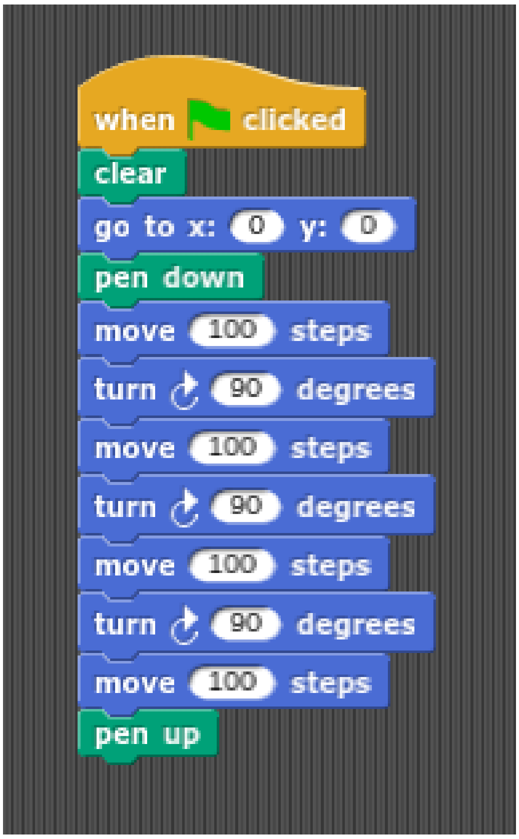
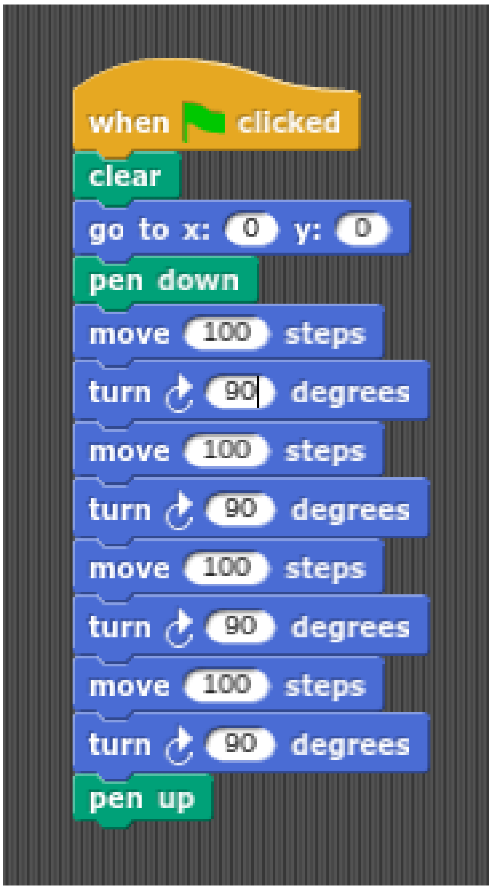
An introduction to programming

Here is a simple programme for drawing a square; we will start with this and try to make more complicated drawings.



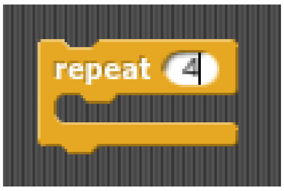
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
pen up
```

Enter this and make sure it draws a square! One thing about this program is that after it draws the square the arrow ends up pointing a different direction to the direction it started in; can you fix this?



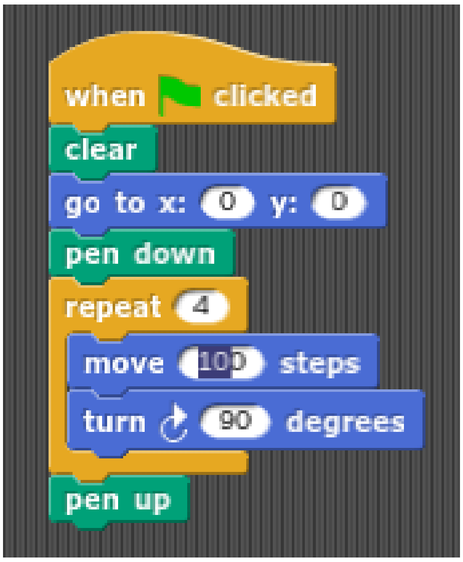
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
move 100 steps
turn ↻ 90 degrees
pen up
```

Repeat



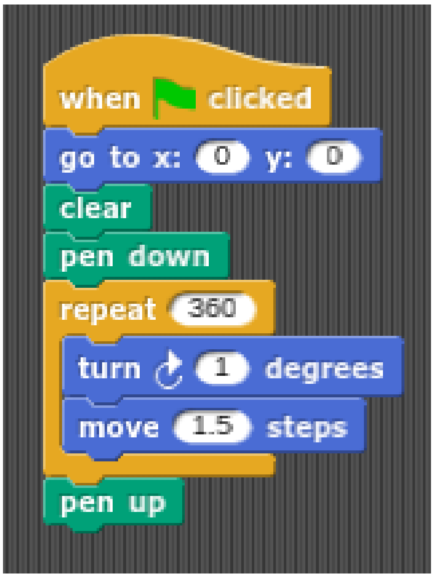
```
repeat 4
```

Can you use that to make the programme more succinct and readable?



