go CODE GIRL
ALLEZ CODER LES FILLES

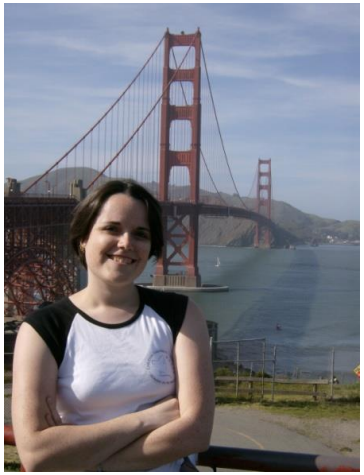
uOttawa and Carleton 2015

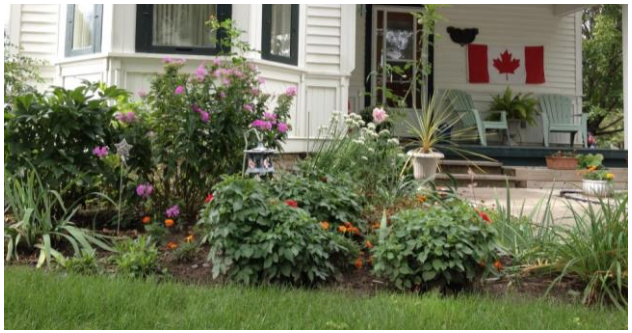
The World is Made with Code

https://www.youtube.com/watch?v=Bo11JJgj1cU

https://www.madewithcode.com/

About Me!





Who Are You?

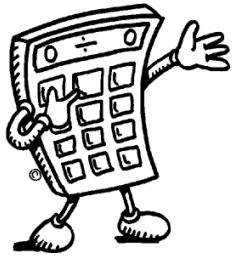
What school do you go to?

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?



Solving Problems!

ONWIE.CA

ONWIE.CA

ONWiE ONTARIO NETWORK OF WOMEN IN ENGINEERING

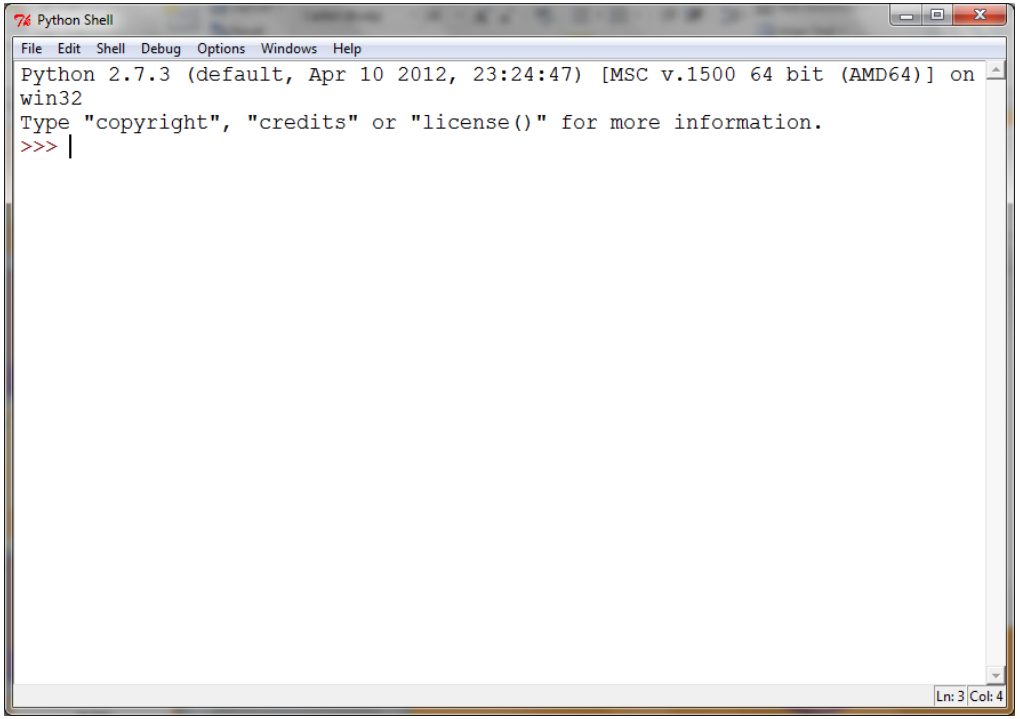
Why Learning to Code is Awesome

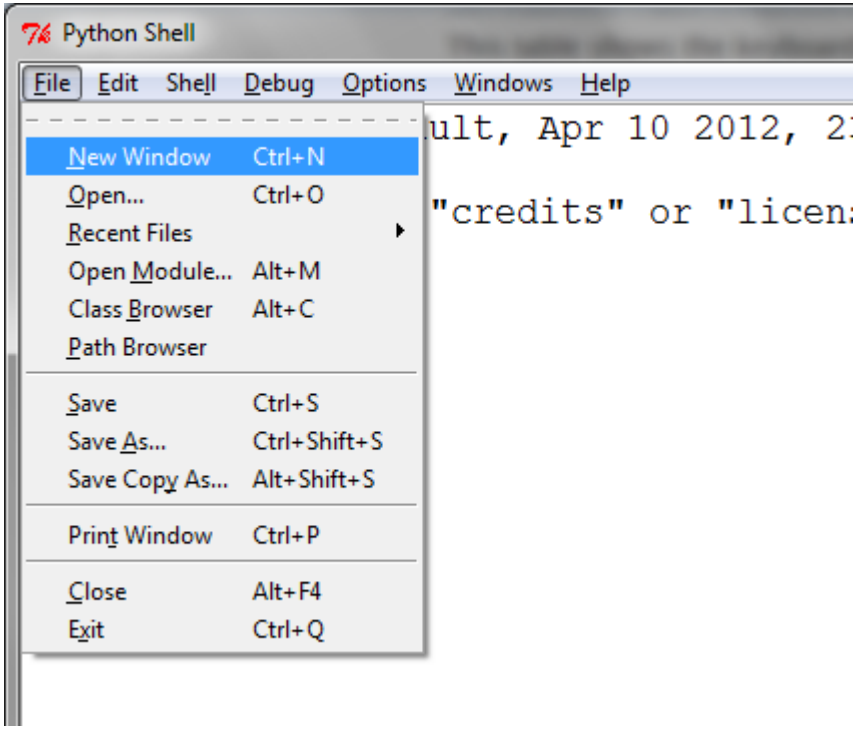
How to Think Like a Computer

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

Turtle Graphics

Open IDLE


Python Shell
File Edit Shell Debug Options Windows Help
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.
>>>
Ln: 3 Col: 4

