Turtle Graphics

- The **turtle** moves and draws with commands that are relative to its own position, such as:
  1. "Move forward 10 spaces" and
  2. "Turn left 90 degrees".

- The Turtle moves on a 2D surface and carries a pen.

- The state of the turtle has three attributes:
  1. Position
  2. Orientation
  3. Color
The Window and the Turtle

```python
import turtle

turtle.setup(300,300) #create a window 300x300 pixels.
window = turtle.Screen()
window.bgcolor("lightgreen")
window.title("Hello, Tess!")

tess = turtle.Turtle()
tess.shape("turtle")
tess.color("blue")
tess.pensize(3)
tess.forward(60) #60 pixels
tess.left(90) #90 degrees
tess.forward(120)

window.exitonclick()
```

Turtle's Perspective: Left & Right

```
left(30)
right(30)
```
What Pattern Does This Make?

# Code for setting up window and turtle goes here.

tess.forward(25)
tess.right(90)
tess.forward(50)
tess.right(90)
tess.forward(75)
tess.right(90)
tess.forward(100)
tess.right(90)
tess.forward(125)
tess.right(90)
tess.forward(150)
tess.right(90)
tess.forward(175)

window.exitonclick()

This code is very repetitive. What is a better way to make a bigger spiral?

Leverage Patterns with while Loop

# Code for setting up window and turtle goes here.

tess.forward(25)
tess.right(90)
tess.forward(50)
tess.right(90)
tess.forward(75)
tess.right(90)
tess.forward(100)
tess.right(90)
tess.forward(125)
tess.right(90)
tess.forward(150)
tess.right(90)
tess.forward(175)

distance = 25
while (distance <= 175):
    tess.forward(distance)
tess.right(90)

distance = distance + 25

window.exitonclick()
import turtle
turtle.setup(300, 300)  # Create a window 300x300 pixels.
window = turtle.Screen()
window.bgcolor("lightgreen")
window.title("Hello, Tess!")
tess = turtle.Turtle()
tess.shape("turtle")
tess.color("blue")
tess.pensize(3)
distance = 10  # was 25
while (distance <= 240):  # was 175
    tess.forward(distance)
tess.right(90)  # unchanged
    distance = distance + 10  # was 25
window.exitonclick()
How Can This Image be Produced?

```python
myTurtle.left(90)
myTurtle.forward(50)
myTurtle.right(60)
myTurtle.forward(50)
myTurtle.left(60)
myTurtle.forward(50)
```

Quiz: Which is the Turtle's Path?

```python
myTurtle.left(90)
myTurtle.forward(50)
myTurtle.right(60)
myTurtle.forward(50)
myTurtle.left(60)
myTurtle.forward(50)
```
Quiz: Which is the Turtle's Path?

```python
myTurtle.left(90)
myTurtle.forward(50)
myTurtle.left(45)
myTurtle.forward(50)
myTurtle.left(45)
myTurtle.forward(50)
```

![Options (a) to (d)]

Creating a List of Turtles (1 of 2)

```python
import turtle
global turtle

def makeTurtles():
    #Shown on next slide

turtle.setup(900,700)
window = turtle.Screen()
window.bgcolor("white")
window.title("List Of Turtles")
makeTurtles()

window.exitonclick()
```
Creating a List of Turtles

```python
def makeTurtles():
global turtle
myList = []
for i in range(10):
    # make a turtle and append it to myList
    myList = myList + [turtle.Turtle()]

    # Turn and move the turtle created in this iteration.
    myList[i].right(36*i)
    myList[i].forward(100)
```

What does this program draw?

Moving All Turtles in a List

```python
def boo():
    myList = []
    for i in range(10):
        myList = myList + [turtle.Turtle()]

        myList[i].right(36*i)
        myList[i].forward(100)

    for turtle in myList:
        turtle.right(25)
        turtle.forward(150)
```

What does this program draw?
Quiz: Moving Turtles in a List

def boo():
    myWorld = makeWorld(200,200)
    myList = []
    for i in range(10):
        myList = myList + [turtle.Turtle()]
        myList[i].right(36*i)
        myList[i].forward(100)
    for turtle in myList:
        turtle.right(120)
        turtle.forward(50)
        turtle.right(45)
        turtle.forward(50)

Quiz: Moving Turtles in a List

def boo():
    turtle.setup(200,200)
    myList = []
    for i in range(10):
        myList = myList + [turtle.Turtle()]
        myList[i].right(36*i)
        myList[i].forward(100)
    for turtle in myList:
        turtle.right(120)
        turtle.forward(50)
        turtle.right(120)
        turtle.forward(50)
Creating a Turtle Animation

```python
import time
turtle.setup(800,800)
myWorld = turtle.Screen()
myList = []
for i in range(10):
    myList = myList + [turtle.Turtle()]
    myList[i].right(36*i)
    myList[i].forward(100)

for i in range(100):
    for turtle in myList:
        turtle.right(10)
        turtle.forward(10)
        time.sleep(0.1)
```

Walk and Turn: Starting in a Circle

```python
turtle.setup(800,650)
n = 3
for i in range(n):
    list = list + [turtle.Turtle()]
    list[i].turn((360/n)*i)
    list[i].forward(300)

for i in range(1000):
    for i in range(len(list)):
        friend = i + 1
        if (friend == len(list)): friend = 0
        list[i].turnToFace(list[friend])
        list[i].forward(3)
```
The `random.uniform` function in Python returns a uniformly distributed random float between `>=min` and `<max`. For example:

```python
def boo():
    import random
    for i in range(20):
        x = random.uniform(0, 800)
        print(x)
```

Returns a uniformly distributed random float between `>=0.0` and `<800.0`.

