Computational Physics

More on Objects in Python

Prof. Paul Eugenio
Department of Physics
Florida State University
14 Mar 2019

http://hadron.physics.fsu.edu/~eugenio/comphy

Recall

Class defines an Object

```
# Defining a class
class ClassName:
""" Doc string information
"""
    [statement 1]
    [statement 2]
    [statement 3]
    [etc.]
```

An object is an instance of a class just like a float variable is an instance of a float data type

```
myClass = ClassName("Physics")
goldenRation = float("1.618")
```

Simple Example: A Circle

Class Definition of Simple Circle

```
class Circle:
    def __init__(self, radius=1):
        self.radius = radius

    def area(self):
        return np.pi*self.radius**2

    def circumference(self):
        return 2*np.pi*self.radius

    def __add__(self, other):
        return Circle(self.radius + other.radius)

    def print(self):
        print( "Hello, I am a circle" )
        print( "my radius is",self.radius )
        print( "My area is", self.area() )
        print( "My circumference is", self.circumference() )
```

Simulating Floating Garbage

```
class Garbage():
    """
     Garbage is an object which simulates the random floating of trash in
     a current-less ocean
    """
    def __init__(self, x=0, y=0):
        """ Each piece of garbage has an (x,y) position. """
        self.x = x
        self.y = y


    def move(self, x_increment=0, y_increment=0):
        """
         Move the garbage according to the parameters given.
         Default behavior is to stay put
        """
        self.x =  self.x + x_increment
        self.y =  self.y + y_increment


    def get_distance(self, other):
        """
         Calculates the distance from this piece to another piece,
         and returns that value.
        """
        distance = np.sqrt( (self.x - other.x)**2 + (self.y - other.y )**2 )
        return distance
```

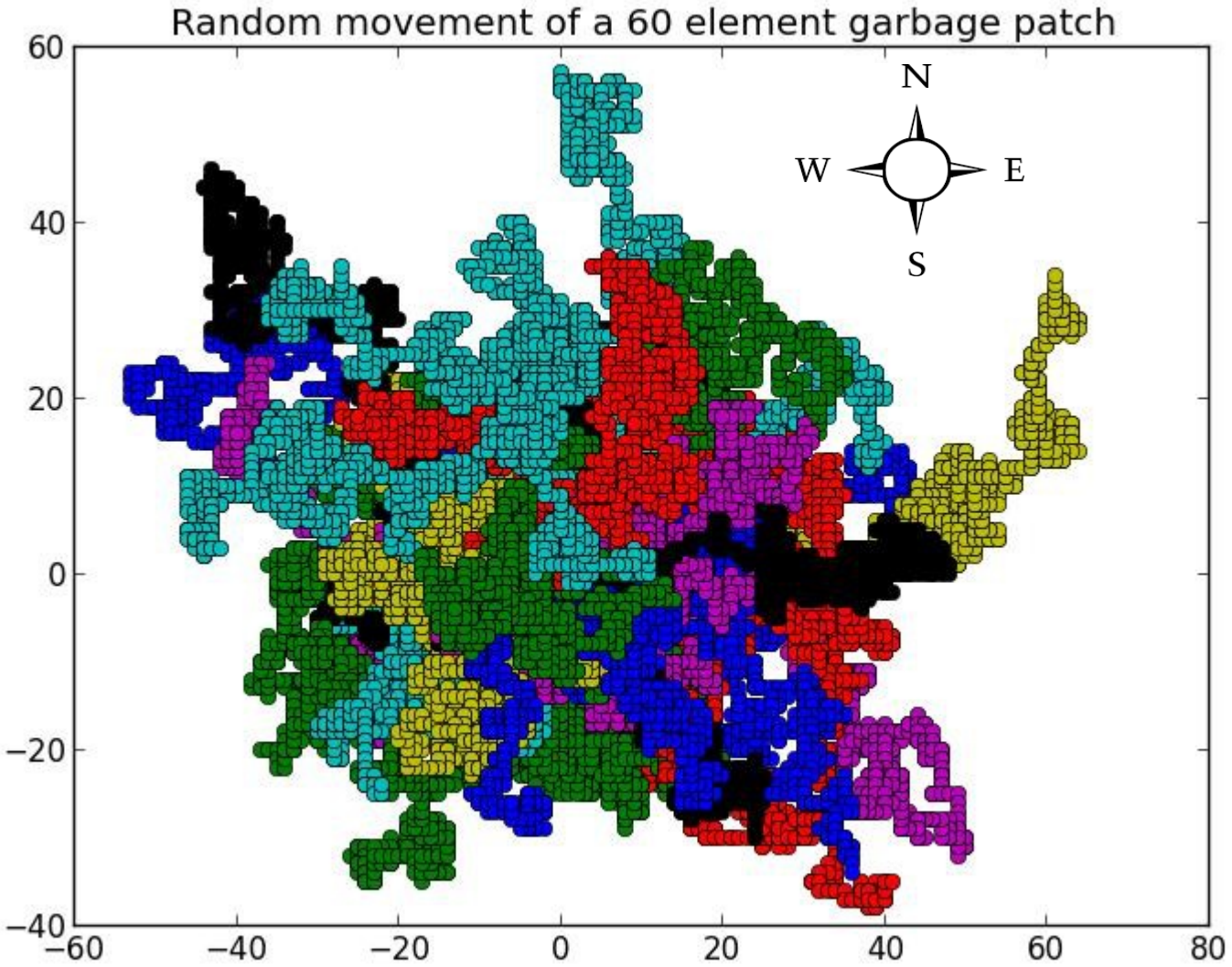
Floating Garbage

```
class Garbage():

    ...

    def float(self):
        """
         random floating movement: moves one unit East, North, West, South,
         or stays put
        """
        direction = np.random.randint(5)
        dx, dy = 0, 0  # default is not to move in any direction

        if direction == 1:          # move East
            dx = 1
        elif direction == 2:        # move North
            dy = 1
        elif direction == 3:        # move West
            dx = -1
        elif direction == 4:        # move South
            dy = -1

        self.move(dx, dy)
```

Floating Garbage

```
#
# main
#

# Create trash for a garbage patch
garbagePatch = []
for k in range(60):
    garbagePatch += [Garbage()]


# time interval
maxTime = 1000
tRange = range(maxTime)

# plot the location of the trash as it randomly floats
for garbage  in garbagePatch:
    x,y = [],[]
    for t in tRange:
        garbage.float()
        x += [trash.x]
        y += [trash.y]
    plt.plot(x,y,"o")

plt.savefig("garbagePatch.jpg")
plt.show()
```

Floating Garbage



http://hadron.physics.fsu.edu/~eugenio/comphy/examples/garbagepatch.py

A Root Finding Object

```
class Newton:
    """
    Root finding method using Newton's method
        x = x_old + f(x) / df(x)/dx
    """

    def __init__(self, f, dfdx, precision=0.1):
        """
            Constructor needs functions for f(x) & df(x)/dx
        """
        self.f, self.dfdx = f, dfdx
        self.precision = precision

    def getRoot(self, x ):
        """
        Root finding method using Newton's method
        """

        lastX, count = float("inf"), 0
        while (x - lastX)**2 > self.precision**2:
            lastX = x
            count += 1
            x = lastX - self.f(lastX)/self.dfdx(lastX)
        return [x, count]
```

Using the Root Finding Object

create object rto from the class Newton defined in module findroot

```
rto = findroot.Newton(P, dPdx)

rto.precision = 1e-10        # set root finding precision



findRoots = True
while findRoots:

    guess = raw_input("Enter starting guess for root['q' to quite]: ")

    if guess != 'q' and guess != '' :
        guess = float(guess)
        print( "guess: ",guess , "root:", rto.getRoot(guess)[0], sep='\t' )
    else:
        findRoots = False
        print("bye")
```

Have the object find a root and print
the root value ignoring number of iteration steps

Let's get working