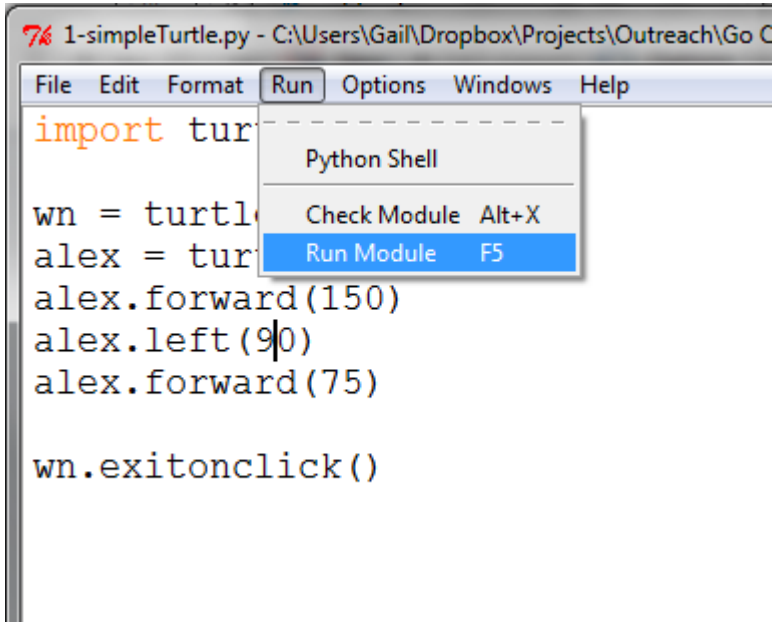

Python Shell
File Edit Shell Debug Options Windows Help
ult, Apr 10 2012, 2
"credits" or "licen
New Window Ctrl+N
Open... Ctrl+O
Recent Files
Open Module... Alt+M
Class Browser Alt+C
Path Browser
Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S
Print Window Ctrl+P
Close Alt+F4
Exit Ctrl+Q

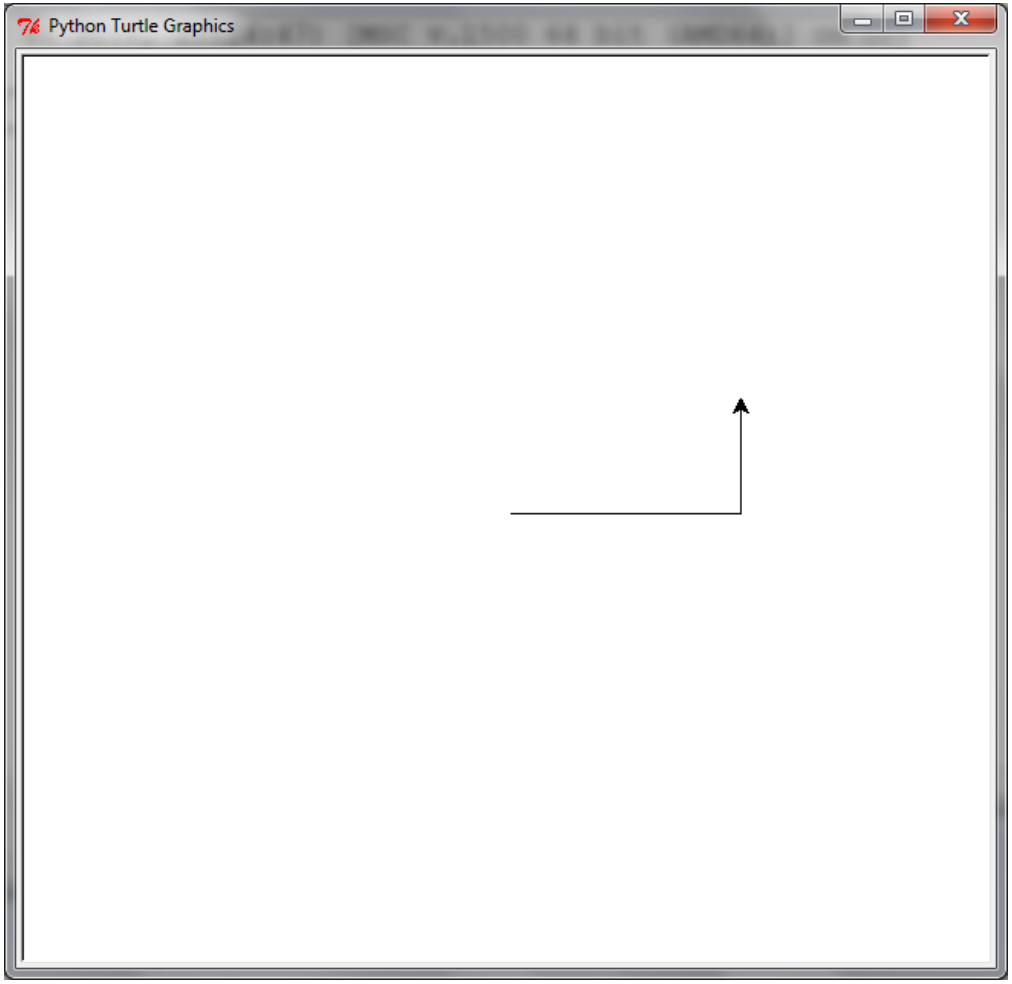
Type this in the new window:

```
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 it `turtle.py`!


1-simpleTurtle.py - C:\Users\Gail\Dropbox\Projects\Outreach\Go C
File Edit Format Run Options Windows Help
import tur
wn = turtl
alex = tur
alex.forward(150)
alex.left(90)
alex.forward(75)
wn.exitonclick()
Python Shell
Check Module Alt+X
Run Module F5


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.S

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

wn.exitonclick()
```

Create a new `Turtle` to draw with and name it (I called mine `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 your turtle's movement code, and add new movements.

What pictures can you make?

