ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 10 - Classes

Defining a Class (Example)

```
cp = 1007.0     # Specific heat at constant pressure (J kg^-1 K^-1)
P0 = 100000.0   # Reference pressure (Pascals)
R = 8.3145      # Universal gas constant (J mol^-1 K^-1)
Md = 0.02996    # Molar mass of dry air (kg mol^-1)
Mw = 0.01801    # Molar mass of water (kg mol^-1)

class dry_parcel:
    def __init__(self,p,T,m):
        self.p = float(p)      # Pressure (Pascals)
        self.T = float(T)      # Temperature (K)
        self.m = float(m)      # mass (kg)

    def moles(self):
        '''Returns number of moles in parcel.'''
        return self.m/Md

   def volume(self):
        '''Returns volume of parcel (m^3)'''
        return self.moles()*R*self.T/self.p

    def rho(self):
        '''Returns density of parcel (kg/m^3)'''
        return self.m/self.volume()
```

Save to file ‘Air_Parcel.py’

Defining a Class (Example)

```
class dry_parcel:
    def __init__(self,p,T,m):
        self.p = float(p)      # Pressure (Pascals)
        self.T = float(T)      # Temperature (K)
        self.m = float(m)      # mass (kg)

    def moles(self):
        '''Returns number of moles in parcel.'''
        return self.m/Md

    def volume(self):
        '''Returns volume of parcel (m^3)'''
        return self.moles()*R*self.T/self.p

    def rho(self):
        '''Returns density of parcel (kg/m^3)'''
        return self.m/self.volume()
```

Attributes

Methods

Using a Class

```
>>> import Air_Parcel as ap
>>> a = ap.dry_parcel(85000,260,3)
>>> a.p
85000.0
>>> a.T
260.0
>>> a.m
3.0
>>> a.moles()
100.13351134846462
>>> a.volume()
2.5466543626796514
>>> a.rho()
1.1780161626815062
```

Attributes Can Be Altered

```
>>> a = ap.dry_parcel(85000,260,3)
>>> a.T
260.0
>>> a.volume()
2.5466543626796514
>>> a.T = 280.0
>>> a.volume()
2.742550852116548
```

Inheritance

- New classes (child or subclasses) can inherit attributes and methods from existing classes (parent classes).
- The child class can also have additional attributes and methods that the parent class didn’t have.
- The child class can also have modified methods from the parent class.

Inheritance Example

```
class moist_parcel(dry_parcel):
    def __init__(self,p,T,m,r):
        dry_parcel.__init__(self,p,T,m)
        self.r = float(r)      # Mixing ratio (g/kg)

    def moles(self):
        '''Returns number of moles in parcel.'''
        return self.mass_dry()/Md + self.mass_vapor()/Mw

    def rho(self):
        '''Returns density of parcel (kg/m^3)'''
        return self.p/(Rd*self.Tv())

    def Tv(self):
        '''Returns virtual temperature in Kelvin.'''
        return self.T*(1 + 0.61*self.q/1000.0)
```

Inheritance

```
class moist_parcel(dry_parcel):
    def __init__(self,p,T,m,r):
        dry_parcel.__init__(self,p,T,m)
        self.r = float(r)      # Mixing ratio (g/kg)

    def moles(self):
        '''Returns number of moles in parcel.'''
        return self.mass_dry()/Md + self.mass_vapor()/Mw

    def rho(self):
        '''Returns density of parcel (kg/m^3)'''
        return self.p/(Rd*self.Tv())

    def Tv(self):
        '''Returns virtual temperature in Kelvin.'''
        return self.T*(1 + 0.61*self.q/1000.0)
```

Inherits from dry_parcel

Calls dry_parcel initialization method

Adds another attribute

Redefines moles() and rho() methods

Adds another method

In-class Exercise

- Create a module called Spheres (note plural).
- The module should contain a class definition for a ‘sphere’ class that:
  - is initialized with a radius (in arbitrary units)
  - has an attribute called ‘radius’ that is the radius of the sphere.
  - has a function called area() that returns the surface area of the sphere
  - has a function called volume() that returns the volume of the sphere.
- The module should also contain a function called merge() that if given two spheres merges them into a new sphere of the combined volume of the two.

Spheres Example

```
>>> import Spheres as s
>>> a = s.sphere(5)
>>> a.area()
314.1592653589793
>>> a.volume()
523.5987755982989
>>> a.diameter()
10.0
>>> a.radius
5.0
>>> b = s.sphere(10)
>>> b.area()
1256.6370614359173
>>> b.volume()
4188.790204786391
>>> b.diameter()
20.0
>>> b.radius
10.0
```

```
>>> c = s.merge(a,b)
>>> c.area()
1359.2881964162366
>>> c.volume()
4712.388980384687
>>> c.diameter()
20.800838230519037
>>> c.radius
10.400419115259519
```