

Logical Thinking - Comparative Operators

==	Equal to
!=	Not equal to
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer division
%	Remainder
**	Exponent

<https://edublocks.org>



Trinity TV
For more help, visit Trinity TV and watch the following videos:
Trinity TV > Year 9 > Computer Science > Term 1

Key Terms

EduBlocks	A visual block based programming tool that helps to introduce text based programming languages.
Python	A text based programming language.
Programming Code	The process of writing computer programs. The instructions that you write to program a computer.
Algorithm	A set of rules / instructions.
Sequence	Parts of the code that run in order and the instructions for our code.
Selection	Using logical tests to change the flow of the sequence.
Iteration	Using loops to repeat sequences of code. Code is repeated (looped) while something is true or for a number of times.
Variable	A value that can be changed (speed, lives, score). Function Inbuilt code that performs a specific task.
Data Type: String	A sequence of characters that can include letters, numbers, symbols.
Data Type: Integer	Whole numbers, no decimal point.
Data Type: Float	Decimal Numbers.
While Loop	A "While" Loop is used to repeat a specific block of code an unknown number of times, until a condition is met.
For Loop	For loop is a programming language conditional iterative statement which is used to check for certain conditions and then repeatedly execute a block of code as long as those conditions are met.
IF, Else, Elif	The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed.
Functions	A function is a command which contains the steps needed to perform a task.
Subroutines	A set of instructions designed to perform a frequently used operation within a program.

Lesson 2 - Drawing Patterns

```

from turtle import *
turtle = Turtle()
screen = Screen()
turtle.speed(100)
while True:
    for i in range(3):
        turtle.forward(50)
        turtle.left(120)
    turtle.right(10)
    
```

Patterns are repeating sequences of code.
Here we modify the triangle code to draw a repeating, rotating pattern. The while True loop will run forever, and the for loop will draw the triangle.
Each time the loop iterates we move the Turtle 10 pixels.

Lesson 2 - An example Pattern

```

from turtle import *
turtle = Turtle()
screen = Screen()
turtle.speed(100)
while True:
    turtle.pencolor(255,0,0)
    turtle.width(2)
    for i in range(3):
        turtle.forward(50)
        turtle.left(120)
    turtle.right(10)
    turtle.pencolor(0,0,255)
    turtle.width(2)
    for i in range(8):
        turtle.forward(50)
        turtle.left(45)
    turtle.right(10)
    
```

Here two for loops are used. The first draws a red triangle at double thickness.
The second loop draws a blue octagon, an eight sided shape.
Did you spot the block to increase the speed of the Turtle?

Lesson 3 - User Inputs Data Types

In this sequence of code we use logic to draw one of two shapes on the screen.

If the user input is square, then a for loop is used to draw the shape on the screen.

Else If the user input is circle, then a circle is drawn.

If we type in something else, then the else condition will activate and apologise to the user.

```

from turtle import *
turtle = Turtle()
screen = Screen()

while True:
    if input("What shape shall I draw?") == "square":
        for i in range(4):
            turtle.forward(50)
            turtle.left(90)
    elif input("What shape shall I draw?") == "circle":
        turtle.circle(50)
    else:
        print("I'm sorry I don't know that shape. Try again.")
    
```

Lesson 4 - Variables

We have captured the users colour choice. But how we do we use it?

We need to use conditional tests and logic to make this work.

The green blocks are found in Logic!

We've created the test for red, can you finish the code?

Red: 255,0,0

Green: 0,255,0

Blue 0,0,255

Run the code, what happens?

```

from turtle import *
turtle = Turtle()
screen = Screen()

sides = int(input("How many sides?"))
colour = input("What colour pen should I use? red, green or blue?")

if colour == "red":
    turtle.pencolor(255,0,0)
elif True:
    # Complete this code
elif True:
    # Complete this code

for i in range(sides):
    turtle.forward(50)
    turtle.left(360 / sides)
    
```

Lesson 5 - Functions

- From the Turtle blocks we need to drag:

- from turtle import *
- turtle = Turtle()
- screen = Screen()

Your code should look like this.
Click Run to test!

```

from turtle import *
turtle = Turtle()
screen = Screen()

def sides(n):
    for i in range(n):
        turtle.forward(30)
        turtle.left(360 / n)

while True:
    sides(int(input("How many sides does the shape have?")))
    
```

Lesson 5 - Why are Functions Useful

Why are functions useful?

Functions are powerful tools. They are subroutines, small sequences of code inside the main code.

We can call the function, and come out of the main code, do the function, then come back to the code.

They enable us to reuse sections of code.

They keep our code tidy, and with fewer lines to write.

In our code we can draw any shape using one section of code.



Lesson 6 - Project

```

from turtle import *
turtle = Turtle()
screen = Screen()
screen.bgcolour(0,0,0)
turtle.speed(100)

if input("Would you like to play? Answer y or n") == "y":
    def star(r):
        for i in range(5):
            turtle.forward(45)
            turtle.left(144)
            turtle.left(45)

    def circle(r):
        turtle.circle(r)
        turtle.right(45)

    number = int(input("What is the radius of the circle?"))

    for i in range(4):
        turtle.pencolor("red")
        star(10)
        turtle.pencolor("blue")
        star(10)

    for i in range(8):
        turtle.pencolor("green")
        circle(number)

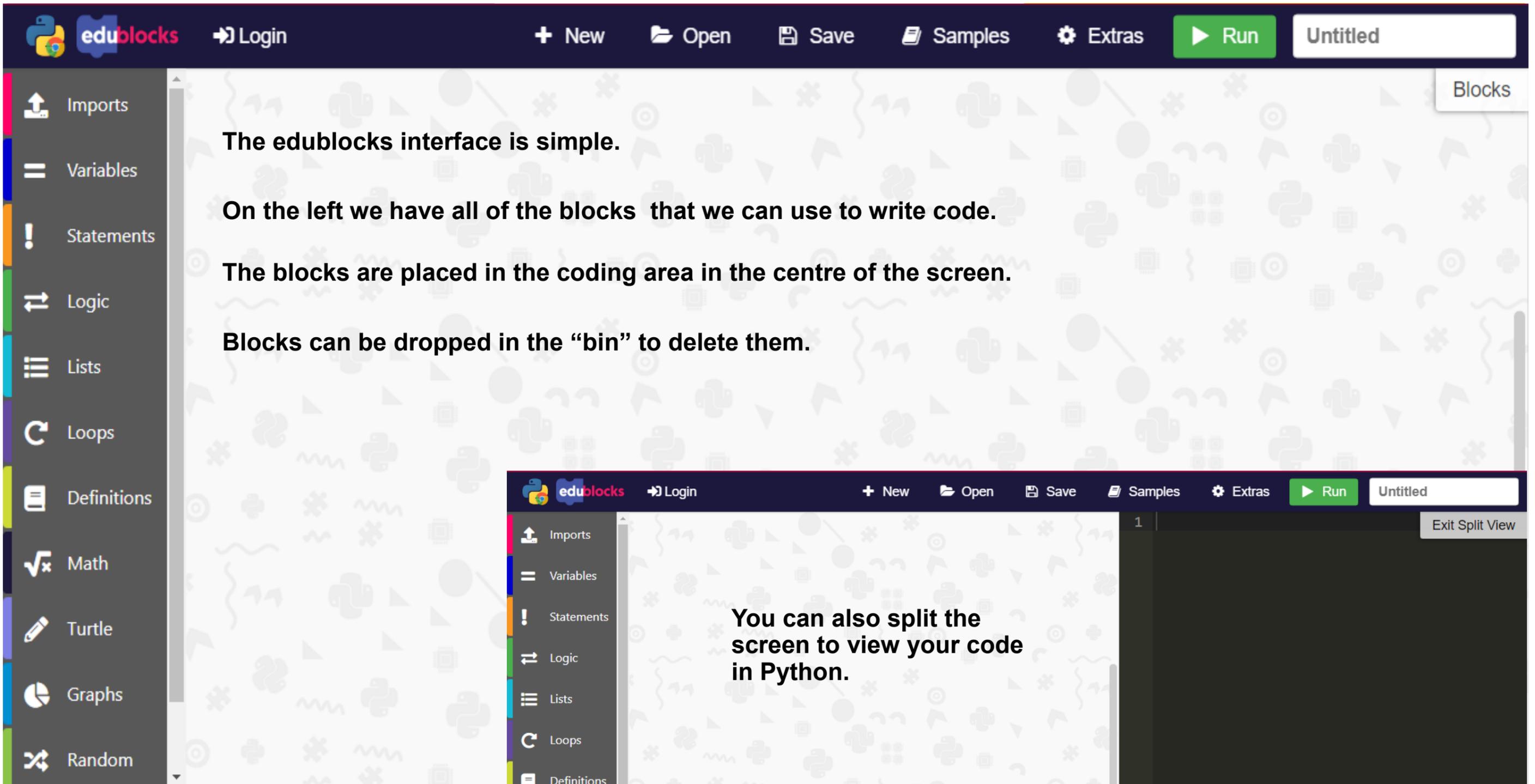
    else:
        print("Ok bye!")
    
```