Can you get your turtle 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)
```

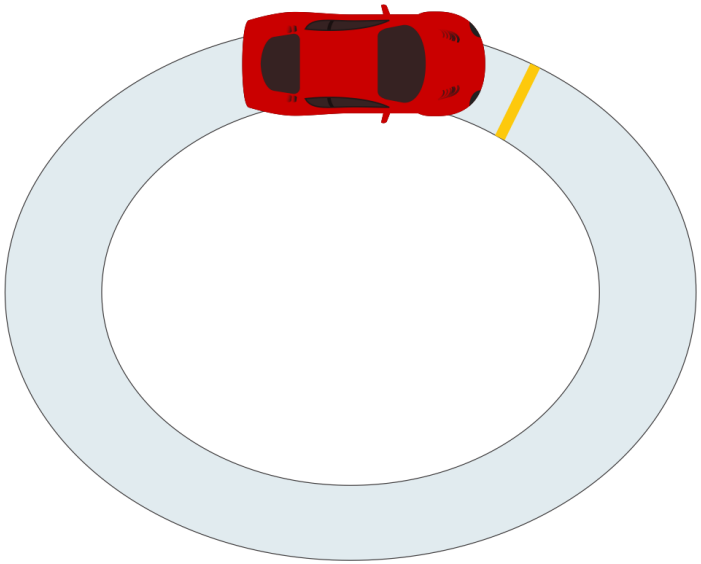
One way to draw a pentagon…

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

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

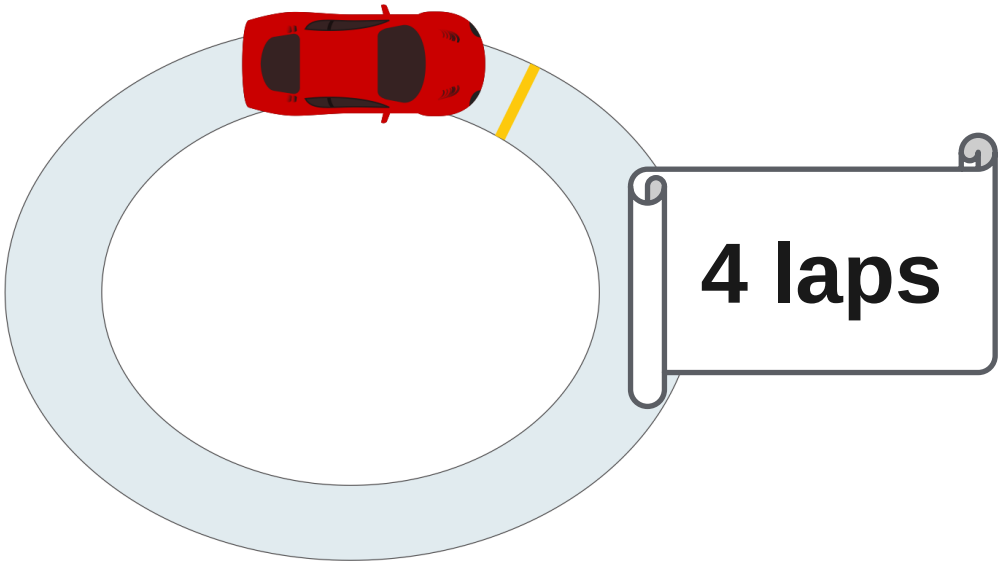
```
alex.left(72)
alex.forward(100)
alex.left(72)
```

Loops



Drive the same track multiple times

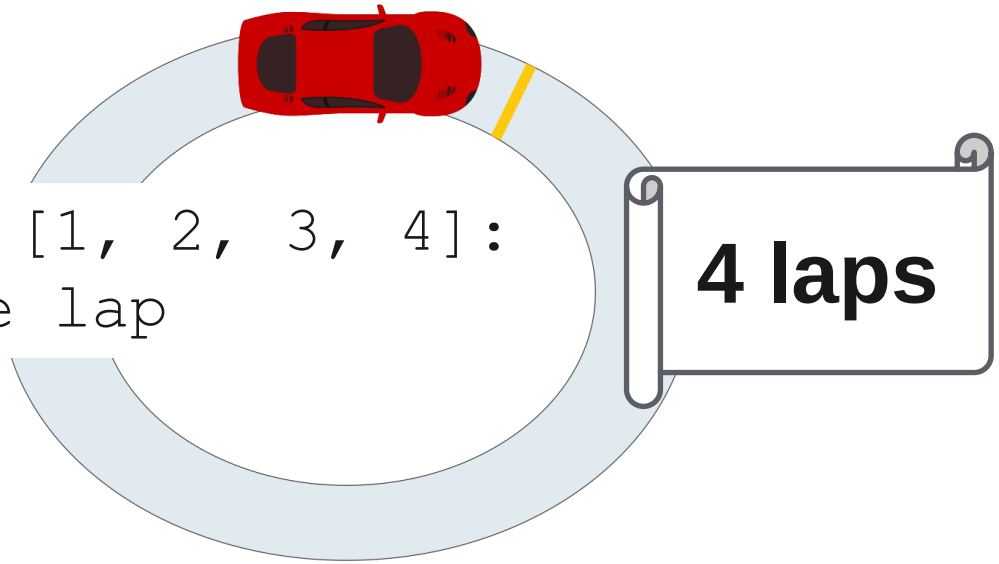
For Loops



Drive the same track exactly four times

For Loops

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



Drive the same track exactly four times

For Loops

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

For loops

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)
```

For Loops

This is a list representing the lap numbers.

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

For Loops

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)
```

For Loops

```
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

For Loops

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

For Loops

```
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)

For Loops

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

For Loops

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

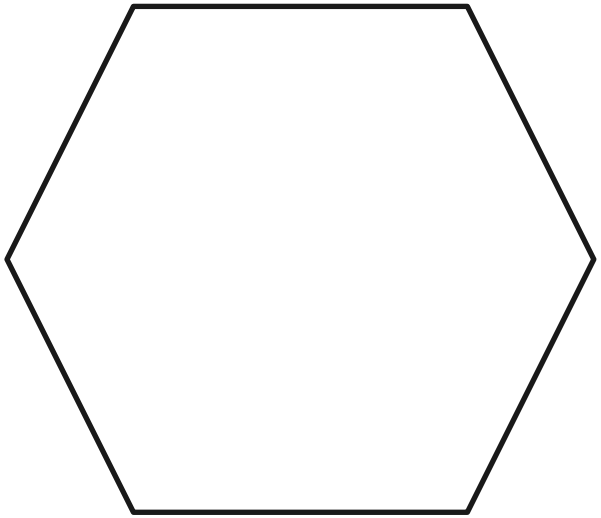
```
for sideNum in range(5):
    alex.forward(100)
    alex.left(72)
```

For Loops

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

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

Try drawing a hexagon instead!



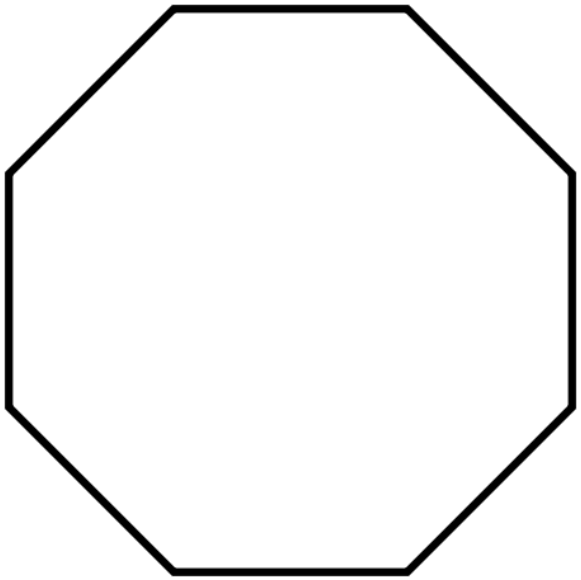
What other cool shapes or designs can you make?

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…

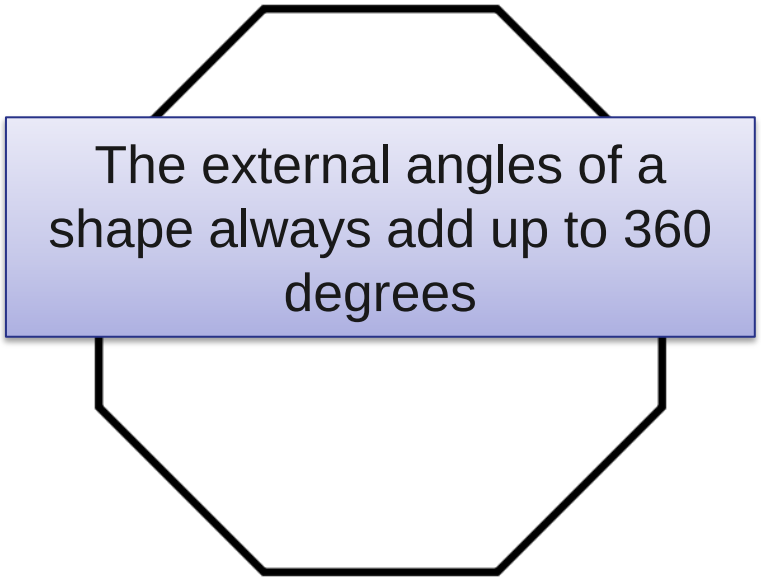
```
for sideNum in range(5):
    alex.forward(100)
    alex.left(72)
```

What if we wanted to draw an octagon?

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

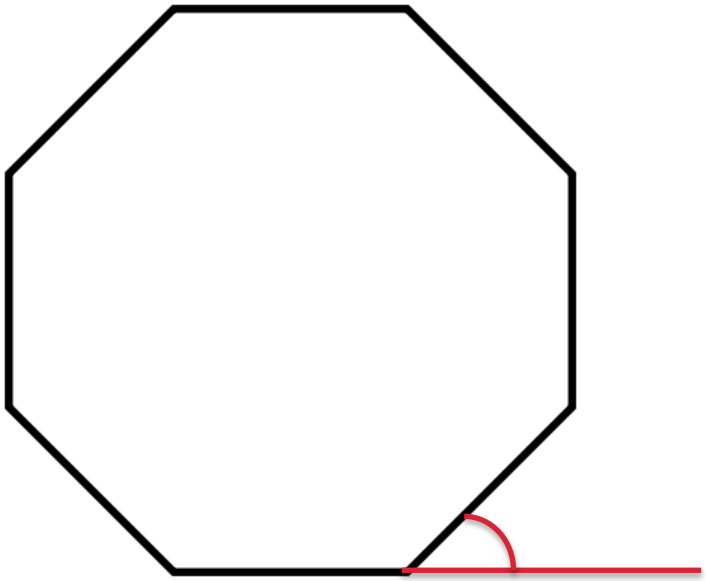
…and this angle has to change.

What if we wanted to draw an octagon?

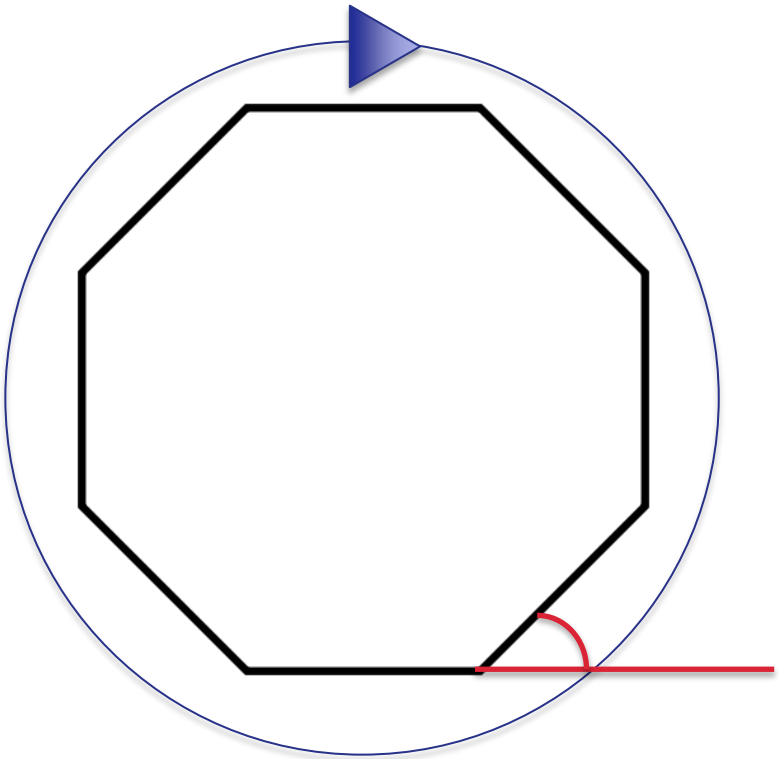


The external angles of a shape always add up to 360 degrees

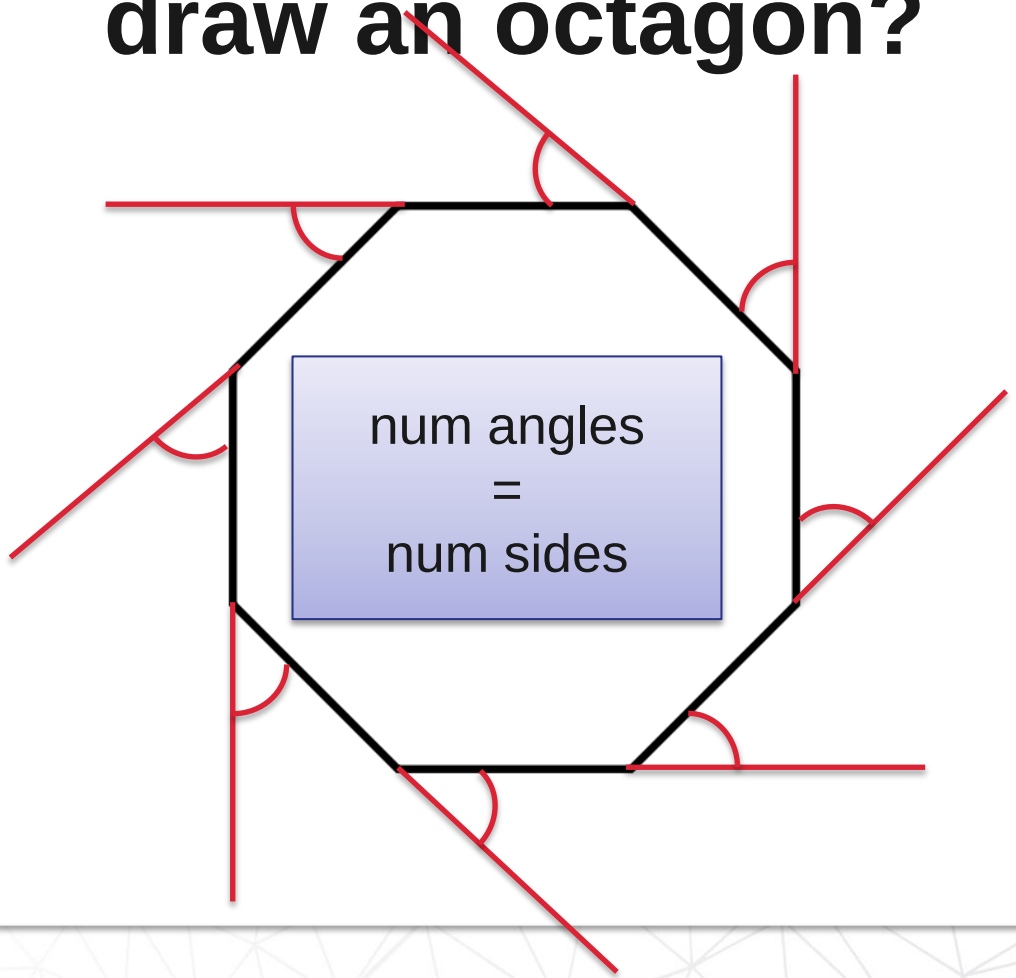
What if we wanted to draw an octagon?



What if we wanted to draw an octagon?



What if we wanted to draw an octagon?



What if we 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)
```

