Graphics
Turtle grapics module - provides a toolkit

work area is the screen
the turtle is the icon
turtle is located in the center of the screen
works on x,y coordinates

turtle starts at 0,0 (Cartesian system)
inital position of the turtle is the origin (0,0) which also called the home

heading - direction the turtle is going north, east, south, west expressed in degrees

turtle's initial heading is 0 and 0 is due east
90 degrees is due north

color - default is black, can be changed to anyone of 16 million colors

width - inital width is pixels

down - true or false , means that the drawing pen is up or down

true - pen is down
false - pen is up

all of these are the state of the turtle

Turtle definitions
Every data value in python is an objects
The types of objects are called classes
Included in a class are the methods (operations) that apply to objects of that class

A Turtle is an object - it has methods (operations)
a set of methods of a given class of objects is called its interface

A method with turtle

define a function drawSquare
the function expects a turtle object, a pair of integers(coordinates) for upperleft hand corner, length of the line we want to draw

Begin by lifting the turtle and positioning it at the lefthand corner coordinates we specified
Then put the pen down
Change the heading of the turtle -270 degrees (south)
Give the length of the line

```
def drawSquare(t,x,y,length):
  """Draws a square with the given turtle t, an upperleft corner point(x,y) and a side's length"""
```

```
t.up()          #lifting the pen
t.goto(x,y)                    #changing direction
t.setheading(270)  #change the turtle heading
t.down()           #put pen on paper
for count in range(4):
  t.forward(length)
  t.left(90)
```

2 other important classes:
Screen - the turtle's associated window
Canvas - the area in which the turtle can move and draw lines

Set up the Turtle.cfg (configuration file)
cfg file - is a text file that contains the inital settings of several attributes of turtle, screen and canvas

Python creates a default .cfg file for you

To create your own, set up a text file with the attributes you want to change, save it with .cfg extension and save the file in the current working directory

```
width = 300
height = 200
using_IDLE = True
colormode = 255
```

Instantiation - process of creating an object

syntax to instantiate an object:
<variable name> = <class name>(<any arguments>)
the expression on the right of the = is called a constructor

arguments are optional

The Turtle class is defined in the turtle module

```
>>>from turtle import Turtle
>>>t=Turtle()
```

Example - write a program:
to draw an uppercase T with a black vertical line and a red crossbar

```
from turtle import Turtle
>>> t=Turtle()
>>> t.width(2)
>>> t.left(90)
>>> t.forward(30)
>>> t.left(90)
>>> t.up()
```

```
>>> t.forward(10)
>>> t.setheading(0)
>>> t.pencolor("red")
>>> t.down()
>>> t.forward(20)
>>>t.hideturtle()
```

Turtle continued
Drawing a square

```
from turtle import Turtle
t=Turtle()

def drawSquare(t,x,y,length):
  """Draws a square with a given turtle t, an upperleft corner point(x,y) and a side's length"""
  t.up()
  t.goto(x,y)
  t.setheading(270)
  t.down
  for count in range(4):
     t.forward(length)
           t.left(90)
```

a more general way of drawing a square:

```
def square(t,length):
   """Draws a square with a given length"""
         for count in range(4):
       t.forward(length)
       t.left(90)
```

Draw a hexagon

```
def hexagon(t,length):
    """Draws a hexagon with the given length"""
          for count in range(6):
              t.forward(length)
                    t.left(60)
```

Draw a radial hexagon

```
def radialHexagons(t,n,length):
   """Draws a radial pattern of n hexagons with the given length"""
          for count in range(n):
              hexagon(t,length)
                    t.left(360/n)
```

```
from polygons import *     #Import all functions
from turtle import Turtle
t=Turtle()
t.pencolor("blue")
t.hideturtle()
square(t,50)
hexagon(t,50)
t.clear()
radialhexagons(t,10,50)

def radialPattern(t,n,length,shape):
  """Draws a radial pattern of n shapes with the given length"""
    for count in range(n):
            shape(t,length)
                t.left(360/n)


t=Turtle()
radialPattern(t,n=10,length=50, shape=square)
t.clear()
radialPattern(t,n=10,length=50, shape=hexagon)
```