Trinity TV

For more help, visit Trinity TV and watch the following videos:

Trinity TV > Year 9 > Computer Science > Term 1

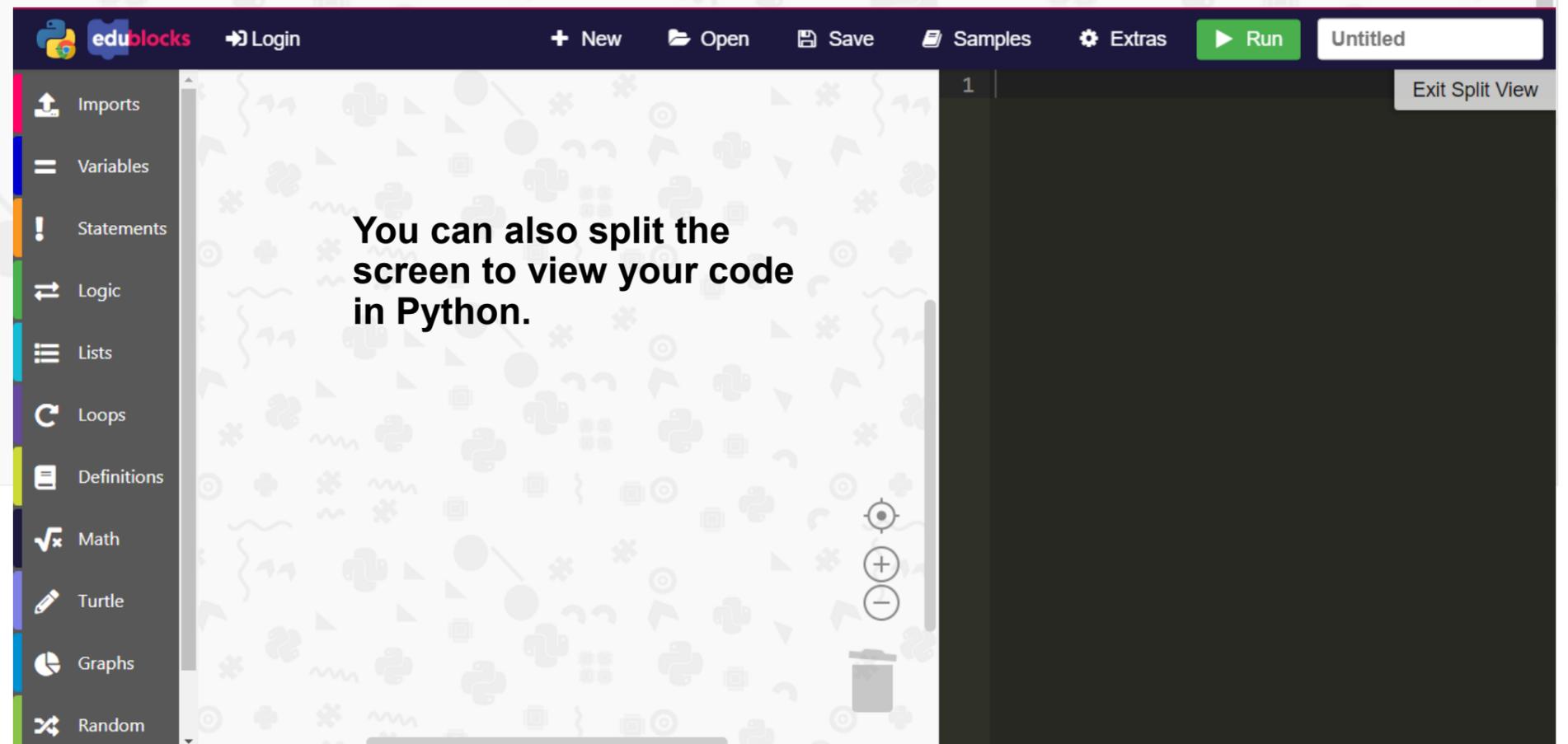


The edublocks interface is simple.

On the left we have all of the blocks that we can use to write code.

The blocks are placed in the coding area in the centre of the screen.

Blocks can be dropped in the “bin” to delete them.



You can also split the screen to view your code in Python.



Trinity TV

For more help, visit Trinity TV and watch the following videos:

Trinity TV > Year 9 > Computer Science > Term 1