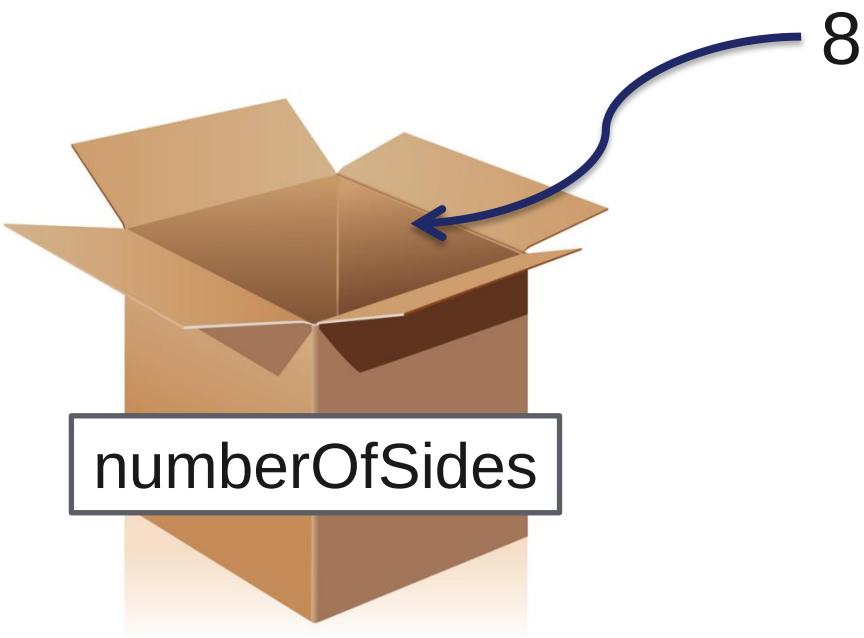
Code to draw an octagon

```
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



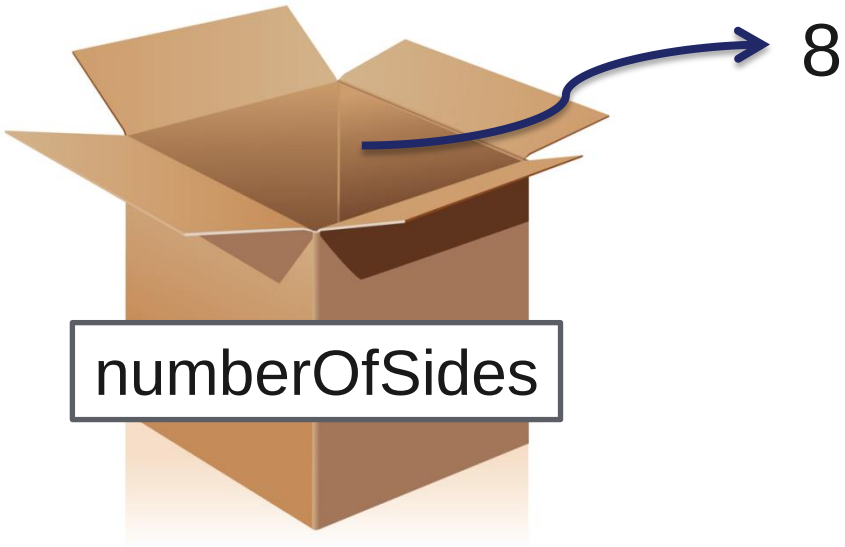

```
numberOfSides = 8
```

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




```
range(numberOfSides)
```

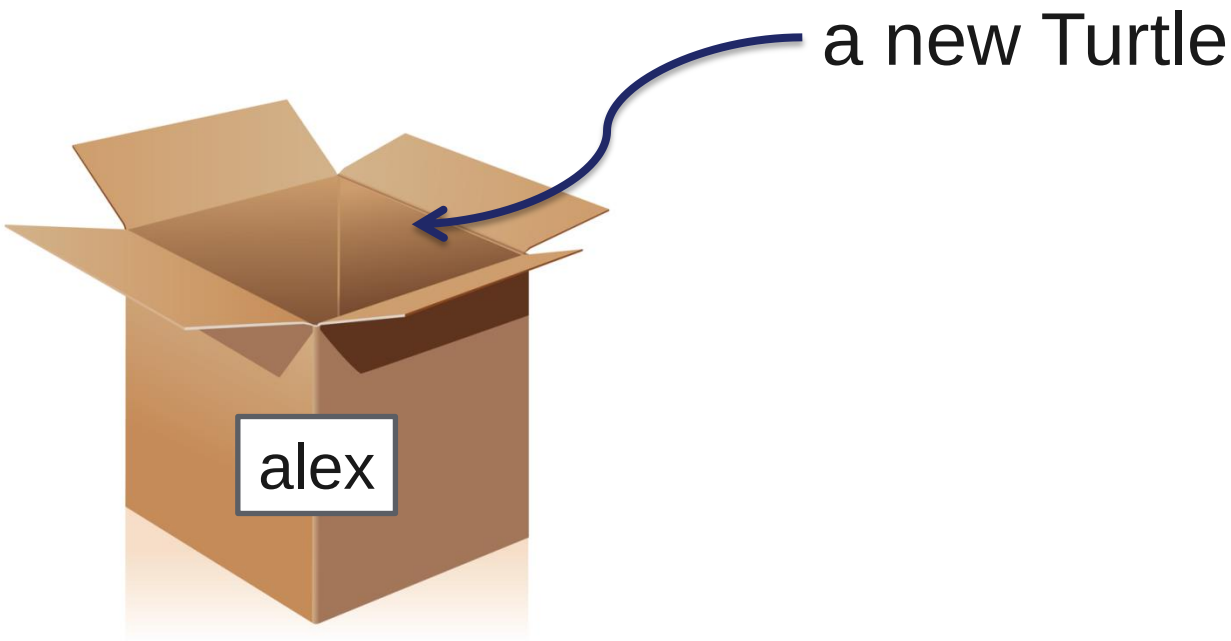
What do we have to do to change the number of sides in our shape?

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

Can you get your turtle to draw a shape with ten sides?

How about a circle?

We have used variables already!




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

We have used variables already!

the turtle we made before

ask the turtle to move

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

More Turtle Commands

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)
```

https://docs.python.org/2.7/library/turtle.html

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.

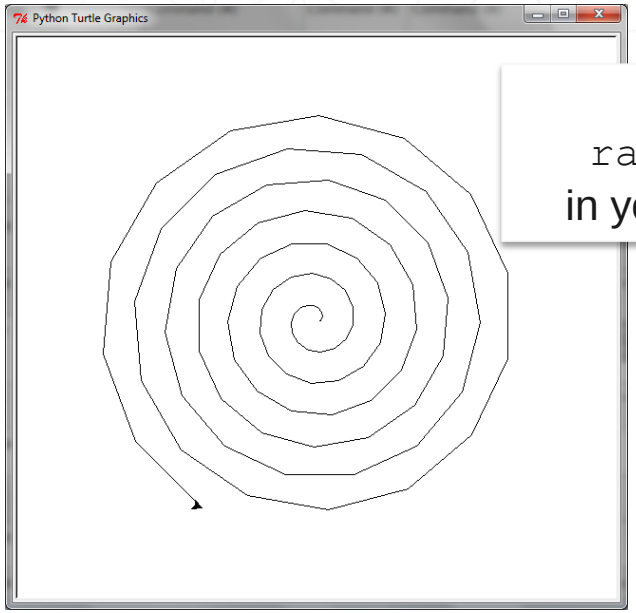
Using print statements

Example:

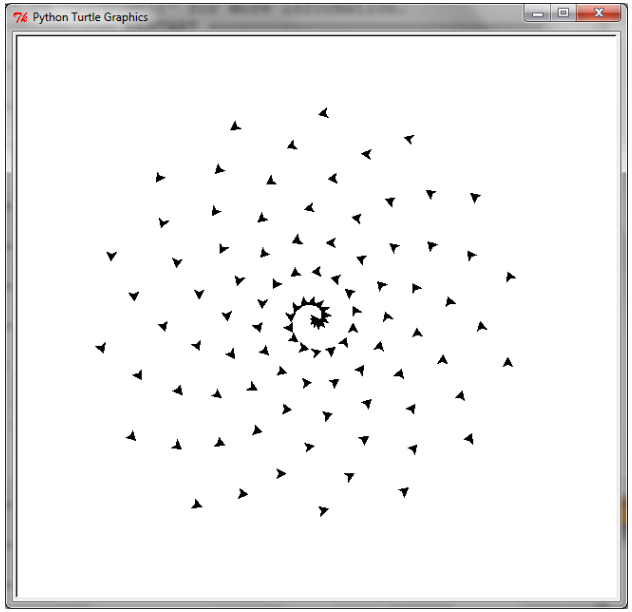
```
print(range(5,30,2))
```

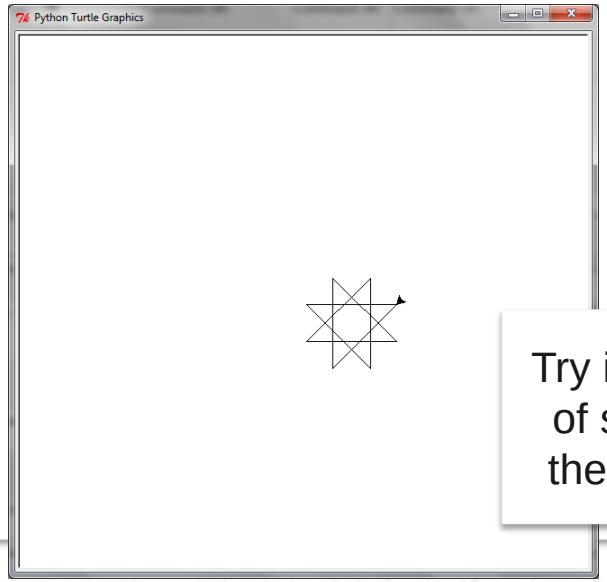
(Note: only works on Python 2)

