# Part 1

## Object Oriented Programming

### Replace pass with the appropriate code in the Line class methods to accept coordinates as a pair of lists and return the slope and distance of the line.

In [4]:
```
import math
class Line(object):
    def __init__(self,coor1,coor2):
        self.coor1 = coor1 #Initializing values
        self.coor2 = coor2
    def distance(self):
        a1,b1=self.coor1#Initializing the values
        a2,b2=self.coor2
        return math.sqrt((b2-b1)**2+(a2-a1)**2)
    def slope(self):
        a1,b1=self.coor1
        a2,b2=self.coor2
        result=((b2-b1)/(a2-a1))#storing the calculated values into the variable result
        print(type(result))#printing the result value along with
        return result
coordinate1=[3,2]
coordinate2=[8,10]
li=Line(coordinate1,coordinate2)
li.distance()
```

Out[4]: 9.433981132056603

In [5]:
```
li.slope()
```

```
<class 'float'>
```

Out[5]: 1.6

In [6]:
```
coordinate1=[5,4]
coordinate2=[7,9]
li=Line(coordinate1,coordinate2)
```

In [7]:
```
li.distance()
```

Out[7]: 5.385164807134504

In [8]:
```
li.slope()
```

```
<class 'float'>
```

Out[8]: 2.5

### 2.Replace pass with the appropriate code in the Cylinder class methods to return the volume and the surface area of the cylinder.

In [13]:
```
import math
class Cylinder(object):
    def __init__(self,height=1,radius=1):
        self.p=3.14
        self.radius=radius
        self.height=height
    def volume(self):
        volume=self.p*math.pow(self.radius,2)*self.height
        return volume
    def surface_area(self):
        surface_area=2*self.p*math.pow(self.radius,2)+2*self.p*self.height*self.radius
        return (surface_area)
```

In [14]:
```
c = Cylinder(2,3)
c.volume()
```

Out[14]: 56.52

In [15]:
```
c.surface_area()
```

Out[15]: 94.2

In [16]:
```
c = Cylinder(4,5)
c.volume()
```

Out[16]: 314.0

In [17]:
```
c.surface_area()
```

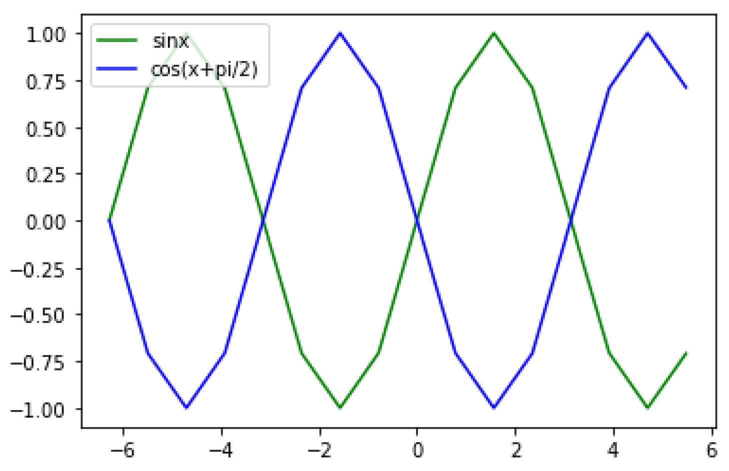
Out[17]: 282.6

## Part 2: 2D Plots

Create a line plot of sin(x) and cos(x + π/2) for -2π < x < 2π where x increases at intervals of π/4.

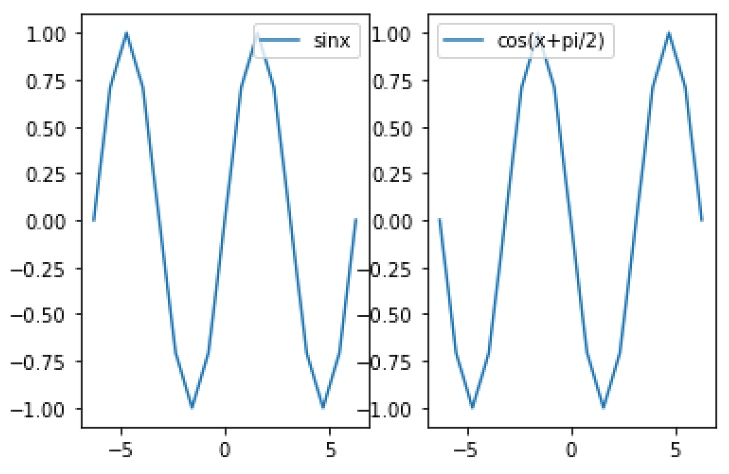
In [18]:
```
import numpy as np
import matplotlib.pyplot as plt
pi_value=3.14
```