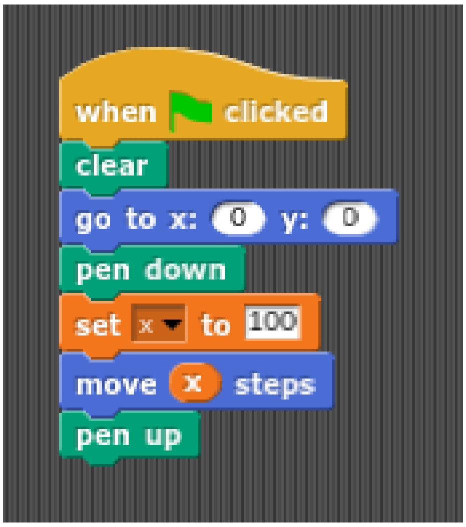
```
when ⚑ clicked
clear
go to x: (0) y: (0)
pen down
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
pen up
```

Can you make a programme to draw something that looks like a circle but going forward a tiny bit and turning again and again?



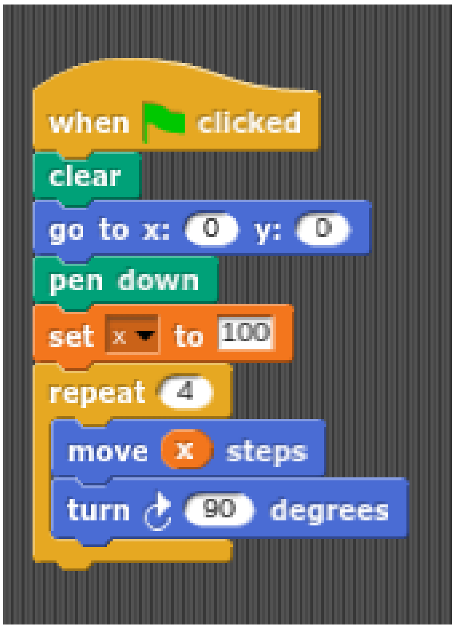
```
when ⚑ clicked
go to x: (0) y: (0)
clear
pen down
repeat (360)
    turn ↻ (1) degrees
    move (1.5) steps
pen up
```

Now, look at this programme



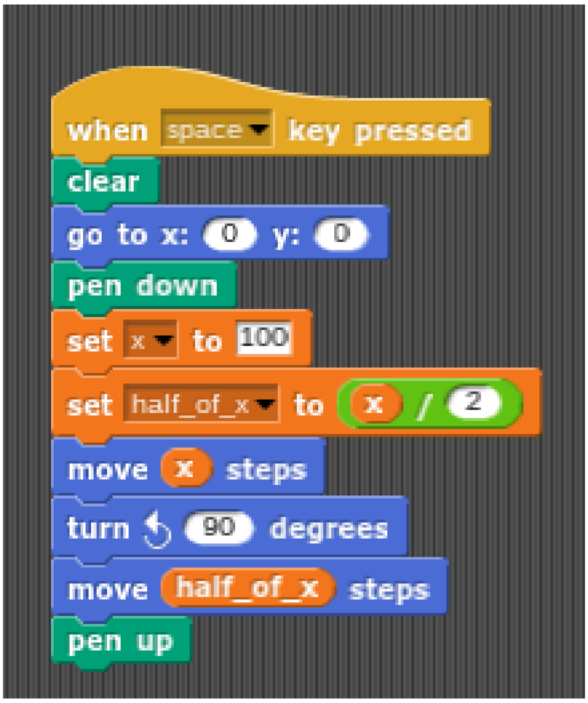
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 100
move x steps
pen up
```

Do the same to your square programme!



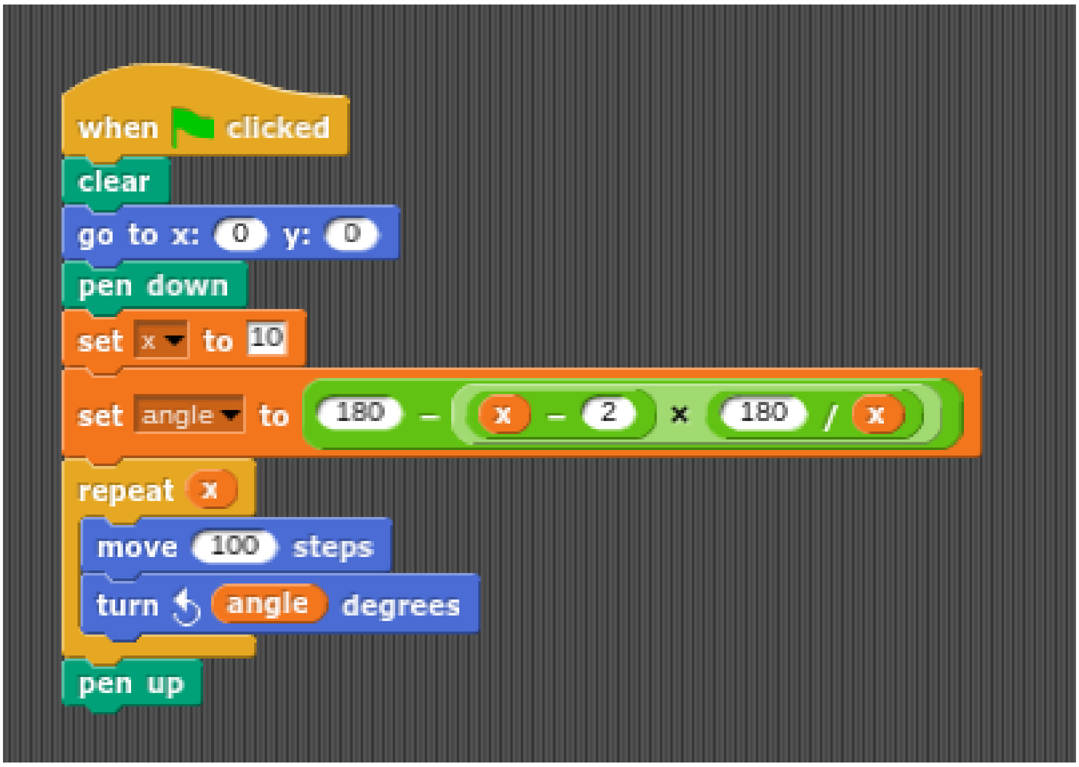
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 100
repeat 4
  move x steps
  turn ↻ 90 degrees
```

This programme does something slighly more useful with a variable.