Challenges



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





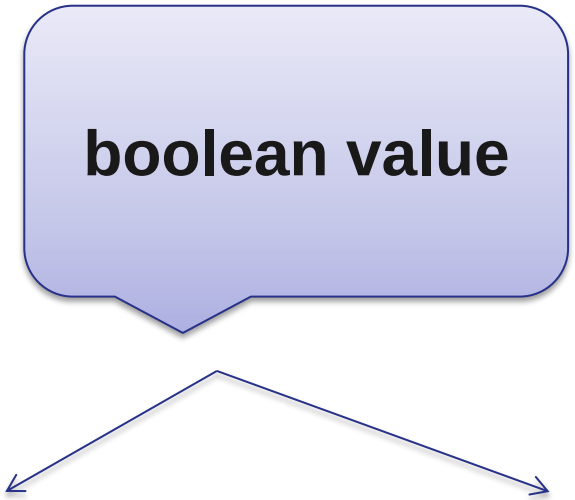
Try it with a variable number of sides and angle to turn, then change the variables!

True, False, If

boolean

Yes/ True *or* **No/ False**

If/Else Statements



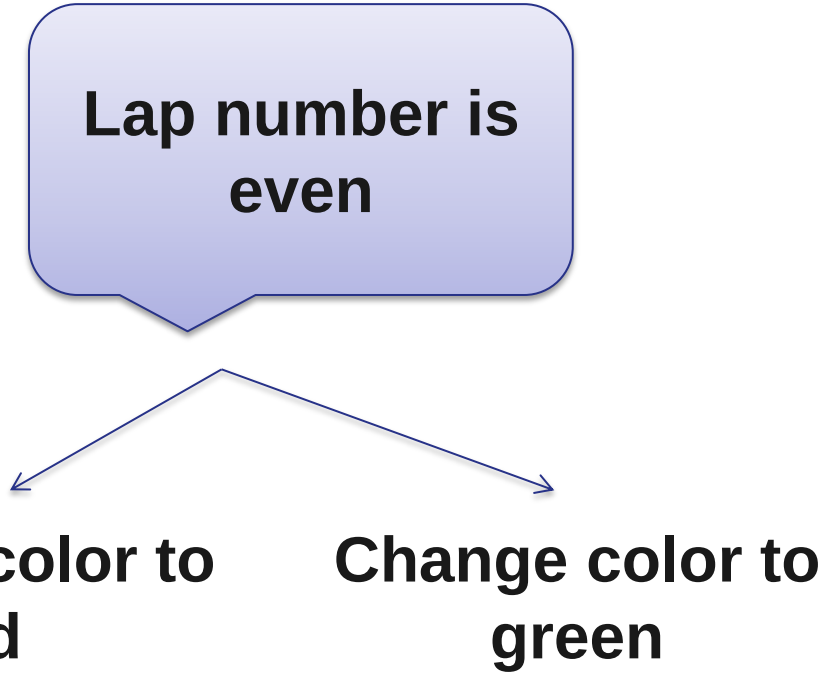
If/Else Statements

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

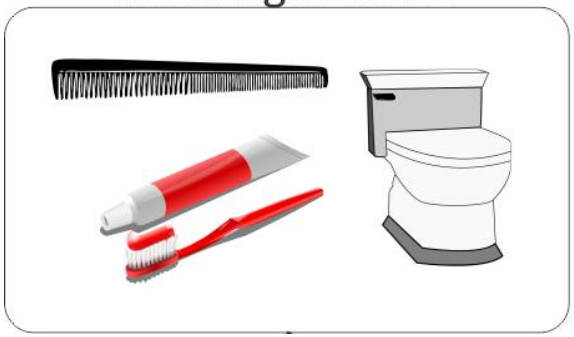
Ask the User a Question

Let's say we have a way to ask the user a yes or no question. How do we check the answer?

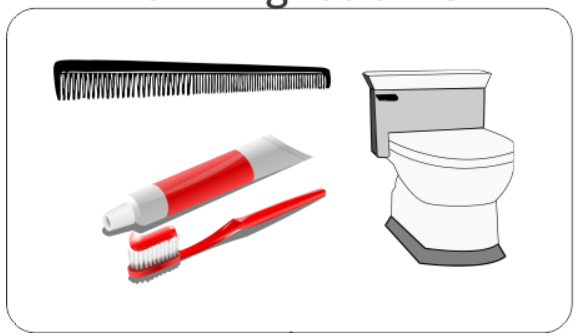
```
import turtle

print("Should we draw with red?")
colorAnswer = raw_input()

numSides = 5

wn = turtle.Screen()
alex = turtle.Turtle()

if colorAnswer == "yes" or colorAnswer == "y":
    alex.color("red")
else:
    alex.color("blue")

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

wn.exitonclick()
```

