1. SQUARES

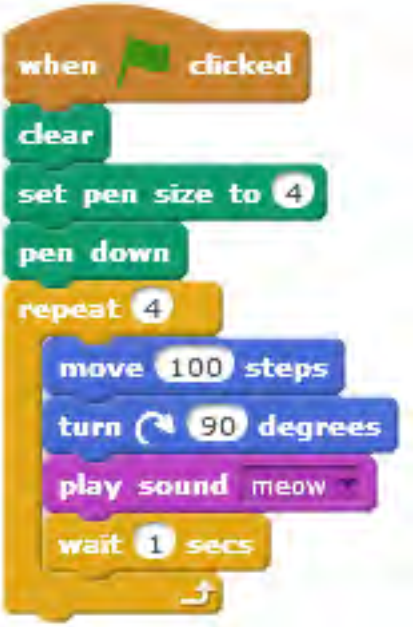
A. Let's write some code to draw a square:

- go to the **Scratch** website: http://scratch.mit.edu
- click on **Create** on the menu

CODING HELP

- The **Cat** is a **Sprite**. A **Sprite** is an image that you can manipulate using code.
- To "hide" the **Cat Sprite**: In the **Sprites** area (bottom-left), right-click on the **Cat** and select "hide"

B. Let's add a repeat block to make the code more efficient.



C. Make your own code block. In the **Scripts** menu, click on **More Blocks**

- click on **New Block**
- in the dialogue box that pops up, type "draw square" for the name of the new block, as shown below

- click OK to get this

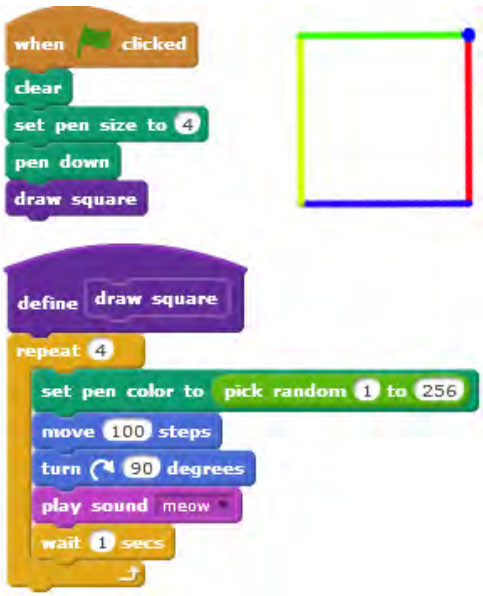
- drag the repeat block and place it underneath

- drag and drop 

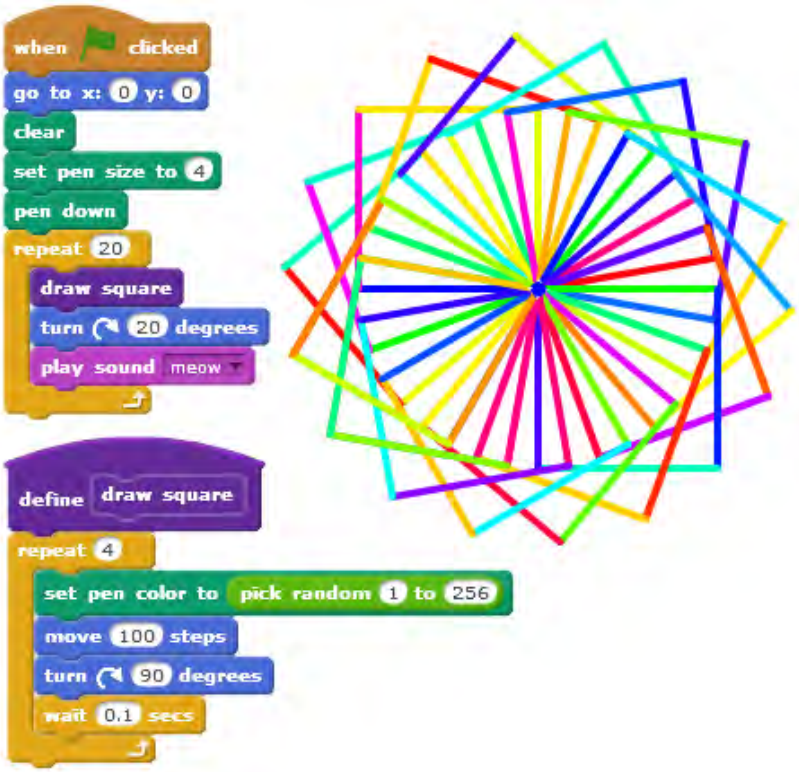
where the repeat block used to be.
- it will look as shown on the right:

Excerpts from Gadanidis, G. (2015). Coding for Young Mathematicians. Available at http://worldiscoveries.ca/technology/18155

D. Add color.



E. Add a turn.



F. Coding challenges.

- draw a circle
- draw half a circle
- draw two half circles
- draw lots of circles
- draw a flower
- draw spirals
- what else?

2. GROWING PATTERNS

A. Natural numbers.

The **Python** code below prints the first 10 Natural numbers.

```
stage = 1
while stage <= 10:
    print (stage)
    stage = stage + 1
```

```
1
2
3
4
5
6
7
8
9
10
```

TRY IT!

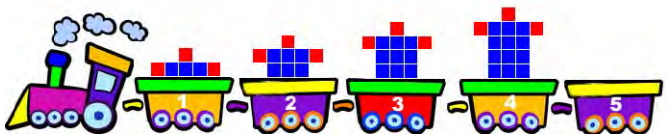
- Go to http://cscircles.cemc.uwaterloo.ca/console
- Type in the code above.
- Click on **Run** to see the **output**.

B. Edit the code to print:

- even numbers
- odd numbers
- multiples of 7
- natural numbers in reverse order
- what else?

C. Constants and variables.

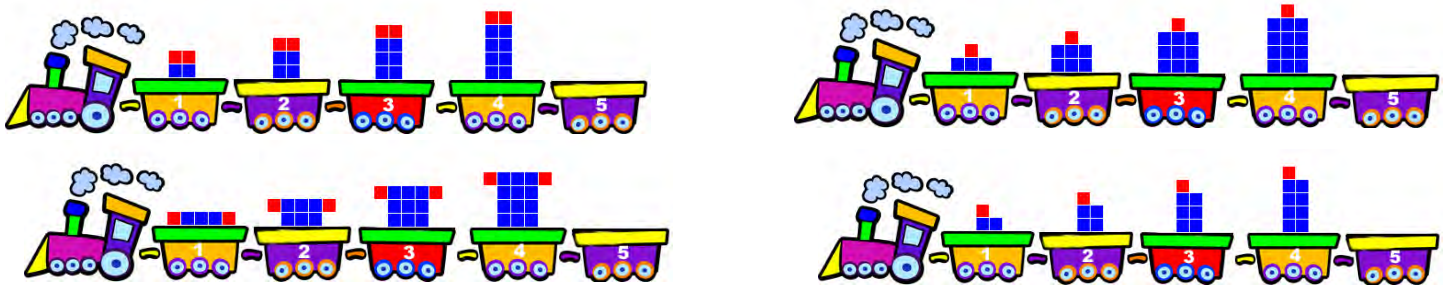
The Python code below models this pattern.
This pattern is the linear function y = 3x + 3. Do you see why?



```
stage = 1
while stage <= 10:
    blue = 3 * stage
    red = 3
    sum = blue + red
    print ("stage", stage, ":", "blue =", blue, "red =", red, "sum =", sum)
    stage = stage +1
```

```
stage 1 : blue = 3 red = 3 sum = 6
stage 2 : blue = 6 red = 3 sum = 9
stage 3 : blue = 9 red = 3 sum = 12
stage 4 : blue = 12 red = 3 sum = 15
stage 5 : blue = 15 red = 3 sum = 18
stage 6 : blue = 18 red = 3 sum = 21
stage 7 : blue = 21 red = 3 sum = 24
stage 8 : blue = 24 red = 3 sum = 27
stage 9 : blue = 27 red = 3 sum = 30
stage 10 : blue = 30 red = 3 sum = 33
```

Edit the code to model the patterns below.



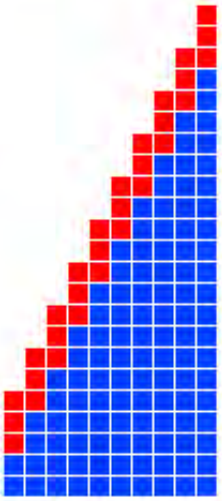
Excerpts from Gadanidis, G. (2015). Coding for Young Mathematicians. Available at http://worldiscoveries.ca/technology/18155

D. Growing patterns with Scratch.

Build this code.

Edit it to simulate the patterns show page 3.

What other patterns can you simulate?



3. PROBABILITY

A. The Python code simulates rolling a number cube 100 times and keeping track how often the numbers 1-6 come up. Can you edit it to simulate the possible sums when rolling two dice?

```
import random
roll1 = 0
roll2 = 0
roll3 = 0
roll4 = 0
roll5 = 0
roll6 = 0
for x in range(1,101):
   roll = random.randint(1,6)
   if roll == 1:
      roll1 = roll1 + 1
   elif roll == 2:
      roll2 = roll2 + 1
   elif roll == 3:
      roll3 = roll3 + 1
   elif roll == 4:
      roll4 = roll4 + 1
   elif roll == 5:
      roll5 = roll5 + 1
   elif roll == 6:
      roll6 = roll6 + 1
print("TOTALS")
print("1 was rolled",roll1,"times")
print("2 was rolled",roll2,"times")
print("3 was rolled",roll3,"times")
print("4 was rolled",roll4,"times")
print("5 was rolled",roll5,"times")
print("6 was rolled",roll6,"times")
```

```
TOTALS
1 was rolled 19 times
2 was rolled 17 times
3 was rolled 23 times
4 was rolled 12 times
5 was rolled 16 times
6 was rolled 13 times
```