Attributes of the Turtle- position, heading, color
mutator methods - color, meaning they change the internal state of the Turtle

accessor methods - things like position

```
>>>from turtle import Turtle
>>>t=Turtle()
>>>t.position
(0.0,0.0)            returns turtle's current position
>>>t.heading(0
0.0
>>>t.isdown()
True
```

Screen and Canvas - two classes

The Screen's object attributes include width, height, background color

```
t.screen.bgcolor("orange")
```

height and width to control a screen's boundaries

Ex. - change the background color from white to orange and then prints coordinates of upper left and lower right corners of the window

```
>>>from turtle import Turtle
```

```
>>>t=Turtle()
>>>t.screen.bgcolor("orange")
>>>x = t.screen.window.width()//2
>>>y = t.screen.window.height()//2
>>>print((-x,y),(x,-y))

from turtle import Turtle
import random

def randomWalk(t,turns,distance=20):
  """Turns a random # of degrees and moves a distance for a fixed number of turns"""
  for x in range(turns):
    if x%2 == 0
              t.left(random,randint(0,270))
            else:
      t.right(random.randint(0,270)
            t.forward(distance)

def main():
  t=Turtle()
  t.shape("turtle")
  randomWalk(t,40,30)

if _name_ == "_main_":
  main()
```

Colors
red, green, blue

RGB - red,green,blue

full saturation of red,green,blue = white
lack of color = black

RGB for black = 0,0,0
white = 255,255,255
red = 255,0,0
green = 0,255,0
blue = 0,0,255
turquoise = 36,255,255

Hexadecimal
2 digits for red
2 digits for green
2 digits for blue

red = #FF0000
green = #00FF00

blue = #0000FF
black = #000000
white = #FFFFFF

pencolor - changes the drawing color
fillcolor - this fills a color in

```
import turtle
t=Turtle()
t.fillcolor('blue')
t.begin_fill()
for --- in range(4):
  t.forward(150)
  t.right(90)
t.end_fill

t.fillcolor('#FFA500')
```

Image Processing
1. Capture of images
2. storing of images
3. image manipulation

analog clock vs a digital clock

10 pixels per linear millimeter (250 pixels per inch and 62,500 pixels in a square inch)

3 x 5 image - 937,500 pixels

Image File Formats

RAW - generated by digital camera
- uncompressed, raw image data
    - storing unaltered image data
    - Very large file sizes
    - most true to life
    - higher quality image
    - no data is lost
    - not used in this format in software or on the web

BMP - bitmap image
- not used with software or web
- paint programs

Compressed file formats:

GIF
Graphics Interchange Format

animation
compression was lossless
first image published on the World Wide Web was a GIF

algorithm that was developed LZW

under patent and owned by a company named Unisys Corp.

1995 - charging a royalty for the LZW compression

PNG - had it's own algorithm
JPG JPEG - Joint Photographic Experts Group- had it's own algorithm
developed in the 1990's

JPG - portable
- most all image processing software works with
- compatible with most hardware devices
- quick storage
- lossy compression
- no animation

PNG - Portable Network Graphics
developed to be an improvement over GIF and support more than 256 colors
lossless
wider range of color support
no animation

TIFF - printing industry
PSD - native file format for Adobe Photoshop

To work with these files to convert to JPG or GIF -
GIMP - open source, free
Photoshop - subscription service

Image Manipulation Operations
- Rotate an image
- Convert an image from color to grayscale
- Apply color filtering to an image
- Highlight a particular area in an image
- Blur all or part of an image
- Sharpen all or part of an image
- Control the brightness of an image
- Perform edge detection on an image
- Enlarge or reduce an image's size
- Apply color inversion to an image
- Crop an image
- Morph an image into another image

Python has a module to allow us to work with images.

Properties of an image

coordinates of the pixels (0,0)
image starts at upperleft hand corner

0,0 to width-1, height-1

along with width and height, set of color values
color values are expressed as tuples

Images Module
small module high-level Python resources for image processing. Non-standard

package is named images.py

Additional Image processing modules

Pillow
OpenCV - very high ranking
NumPy

turtle.setposition(x,y)

turtle.speed(number)

Radial Turtle exercise

```
import turtle
ninja = turtle.Turtle()

ninja.speed(10)

for i in range(180):
   ninja.forward(100)
        ninja.right(30)
        ninja.forward(20)
        ninja.left(60)
        ninja.forward(50)
        ninja.right(30)

        ninja.penup()
        ninja.setposition(0,0)
        ninja.pendown()

        ninja.right(2)
```

turtle.done()

Image module has a class named Image
representing an image as a 2 dimensional array of rgb values

Methods for Image class are on myTCC

This module only works with GIF images
To convert an image from JPG or PNG to Gif
use Gimp or Photoshop, Paint - to open the JPG or PNG and then turn right around and save as a GIF

The code:
1. Imports the Image class from the images module
2. Instantiate this class using the file named smokey.gif
3. Draws the image

```
from images import Image
image = Image("smokey.gif")
image.draw()
```

```
image.getWidth()
```

```
image.getHeight()
```

print(image) - to get image string representation

image.getPixel(0,0) - brings back a tuple with the rgb values of that location

image.draw() - this would draw a new blank image

image = blank(300,300) - specifying the images size

draw an empty image

then draw another image and replace pixels along the y axis

Example to draw an empty image and then
draw a blue line in the center of the window:

```
from images import Image
image=Image(300,300)
image.draw()
blue=(0,0,255)
y=image.getHeight()//2
for x in range(image.getWidth()):
   image.setPixel(x, y-1, blue)
   image.setPixel(x,y,blue)
```

```
   image.setPixel(x,y+1,blue)

image.draw()
```

to save the image created:
image.save("image_name goes here")

Loop pattern for a grid
use a nested loop to access the 2 dimensional image
outer loop and an inner loop
each loop has its own control variable
outer loop iterates over 1 coordinate, while the inner loop iterates over the other coordinate

for Example

```
width = 2
height = 3
for y in range(height):
   for x in range(width):
            print((x,y), end = " ")
         print()
```

what this prints out is:
(0,0) (1,0)
(0,1) (1,1)
(0,2) (1,2)

This is called row-major traversal

This template is used over and over to traverse an image

```
for y in range(height):
   for x in range(width):
            <do something at position(x,y)>
```

Tuples
pixels RGB values are stored as tuples
Python allows assignment of one tuple to another

getPixel - returns a tuple of integers representing RGB values at the pixel position(x,y)

```
image = Image("smokey.gif")
(r,g,b) = image.getPixel(0,0)
```

r 194
g 221
b 114

```
image.setPixel(0,0(r+10, g+10, b+10))

def average(triple):
   (a,b,c) = triple
          return(a+b+c)//3

average((40,50,60))
```

the answer is : 50

Converting an image to Black and White
for each pixel, the algorithm that computes the average of the red, green, blue values
then the algorithm resets the pixel's color values to 0(black) if the average is closer to 0
or to 255(white) if the average is closer to 255

function to convert image pixels:

```
def blackAndWhite(image):
  """Converts the argument image to black and white"""
  blackPixel = (0,0,0)
  whitePixel = (255,255,255)
  for y in range(image.getHeight()):
     for x in range(image.getWidth()):
            (r,g,b) = image.getPixel(x,y)
            average = (r+g+b)//3
            if average < 128:
              image.setPixel(x,y,blackPixel)
            else:
              image.setPixel(x,y,whitePixel)
```

Program to convert pixels in image to black and white:

```
from images import Image

def blackAndWhite(image):
  """Converts the argument image to black and white"""
  blackPixel = (0,0,0)
  whitePixel = (255,255,255)
  for y in range(image.getHeight()):
     for x in range(image.getWidth()):
            (r,g,b) = image.getPixel(x,y)
            average = (r+g+b)//3
            if average < 128:
              image.setPixel(x,y,blackPixel)
            else:
              image.setPixel(x,y,whitePixel)

def main(filename="smokey.gif"):
```

```
    image = Image(filename)
            print ("Close the image window to continue")
            image.draw()
            blackAndWhite(image)
            print ("Close the image window to continue")
            image.draw()

if _name_ = "_main_":
    main()
```