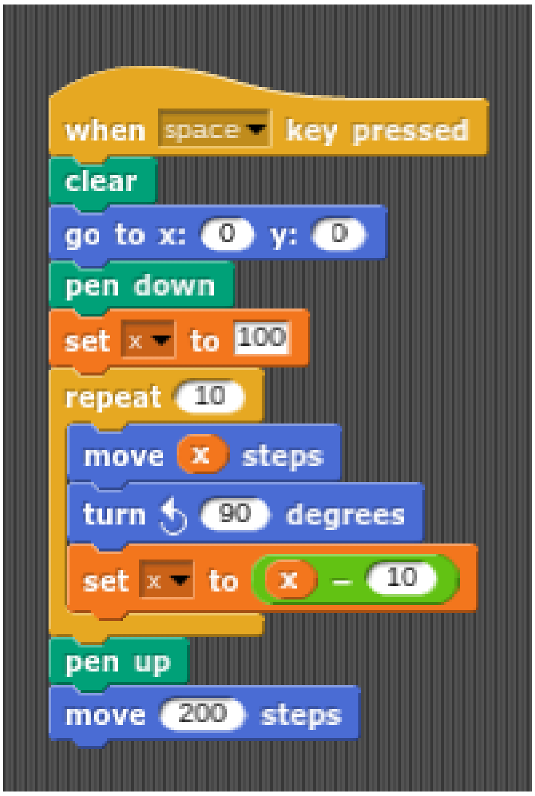
```
when space key pressed
clear
go to x: 0 y: 0
pen down
set x to 100
set half_of_x to x / 2
move x steps
turn ↺ 90 degrees
move half_of_x steps
pen up
```

Try modifying your programme in a similar way so that it draws an n-gon.



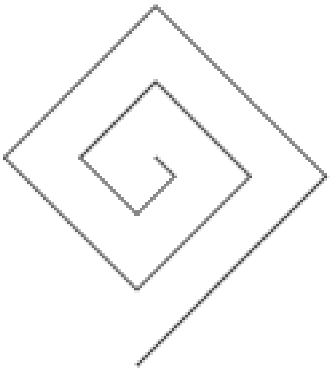
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 10
set angle to 180 - ((x - 2) × (180 / x))
repeat x
    move 100 steps
    turn ↺ angle degrees
pen up
```

In this programme the variable is changed in the *loop* so the line is shorter each time:

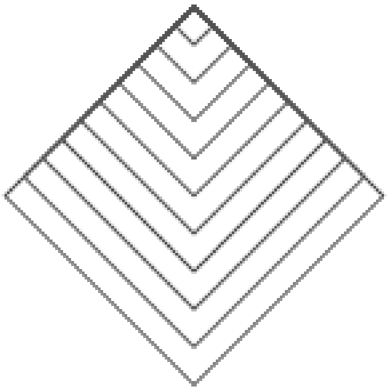


```
when space key pressed
clear
go to x: 0 y: 0
pen down
set x to 100
repeat 10
    move x steps
    turn ↺ 90 degrees
    set x to x - 10
pen up
move 200 steps
```

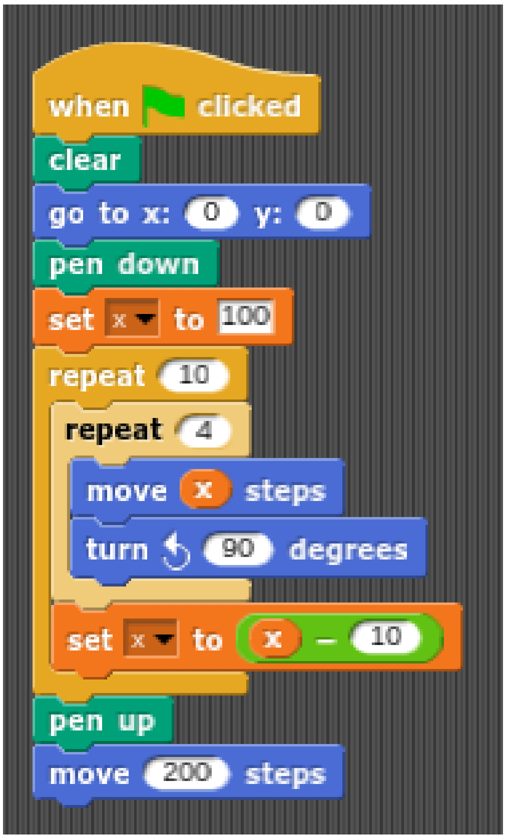
giving a spiral



Try modifying your programme in the same way so that you get smaller and smaller squares retreating into one corner, like this

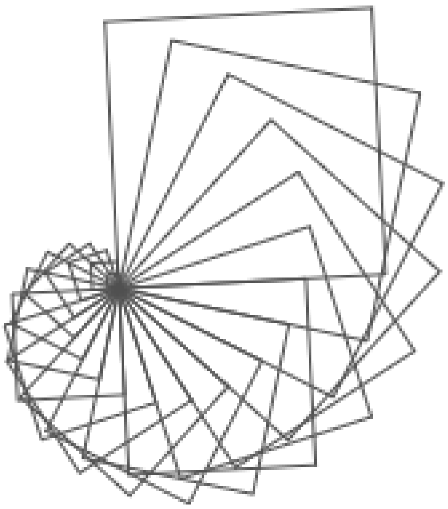


so

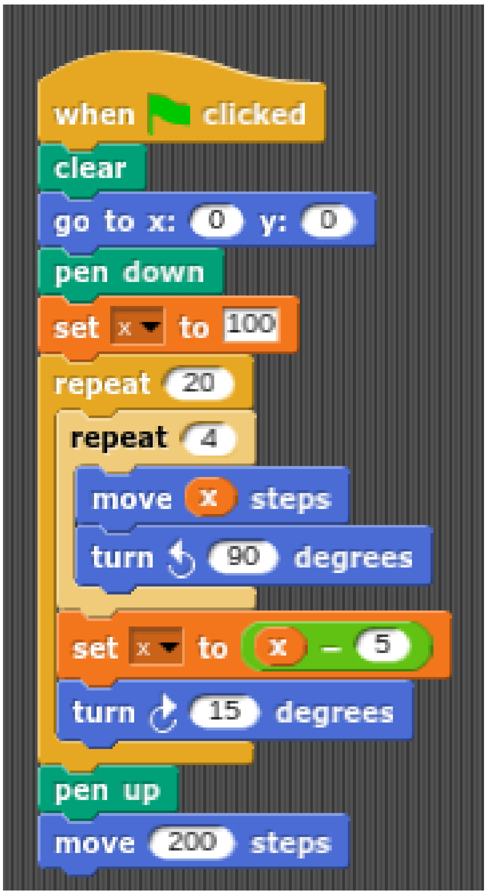


```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 100
repeat 10
  repeat 4
    move x steps
    turn ↺ 90 degrees
  set x to x - 10
pen up
move 200 steps
```

If you want to you can try playing with you program a bit to give other patterns, like this

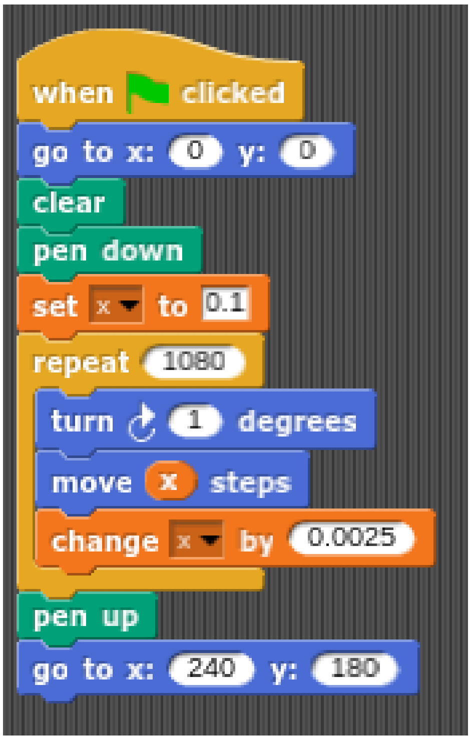


so



```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 100
repeat 20
  repeat 4
    move x steps
    turn ↺ 90 degrees
  set x to x - 5
  turn ↻ 15 degrees
pen up
move 200 steps
```

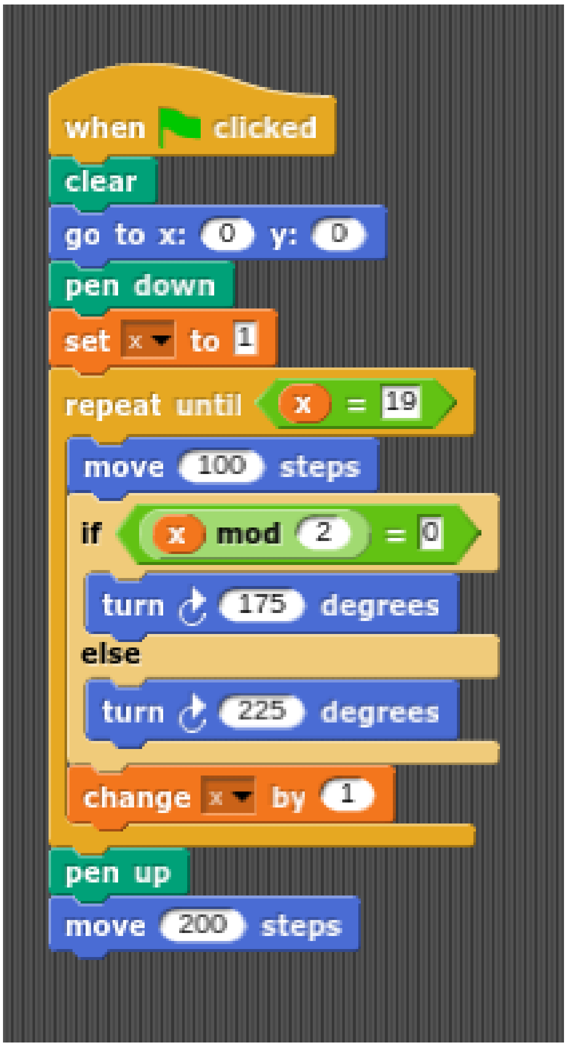
Try modifying your circle programme to draw a round spiral.



```
when ⚑ clicked
go to x: 0 y: 0
clear
pen down
set x to 0.1
repeat 1080
    turn ↻ 1 degrees
    move x steps
    change x by 0.0025
pen up
go to x: 240 y: 180
```

This next programme draws a star;

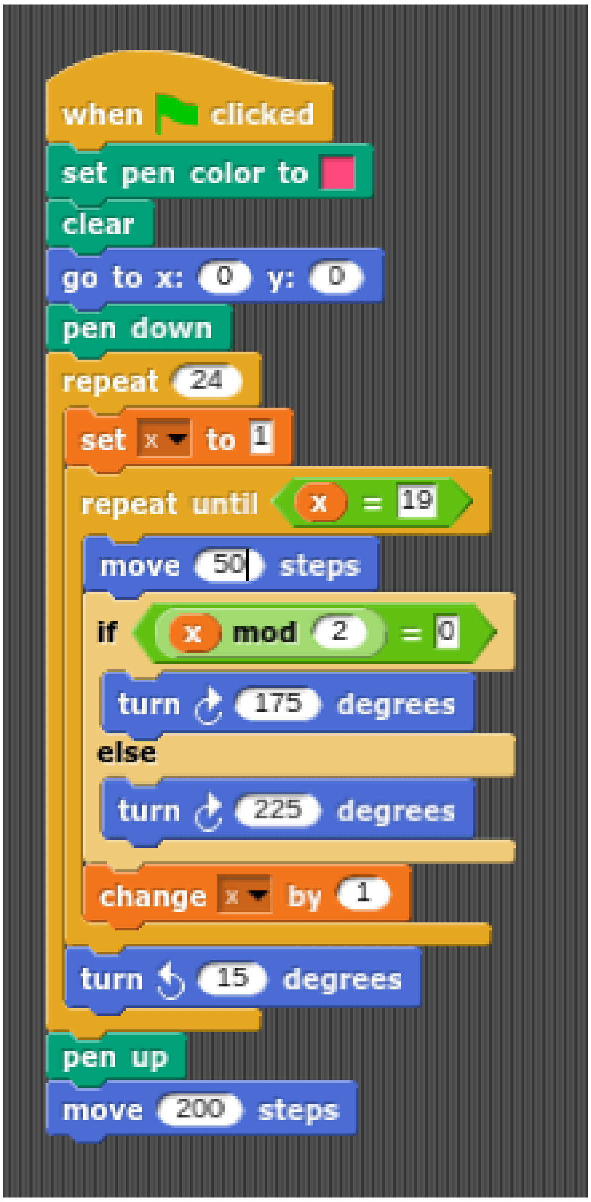
This next programme draws a star



```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 1
repeat until x = 19
    move 100 steps
    if x mod 2 = 0
        turn ↻ 175 degrees
    else
        turn ↻ 225 degrees
    change x by 1
pen up
move 200 steps
```

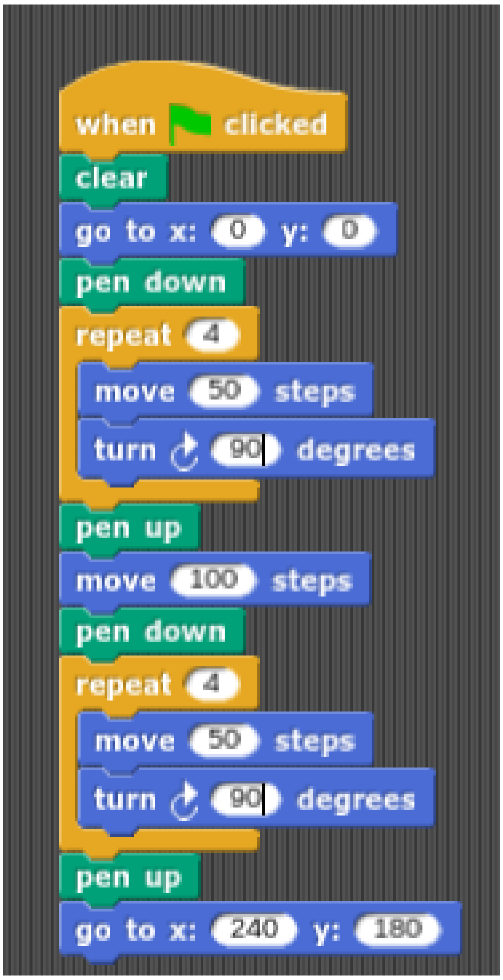
You can mess with programme a bit, maybe changing the angles or putting the whole thing in a loop to give something like this

so



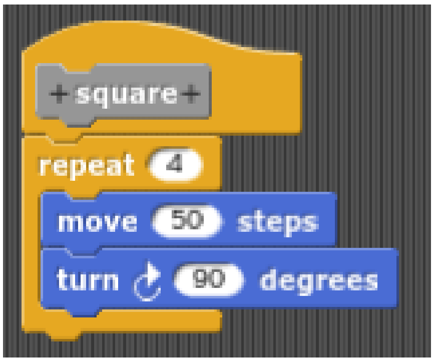
```
when ⚑ clicked
set pen color to 🟥
clear
go to x: 0 y: 0
pen down
repeat 24
    set x to 1
    repeat until x = 19
        move 50 steps
        if x mod 2 = 0
            turn ↻ 175 degrees
        else
            turn ↻ 225 degrees
        change x by 1
    turn ↺ 15 degrees
pen up
move 200 steps
```

Imagine you want to use the same commands a few times; in this programme for example we draw a square, move over a bit and draw another:



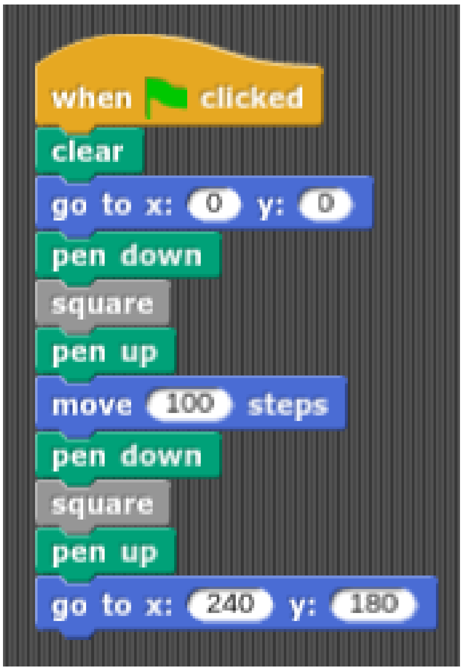
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up
move 100 steps
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up
go to x: 240 y: 180
```

Here we will make a block called square that draws a square: the block commands are at the bottom of the variable menu:



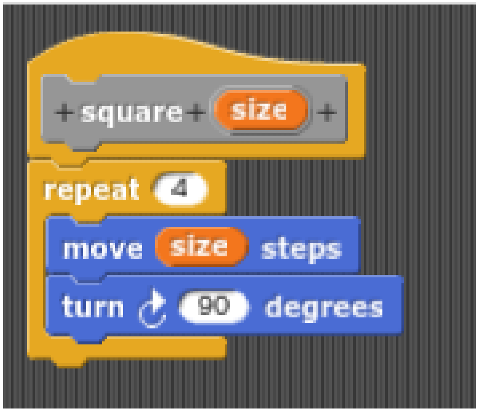
```
+square+
repeat 4
    move 50 steps
    turn ↻ 90 degrees
```

This block draws a square, so our two square code becomes a bit neater, quicker to input and easier to read:



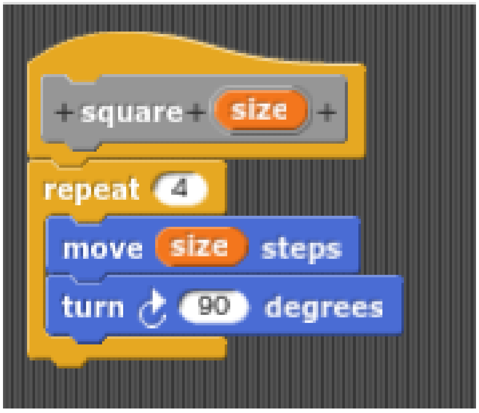
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
square
pen up
move 100 steps
pen down
square
pen up
go to x: 240 y: 180
```

You can make your blocks more flexible by adding *arguments*; these are variables that work inside the block that you can send from the main programme, you make them by clicking the plus by the block name.



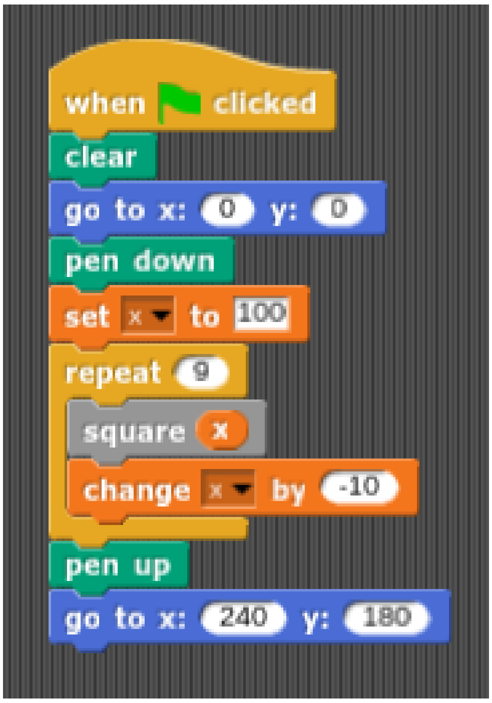
```
+square+ size +
repeat 4
    move size steps
    turn ↻ 90 degrees
```

and



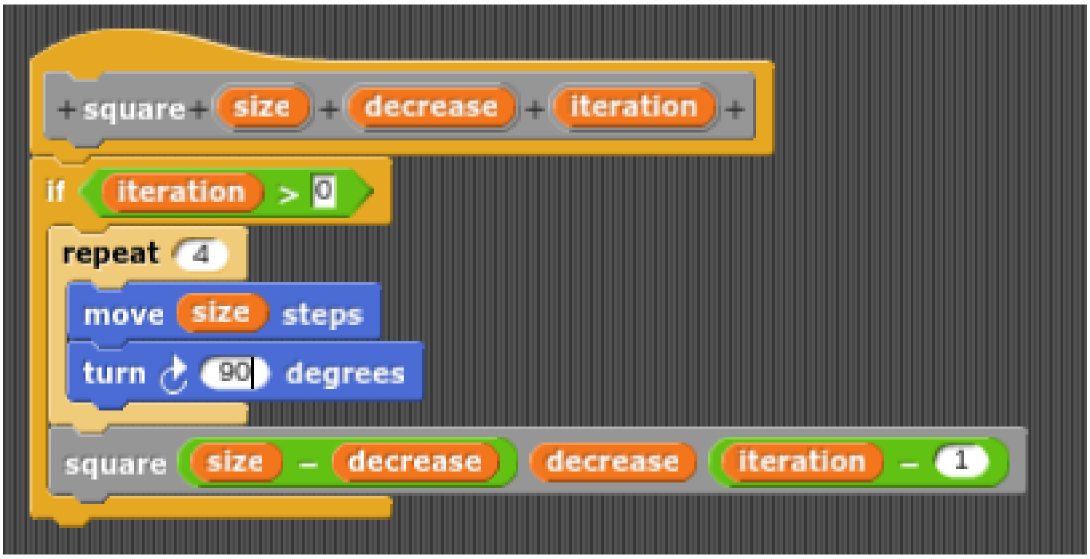
```
+square+ size +
repeat 4
    move size steps
    turn ↻ 90 degrees
```

draws the two squares different sizes. Try rewriting your original shrinking square code with blocks.



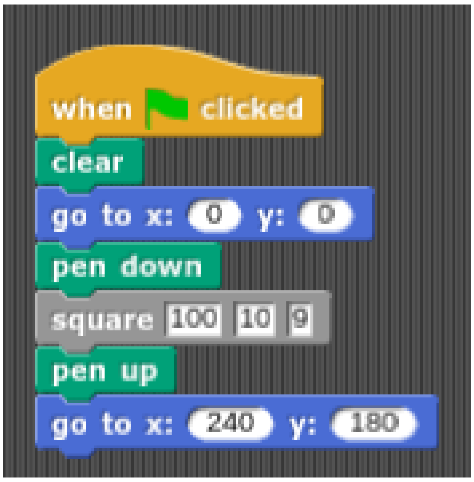
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
set x to 100
repeat 9
    square x
    change x by -10
pen up
go to x: 240 y: 180
```

If they finish this, maybe try recursion:



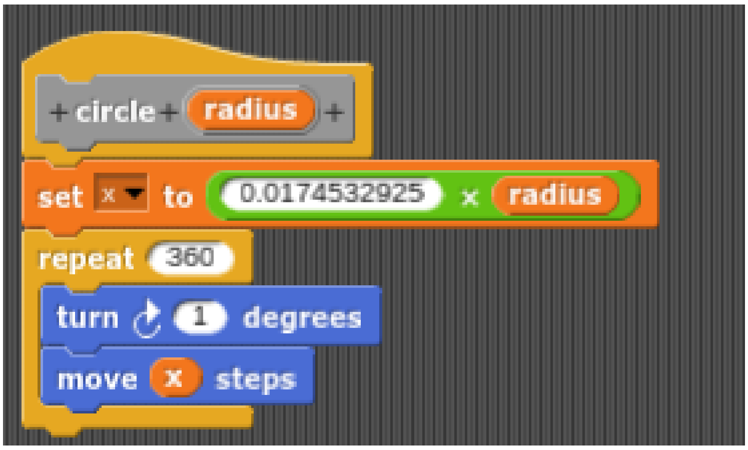
```
+ square + size + decrease + iteration +
if iteration > 0
    repeat 4
        move size steps
        turn ↻ 90 degrees
    square size - decrease decrease iteration - 1
```

draws the two squares different sizes. Try rewriting your original shrinking square code with blocks.



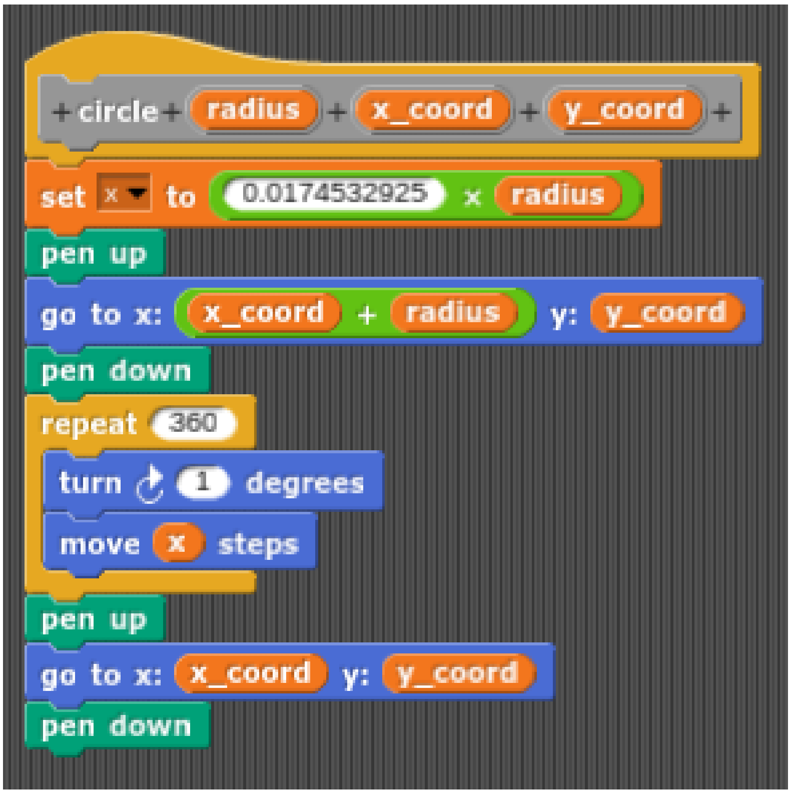
```
when ⚑ clicked
clear
go to x: 0 y: 0
pen down
square 100 10 9
pen up
go to x: 240 y: 180
```

Try writing a block to make a circle with the radius as the argument



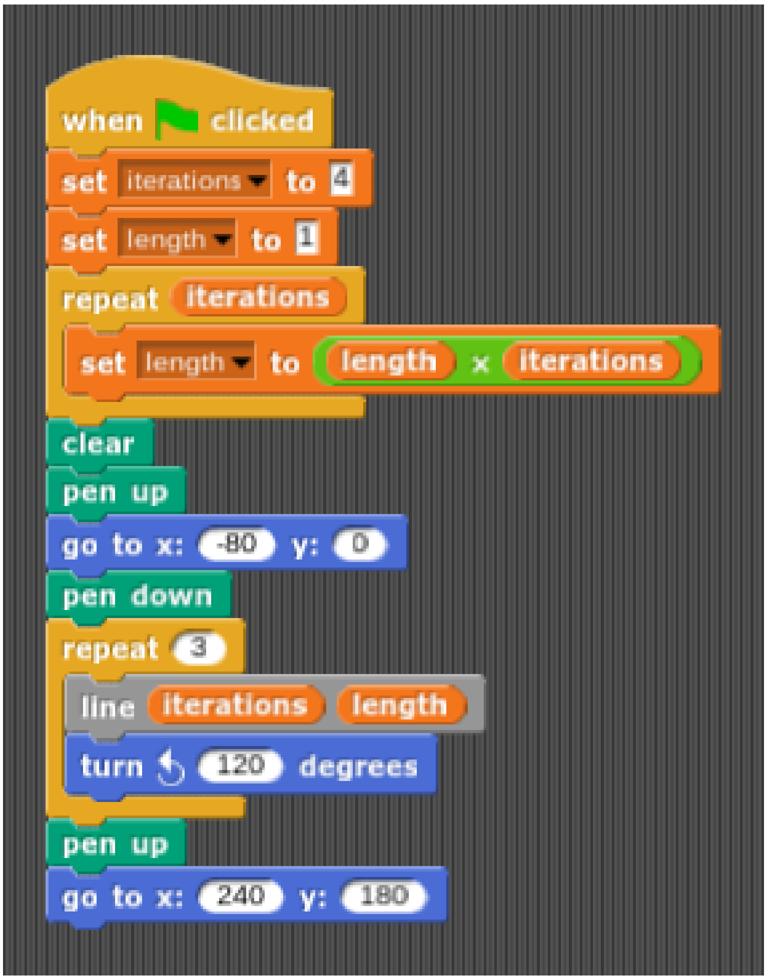
```
+ circle + radius +
set x to 0.0174532925 × radius
repeat 360
    turn ↻ 1 degrees
    move x steps
```

Can you improve your circle block so that you also send the x and y location of the center of the circle?



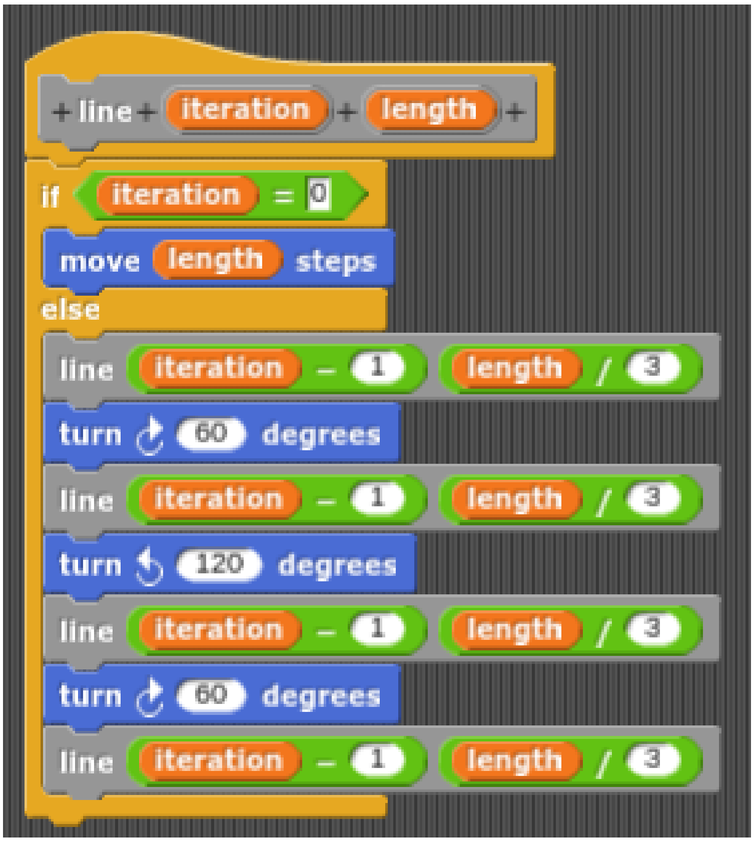
```
+circle+ radius + x_coord + y_coord +
set x to (0.0174532925 × radius)
pen up
go to x: (x_coord + radius) y: y_coord
pen down
repeat 360
    turn ↻ 1 degrees
    move x steps
pen up
go to x: x_coord y: y_coord
pen down
```

If anyone is finished with all of this maybe go on to this



```
when ⚑ clicked
set iterations to 4
set length to 1
repeat (iterations)
    set length to ((length) x (iterations))
clear
pen up
go to x: (-80) y: (0)
pen down
repeat (3)
    line (iterations) (length)
    turn ↺ (120) degrees
pen up
go to x: (240) y: (180)
```

with block



```
+ line + iteration + length +
if iteration = 0
  move length steps
else
  line iteration - 1 length / 3
  turn ↻ 60 degrees
  line iteration - 1 length / 3
  turn ↺ 120 degrees
  line iteration - 1 length / 3
  turn ↻ 60 degrees
  line iteration - 1 length / 3
```

draws