Learn to Program With Python

Day 1: Drawing with Turtles

This course uses unofficial curriculum created for Girl Develop It! Ottawa by Gail Carmichael.

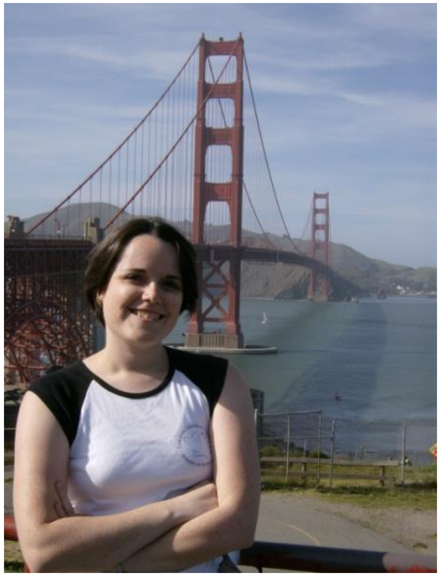
Girl Develop It

GDI

don't be shy. develop it.

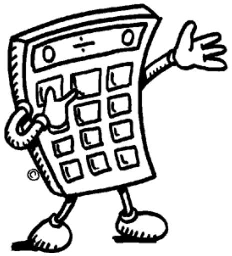
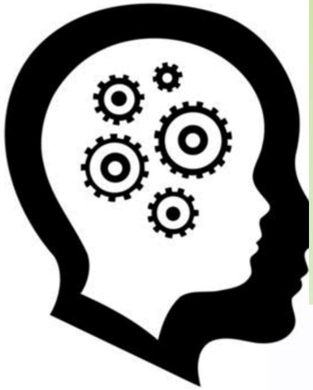
http://www.gailcarmichael.com

About Me



What is computer science?

What is computer science?

Solving problems!

Computational Thinking

Learning to Code is Awesome

https://www.youtube.com/watch?v=nKIu9yen5nc

Thinking Like a Computer

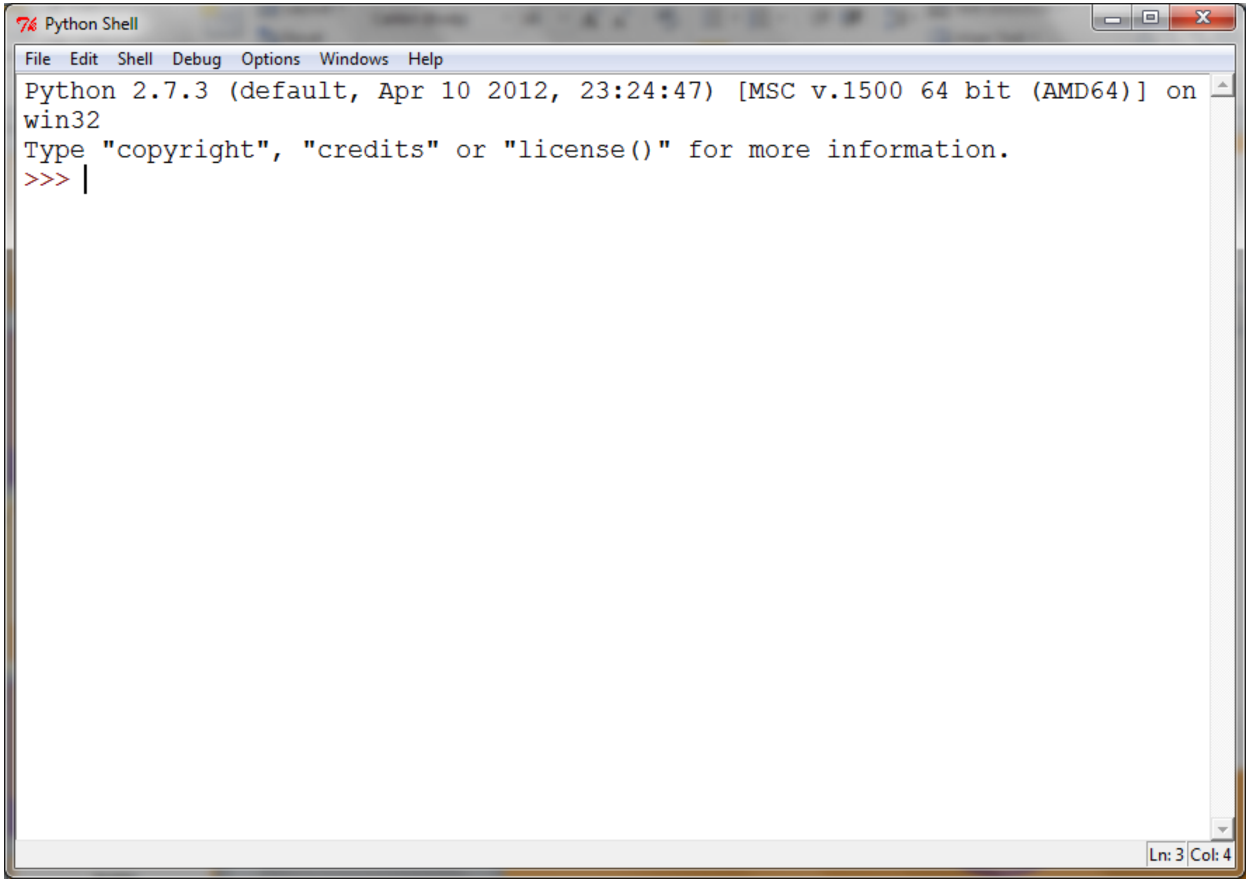
http://csunplugged.org/programming-languages

TURTLE GRAPHICS

Access these slides online:

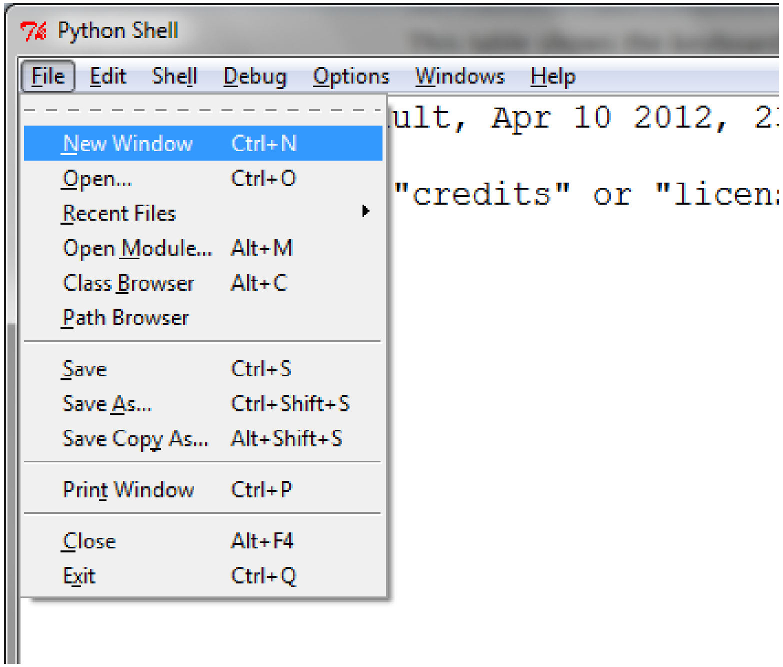
http://gailcarmichael.com/learn-python

Open IDLE



```
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.
>>> 
```

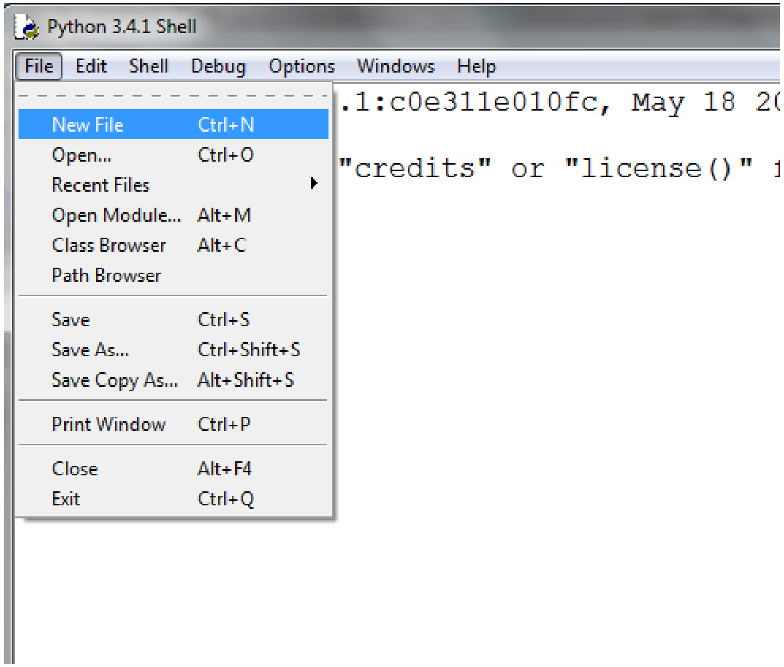
File, New Window



```
ult, Apr 10 2012, 2
"credits" or "licen
```

(Python 2)

File, New File



```
.1:c0e311e010fc, May 18 2
"credits" or "license()" 
```

(Python 3)

Type this in the new window:

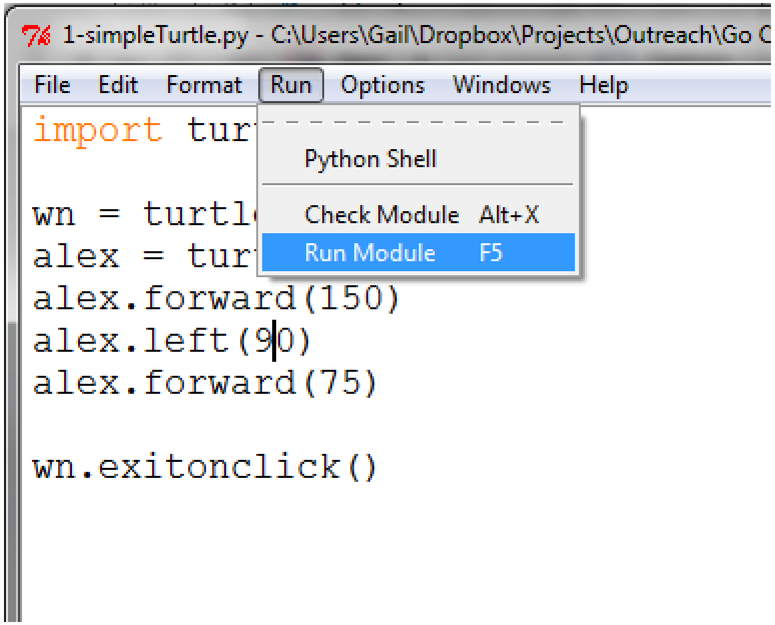
```
import turtle

wn = turtle.Screen()

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

wn.exitonclick()
```

Save, Save As

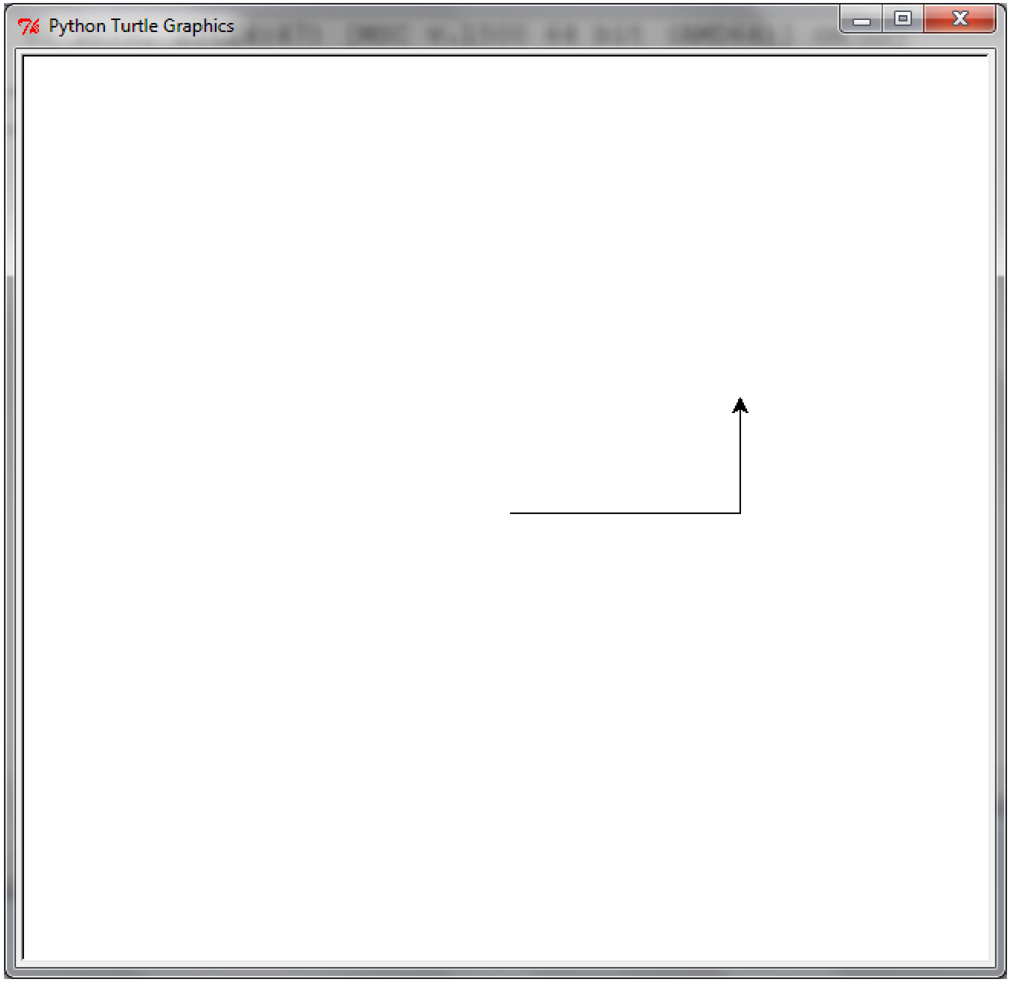
Note: make sure you add `.py` to the end of your file!

(And don’t name the file turtle.py)

Run, Run Module



```
import tur

wn = turtl
alex = tur
alex.forward(150)
alex.left(90)
alex.forward(75)

wn.exitonclick()
```


Python Turtle Graphics

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.Scr

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

wn.exitonclick()
```

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

```
import turtle

wn = turtle.Screen()

alex = turtle.Turtl
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

One way to draw a pentagon...

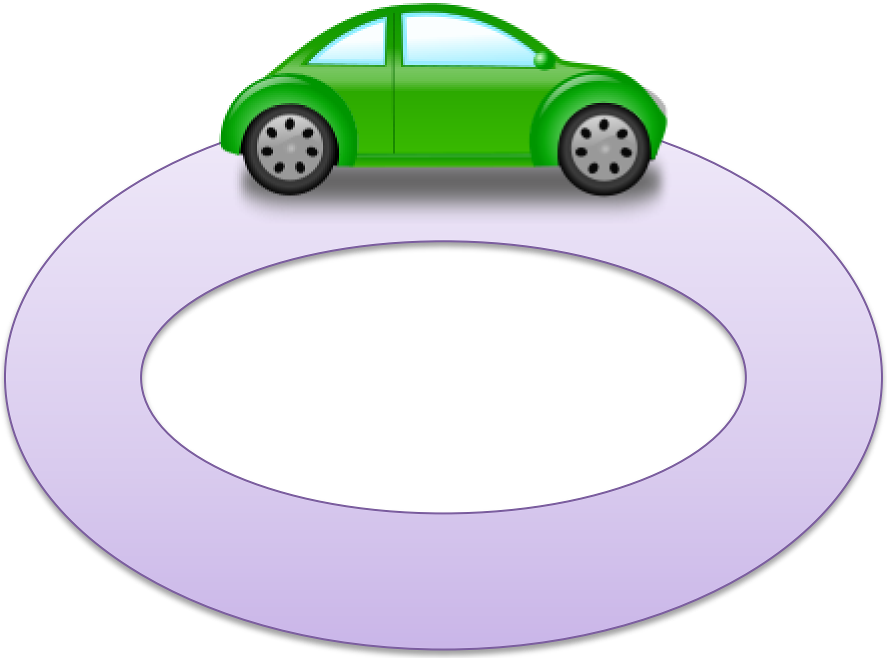
```
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
alex.forward(100)
alex.left(72)
```

One way to draw a pentagon…

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

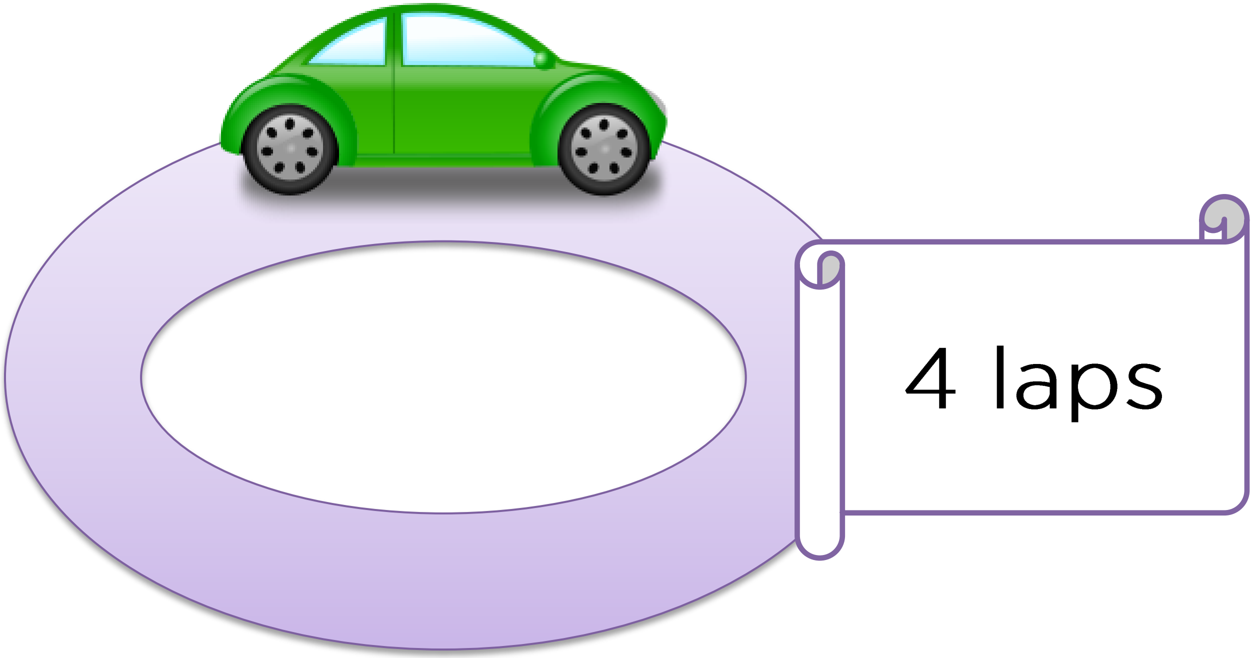
Can we avoid writing the same lines of code over and over?

Loops



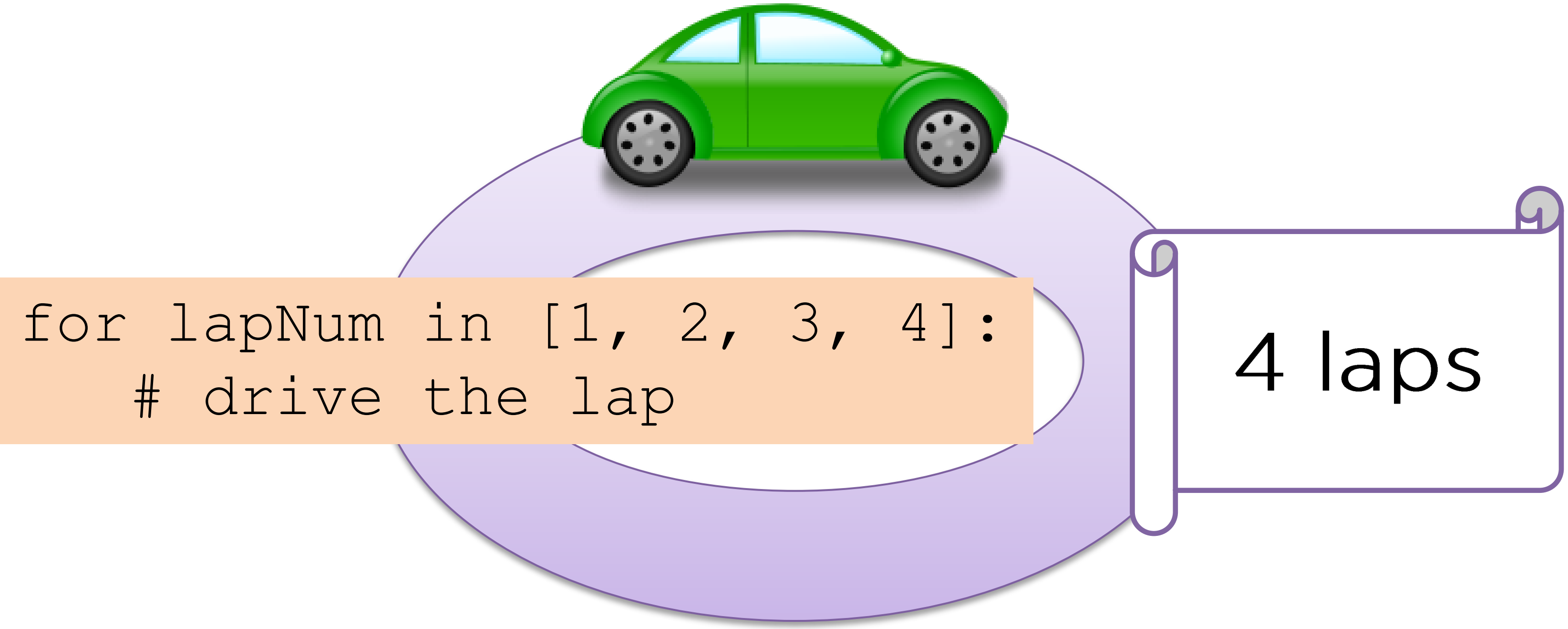
Drive the same track multiple times

for loop



Drive the same track exactly four times

for loop



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

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)
```

Using a for loop to draw a
agon

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 pen

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 pentag

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!

How many lines of code did you have to change?

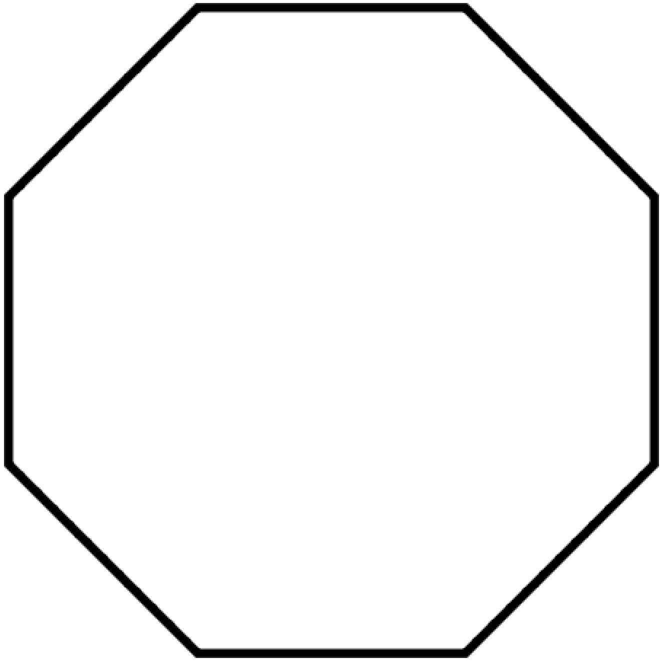
What other cool shapes or designs can you make?

VARIABLES

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…

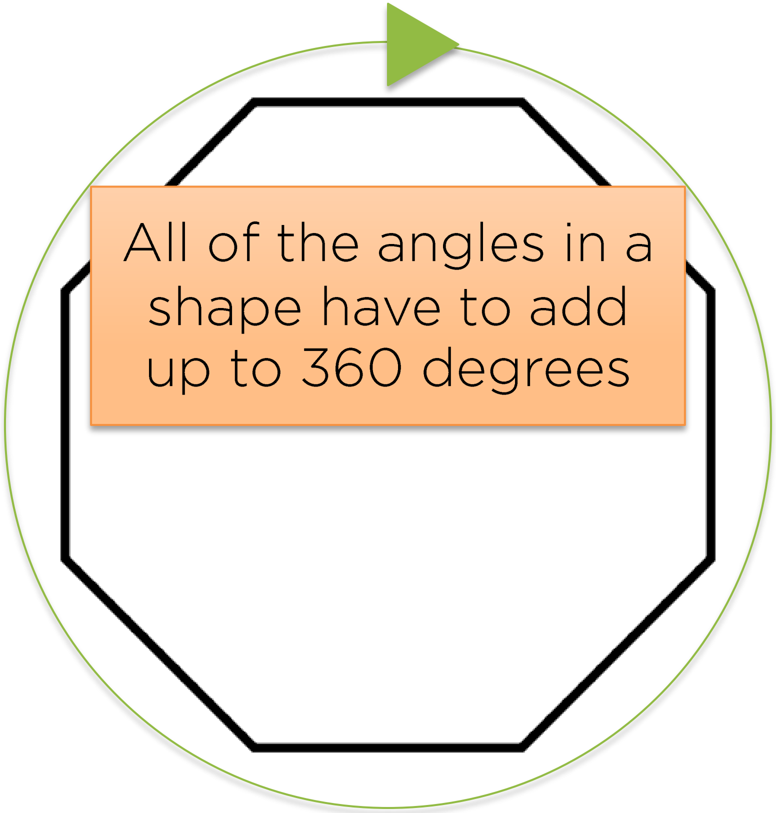
```
for sideNum in range(5):
    alex.forward(100)
    alex.left(72)
```

What if we wanted to draw an octagon?

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

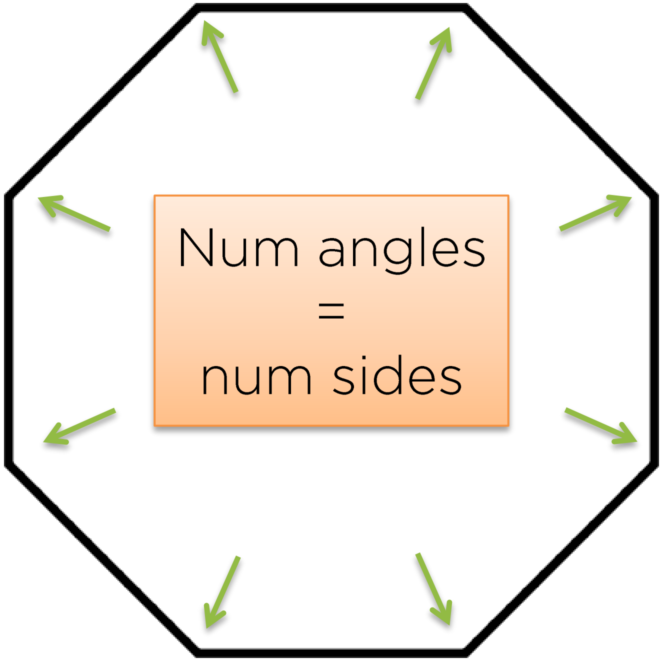
...and this angle has to change.

What if we wanted to draw an octagon?



All of the angles in a shape have to add up to 360 degrees

What if we wanted to draw an octagon?



Num angles
=
num sides

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)
```

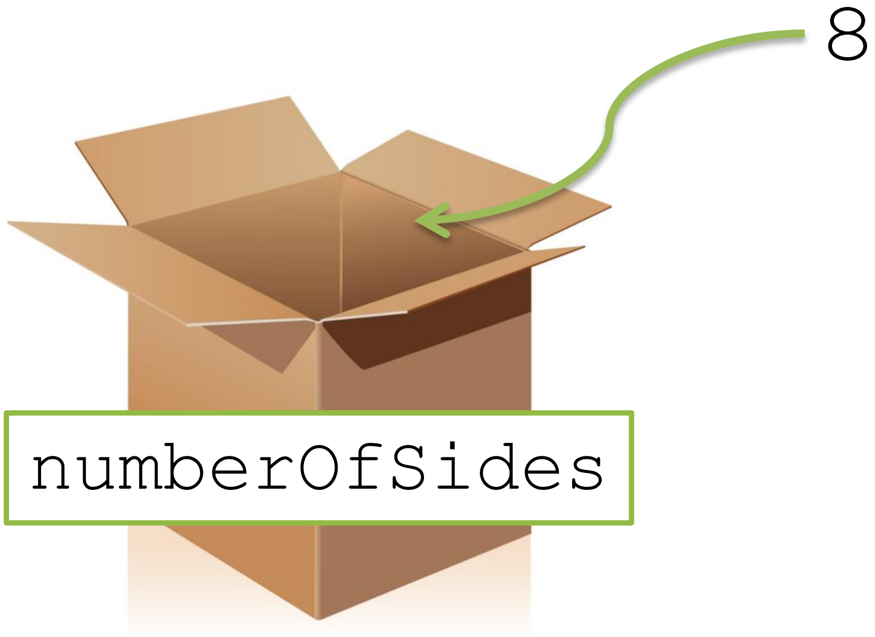
Code to draw an octagon

```
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

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

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

Code to draw an octagon

8
numberOfSides

```
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. Try it!

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

How about a circle?

We have used variables already!

a new Turtle
alex

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

We have used variables already!

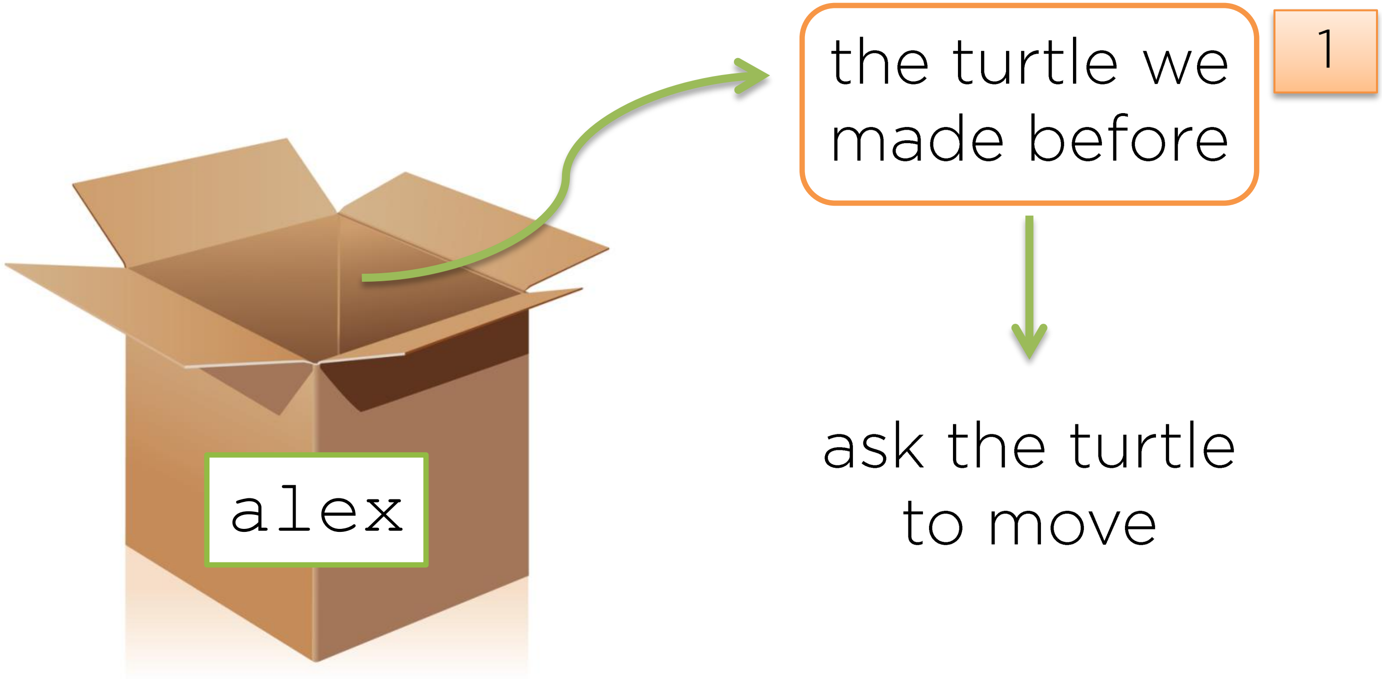
the turtle we made before

↓

ask the turtle to move

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

We have used variables already!



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

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

```
alex.shape(“turtle”)
alex.reset()
alex.shape(“square”)
alex.penup()
alex.pendown()
```

```
alex.up()
alex.backward(someNumber)
alex.color(“red”)
alex.pensize(someNumber)
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

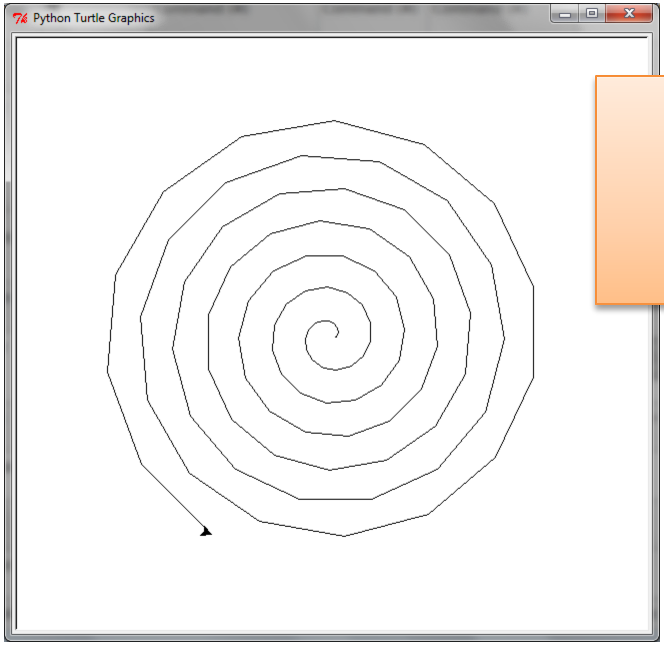
```
for aColor in ["red", "blue", "yellow",
               "green", "purple"]:
    alex.color(aColor)
    alex.forward(100)
    alex.left(72)
```

Using a color variable in a loop

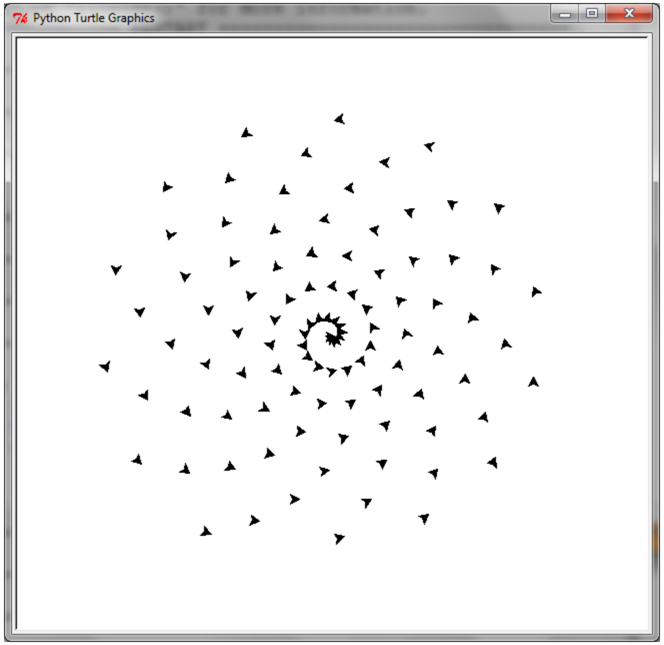
```
for aColor in ["red", "blue", "yellow",
               "green", "purple"]:
    alex.color(aColor)
    alex.forward(100)
    alex.left(72)
```

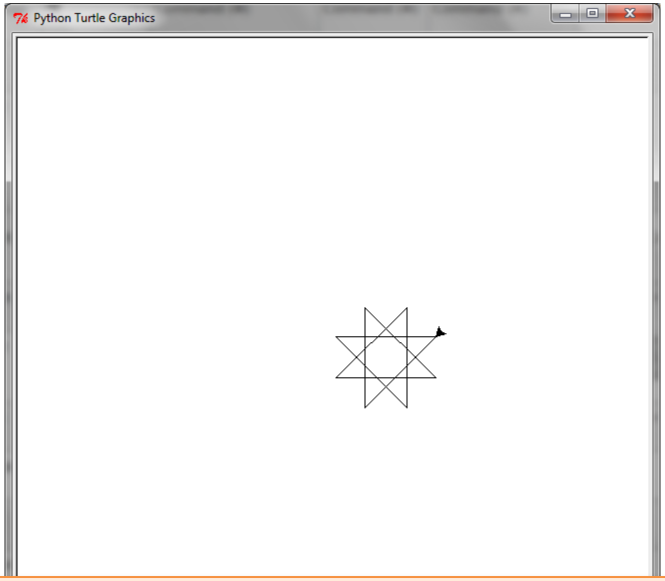
Since the value in the `aColor` box changes each lap, we set a new color to draw with each time

CHALLENGES: CAN YOU DRAW THIS?



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





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

TRUE, FALSE, AND IF

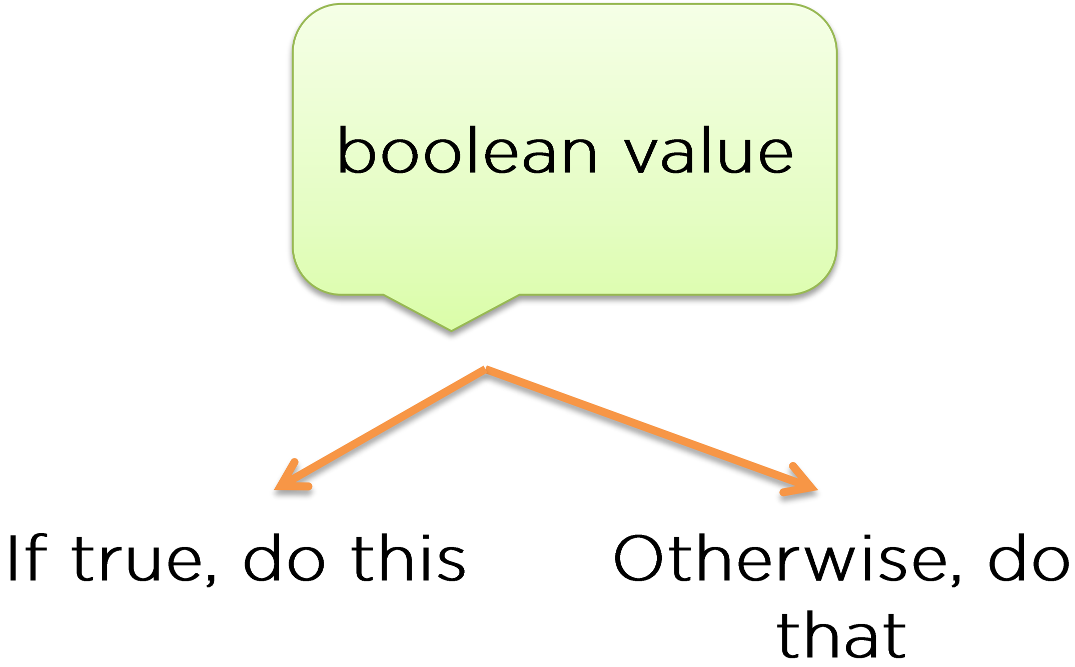
boolean

Yes/
True

or

No/
False

If/Else Statements

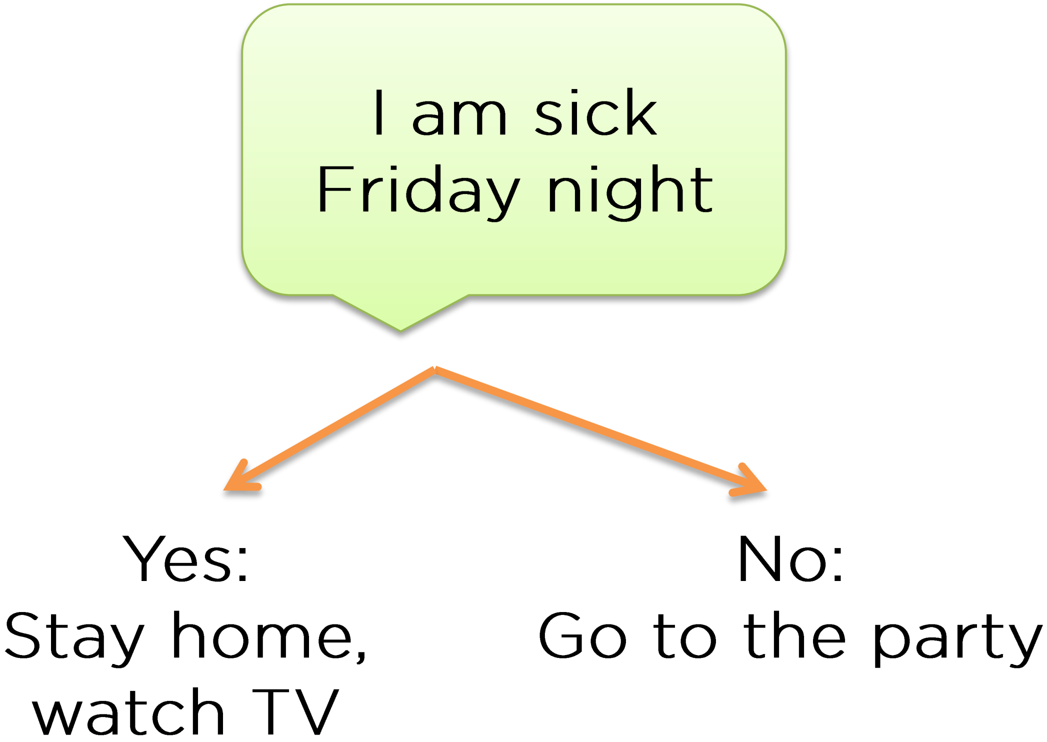


boolean value

If true, do this

Otherwise, do that

If/Else Statements

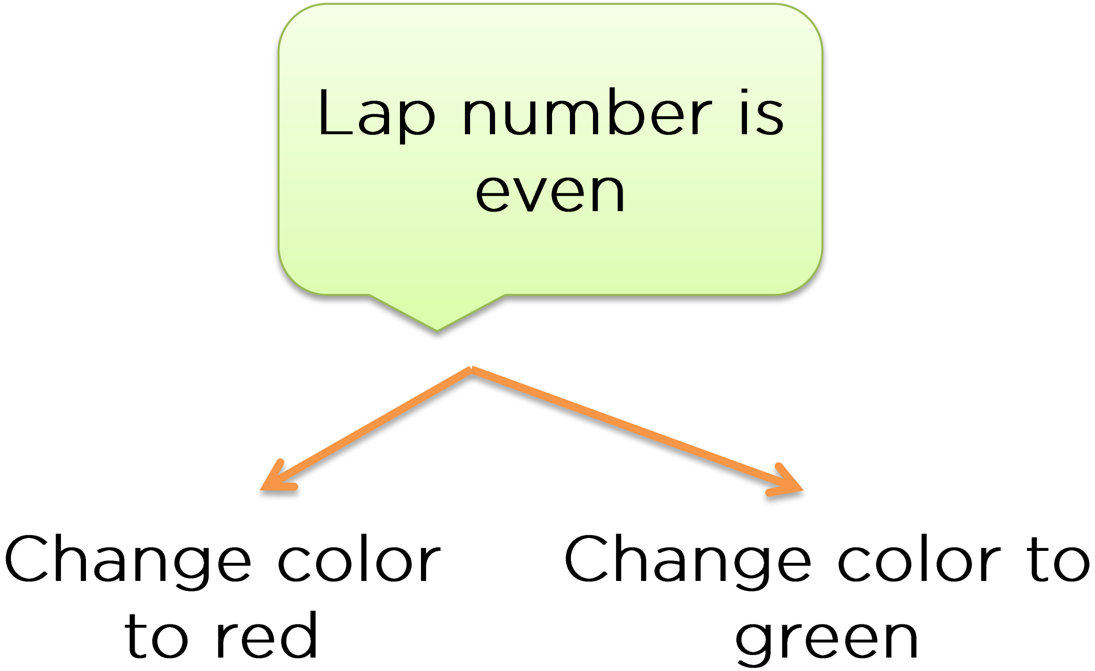


I am sick Friday night

Yes:
Stay home,
watch TV

No:
Go to the party

If/Else Statements



Lap number is even

Change color to red

Change color to green

If/Else Statements

“Mod” operator (%) means remainder:

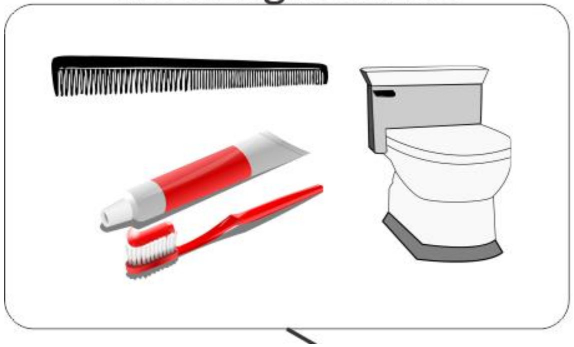
5 % 1 = 0
5 % 2 = 1
5 % 3 = 2
5 % 4 = 1
5 % 5 = 0

Type this and see what happens:

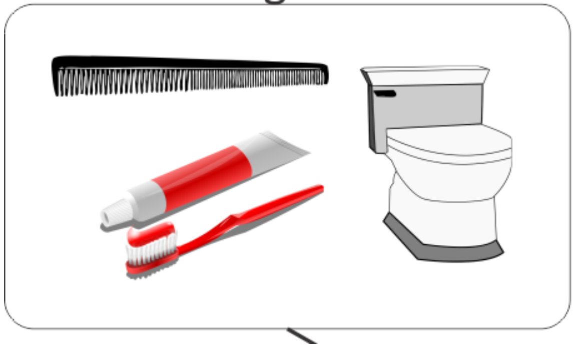
```
for sideNum in range(9):
    if sideNum % 2 == 0:
        alex.color("red")
    else:
        alex.color("green")
    alex.forward(100)
    alex.left(225)
```

FUNCTIONS

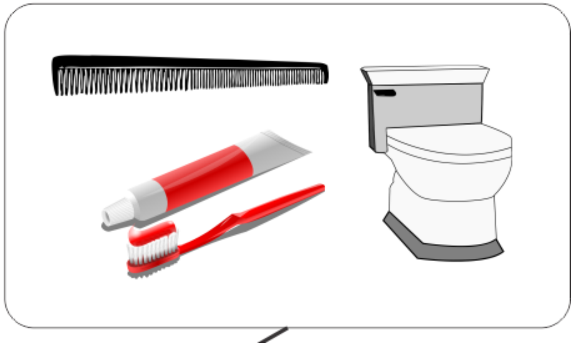
morningRoutine

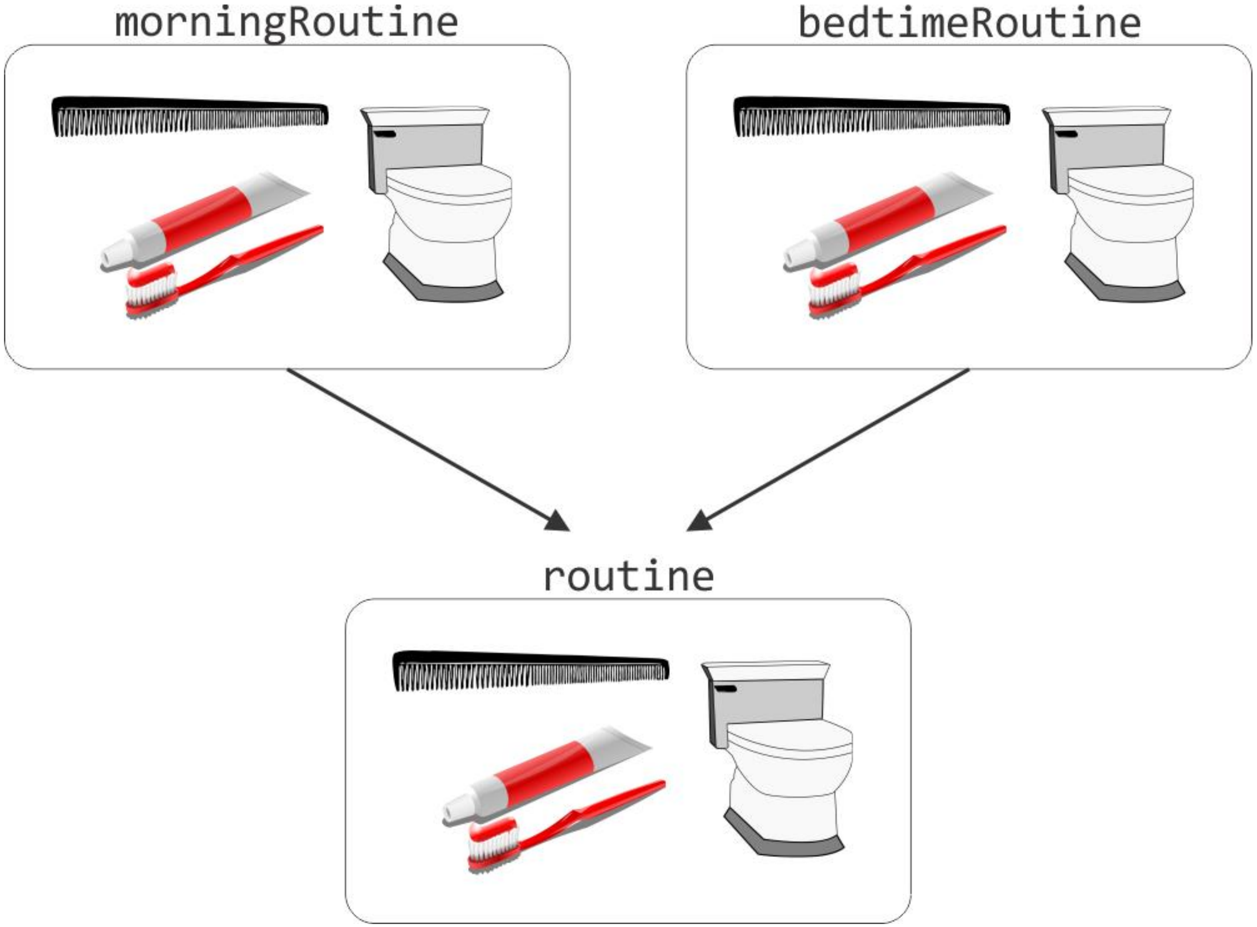


morningRoutine



bedtimeRoutine





`morningRoutine`

`bedtimeRoutine`

`routine`

Defining our own routine

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

Defining our own routine

Indicates we want to start our routine (aka “function”) definition

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

Defining our own routine

Our routine will be called `drawSquare`

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

Defining our own routine

Routine names are followed by brackets

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

Defining our own routine

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

Indentation indicates what code to run when we run the routine (i.e. “call the function”)

Defining our own routine

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

This code will never run until we ask it to

Running the routine

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare()
```

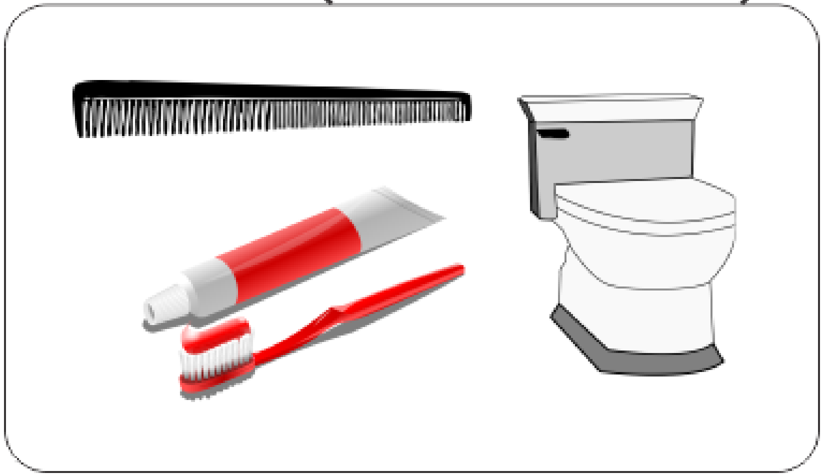
Running the routine

```
def drawSquare():
    alex.penup()
    alex.goto(50, 50)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)
```

```
drawSquare()
```

Run the routine (i.e. “call the function”)

```
routine(doThisFirst)
```



Customizing the routine

```
def drawSquare(x, y):
    alex.penup()
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare(50, 50)
drawSquare(200, 200)
```

Customizing the routine

Routine parameters: variables that will be filled when we run the routine

```
def drawSquare(x, y):
    alex.penup()
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare(50, 50)
drawSquare(200, 200)
```

Customizing the routine

Parameters can be used inside the routine as if they were regular variables

```
def drawSc
    alex.p
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare(50, 50)
drawSquare(200, 200)
```

Customizing the routine

```
def drawSquare(x, y):
    alex.penup()
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
                    (50)
                    0)

drawSquare(50, 50)
drawSquare(200, 200)
```

The drawSquare routine can now be called with specific values for the parameters

Customizing the routine

```
def drawSquare(x, y):
    alex.penup()
    alex.goto(x, y)
    alex.pendown()
    for side in range(4):
        alex.forward(50)
        alex.right(90)

drawSquare(50, 50)
drawSquare(200, 200)
```

When called a second time, all new values are used for the parameters

Exercises

Where have you already been calling previously-defined routines (“functions”)?

After adding parameters to the routine (“function”) definition, can you run the routine without providing any values for those parameters?

What happens if you move the calls to your routine above the function definition?

Can you add a parameter to `drawSquare` called `size`, and then draw two squares of different sizes?

Can you add a parameter for colour so you can draw two squares of different colors?

Challenge

Draw a complex picture with repeating elements by creating functions for the individual pieces!

For example, if you draw a car, you can have a function for the wheels so you only have to write the code for them once.

Hint: Sketch the picture on paper first, then break it down into parts. Make a routine for each part.