In [21]:
```
l=np.arange(-2*pi_value,2*pi_value,pi_value/4)
j1=np.sin(l)
j2=np.cos(l+pi_value/2)
plt.plot(l,j1,color="green",label="sinx")
plt.plot(l,j2,color="blue",label="cos(x+pi/2)")
plt.legend()
plt.show()
```




### Using the same info as above, make a subplot with 2 different graphs- one graph for sin(x) and one graph for cos(x+π/2)

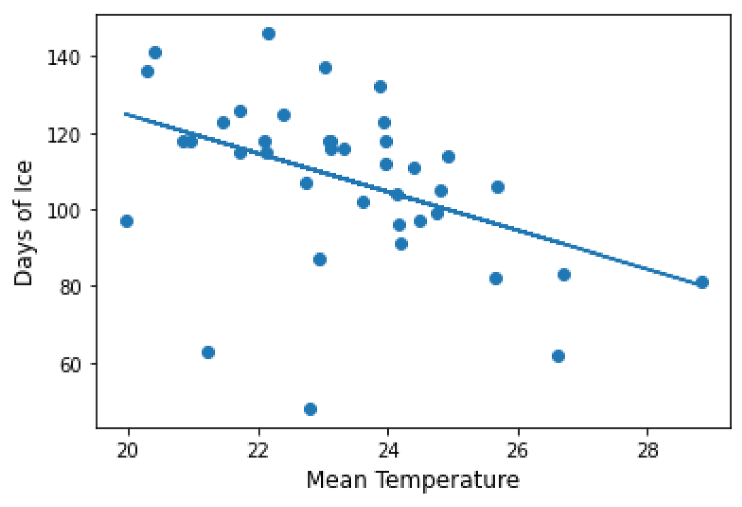
In [22]:
```
k=np.array([ -6.283, -5.498, -4.712, -3.927, -3.142, -2.356, -1.571, -.7854, 0, .7854, 1.571, 2.356, 3.142, 3.927, 4.712, 5.498, 6.283])
sinx=[0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0]
cosx=[0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0, -.70711, -1, -.70711, 0, .70711, 1, .70711, 0]
j1=np.sin(k)
j2=np.cos(k+pi_value/2)
fig,(a1,a2)=plt.subplots(1,2)
a1.plot(k,j1,label="sinx")#plotting the graphs and setting the labels
a1.legend()
a2.plot(k,j2,label="cos(x+pi/2)")
a2.legend()
plt.show()
```




# 2.Scatter Plot:

### Using the following data about winter temperatures affecting the number of days for lake ice at Lake Superior, construct a scatter plot to display the data. Include a line of best fit.

In [25]:
```
temperaturemean=np.array([22.94, 23.02, 25.68, 19.96, 24.80, 23.98, 22.10, 20.30, 24.20, 22.74, 24.16, 24.94, 22.40, 22.14, 20.84, 25.66, 21.73, 24.49, 24.13, 22.17, 21.73, 20.41,
D_o_i=np.array([87, 137, 106, 97, 105, 118, 118, 136, 91, 107, 96, 114, 125, 115, 118, 82, 115, 97, 104, 146, 126, 141, 111, 123, 118, 83, 48, 118, 116, 81, 116, 123, 112, 99, 102,
plt.scatter(temperaturemean,D_o_i)
plt.ylabel('Days of Ice', fontsize=12) #labelling the x-axis
plt.xlabel('Mean Temperature',fontsize=12)  #labeling the y-axis
def b_f(A, B): #calculates the line Equation that fits in the most of the data
    x = sum(A)/len(A)
    y = sum(B)/len(B)
    n = len(A) # or len(Y)
    num = sum([xi*yi for xi,yi in zip(A,B)]) - n * x * y
    d = sum([xi**2 for xi in A]) - n * x**2
    c = num / d
    d = y - c * x
    return d, c
d, c = b_f(temperaturemean, D_o_i)
y_fit = [d + c * xi for xi in temperaturemean]
plt.plot(temperaturemean, y_fit)
plt.show()
```




In [ ]: