSugar = 256.9
5 numberOfClasses = 6
6 heightOfHeadteacher = 3.45
7 clevernessOfCoders = 1000000
8 nextWeeksLotteryNumbers = 110426333645
```



In Python variables can be re-assigned new values at any time.

In real life, some variables have a fixed value (for example the number of days in a month doesn't change) and should not be changed. Python has a way to handle this too.

Tasks

- Add another line of code that changes the value of ‘my_number’ to a whole number between 100 and 200.
- Create a variable called ‘NUM_DAYS_IN_JANUARY’ and assign it the correct value. In Python we often use capital letters to indicate that a variable has a fixed value which another programmer shouldn't change later in the code.

Solution

```
1 age = 12
2 year_born = 2004
3 my_number = 45
4 weightOfSugar = 256.9
5 numberOfClasses = 6
6 heightOfHeadteacher = 3.45
7 clevernessOfCoders = 1000000
8 nextWeeksLotteryNumbers = 110426333645
9 my_number = 167
10 NUM_DAYS_IN_JANUARY = 31
11
```



Another type of value in Python is 'String' - which we can use to represent text. In Python, strings must start and end with quotation marks. You can use double or single quotation marks.

Tasks

- Create a variable called 'welcome_message' and assign it the value "Hello". Remember, in Python strings have to start and end with quotation marks.
- Create a variable called 'favFood' and assign it your favourite food. For example, if you like cake, write favFood = "cake".
- Make a few more variables and assign them string values. Give them meaningful names. Check that you can see the values of all your variables in the debug panel (bottom-right)

Solution

The image shows a code editor with two panels. The left panel displays block-based code with 'set' blocks for each variable. The right panel displays the equivalent Python text code.

```
1 age = 12
2 year_born = 2004
3 my_number = 45
4 weightOfSugar = 256.9
5 nextWeeksLotteryNumbers = 110426333645
6 my_number = 167
7 NUM_DAYS_IN_JANUARY = 31
8 welcome_message = 'Hello'
9 favFood = 'scrambled snake'
10 bestSubject = 'Computing'
11 name = 'Gruffalo'
12
```



Another way to assign a value to a variable is to ask the user to give you some input. We can use the 'input' function in Python to ask the user to input a string.

Tasks

- Ask the user what their name is by using an input statement that says, "What is your name?". You can use the 'input with message' block or write the code 'input("What is your name?")'.
- Assign their response to a variable called 'user_name'.
- Print the value of the variable 'user_name' using the 'print' function.
- Run your code and enter your name. You should be able to see the value of your variable in the debug panel (bottom-right) and printed in the output panel (bottom-left)

Solution

```

1 age = 12
2 my_number = 45
3 year_born = 2004
4 weightOfSugar = 256.9
5 nextWeeksLotteryNumbers = 110426333645
6 my_number = 167
7 NUM_DAYS_IN_JANUARY = 31
8 welcome_message = 'Hello'
9 favFood = 'scrambled snake'
10 bestSubject = 'Computing'
11 name = 'Gruffalo'
12 user_name = input('What is your name?')
13 print(user_name)
14

```



Look at this code - it has quite a few problems. It is supposed to ask for the user's favourite hobby and then print it. The first problem is that it doesn't work - it has some 'syntax errors' which you must correct.

The second problem is that the variable is named very badly. Change the variable name to a better one. You can change this in the blocks by clicking on the variable and selecting 'Rename the variable' or you can edit the Python code.

Tasks

- Fix the errors in this code so that it works. It should ask for your favourite hobby, store the response in a variable and then print the variable.
- Change the variable name to a better one. Decide yourself whether to use snake-case or camel-case.
- Run the code and check that it works.

Solution

```

1 UsersFaveHobby=input('What is your favourite hobby?')
2 print(UsersFaveHobby)
3
4
5
6 Users_Fave_Hobby=input('What is your favourite hobby?')
7 print(Users_Fave_Hobby)

```



So far, we have seen the data types 'Number' and 'String'. Another kind of variable is a 'Text field', an editable piece of text that appears on the screen.

Let's make a general knowledge quiz about a famous building, using 'Text field' and 'String' variables.

Tasks

- Go into design mode and find the three blue text fields for the answers. They are badly named at the moment. Rename them using sensible names - these are the variable names that you must use in your code.
- Ask the user the first question, "The Eiffel tower is the tallest building in which city?". Use the 'input' function. Store the response in a sensibly named variable.
- Make the first blue text field show the value of the variable. You can use the 'text set to' block, or in Python use (for example) `textfield.text = answer`.
- Ask the user the second and third questions and save the responses in two variables. Make the other two text fields show the value of the other two variables.
- Run your program. Do some research and see if you can find the answers.

Solution

The image shows a Scratch script on the left and the corresponding Python code on the right. The Scratch script consists of three 'Input with message' blocks followed by 'set to' blocks for three text fields named 'answer1', 'answer2', and 'answer3'. The Python code implements the same logic: it asks three questions, stores the answers in variables, and then sets the text of three text fields to the corresponding variables.

```

1 answer1 = input('The Eiffel tower is the tallest building in which city?')
2 txtAnswer1.text = answer1
3 answer2 = input('In what year did its construction start?')
4 txtAnswer2.text = answer2
5 answer3 = input('How high is it in metres?')
6 txtAnswer3.text = answer3

```



Challenge

Have a go at these challenges to improve your quiz.

Suggestions

- Allow the user to enter the questions as well as the answers.
- Store the correct answers in variables in your code. You could make these constants because they shouldn't change
- See if you can use 'if' statements to check if the user has answered correctly.

Lesson 6 – Repeat while

Stage 1- while loop

In Python, a 'while' loop looks like this:

```
1 while <test>
2   <statements>
```

The 'test' can be any expression and while it is true the statements will execute repeatedly. If you write 'while True': then the loop will execute forever because the test is always true.

Tasks

- Add a 'while True' loop.
- Inside the loop, print("Hello")
- Check that it prints 'Hello' forever, until you click STOP in the top menu bar.

Solution



The image shows a Scratch script on the left and its Python equivalent on the right. The Scratch script starts with a 'repeat while' block where the condition is 'True'. Inside the loop, there is a 'print' block with the text 'Hello'. The Python code on the right is:

```
1 while True:
2   print("Hello")
3
```

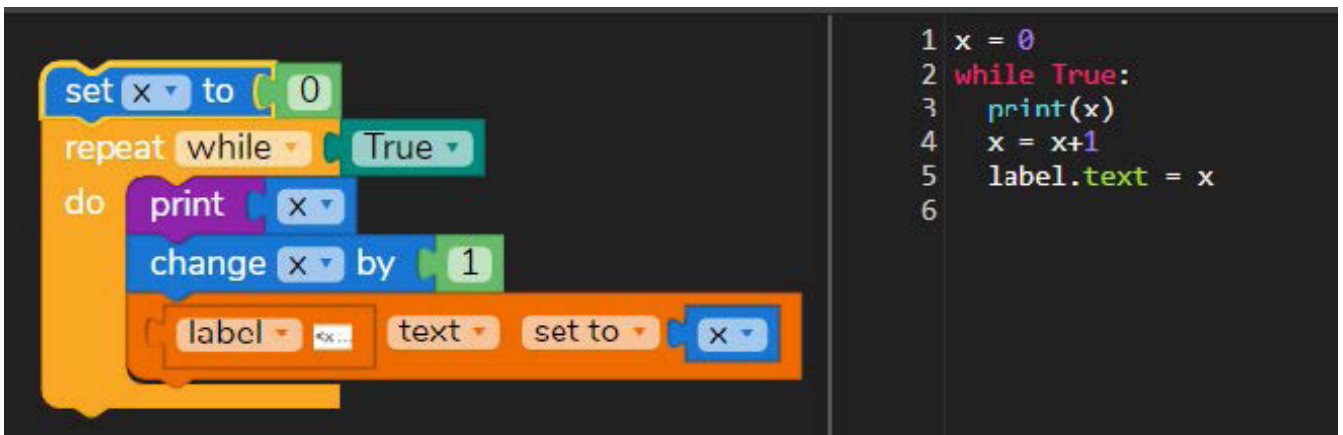
Stage 2- Changing a variable

We can do more than inside a while loop than just printing. Let's change the value of a variable.

Tasks

- After the 'print' statement, write code to increase 'x' by one. You can use the 'change variable by' block, or in Python, you can write 'x = x + 1'.
- Go into design mode and find the text field called 'label'. Set the 'text' property of label to 'x'. You can use the 'text set to' block or in Python you can write 'label.text = x'.
- Run your code and check that the text field shows 0, 1, 2, 3, 4, 5, ... etc

Solution



The image shows a Scratch script on the left and its Python equivalent on the right. The Scratch script starts with a 'set x to' block with the value 0. It then enters a 'repeat while' loop with the condition 'True'. Inside the loop, there are three blocks: 'print x', 'change x by 1', and 'label text set to x'. The Python code on the right is:

```
1 x = 0
2 while True:
3   print(x)
4   x = x+1
5   label.text = x
6
```




Stage 3- Using an 'if' statement

An 'if' statement is used for decision making. It tells the code to check a condition and to execute code if the condition is true.

Tasks

- Inside your loop, add an 'if' statement at the end. It should check if 'x' is greater than 100.
- If so, set the background colour to 'red'. You can use the 'set color to' block or write 'background.set_color("red")'
- Run your code and wait until 'x' is greater than 100. Check that the background goes red.

Solution



The image shows the Scratch code blocks on the left and the corresponding Python code on the right. The Scratch code starts with 'set x to 0', followed by a 'repeat while True' loop. Inside the loop, there is a 'print x' block, a 'change x by 1' block, a 'label text set to x' block, and an 'if x > 100' block. The 'if' block contains a 'background set background color to red' block. The Python code is as follows:

```

1 x = 0
2 while True:
3     print(x)
4     x = x+1
5     label.text = x
6     if x > 100:
7         background.set_bg_color("red")
8

```



Stage 4- Using an 'elif' statement

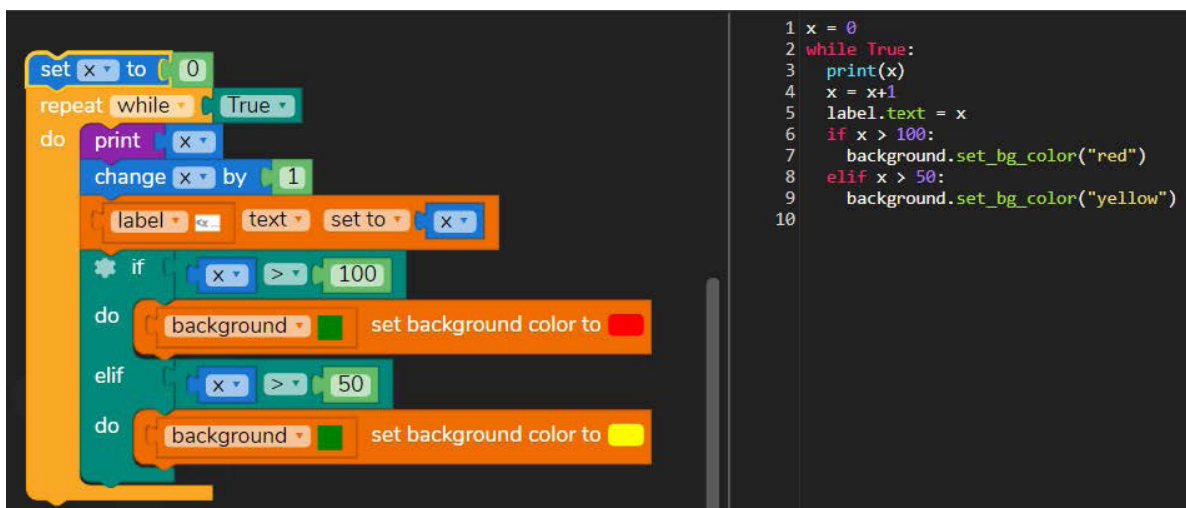
'if' is often followed by one or more 'elif' statements which let you check other conditions and execute different code when each condition is met.

Each condition will be tested one by one. The first one that passes will have its statements executed.

Tasks

- After the 'if' statement add an 'elif' statement. Inside, check if 'x' is greater than 50. If so, set the background colour to 'yellow'.
- Run your code. Check that the screen goes yellow when x goes above 50 and red when it goes above 100.

Solution



The image shows the Scratch code blocks on the left and the corresponding Python code on the right. The Scratch code is similar to Stage 3 but includes an 'elif x > 50' block with a 'background set background color to yellow' block. The Python code is as follows:

```

1 x = 0
2 while True:
3     print(x)
4     x = x+1
5     label.text = x
6     if x > 100:
7         background.set_bg_color("red")
8     elif x > 50:
9         background.set_bg_color("yellow")
10

```




The code in this stage contains a 'logical' error. It runs but it doesn't do what it is supposed to. It is supposed to say 'LOW' when 'x' is less than or equal to 50, 'MEDIUM' when x is greater than 50 and 'HIGH' when x is greater than 100.

At the moment it never says 'HIGH'. Why?

Tasks

- Fix the code. It should say 'LOW' when 'x' is less than or equal to 50, 'MEDIUM' when x is greater than 50 and 'HIGH' when x is greater than 100. At the moment it never says 'HIGH'
- Run your code, wait until 'x' is greater than 100 and check that it says 'HIGH'

Solution

The image shows the solution code in two formats: Scratch blocks and Python code.

Scratch Code:

- set x to 0
- repeat while True
- do
 - label text set to x
 - change x by 1
 - if x ≤ 50
 - do status text set to "LOW"
 - elif x > 100
 - do status text set to "HIGH"
 - elif x > 50
 - do status text set to "MEDIUM"

```
1 x = 0
2 while True:
3     label.text = x
4     x = x+1
5     if x <= 50:
6         status.text = "LOW"
7     elif x > 100:
8         status.text = "HIGH"
9     elif x > 50:
10        status.text = "MEDIUM"
11
```



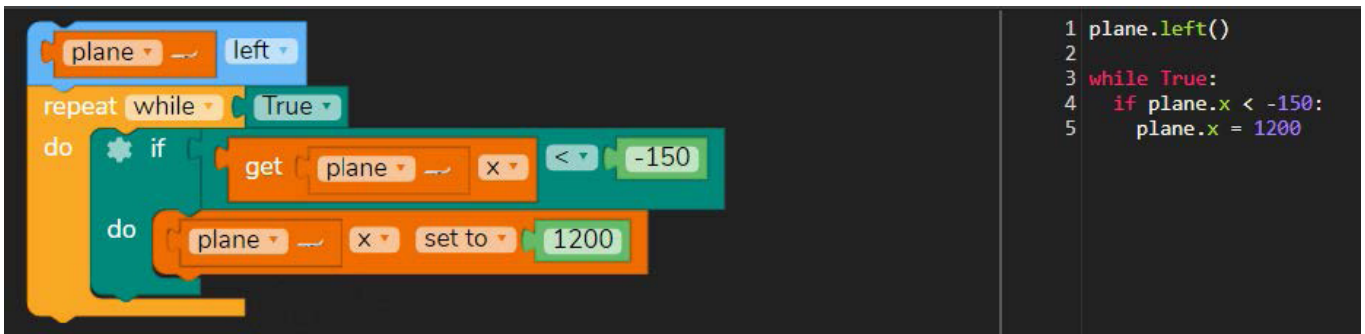
We can control sprites inside 'while' loops too - this is often used in games where it is called a 'game loop'. In design mode there is an aeroplane called 'plane'. There's only one plane but we can use a while loop to simulate a lot of planes flying overhead.

When the plane has moved far off the screen on the left, we can position it far off the screen to the right. Then it will look like a different plane.

Tasks

- Make the plane move left.
- Add a 'while True' loop. Inside check if the plane's 'x' position is less than -150 (minus 150, or 150 pixels off the screen to the left). If so, set the plane's 'x' position to 1200. Do not change its direction.
- Run your program and check that it looks like a lot of planes are flying overhead.

Solution



The image shows the solution in two parts: Scratch code blocks on the left and Python code on the right.

Scratch Code:

- A blue 'left' block with 'plane' as the target.
- An orange 'repeat while' block with 'True' as the condition.
- Inside the loop, a green 'if' block with a gear icon.
- The 'if' block's condition is 'get plane x < -150'.
- Inside the 'if' block, an orange 'do' block with 'plane x set to 1200'.

Python Code:

```
1 plane.left()
2
3 while True:
4     if plane.x < -150:
5         plane.x = 1200
```



We'll use an 'if' statement to check the car's position every frame. If it has gone too far right, we'll make it go back left.

Then we need to use an 'elif' statement to check if it has gone too far left - if so, we need to move it right.

Tasks

- At the top of your program, make the car move right.
- Inside the while loop, add another 'if' statement that will check if the car's 'x' position is greater than 1000. If so, make the car move left.
- Add an 'elif' statement to check if the car's 'x' position is less than 100. If so, make the car move right.
- Run your program and check that the car moves forwards and backwards forever. Click 'STOP' when you're ready.

Solution

The image shows a Scratch script on the left and its corresponding Python code on the right. The Scratch script starts with 'plane left' and 'car right' blocks. A 'repeat while True' loop contains an 'if' block: 'if get plane x < -150', followed by a 'do' block 'plane x set to 1200'. Another 'if' block: 'if get car x > 1000', followed by a 'do' block 'car left'. An 'elif' block: 'elif get car x < 100', followed by a 'do' block 'car right'. The Python code on the right is:

```
1 plane.left()
2 car.right()
3
4 while True:
5     if plane.x < -150:
6         plane.x = 1200
7     if car.x > 1000:
8         car.left()
9     elif car.x < 100:
10        car.right()
```



So far, we have only checked when the 'x' positions of our objects are greater or less than a number. We can also check against the positions of other objects.

This spooky scene is not working properly. The witch should move left, and when she reaches the spell book it should disappear, and she should move right.

Tasks

- Debug this code so that the witch moves towards the spell book. When she reaches it make it disappear and make her move right.

Solution

The solution is shown in two parts: Scratch code blocks and Python code.

Scratch Code:

- witch left
- repeat while True
- do if (get witch x) ≤ (get book x)
- do book hide
- witch right

Python Code:

```
1 witch.left()
2 while True:
3     if witch.x <= book.x:
4         book.hide()
5         witch.right()
6
```



The potion starts off as hidden. When she gets the spell book the potion should appear, and she should collect that as well. When she has both we need to break out of the while loop - its job is done.

To break out of a while loop without having to click the STOP button we can use the 'break' statement.

Tasks

- Inside your while loop add code so that when the book disappears, the potion is made visible.
- Add an 'elif' statement to check when she reaches the potion - when she does make it disappear.
- When she has the potion add a 'break' statement to break out of the while loop.
- Add code at the end of your program, outside of the while loop, make the witch fly upwards.

Solution

The image shows a Scratch script on the left and its equivalent Python code on the right. The Scratch script starts with a 'witch left' block, followed by a 'repeat while True' loop. Inside the loop, there is an 'if' block that checks if the witch's x-coordinate is less than the book's x-coordinate. If true, it performs 'book hide', 'potion show', and 'witch right'. An 'elif' block checks if the witch's x-coordinate is greater than the potion's x-coordinate. If true, it performs 'potion hide' and 'break out of loop'. After the loop, there is a 'witch up' block.

```
1 witch.left()
2 while True:
3     if witch.x < book.x:
4         book.hide()
5         potion.show()
6         witch.right()
7     elif witch.x > potion.x:
8         potion.hide()
9         break
10 witch.up()
```



Have a go at these challenges to improve your quiz

Suggestions

- Use the `scaleX` property to make the witch face the right direction when she is moving.
- Add some more objects to the scene and make them move inside the game loop. They could move left and right or up and down.
- Use 'if' and 'elif' statements to test the position of different objects to make them interact with each other.
- Use the 'random integer from' block or the 'random.randint' function to generate a random number. Make it so that your program does different things depending on the value of certain random numbers.