---

The `moveTo` function in Python's `turtle` module is used to move the turtle to a specified position. It accepts integer arguments only.

```python
def boo():
    import turtle
    turtle.setup(800, 650)
    x = random.uniform(0, 800)
    y = random.uniform(0, 100)
    tess = turtle.Turtle()
    tess.moveTo(x, y)
```

Loading Program:
>>> boo()
The error was: 1st arg can't be coerced to int
Inappropriate argument type.
Truncate Float Type to Integer: \texttt{int(x)}

```python
def boo():
    import random
    turtle.setup(800, 200)
    list = []
    for i in range(20):
        list = list + [turtle.Turtle()]
        x = int(random.uniform(0, 800))
        y = int(random.uniform(0, 100))
        list[i].moveTo(x, y)
```

Move Turtle to Random Location

```python
def boo():
    import random
    turtle.setup(400, 300)
    list = []
    for i in range(20):
        list = list + [turtle.Turtle()]
        x = int(random.uniform(0, 400))
        y = int(random.uniform(0, 300))
        list[i].penUp()
        list[i].moveTo(x, y)
        list[i].penDown()
```
Give Turtle Random Heading

```python
def boo():
    import random
    myWorld = makeWorld(400,300)
    list = []
    for i in range(20):
        list = list + [turtle.Turtle()]
        angle = int(random.uniform(0, 360))
        list[i].turn(angle)
        x = int(random.uniform(0, 400))
        y = int(random.uniform(0, 300))
        list[i].penUp()
        list[i].moveTo(x, y)
        list[i].penDown()
```

Walk and Turn: Random Start (1 of 2)

```python
def walkAndTurnTurtles():
    import time
    import random
    myWorld = makeWorld(800,650)
    list = []
    for i in range(25):
        list = list + [turtle.Turtle()]
        angle = int(random.uniform(0, 360))
        list[i].turn(angle)
        x = int(random.uniform(0, 800))
        y = int(random.uniform(0, 650))
        list[i].penUp()
        list[i].moveTo(x, y)
        list[i].penDown()
```
for i in range(1000):
    for i in range(len(list)):
        friend = i + 1
        if (friend == len(list)): friend = 0
        list[i].turnToFace(list[friend])
        list[i].forward(3)
        time.sleep(0.001)
        myWorld.repaint()

= A single equals, =, is an assignment.
== A double equals, ==, is a boolean (true/false) operator.

def foo(grade):
    if (grade < 70): print 'See you fall 2012'
    elif (grade <= 79): print 'You get a C'
    elif (grade <= 89): print 'You get a B'
    else: print 'You get an A'

One and only one of the print statements executes.

>>> foo(71)
You get a C

>>> foo(92)
You get an A
**if, elif and else Blocks**

```python
def foo(grade):
    if (grade < 70):
        print 'See you fall 2012'
    elif (grade <= 79):
        print 'You get a C'
    elif (grade <= 89):
        print 'You get a B'
    else:
        print 'You get an A'
    print 'Done'
```

```python
>>> foo(88)
You get a B
Done
```

**if, elif and else Blocks**

```python
def foo(grade):
    if (grade <= 89):
        print 'You get an A'
    elif (grade <= 79):
        print 'You get a C'
    elif (grade <= 89):
        print 'You get a B'
    else:
        print 'do it again'
    print 'Done'
```

```python
>>> foo(88)
You get a B
Done
```
Quiz: *if, elif* and *else* Blocks

```python
def moo(x, y):
    if (x+y < 10):
        print 'less than 10.'
    elif (x+y < 20):
        print 'less than 20.'
    elif (x+y < 30):
        print 'less than 30.'
    else:
        print '30 or more.'
```

What is the output of the command:
```python
>>> moo(7, 8)
```

(a) less than 20.  
(b) less than 10.  
(c) less than 20.  
less than 20.  
less than 30.

if, *elif* and *else* with *random*

```python
def foo():
    import random

    for i in range(10):
        r = int(random.uniform(1, 4))
        if (r==1):
            print i, ' Peter'
        elif (r==2):
            print i, ' Paul'
        else:
            print i, ' Marry'

>>> foo()
0  Peter
1  Marry
2  Paul
3  Paul
4  Paul
5  Peter
6  Paul
7  Peter
8  Paul
9  Peter
```

r = 1, 2 or 3  
Each with equal probability.
Random Walk

```python
def randomWalk():
    import random
    x = 250
    y = 250
    myPic = makeEmptyPicture(x*2, y*2)
    setColor(getPixel(myPic, x, y), blue)
    for i in range(x*y):
        r = int(random.uniform(1,5))
        if (r==1): y=y-1
        elif (r==2): y=y+1
        elif (r==3): x=x-1
        else: x=x+1
        setColor(getPixel(myPic, x, y), blue)
    myPic.repaint()
```

- This program will crash if either x or y is outside the picture boundary.
- Will this ever happen?

Faster Random Walk: Repaint 1 in 10

```python
def randomWalk():
    import random
    x = 250
    y = 250
    myPic = makeEmptyPicture(x*2, y*2)
    setColor(getPixel(myPic, x, y), blue)
    for i in range(x*y):
        r = int(random.uniform(1,5))
        if (r==1): y=y-1
        elif (r==2): y=y+1
        elif (r==3): x=x-1
        else: x=x+1
        setColor(getPixel(myPic, x, y), blue)
        if ((i%10) == 0): myPic.repaint()
```

- % is python’s modulus operator.
- Only repaint if i is divisible by 10.
Modifying Random Walkers

What changes could be made to the random walkers program to:

- Make the picture and the area walked bigger?
- Add additional walkers with additional colors?
- Make the starting location of each walker random?
- Make the walkers bounce back if they try to enter the same pixel at the same time.
- Make the walkers bounce off any pixel where a walker has already been.
  - If this were done, what complications would need to be considered?