Functions

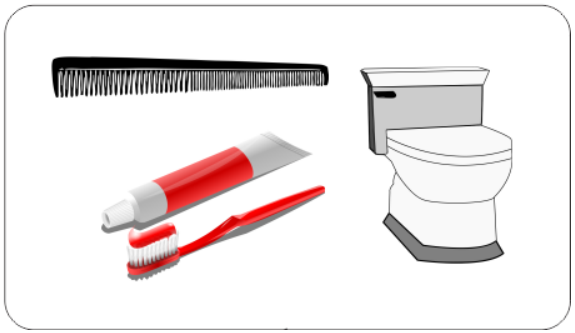
morningRoutine

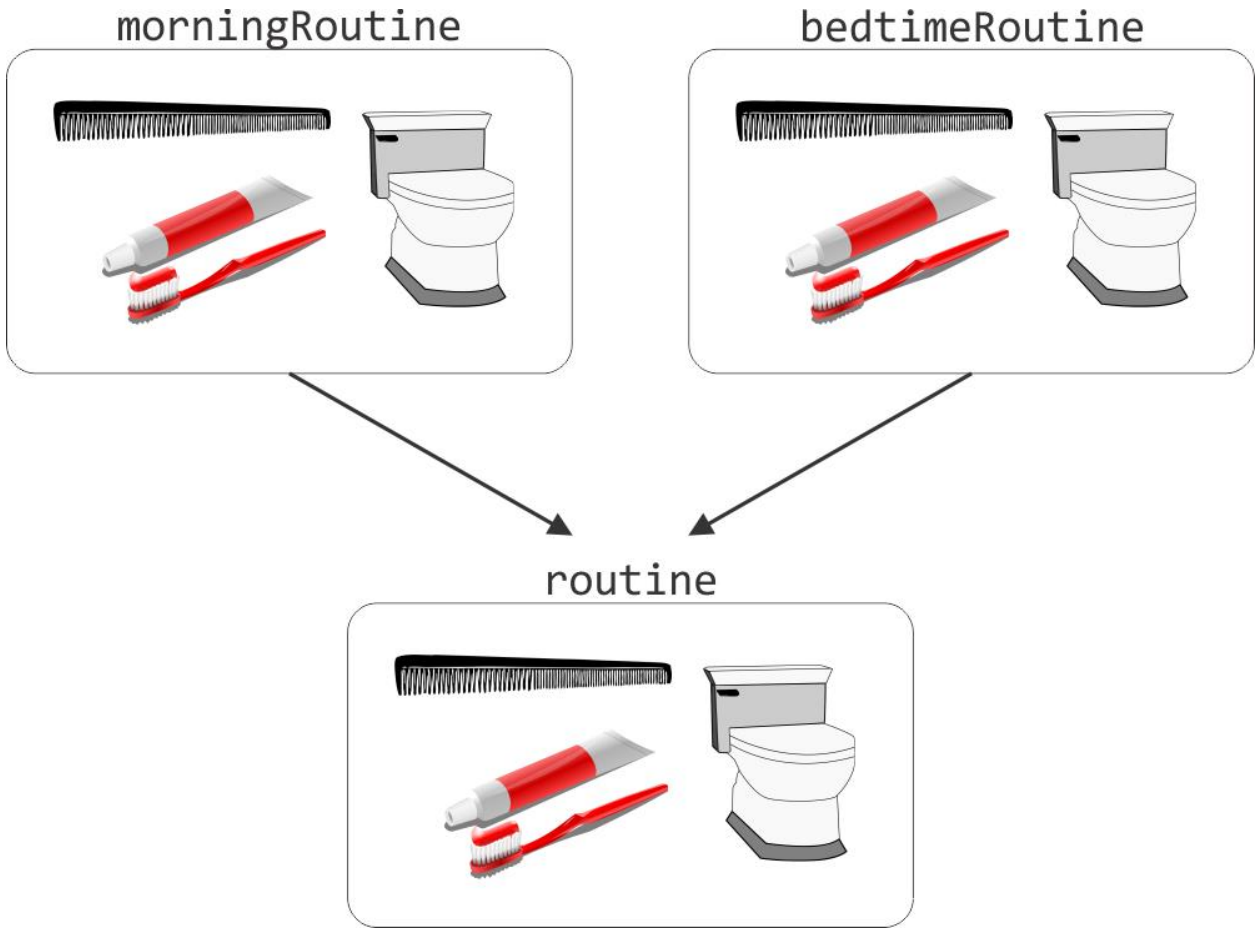


morningRoutine

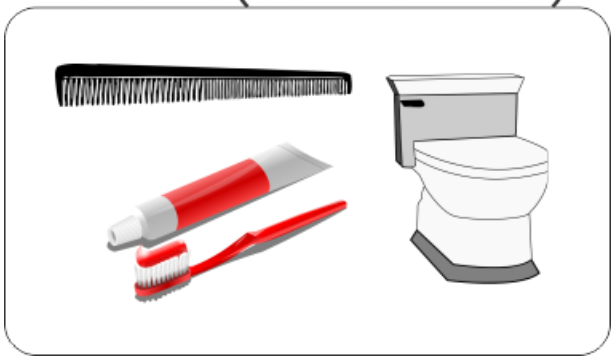


bedtimeRoutine




morningRoutine
bedtimeRoutine
routine

```
routine(doThisFirst)
```



Sometimes we customize our routines (for example, we can specify what comes first, or how long to spend).

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)
```

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
```

We are defining our own routine called `drawPolygon`

```
    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)
```

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto
    alex.pendo

    for sideNu
        alex.forward(sideSize)
        alex.left(360/numSides)
```

Our routine must be customized with a number of sides, how big each side is, and the position to draw the shape in

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)
```

Inside the routine, we first have our turtle lift its pen and go to the location specified above

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)
```

Then we do the shape drawing, again using values specified above

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)

drawPolygon(4, 50, 0, 100)
drawPolygon(10, 100, -230, -300)
drawPolygon(360, 1, 400, 300)
```

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)
```

```
drawPolygon(4, 50, 0, 100)
drawPolygon(10, 100, -230, -300)
drawPolygon(360, 1, 400, 300)
```

After defining our routine, we can invoke it as many times as we like

Image Stamping

```
def drawPolygon(numSides, sideSize, x, y):
    alex.penup()
    alex.goto(x,y)
    alex.pendown()

    for sideNum in range(numSides):
        alex.forward(sideSize)
        alex.left(360/numSides)

drawPolygon(4, 50, 0, 100)
drawPolygon(10, 100, -230, -
drawPolygon(360, 1, 400, 30(
```

Each time, we provide customized values for `numSides`, `sideSize`, `x` and `y`

Image Stamping

Define your own routine that draws something in particular (a house? A person? A flower? Whatever you want!).

Add some customizations to your routine (you can start just with location).

Invoke your routine with different customizations – it's like you are stamping your image!

Thank You!

ONWiE Members Hosting Go Code Girl

UOIT
CHALLENGE INNOVATE CONNECT

uOttawa
Faculté de génie
Faculty of Engineering

CONESTOGA
Connect Life and Learning

McMaster University
Inspiring Innovation and Discovery

VELUT ARBOR ÆVO
UNIVERSITY OF TORONTO

Queen's UNIVERSITY

Thank You to Our Sponsors!

Patrons

hydro one
Partners in Powerful Communities

Callisto Integration

Carleton UNIVERSITY | Faculty of Engineering and Design

Sponsors

ONTARIO POWER GENERATION

THE WOODBRIDGE GROUP®
Mastering Science To Serve Our Customers™

ONTARIO SOCIETY OF PROFESSIONAL ENGINEERS

UNIVERSITY OF TORONTO

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING