

0100100110101  
101001001001  
001101001101  
001100010101  
001010100101

# Welcome!

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

# ABOUT ME



# Who Are You?

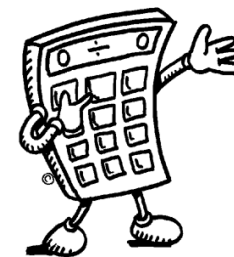
- What school do you go to?
  - What grade are you in?
- What made you come to the workshop?
- What's something interesting we can't tell by looking at you?



# What is Computer Science?

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.



# Solving Problems!



thinking use data capable  
 year include Target propose  
 Musicians Center Kidney binding  
 OneWORK History key Ideas techniques two  
 understand developed microsoft possible  
 real given set topics tools goal environment Performer  
 models programs SCIENCE optimization PROBE well file  
 Parallelmay privacy language  
 Ensemble problem Live plan proposed time cells structures used  
 focused able People impact Algorithms Environments different  
 information organized design sequence  
 recent new parallelism engineering develop distribution patient  
 exchange Performance Carnegie program exchanges algorithm complex store free example robots programming  
 beat together project audio others Studio processing researchers provably current energy System Working technology understanding think  
 Meld Using already research Interactive  
 Music make drug workshop types e.g. robotics molecule model



# Why Learning to Code is Awesome

<https://www.youtube.com/watch?v=nKlu9yen5nc>



# Getting Started Thinking Like a Computer

<http://csunplugged.org/programming-languages>



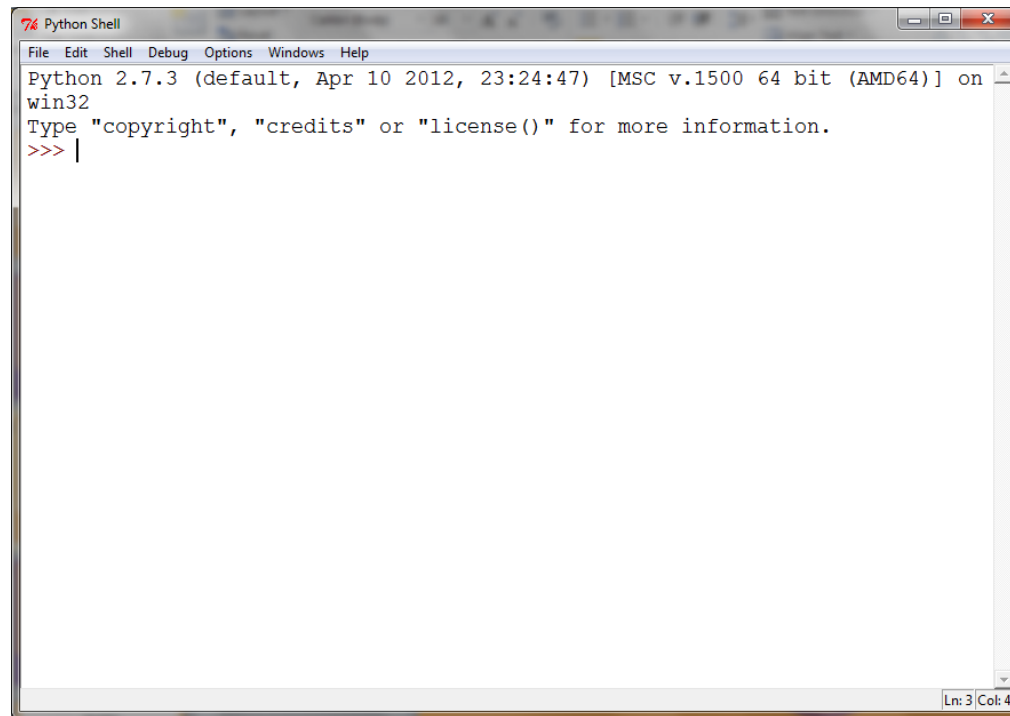


# Turtle Graphics

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

# Open IDLE

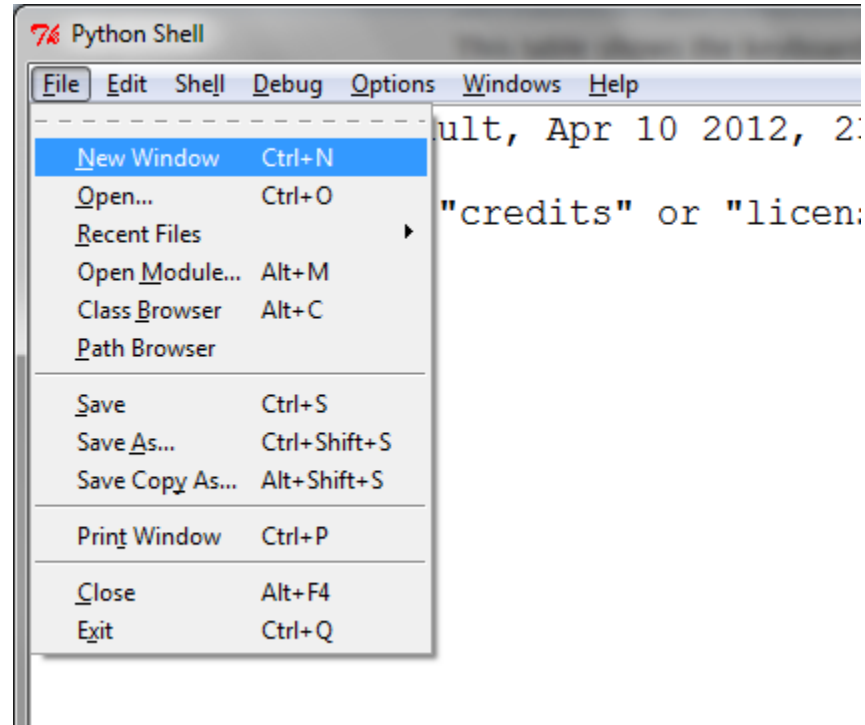


The image shows a screenshot of the Python Shell window. The window title is "Python Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area displays the following information: "Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32". Below this, it says "Type 'copyright', 'credits' or 'license()' for more information." and the prompt ">>> |" is visible. The status bar at the bottom right shows "Ln: 3 Col: 4".

```
Python 2.7.3 (default, Apr 10 2012, 23:24:47) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```



# File, New Window



# Type this in the new window:

```
import turtle

wn = turtle.Screen()

alex = turtle.Turtle()
alex.forward(150)
alex.left(90)
alex.forward(75)

wn.exitonclick()
```

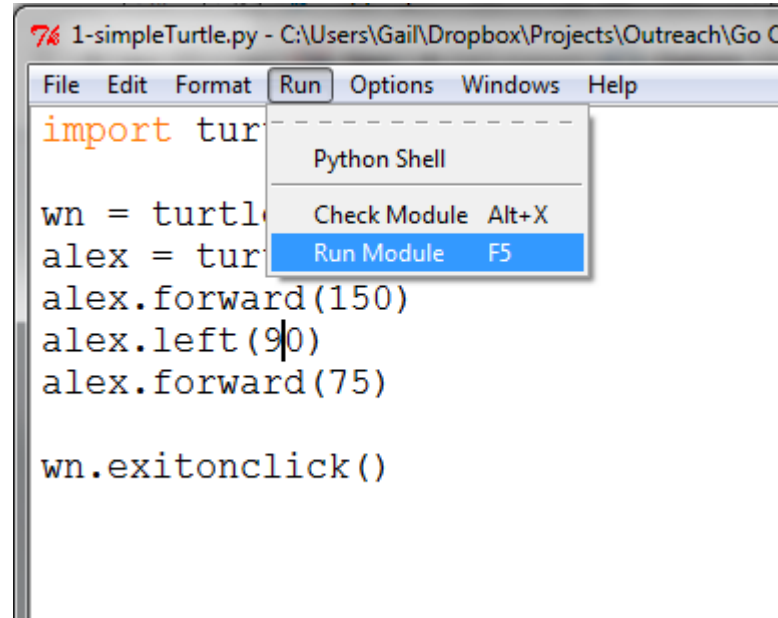


# Save, Save As

Note: make sure you  
add `.py` to the end  
of your file!



# Run, Run Module

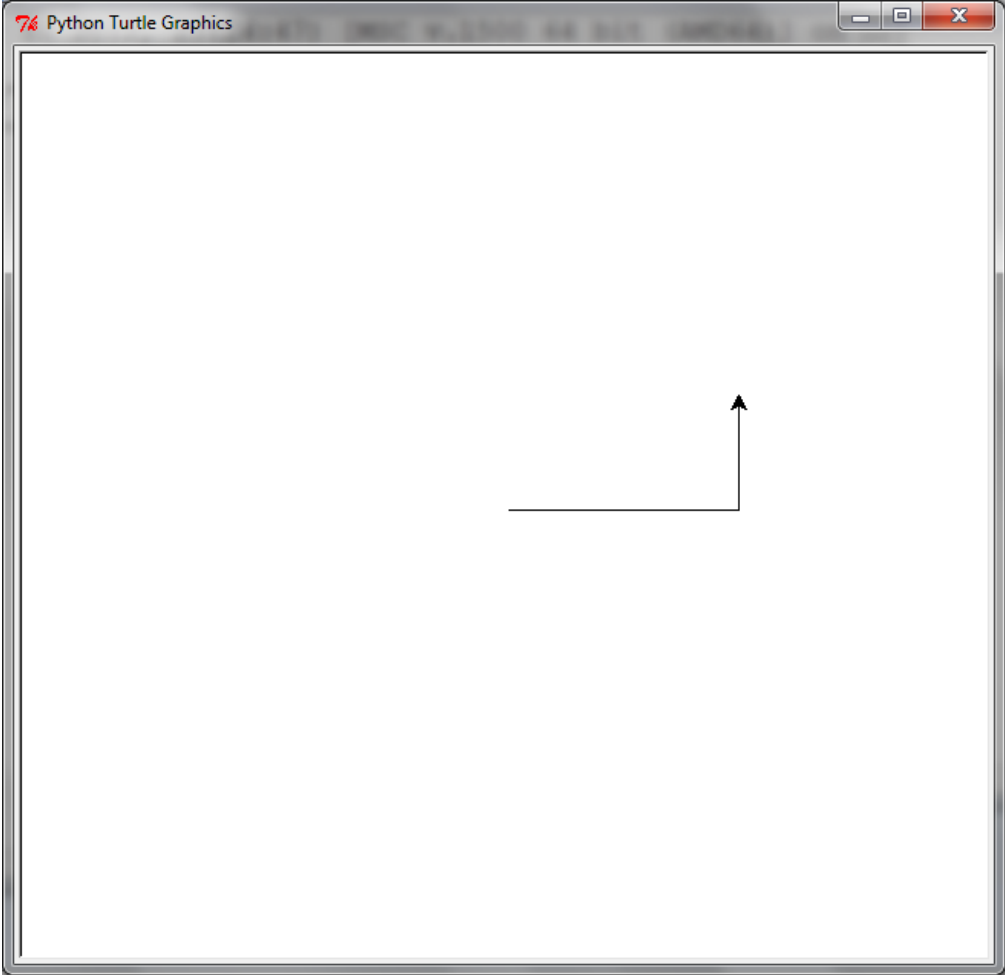


```
7% 1-simpleTurtle.py - C:\Users\Gail\Dropbox\Projects\Outreach\Go C
File Edit Format Run Options Windows Help
import turtle
wn = turtle.Screen()
alex = turtle.Turtle()
alex.forward(150)
alex.left(90)
alex.forward(75)

wn.exitonclick()
```

Python Shell  
Check Module Alt+X  
Run Module F5





Tell Python you want to  
use Turtle Graphics in your  
program

```
import turtle
```

```
wn = turtle.Screen()
```

```
alex = turtle.Turtle()
```

```
alex.forward(150)
```

```
alex.left(90)
```

```
alex.forward(75)
```

```
wn.exitonclick()
```





Create a new window to  
draw with the turtle on;  
refer to the window from  
now on as `wn`

```
import turtle
```

```
wn = turtle.Screen()
```

```
alex = turtle.Turtle()
```

```
alex.forward(150)
```

```
alex.left(90)
```

```
alex.forward(75)
```

```
wn.exitonclick()
```



```
import turtle  
  
wn = turtle.Screen()
```

Ask Turtle Graphics to  
create a new Turtle to  
draw with; call it alex

```
alex = turtle.Turtle()  
alex.forward(150)  
alex.left(90)  
alex.forward(75)  
  
wn.exitonclick()
```



```
import turtle

wn = turtle.Screen()

alex = turtle.Turtle()
alex.forward(150)
alex.left(90)
alex.forward(75)

wn.exitonclick()
```

Ask alex to go forward,  
turn left, and go forward  
again, drawing while she  
moves



```
import turtle

wn = turtle.Screen()

alex = turtle.Turtle()
alex.forward(150)
alex.left(90)
alex.forward(75)
```


```
wn.exitonclick()
```

Tell the program to exit  
when someone clicks on  
the window we named `wn`



Try changing the numbers in alex's movement code, or even add new movements.



Can you get alex to draw a  
square? 

How about a pentagon? 



# Repetition

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

# One way to draw a pentagon...

```
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
```





# One way to draw a pentagon...

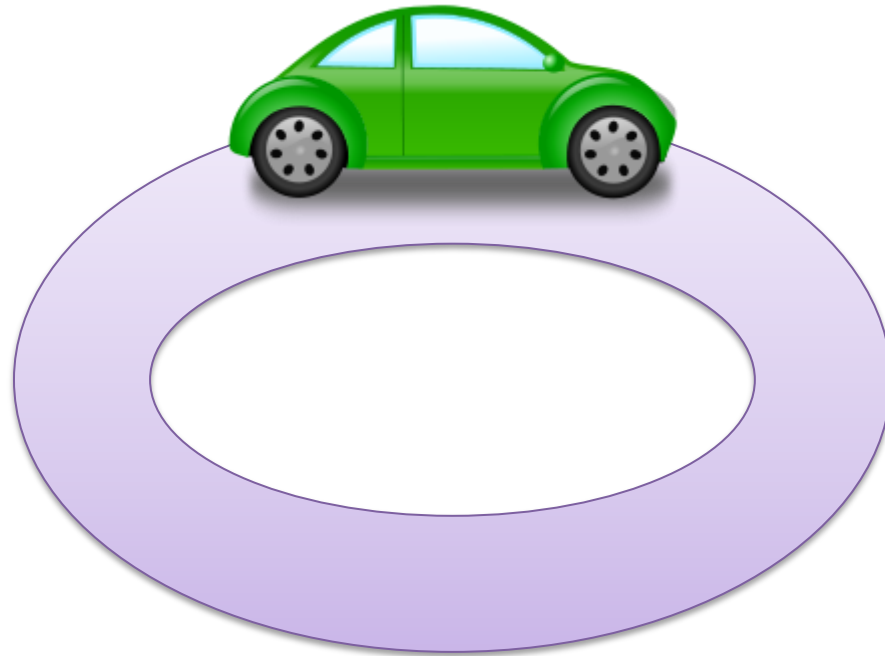
```
alex.forward(100)
alex.left(72)
alex.forward(100)
```

Can we avoid writing the same lines of code over and over?

```
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
```



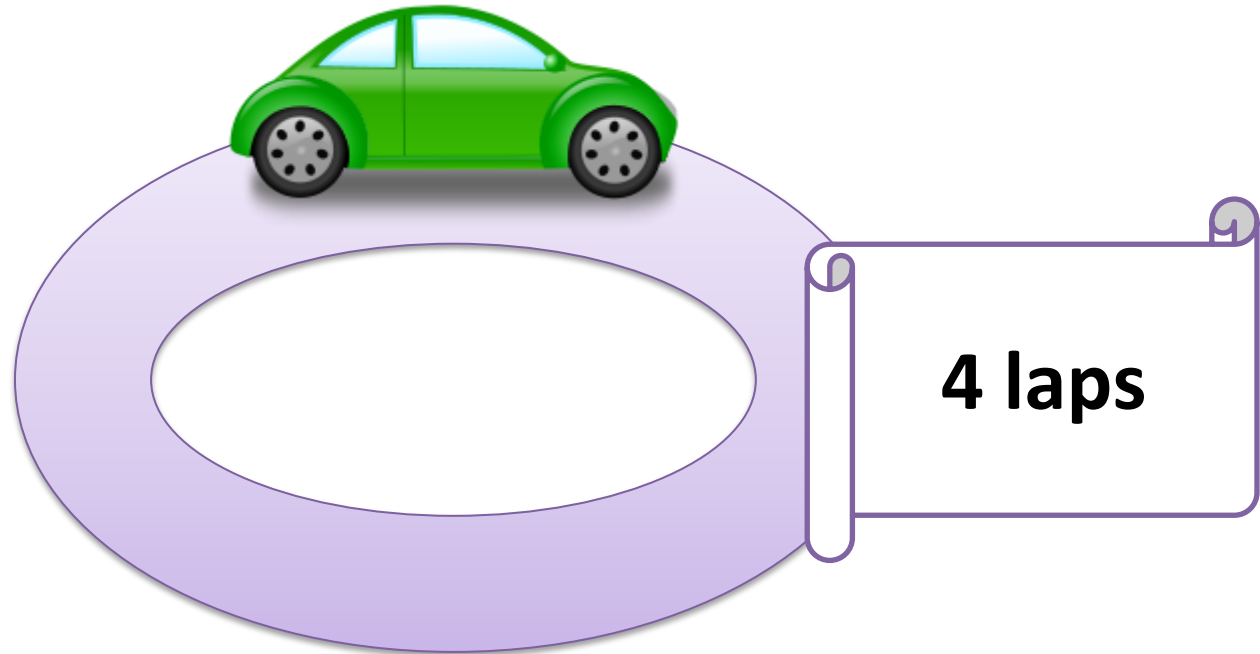
# Loops



Drive the same track multiple times



# for loop



Drive the same track exactly four times



# for loop



```
for lapNum in [1, 2, 3, 4]:  
    # drive the lap
```

**4 laps**

Drive the same track exactly four times



# Using a for loop to draw a pentagon

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```



# Use a for loop to draw a pentagon

This gives a name to the lap numbers as we “drive” around (first it will be 1, then 2, ...)

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```



# Using a for loop to draw a pentagon

This is a list representing the lap numbers.

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```



# Using a for loop to draw a pentagon

The colon says we're ready to specify how to drive each lap

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```





# Using a for loop to draw a pentagon

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```

We use indentation to show what code belongs inside the for loop



# Using a for loop to draw a pentagon

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```



# Using a for loop to draw a pentagon

```
for sideNum in [1, 2, 3, 4, 5]:  
    alex.forward(100)  
    alex.left(72)
```

This is the code that will  
run each lap (5 times in  
this case)



# Shortcut: range

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```



# Shortcut: range

This produces the list  
[0,1,2,3,4]

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```



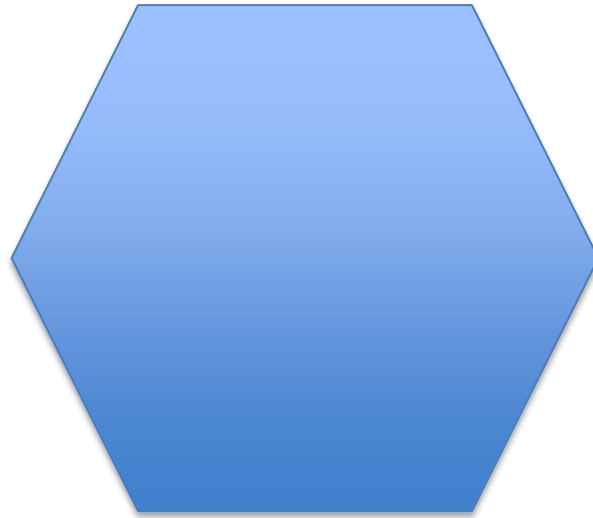
# Shortcut: range

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```

Important:  
We still have 5 laps,  
we're just counting them  
from 0 instead of 1



# Try drawing a hexagon instead!



What other cool shapes or designs can you make?



# Variables

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

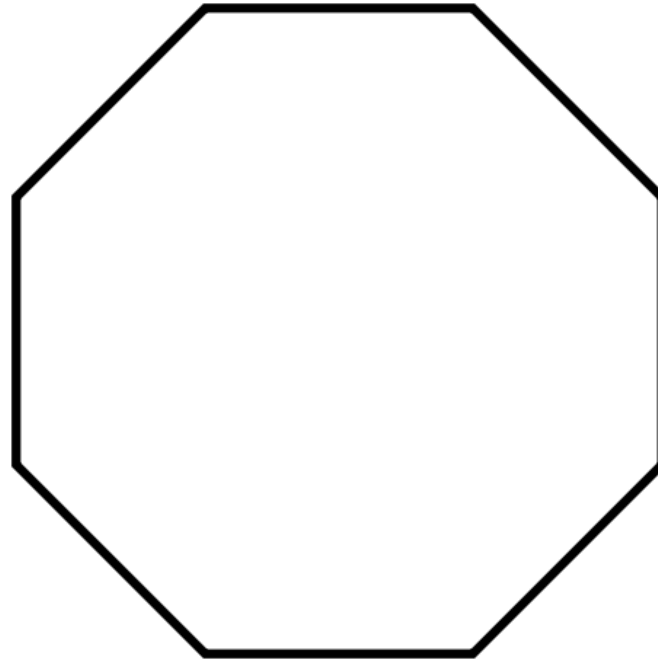


# Remember our shape drawing loop?

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```



# What if we wanted to draw an octagon?



# What if we wanted to draw an octagon?

This number has to change so we can have more sides...

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```



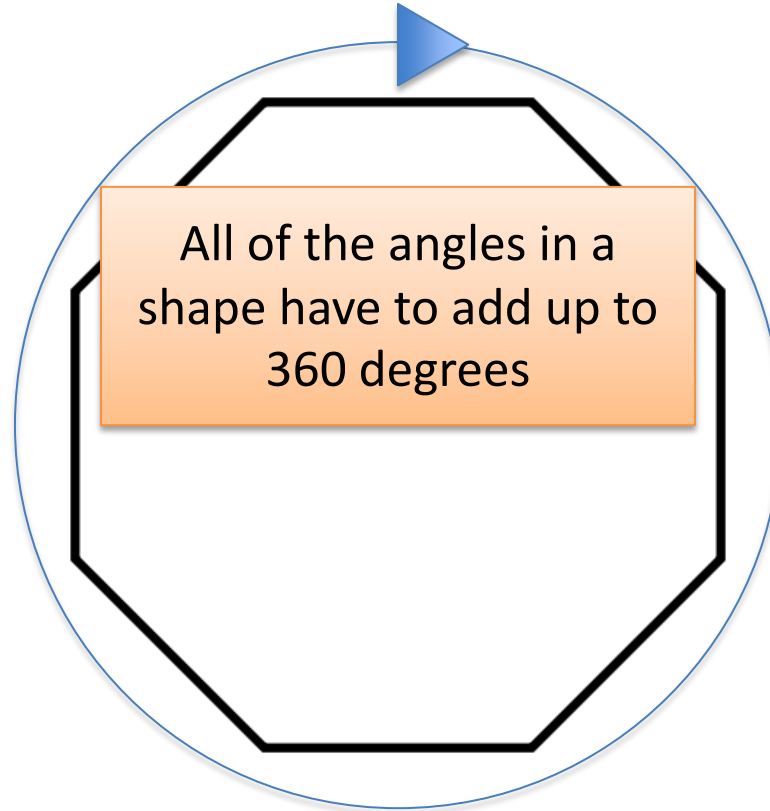
# What if we wanted to draw an octagon?

```
for sideNum in range(5):  
    alex.forward(100)  
    alex.left(72)
```

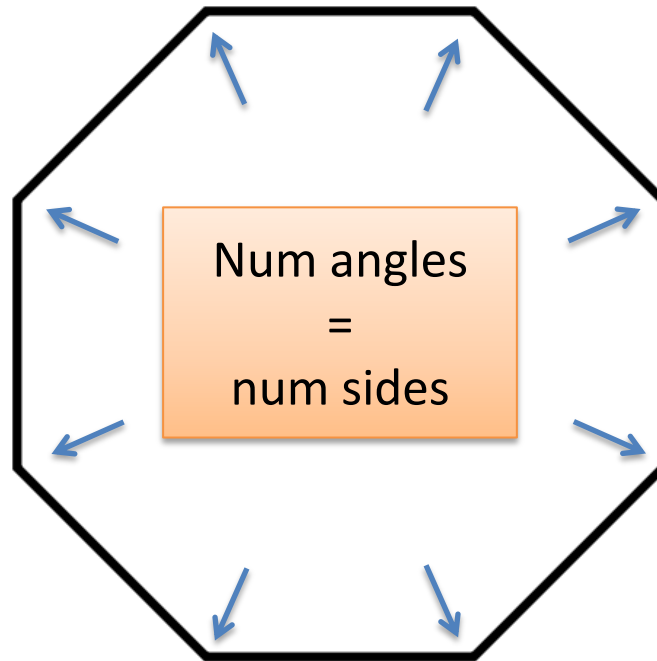
...and this angle has to  
change.



# What if we wanted to draw an octagon?



# What if we wanted to draw an octagon?



# What if we wanted to draw an octagon?

What if we could write the number of sides down and just use that to decide the number of laps and to calculate the angle to turn?



# Variables





# Code to draw an octagon

```
numberOfSides = 8
```

```
for sideNum in range(numberOfSides):  
    alex.forward(100)  
    alex.left(360/numberOfSides)
```



# Code to draw an octagon

Now we have a box labelled  
`numberOfSides`

```
numberOfSides = 8
```

```
for sideNum in range(numberOfSides):  
    alex.forward(100)  
    alex.left(360/numberOfSides)
```



# Code to draw an octagon

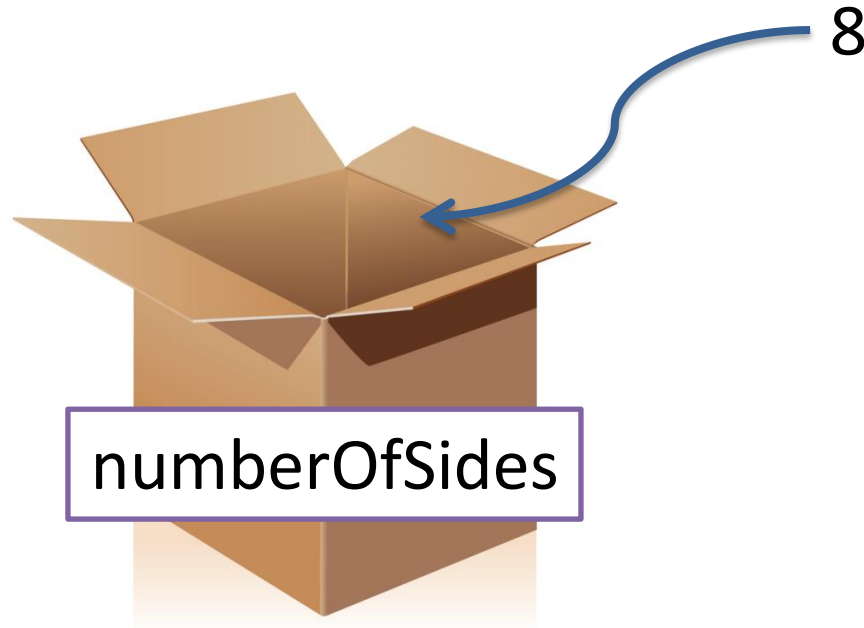
```
numberOfSides = 8
```

This puts 8 into the box

```
for sideNum in range(numberOfSides):  
    alex.forward(100)  
    alex.left(360/numberOfSides)
```



# Code to draw an octagon



```
numberOfSides = 8
```



# Code to draw an octagon

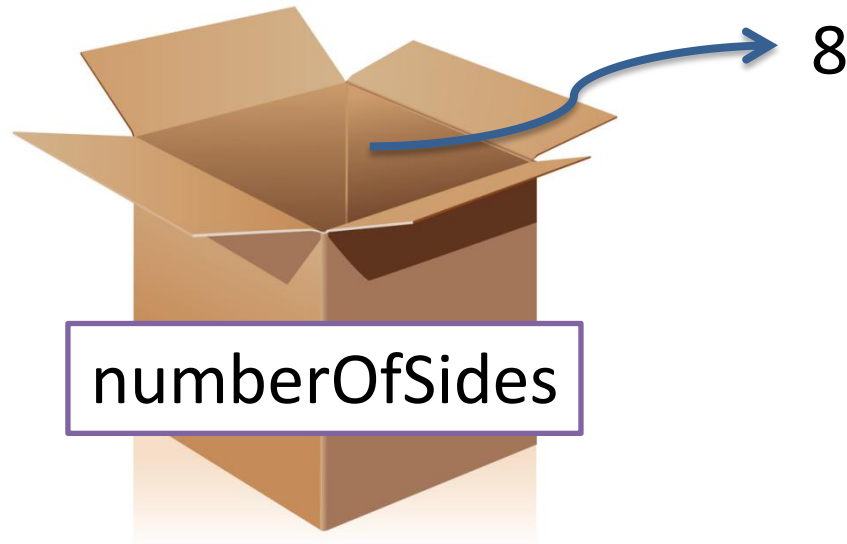
```
numberOfSides = 8
```

This grabs whatever is in the box (in this case, 8)

```
for sideNum in range(numberOfSides):  
    alex.forward(100)  
    alex.left(360/numberOfSides)
```



# Code to draw an octagon



```
range(numberOfSides)
```



# What do we have to do to change the number of sides in our shape?

There's just one line of code to change now!



Can you get alex to draw a  
shape with ten sides?

How about a circle?





# We have used variables already!

a new Turtle



```
alex = turtle.Turtle()
```



# We have used variables already!



the turtle we  
made before



ask the turtle  
to move

```
alex.forward(150)
```



# We have used variables already!



the turtle we  
made before

1

ask the turtle  
to move

1

```
alex.forward(150)
```



# We have used variables already!



the turtle we  
made before



ask the turtle  
to move

2

```
alex.forward(150)
```

2



# More Turtle Commands

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

# Try these commands – experiment and see what designs you can make!

```
alex.shape("turtle")
alex.reset()
alex.backward(someNumber)
alex.up()
alex.color("red")
alex.pensize(someNumber)
alex.penup()
alex.pendown()
alex.stamp()
alex.circle(someNumber)
```



# Type this code and see what it does!

```
for aColor in ["red", "blue", "yellow",  
               "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```



# Using a color variable in a loop

This variable will  
change every lap

```
for aColor in ["red", "blue", "yellow",  
               "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```





# Using a color variable in a loop

Instead of referring to a lap with a number, this time we'll use a color

```
for aColor in ["red", "blue", "yellow",  
              "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```



# Using a color variable in a loop

The for loop will have 5 laps since we have to go through each color one at a time

```
for aColor in ["red", "blue", "yellow",  
              "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```



# Using a color variable in a loop

A word in quotes is called a string –  
it is just text, not a variable

```
for aColor in ["red", "blue", "yellow",  
               "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```



# Using a color variable in a loop

```
for aColor in ["red", "blue", "yellow",  
               "green", "purple"]:  
    alex.color(aColor)  
    alex.forward(100)  
    alex.left(72)
```

Since the value in the aColor box changes each lap, we set a new color to draw with each time



# Using print statements

You can print messages to the console with `print()`. This can help you better understand some code or help find the source of a problem.



# Using print statements

Example:

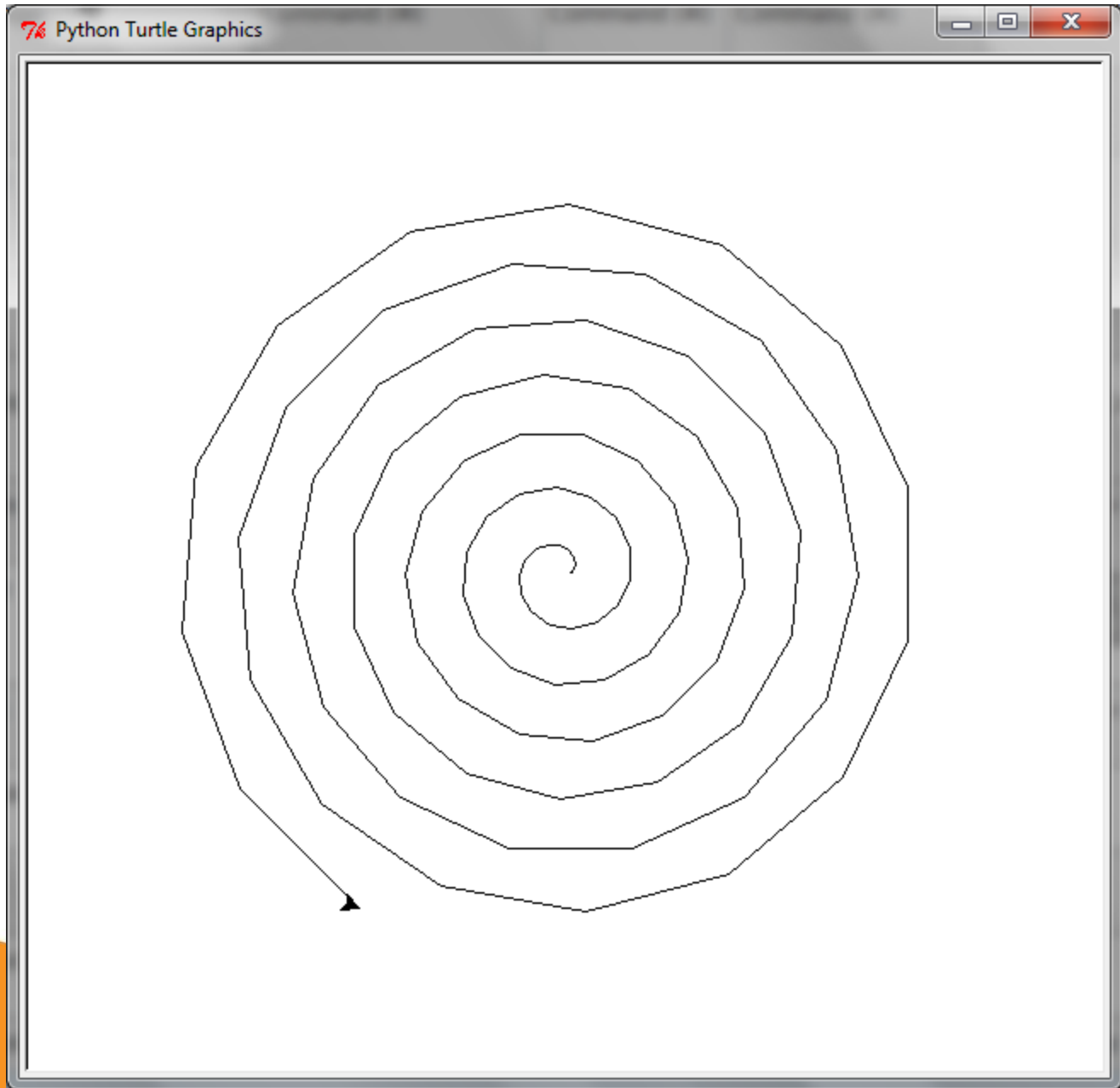
```
print (range (5, 30, 2) )
```



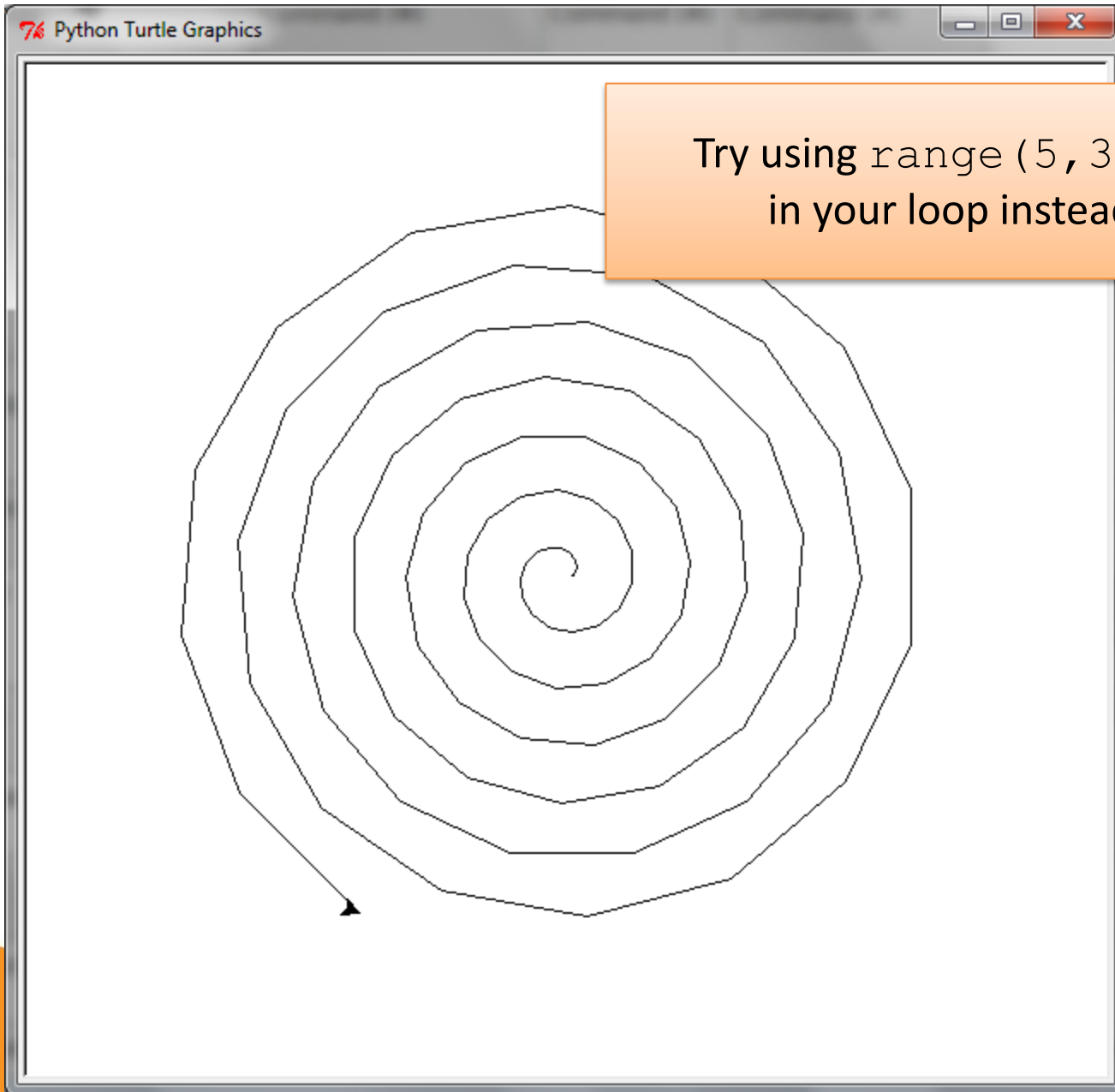
Challenges: Can you draw  
this?

**GO  
CODE  
GIRL**

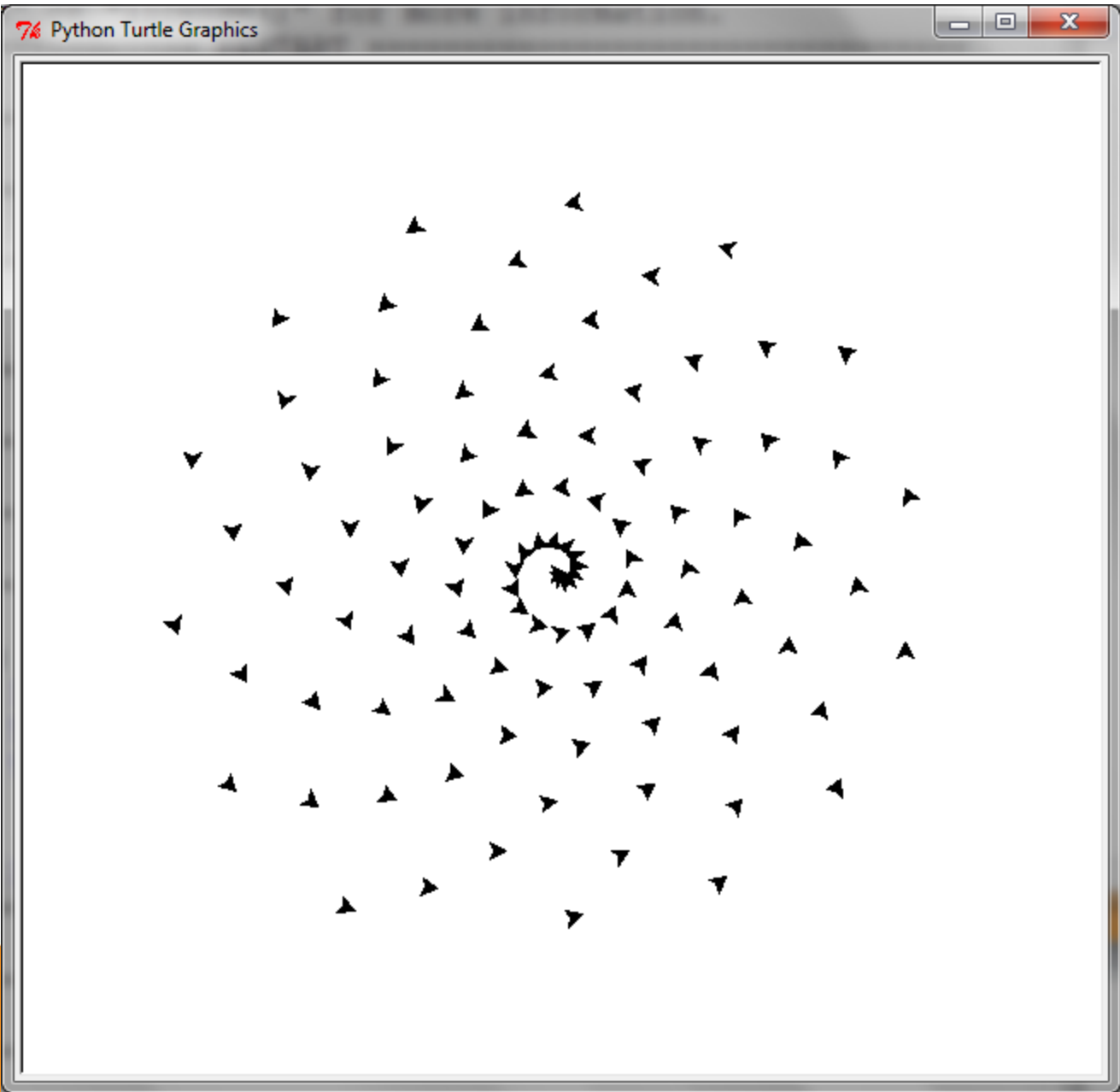
IMAGINE. DESIGN. CREATE.



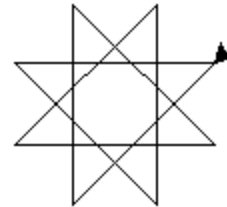




Try using `range(5, 30, 2)`  
in your loop instead!



Try it with a variable number of sides and angle to turn, then change the variables!



# True, False, and If

**GO  
CODE  
GIRL**

IMAGINE. DESIGN. CREATE.

# boolean

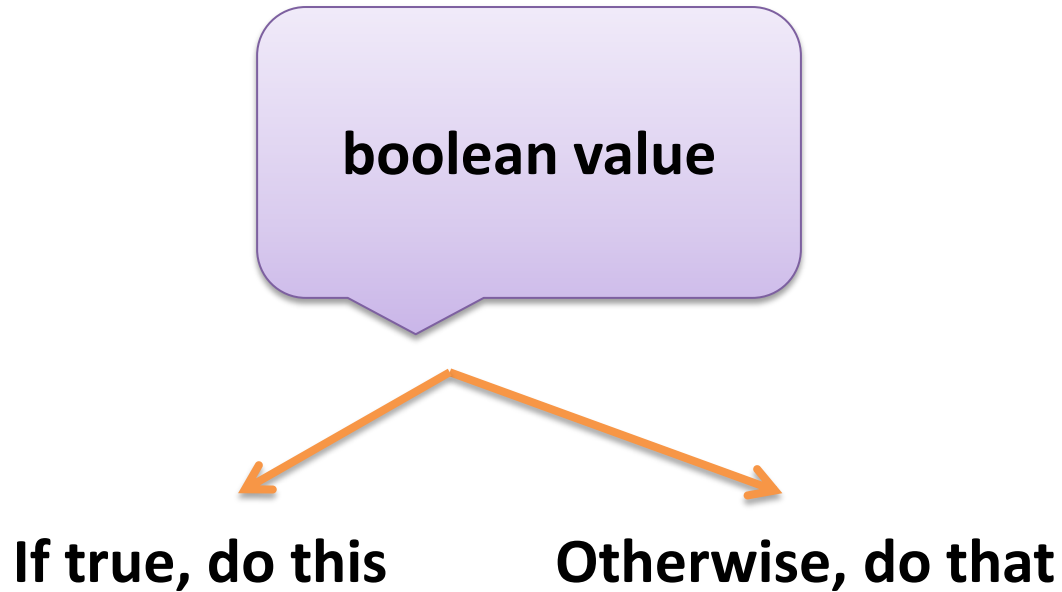
**Yes/  
True**

*or*

**No/  
False**

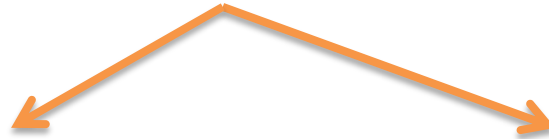


# If/Else Statements



# If/Else Statements

I am sick Friday  
night

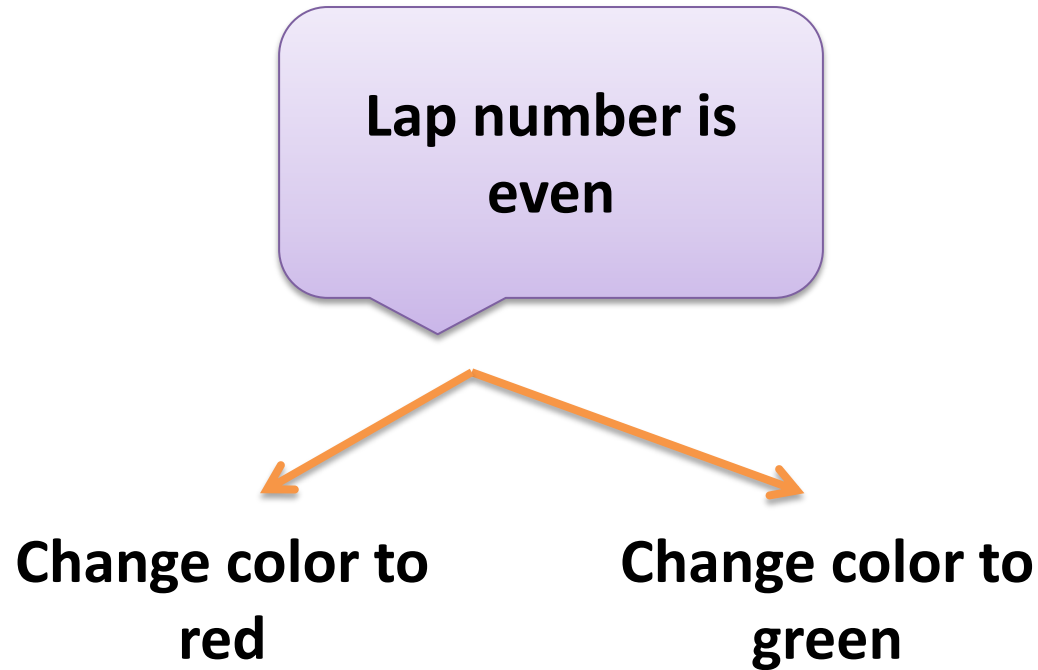


**Yes:**  
Stay home, watch  
TV

**No:**  
Go to the party



# If/Else Statements





# If/Else Statements

“Mod” operator means remainder:

$$5 \% 1 = 0$$

$$5 \% 2 = 1$$

$$5 \% 3 = 2$$

$$5 \% 4 = 1$$

$$5 \% 5 = 0$$



# Type this and see what happens:

```
for sideNum in range(9):  
    if size % 2 == 0:  
        alex.color("red")  
    else:  
        alex.color("green")  
    alex.forward(100)  
    alex.left(225)
```

