MAT541D E-Record

Arnav Sinha

2040811

5EMS

Contents

Topic 1: Introduction to Python

```
1 Date: 23.07.2022
```

Question 1: Who started Pyhton programming?

- Guido Van Rossum

Question 2: Why do we use Python ?

- Open source Library
- Easy syntax
- Beginner friendly

Question 3

Who uses Python ?

- Microsoft
- Apple
- Netflix
- Asus
- HP
- Dell
- Lenova
- Facebook
- TCS

Question 4

What is a program ?

A defined set of instructions.

Question 5

What is order of mathematical operations?

PEMDAS

P : Parenthesis

M,D : equal importance, whoever comes first will be solved first

A,S : equal importance, whoever comes first will be solved first

Question 6

What is a function ?

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

Question 7

What is a module ?

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

Question 8

What is floor division and modulus

In Python, we can perform floor division (also sometimes known as integer division) using the // operator. This operator will divide the first argument by the second and round the result down to the nearest whole number, making it equivalent to the math. floor() function. Python Modulus Operator is an inbuilt operator that returns the remaining numbers by dividing the first number from the second. It is also known as the Python modulo. In Python, the modulus symbol is represented as the percentage (%) symbol. Hence, it is called the remainder operator

Question 10

What are Boolean expressions?

In programming you often need to know if an expression is True or False. You can evaluate any expression in Python, and get one of two answers, True or False. When you compare two values, the expression is evaluated and Python returns the Boolean answer

Question 11

Write a code which accepts user's first and last name and print it with a space between it

In [1]:
```
first_name = input("Enter your first name:")
last_name = input("Enter your last name: ")

print("Hello",first_name, last_name)
```

```
Enter your first name:Arnav
Enter your last name: Sinha
Hello Arnav  Sinha
```

Question 12

Write a program to check whether a number is even or odd

In [2]:
```
n = int(input("Enter a no."))

if n%2 == 0:
    print("The no. is even")
else:
    print("The no. is odd")
```

```
Enter a no.1
The no. is odd
```

Question 13

Write a program to check if two numbers are divisble

In [3]:
```
n = int(input("Enter a no."))
m = int(input("Enter a no."))
if n%m == 0:
    print("First no. is divisible by the second no.")
else:
    print("First no. is not divisble by the second no.")
```

```
Enter a no.1
Enter a no.2
First no. is not divisble by the second no.
```

Question 14

Write a program to find the distance betwen the two 2D points

In [4]:
```
a = int(input("Enter the abscissa of Point 1: "))
b = int(input("Enter the ordinate of Point 1 "))
c = int(input("Enter the abscissa of Point 2 "))
d = int(input("Enter the ordinate of Point 2 "))

list1 = [a, b]
list2 = [c, d]

dlist = [list2[0] - list1[0], list2[1] - list1[1]]
dlist

dist = (dlist[0]**2 + dlist[1]**2)**(1/2)
print("The distance between the two points is",dist)
```

```
Enter the abscissa of Point 1: 1
Enter the ordinate of Point 1 2
Enter the abscissa of Point 2 3
Enter the ordinate of Point 2 4
The distance between the two points is 2.8284271247461903
```

In [5]:
```
a = int(input("Enter the abscissa of Point 1: "))
b = int(input("Enter the ordinate of Point 1 "))
c = int(input("Enter the abscissa of Point 2 "))
d = int(input("Enter the ordinate of Point 2 "))

dlist = [a-c,b-d]
dlist

dist = (dlist[0]**2 + dlist[1]**2)**(1/2)
print("The distance between the two points is",dist)
```

```
Enter the abscissa of Point 1: 1
Enter the ordinate of Point 1 2
Enter the abscissa of Point 2 3
Enter the ordinate of Point 2 4
The distance between the two points is 2.8284271247461903
```

Question 15

Write a program to identify the largest of 3 given numbers

In [6]:
```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))
if (num1 >= num2) and (num1 >= num3):
    largest = num1
elif (num2 >= num1) and (num2 >= num3):
    largest = num2
else:
    largest = num3
print("The largest number is", largest)
```

```
Enter first number: 3
Enter second number: 4
Enter third number: 6
The largest number is 6.0
```

Topic 2: Plotting using Matplotlib

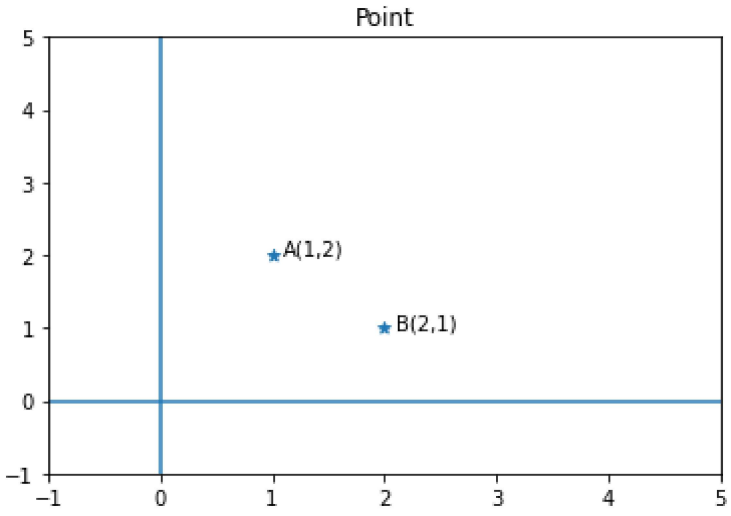
Date: 30.07.2022

Scatter Plot 2D

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

In [8]:

```
x = [1,2]
y = [2,1]

plt.scatter(x,y,marker = "*")
plt.annotate("A(1,2)", (1.1,2))
plt.annotate("B(2,1)", (2.1,1))
plt.axhline()
plt.axvline()
plt.xlim([-1,5])
plt.ylim([-1,5])
plt.title("Point")
None
```

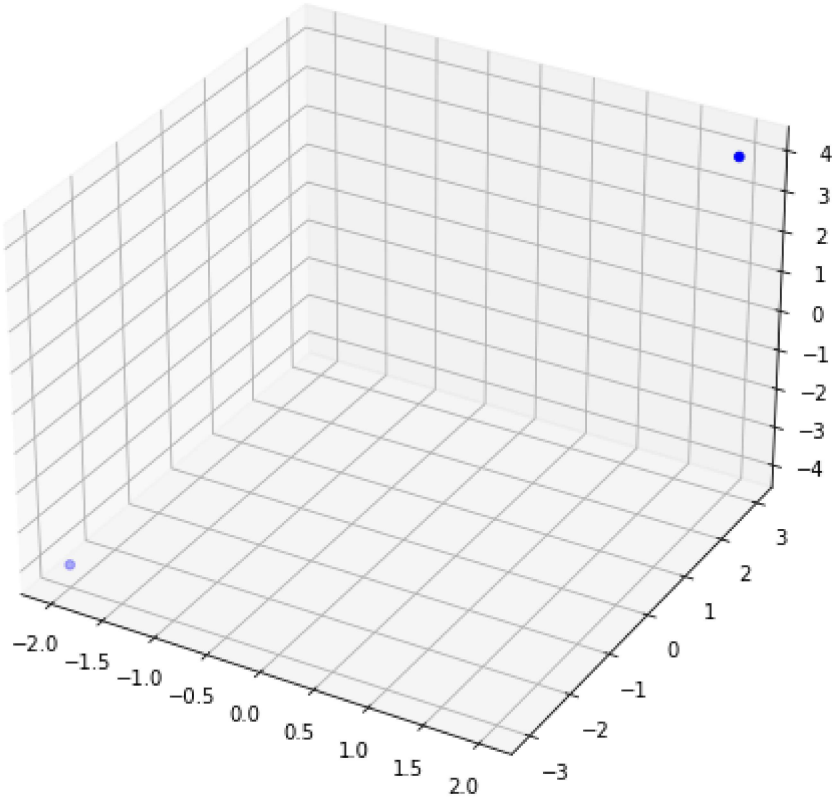


Scatter plot 3D

In [9]:

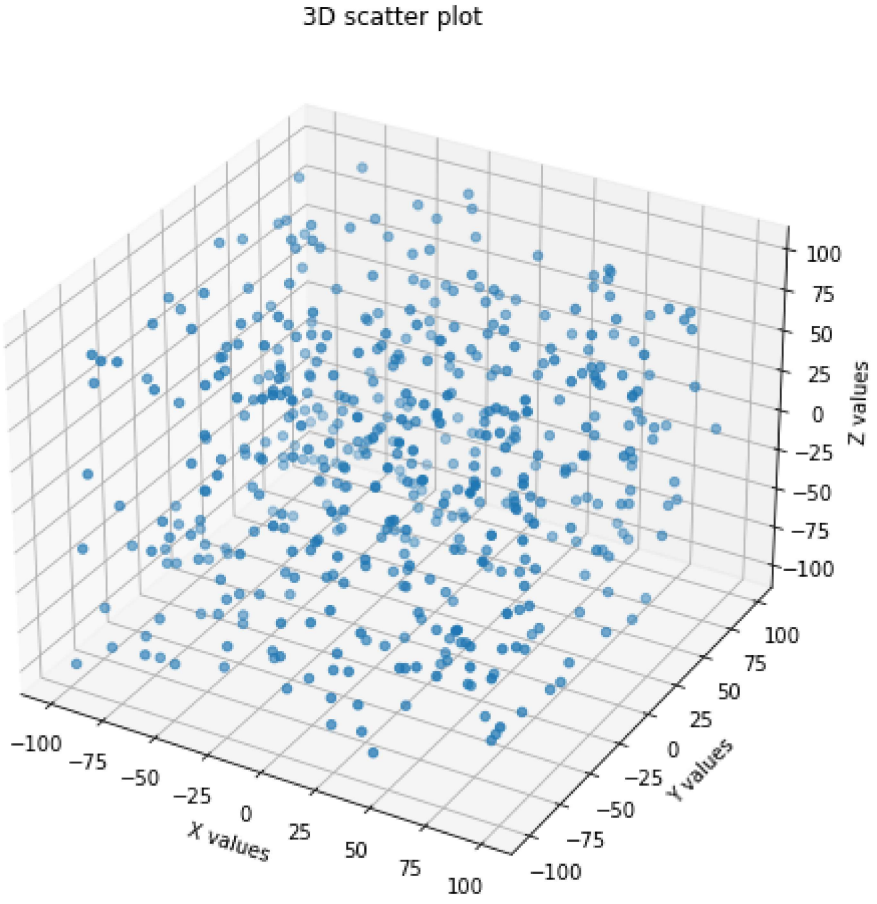
```
#ax = plt.axes(projection = "3d")

fig = plt.figure(figsize = [12,8])
ax = fig.gca(projection = '3d')
xval = [2,-2]
yval = [3,-3]
zval = [4,-4]
ax.scatter3D(xval, yval, zval, color = 'blue')

```

Out[9]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1c945f744c0>



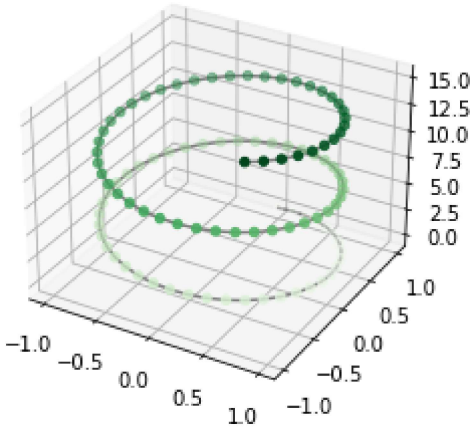
In [10]:

```
1
2  fig = plt.figure(figsize =[12,8])
3  ax = fig.gca(projection = '3d')
4  x_values = np.random.randint(-100,100,(500,))
5  y_values = np.random.randint(-100,100,(500,))
6  z_values = np.random.randint(-100,100,(500,))
7
8  ax.scatter(x_values,y_values,z_values)
9  ax.set_title("3D scatter plot")
10 ax.set_xlabel("X values")
11 ax.set_ylabel("Y values")
12 ax.set_zlabel("Z values")
13 plt.show()
```



In [11]:

```
ax = plt.axes(projection = '3d')
zline = np.linspace(0,15,100)
xline = np.sin(zline)
yline = np.cos(zline)

ax.plot3D(xline,yline,zline,"gray")
ax.scatter3D(xline,yline,zline, c = zline, cmap = "Greens")
None
```

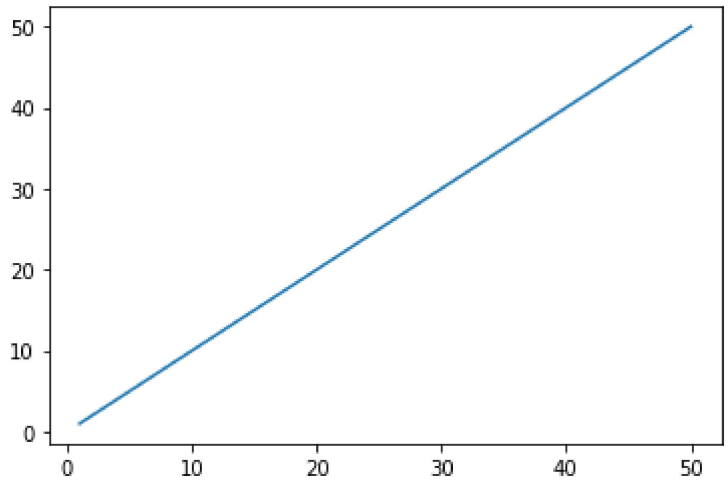


In [12]:

```
#import the library NumPy
import numpy as np

# import the library matplotlib
import matplotlib.pyplot as plt
```

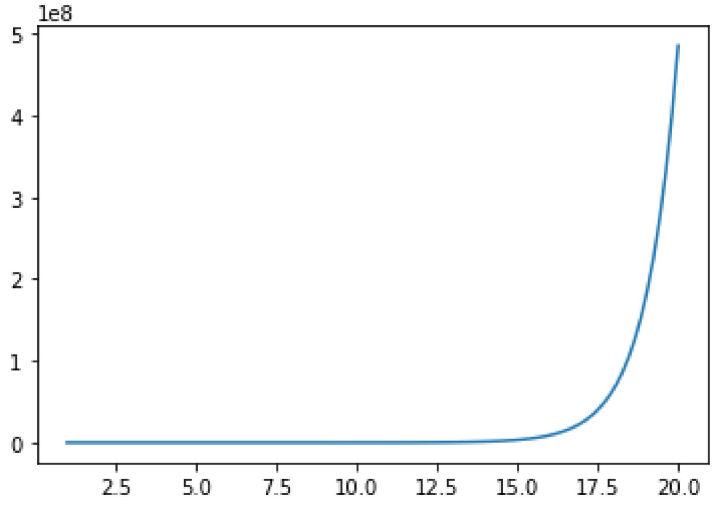
In [13]:

```
X = np.linspace(1, 50, 100)
Y = X

plt.plot(X,Y)

plt.show()
```



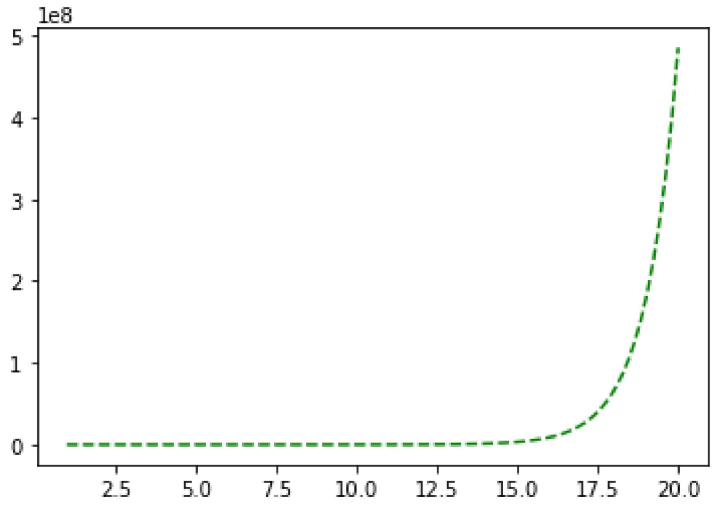
1) Write a python program to get the graph of an exponential function

In [14]:
```
X= np.linspace(1,20,100)
plt.plot(X, np.exp(X))
plt.show()

```



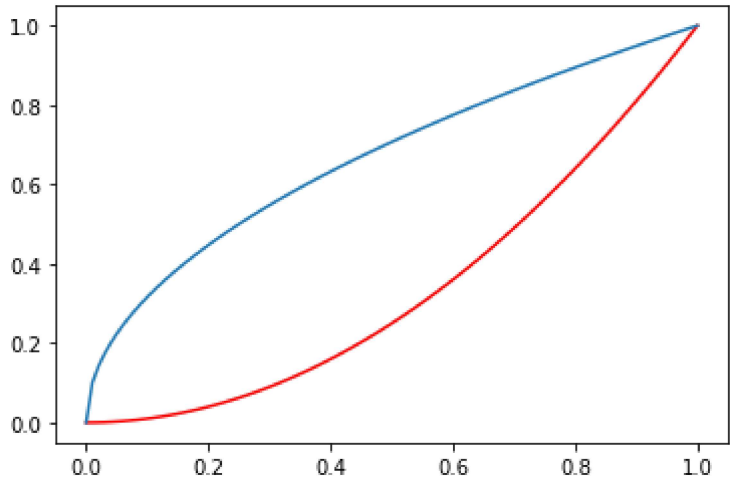
The above plot can be represented not only by a solid line, but also a dotted line with varied thickness. The points can be marked explicitly using any symbol

In [15]:
```
X = np.linspace(1,20,100)
plt.plot(X,np.exp(X),'--',color='g')
plt.show()
```



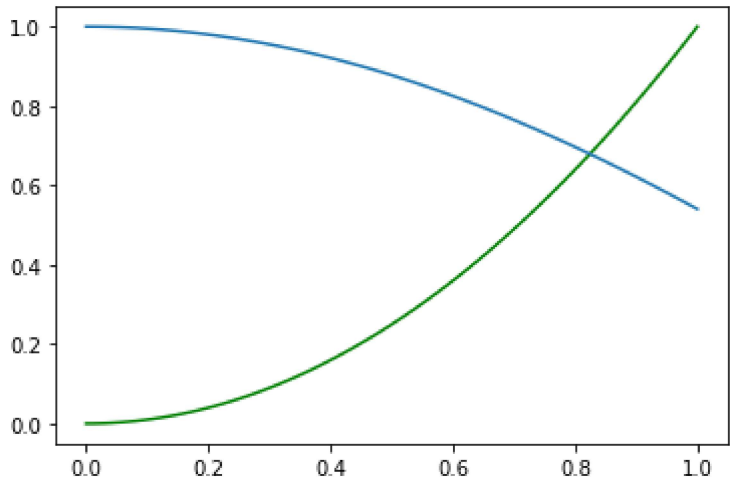
2) Write a python prpogram to plot the graphs of x^2 and $sqrt(x)$ in one plot

In [16]:
```
X = np.linspace(0,1,100)
plt.plot(X,X**2, color='r')
plt.plot(X, np.sqrt(X))
plt.show()
```



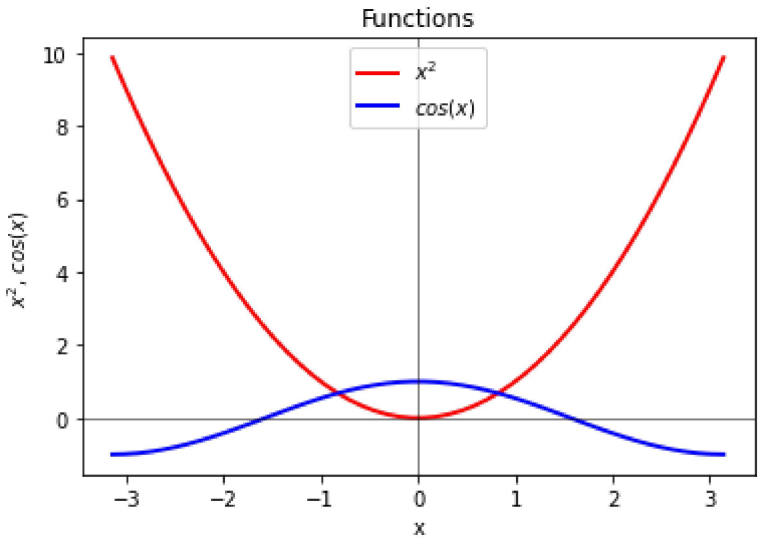
3) Plot the graphs of the functions x2 and cos(x) in a single plot**

In [17]:
```
X = np.linspace(0,1,100)
plt.plot(X,X**2, color='g')
plt.plot(X, np.cos(X))
plt.show()
```



In [18]:

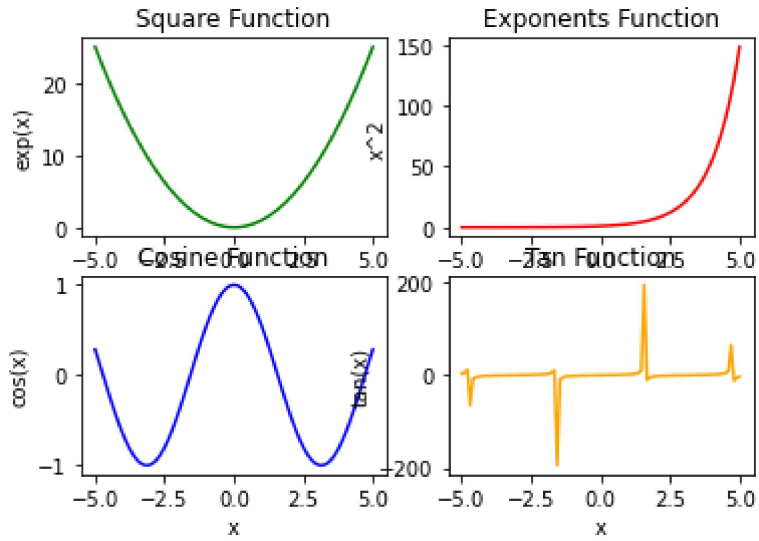
```
x = np.linspace(-np.pi, np.pi)
Y1 = x**2
Y2 = np.cos(x)

plt.plot(x, Y1, lw = 2, color = 'red', label = '$x^2$')
plt.plot(x, Y2, lw = 2, color = 'blue', label = '$cos(x)$')

plt.title('Functions')
plt.xlabel('x')
plt.ylabel('$x^2$, $cos(x)$')
plt.axhline(0, lw = 0.5, color = 'black')
plt.axvline(0, lw = 0.5, color = 'black')
plt.legend()
None
```



4) Plot the graph of i) e^x, ii) $x^2, iii) cos(x), iv) tan(x) as subplots

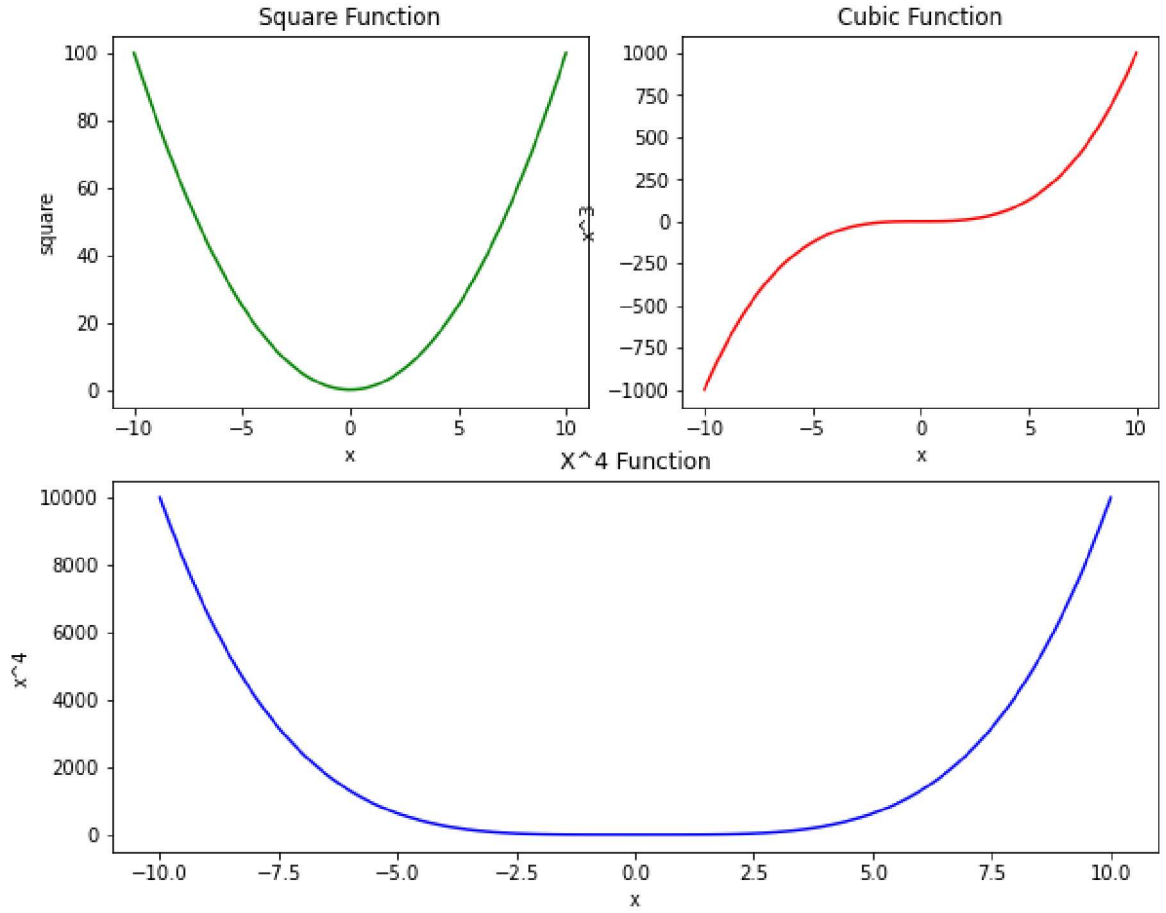
In [19]:

```
X = np.linspace(-5,5,100)

plt.subplot(2,2,1)
plt.plot(X,X**2, color='g')
plt.xlabel('x')
plt.ylabel('exp(x)')
plt.title('Square Function')

plt.subplot(2,2,2)
plt.plot(X, np.exp(X),  color='r')
plt.xlabel('x')
plt.ylabel('x^2')
plt.title('Exponents Function')

plt.subplot(2,2,3)
plt.plot(X, np.cos(X), color='b')
plt.xlabel('x')
plt.ylabel('cos(x)')
plt.title('Cosine Function')
plt.subplot(2,2,4)
plt.plot(X, np.tan(X), color='orange')
plt.xlabel('x')
plt.ylabel('tan(x)')
plt.title('Tan Function')

plt.show()
```



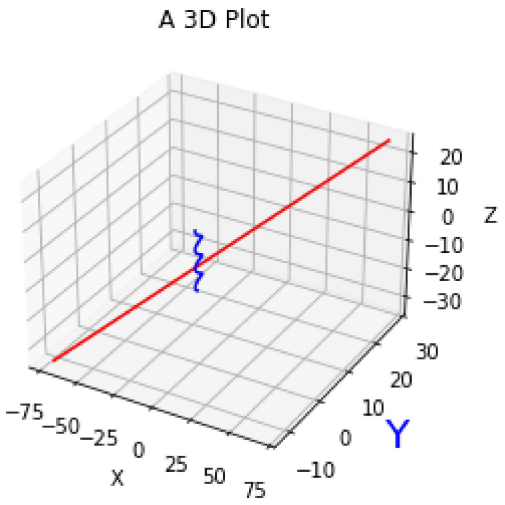
In [20]:

```
import numpy as np
import matplotlib.pyplot as plt


X = np.linspace(-10,10,100)
plt.figure(figsize = (10,8))


plt.subplot(2,2,1)
plt.plot(X,X**2, color='g')
plt.xlabel('x')
plt.ylabel('square')
plt.title('Square Function')

plt.subplot(2,2,2)
plt.plot(X, X**3,  color='r')
plt.xlabel('x')
plt.ylabel('x^3')
plt.title('Cubic Function')

plt.subplot(2,2,(3,4))
plt.plot(X, X**4, color='b')
plt.xlabel('x')
plt.ylabel('x^4')
plt.title('X^4 Function')


plt.show()
```



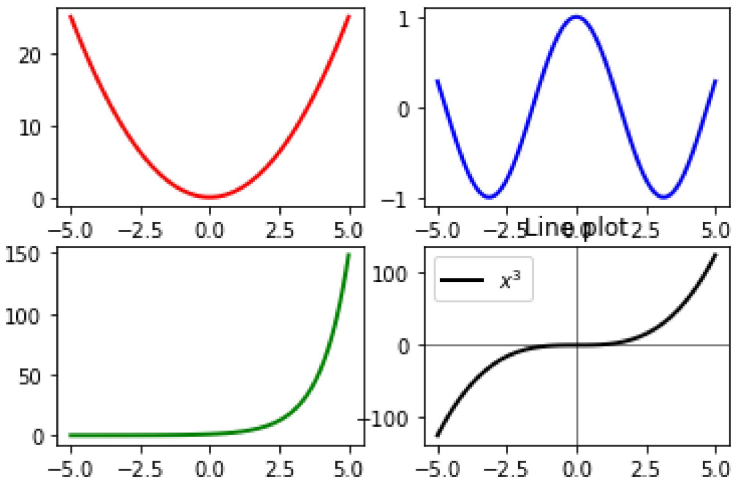
In [21]:

```
ax = plt.axes(projection = '3d')

t = np.linspace(-10,10,100)
xval = -2 +7*t
yval = 8 + 2*t
zval = -5 + 3*t

ax.plot3D(xval, yval, zval, 'red')


x = np.sin(t)
y = np.cos(t)
z = t
ax.plot3D(x,y,z,'blue')


plt.title("A 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y", size = 20, color = 'blue')
ax.set_zlabel("Z", fontsize = 10)

None
```



In [22]:

```
x = np.linspace(-5,5,100)
y = np.square(x)
z = np.cos(x)
a = np.exp(x)
b = x**3


plt.subplot(2,2,1)
plt.plot(x,y,lw = 2, color = 'red', label = "$x^2$")
plt.subplot(2,2,2)
plt.plot(x,z,lw = 2,color = 'blue', label = '$cos(x)$')
plt.subplot(2,2,3)
plt.plot(x,a,lw = 2,color = 'green', label = '$e^(x)$')
plt.subplot(2,2,4)
plt.plot(x,b,lw = 2,color = 'black', label = '$x^3$')


plt.title("Line plot")
plt.legend()
plt.show

plt.axhline(0,lw = 0.5, color = "Black")
plt.axvline(0,lw = 0.5, color = "Black")
None
```



2) Consider the average heights and weights of persons stored in the following lists:

height = [121.9,124.5,129.5,134.6,139.7,147.3, 152.4, 157.5,162.6]

weight= [19.7,21.3,23.5,25.9,28.5,32.1,35.7,39.6, 43.2]

To plot a line chart where:

i) x axis will represent weight and y axis will represent height.

ii) Label x axis as "Weight in kg" and y axis as "Height in cm"

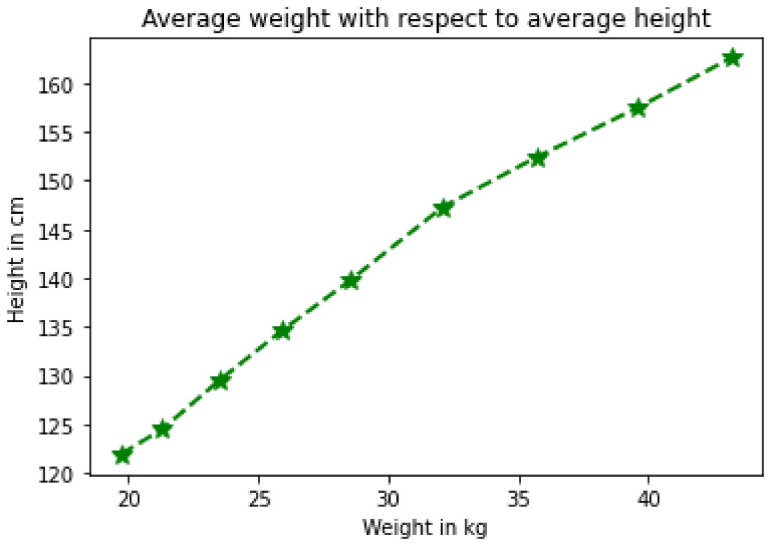
iii) Give the title as "Average weight with respect to average height"

iv) Give line color = green, line style = dashed and line width = 2

v) give marker as * and size of marker as 10.

In [23]:
```
#import the library NumPy
import numpy as np

# import the library matplotlib
import matplotlib.pyplot as plt
```

In [24]:
```
height = [121.9,124.5,129.5,134.6,139.7,147.3,152.4,157.5,162.6]
weight = [19.7,21.3,23.5,25.9,28.5,32.1,35.7,39.6,43.2]


plt.xlabel('Weight in kg')
plt.ylabel('Height in cm')
plt.title('Average weight with respect to average height')

plt.plot(weight, height, marker = '*', markersize=10, color='green', lw=
plt.show()
None
```



6) Scatter Plot

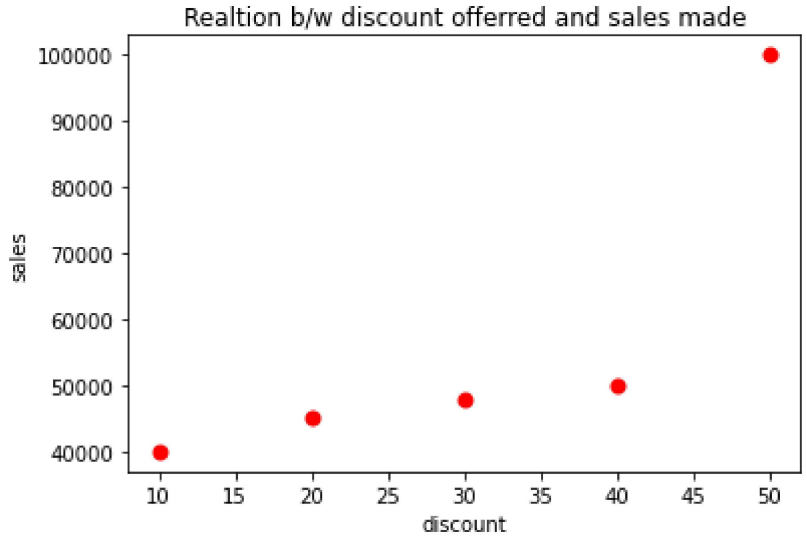
A scatter plot is a set of points plotted on horizontal and vertical aces. The scatter plot can be used to study the conrrelation between the two variables. One can also detect the extreme data points using a scatter plot.

Question

A shop which sells designer Kurtas gave discounts ranging from 10% to 50% over a period of 5 weeks, during the sales season. They made a sales of 40000, 45000, 48000, 50000, 100000 (in Rs) respectively. They recorded sales for each type of discount in an array. Draw a scatter plot to

show a relationship between the discount offered and sales made.

In [25]:
```
discount = [10, 20, 30, 40, 50]
sales = [40000, 45000, 48000, 50000, 100000]

plt.xlabel('discount')
plt.ylabel('sales')
plt.title('Realtion b/w discount offerred and sales made')

plt.scatter(discount, sales, color='red', lw=2)
plt.show()

```



Customizing Scatter Chart

i) Change the width

ii) Colour

iii) Marker

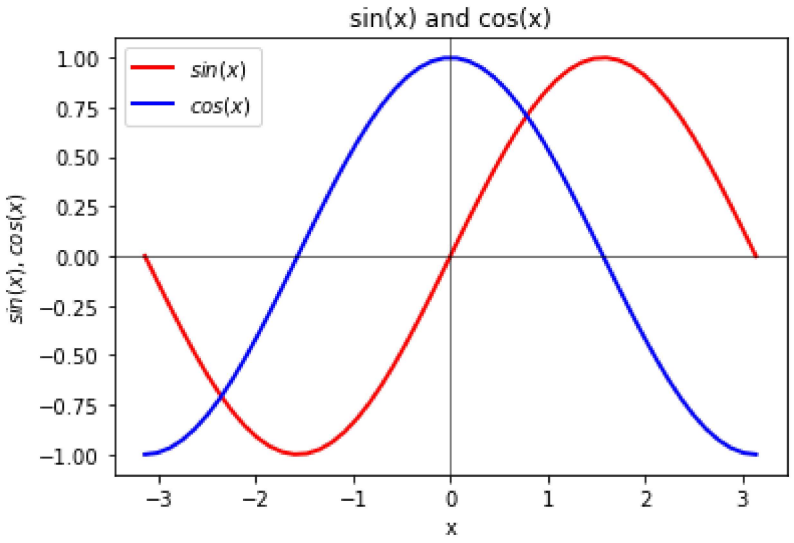
In [26]:

```
discount = [10, 20, 30, 40, 50]
sales = [40000, 45000, 48000, 50000, 100000]

plt.xlabel('discount')
plt.ylabel('sales')
plt.title('Realtion b/w discount offerred and sales made')

plt.scatter(discount, sales, color='purple', lw=5, marker = ',')
plt.show()
```

7) Plot the graph of i) sinx, ii) cosx in a single plot

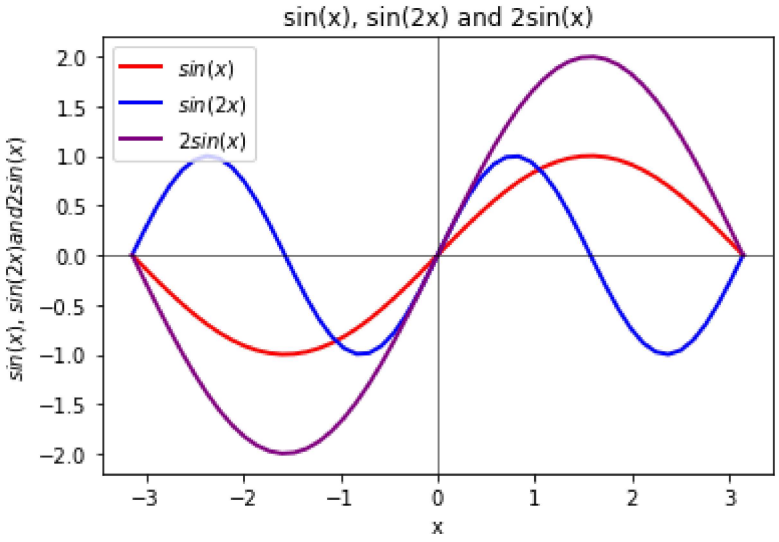
In [27]:

```
x = np.linspace(-np.pi, np.pi)
Y1 = np.sin(x)
Y2 = np.cos(x)

plt.plot(x, Y1, lw = 2, color = 'red', label = '$sin(x)$')
plt.plot(x, Y2, lw = 2, color = 'blue', label = '$cos(x)$')

plt.title('sin(x) and cos(x)')
plt.xlabel('x')
plt.ylabel('$sin(x)$, $cos(x)$')
plt.axhline(0, lw = 0.5, color = 'black')
plt.axvline(0, lw = 0.5, color = 'black')
plt.legend()
None
```



8) Plot the graph of i) sinx ii) sin2x, iii) 2sinx in single plot and use the legend function.

In [28]:
```
x = np.linspace(-np.pi, np.pi)
Y1 = np.sin(x)
Y2 = np.sin(2*x)
Y3 = 2*np.sin(x)

plt.plot(x, Y1, lw = 2, color = 'red', label = '$sin(x)$')
plt.plot(x, Y2, lw = 2, color = 'blue', label = '$sin(2x)$')
plt.plot(x, Y3, lw = 2, color = 'purple', label = '$2sin(x)$')

plt.title('sin(x), sin(2x) and 2sin(x)')
plt.xlabel('x')
plt.ylabel('$sin(x)$, $sin(2x) and 2sin(x)$')
plt.axhline(0, lw = 0.5, color = 'black')
plt.axvline(0, lw = 0.5, color = 'black')
plt.legend()
None
```

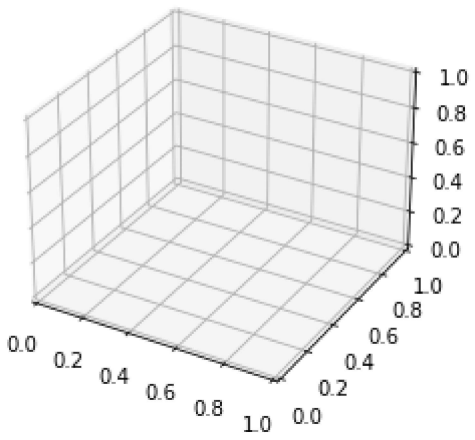


3D - Plots

In [29]:
```
from mpl_toolkits import mplot3d
```

In [30]:
```
import matplotlib.pyplot as plt
from numpy import *
```

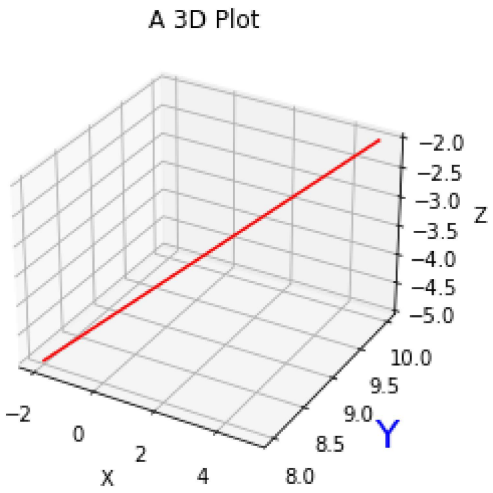
In [31]:
```
1 ax = plt.axes(projection = '3d')
```



9) Plot $x = -2 + 7t, y = 9 + 2t, z = -5 + 3t$

In [32]:

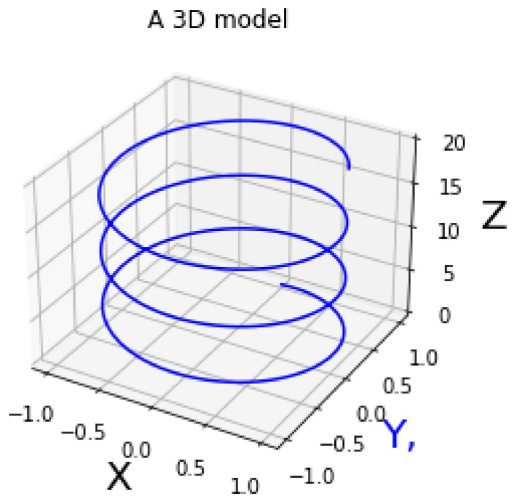
```
ax = plt.axes(projection = '3d')

t = linspace(0,1,10)
xval = -2 +7*t
yval = 8 + 2*t
zval = -5 + 3*t

ax.plot3D(xval, yval, zval, 'red')

plt.title("A 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y", size = 20, color = 'blue')
ax.set_zlabel("Z", fontsize = 10)

None
```



10) Plot $r(t) = (sint)i + (cost)j + tk$

In [33]:
```
ax = plt.axes(projection = '3d')

t = linspace(0,20,1000)
x = sin(t)
y = cos(t)
z = t
ax.plot3D(x,y,z,'blue')

plt.title("A 3D model")

ax.set_xlabel("X", fontsize = 20)
ax.set_ylabel("Y,",size = 20, color = 'blue')
ax.set_zlabel("Z", fontsize = 20)
```

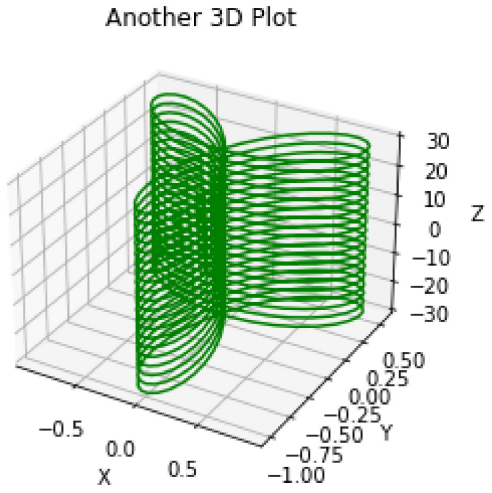
Out[33]: Text(0.5, 0, 'Z')



11) Plot the curve $r(t) = (sin3t)(cost)i + (sin3t)(sint)j + tk$

In [34]:

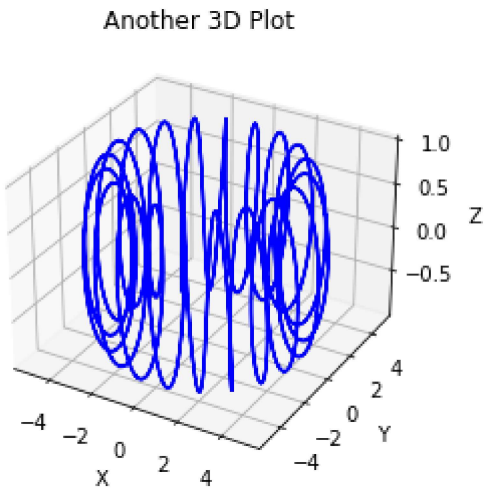
```
ax = plt.axes(projection = '3d')

t = linspace(-30, 30, 1000)
xval = sin(3*t)*cos(t)
yval = sin(3*t)*sin(t)
zval = t

ax.plot3D(xval, yval, zval, 'green',
          markersize = 1, markeredgewidth = 0.1,)


plt.title("Another 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")

None
```



12) Plot the curve $r(t) = (4 + sin20t)(cost)i + (4 + sin20t)(sint)j + (cos20t)k$

In [35]:

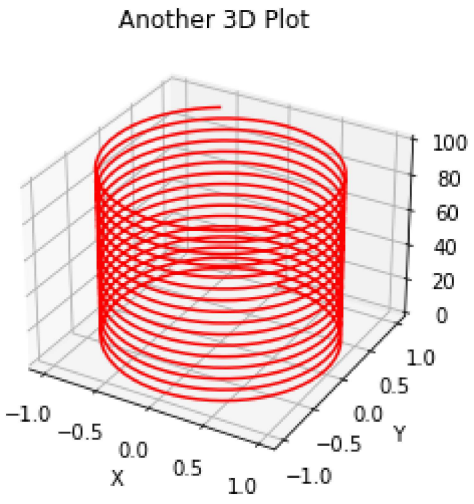
```
ax = plt.axes(projection = '3d')

t = linspace(-2*pi, 2*pi, 1000)
xval = (4 + sin(20*t))*cos(t)
yval = (4 + sin(20*t))*sin(t)
zval = cos(20*t)

ax.plot3D(xval, yval, zval, color = 'blue',  )


plt.title("Another 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")

None
```



Q Plot x= sin(z), y = cos(z)

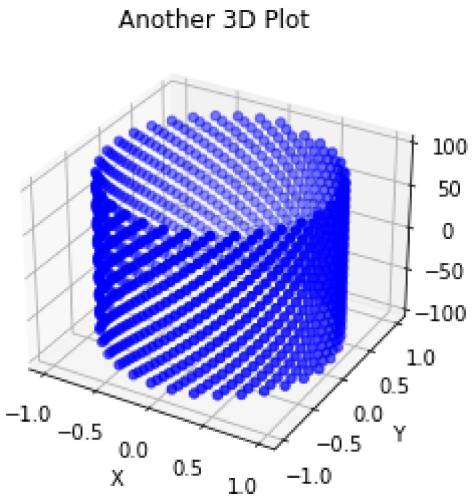
In [36]:

```
ax = plt.axes(projection = '3d')

z = linspace(0, 100, 1000)
xval = sin(z)
yval = cos(z)

ax.plot3D(xval, yval, z, 'red',  )


plt.title("Another 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y")

None
```



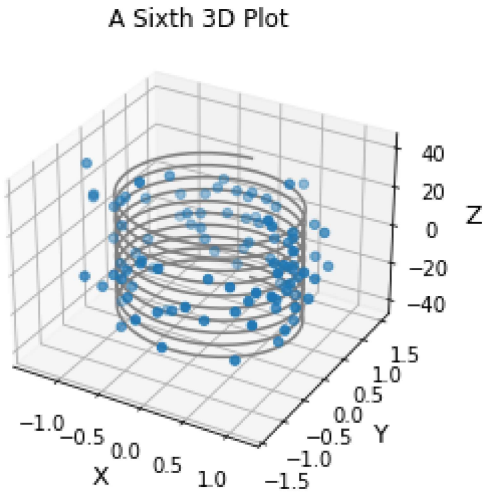
In [37]:

```
ax = plt.axes(projection = '3d')

z = linspace(-30*pi,30*pi,1000)
xval = sin(z)
yval = cos(z)

ax.scatter3D(xval, yval, z, color = 'blue',  )


plt.title("Another 3D Plot")

ax.set_xlabel("X")
ax.set_ylabel("Y")

None
```



In [38]:

```
ax = plt.axes(projection = '3d')

z = linspace(-10*pi, 10*pi, 1000)
xval = sin(z)
yval = cos(z)

Z = 15*random.randn(100)
Y = sin(Z) + 0.2*random.randn(100)
X = cos(Z) + 0.2*random.randn(100)

ax.plot3D(xval, yval, z, 'gray')
ax.scatter3D(X, Y, Z, 'coral')
plt.title("A Sixth 3D Plot")

ax.set_xlabel("X", fontsize = 12.5)
ax.set_ylabel("Y", fontsize = 12.5)
ax.set_zlabel("Z", fontsize = 12.5)
None
```

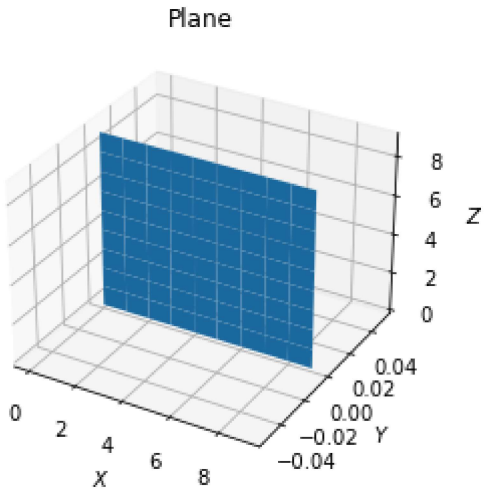


In [39]:

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.pyplot import *
from numpy import *
```

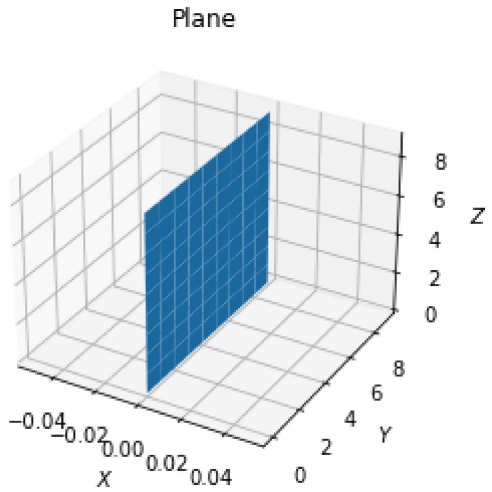
In [40]:

```
xx, zz = meshgrid(range(10), range(10))

y = 0

#plot the surface
ax = figure().gca(projection = '3d')
ax.plot_surface(xx,y,zz)

title("Plane")
ax.set_xlabel('$X$')
ax.set_ylabel('$Y$')
ax.set_zlabel('$Z$')
show()
```



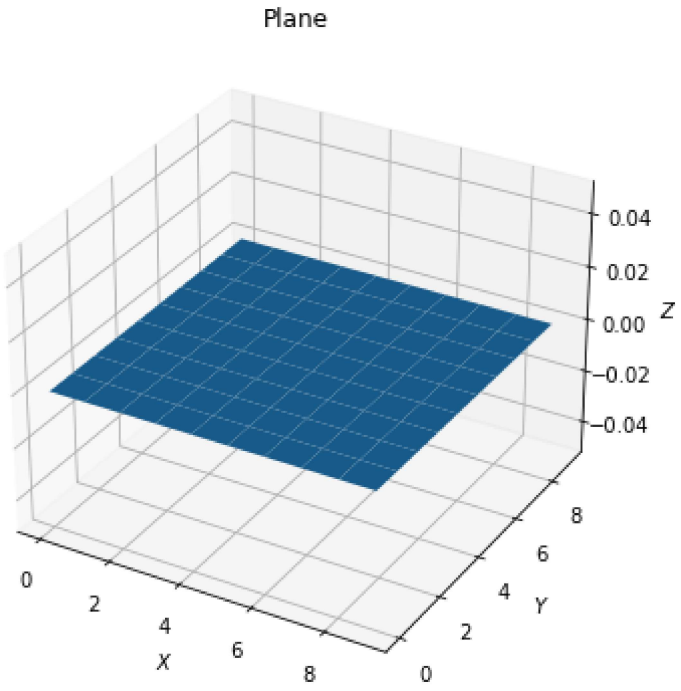
In [41]:

```
zz, yy = meshgrid(range(10), range(10))

x = 0

#plot the surface
ax = figure().gca(projection = '3d')
ax.plot_surface(x,yy,zz)

title("Plane")
ax.set_xlabel('$X$')
ax.set_ylabel('$Y$')
ax.set_zlabel('$Z$')
show()
```



In [42]:

```
xx, yy = meshgrid(range(10), range(10))

z = 0 * xx - 0 * yy - 0

fig = figure(figsize = (6,6))
ax = fig.add_subplot(1,1,1,projection = '3d')
ax.plot_surface(xx, yy ,z)

title("Plane")
ax.set_xlabel('$X$')
ax.set_ylabel('$Y$')
ax.set_zlabel('$Z$')
show()
```

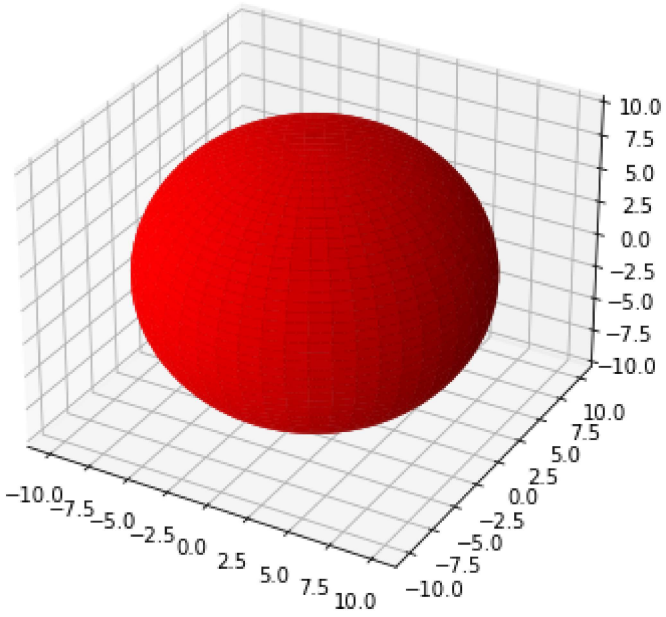


In [43]:

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.pyplot import *
from numpy import *
```

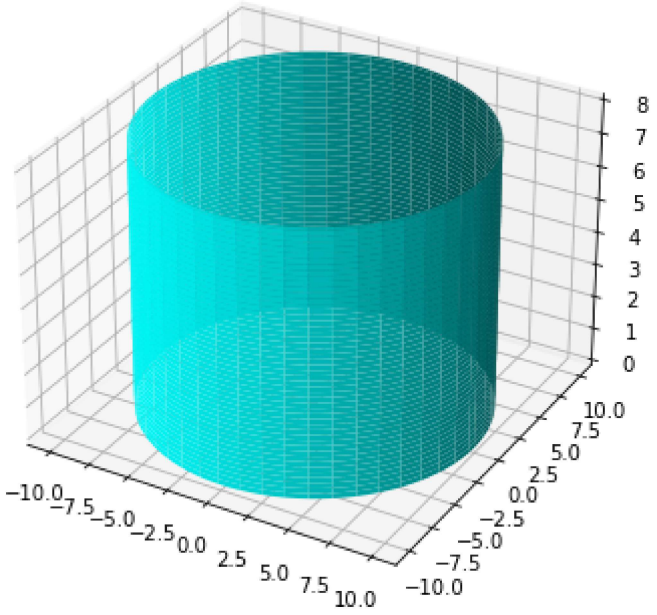
In [44]:

```
fig = figure(figsize=(8,6))

ax = fig.add_subplot(1, 1, 1, projection = '3d')

u = linspace(0, 2*pi, 100)
v = linspace(0, pi, 100)

x = 10*outer(cos(u), sin(v))
y = 10*outer(sin(u), sin(v))
z = 10*outer((ones(size(v))), cos(v))

ax.plot_surface(x,y,z,color='r')
show()
```



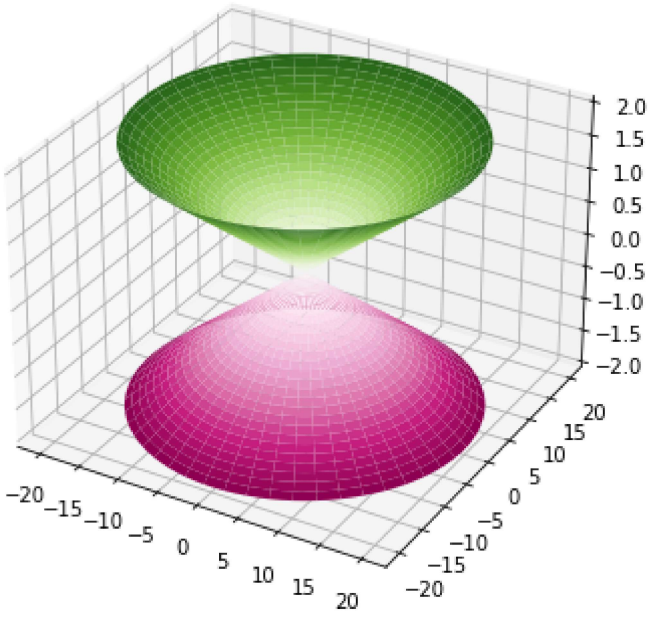
In [45]:

```
fig = figure(figsize = (8,6))

ax = fig.add_subplot(1,1,1, projection = '3d')

u = linspace(0,2*pi, 100)
v = linspace(0,8,100)

x = 10*outer((ones(size(u))),cos(u))
y = 10*outer((ones(size(u))),sin(u))
z = outer(v,ones(size(v)))

ax.plot_surface(x,y,z, color = 'cyan')
show()
```



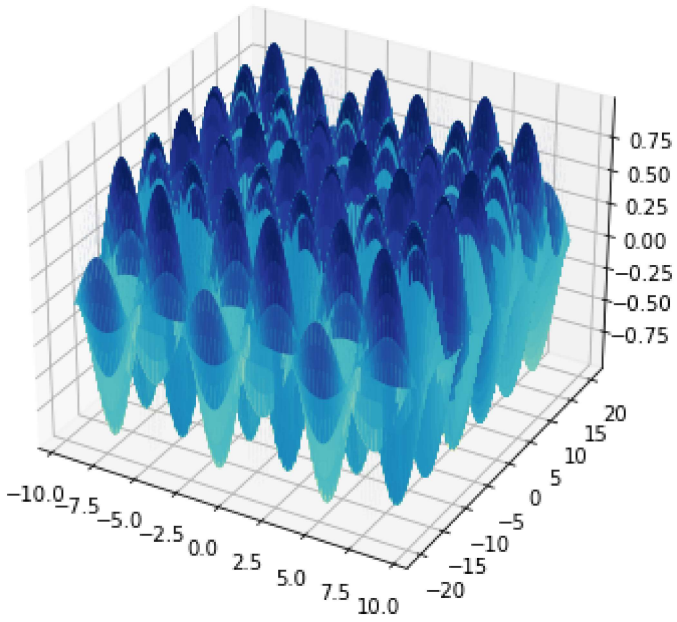
In [46]:

```
fig = figure(figsize = (8,6))

ax = fig.add_subplot(1,1,1, projection = '3d')

u = linspace(0,2*pi, 100)
v = linspace(-2,2,100)

x = 10*outer(v,cos(u))
y = 10*outer(v,sin(u))
z = outer(v,ones(size(u)))

ax.plot_surface(x,y,z, cmap = 'PiYG')
show()
```



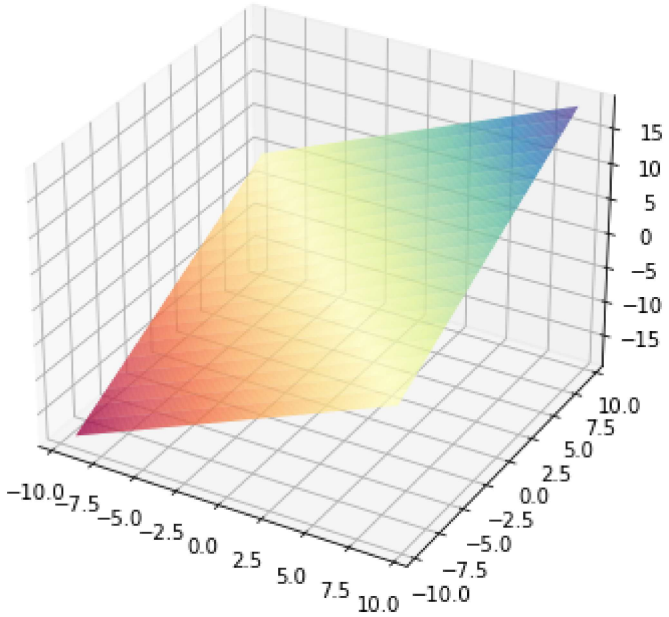
In [47]:

```
fig = figure(figsize = (8,6))

ax = fig.add_subplot(1,1,1, projection = '3d')

x = arange(-3*pi, 3*pi,0.1)
v = arange(-3*pi, 3*pi,0.1)

xx,yy = meshgrid(x,y)
z = sin(xx)*sin(yy)

ax.plot_surface(xx,yy,z, cmap = 'YlGnBu', cstride = 2)
show()
```



In [48]:

```
fig = plt.figure(figsize = (8,6))

ax = fig.add_subplot(1,1,1, projection = '3d')

x = arange(-3*pi,3*pi,0.1)
y = arange(-3*pi,3*pi,0.1)

xx,yy = meshgrid(x,y)
z = xx + yy

ax.plot_surface(xx,yy,z, cmap = cm.Spectral, cstride = 1)
show()
```



Conclusion:

Through this topic, we have learnt how to graph various 2 dimensional and 3 dimensional graphs on python

Topic 3: Vectors and vector operations

Date: 09.11.2022

Vector is a quantity that has both magnitude and direction.

A vector u = <u1, u2, u3> is defined in python using teh array function from the library numpy.

Syntax:

u = np.array([u1,u2,u3])

In [50]:
```
u = np.array([1,2,3])
u
```

Out[50]: array([1, 2, 3])

In [51]:
```
round(10/3,4)
```

Out[51]: 3.3333

In [53]:
```
u = np.array([1,2,3,4])
z = np.array([4,3,2,1])
```

In [54]:
```
u+z
```

Out[54]: array([5, 5, 5, 5])

In [55]:
```
u - z
```

Out[55]: array([-3, -1, 1, 3])

In [56]:
```
u*5
```

Out[56]: array([5, 10, 15, 20])

In [57]:
```
u/z
```

Out[57]: array([0.25   , 0.66666667, 1.5   , 4.   ])

In [58]:
```
u*z
```

Out[58]: array([4, 6, 6, 4])

In [59]:
```
np.dot(u,z)
```

Out[59]: 20

In [60]:
```
5*u - 3*z
```

Out[60]: array([-7, 1, 9, 17])

In [61]:
```
import math

def mag(x):
    return math.sqrt(sum(i**2 for i in x))
```

In [63]:
```
np.linalg.norm(u)
```

Out[63]: 5.477225575051661

In [64]:
```
mag = 0
for i in u:
    mag += i**2
print(np.sqrt(mag))
```

```
5.477225575051661
```

In [65]:
```
u
```

Out[65]: array([1, 2, 3, 4])

In [66]:
```
z
```

Out[66]: array([4, 3, 2, 1])

In [67]:
```
np.dot(u,z)
```

Out[67]: 20

Topic 4: Contour plots and meshgrid

Date : 20.08.2022

Contour Plots

- A contour line or isoline of a function of two vraiables

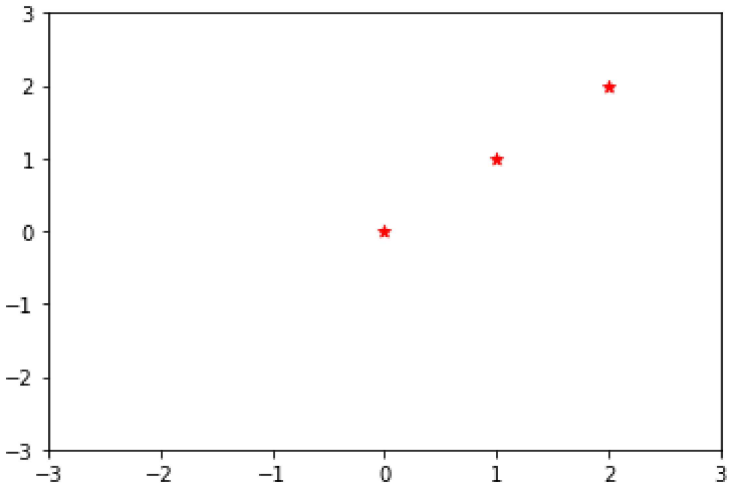
Meshgrid

In [68]:
```
from numpy import *

x = arange(0,3,1)
y = arange(0,3,1)

gx, gy = meshgrid(x,y)
print(gx)
print(gy)
```

```
[[0 1 2]
 [0 1 2]
 [0 1 2]]
[[0 0 0]
 [1 1 1]
 [2 2 2]]
```

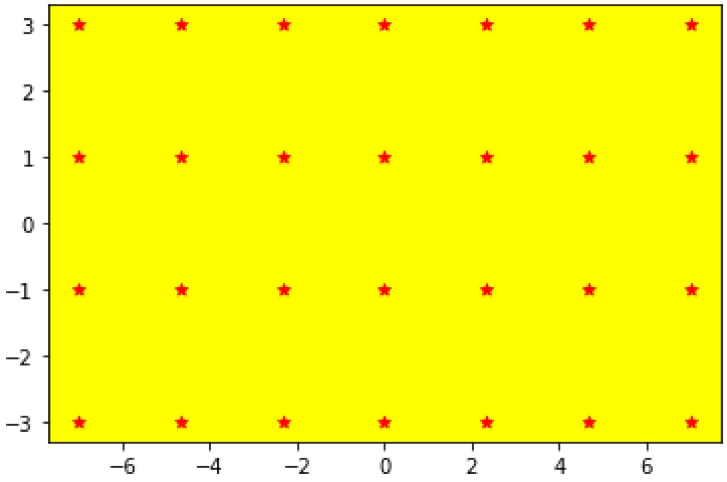
In [69]:
```
import matplotlib.pyplot as plt
fig , ax = plt.subplots()

plt.scatter(x,y, color = "red", marker = "*")
plt.ylim(-3,3)
plt.xlim(-3,3)
```

Out[69]: (-3.0, 3.0)



In [70]:

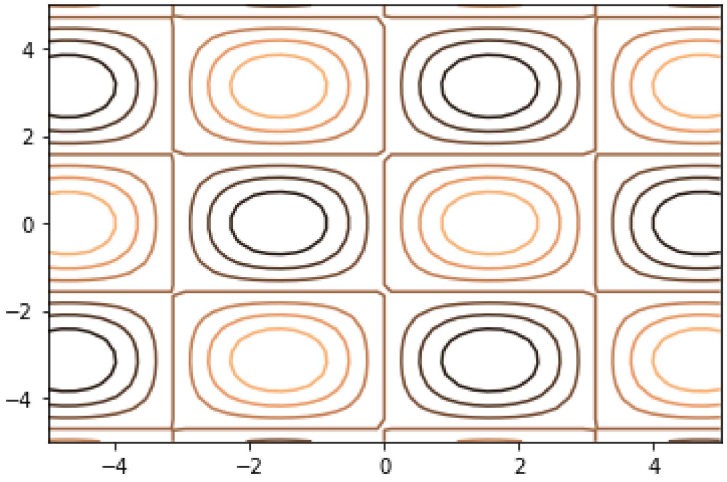
```
import numpy as np
xlist = np.linspace(-7,7,7)
ylist = np.linspace(-3,3,4)
X,Y = meshgrid(xlist,ylist)

fig, ax = plt.subplots()
ax.scatter(X,Y,color = 'red', marker = "*")
ax.set_facecolor("yellow")
None
```



A contour plot can be createad with the plt.contour function

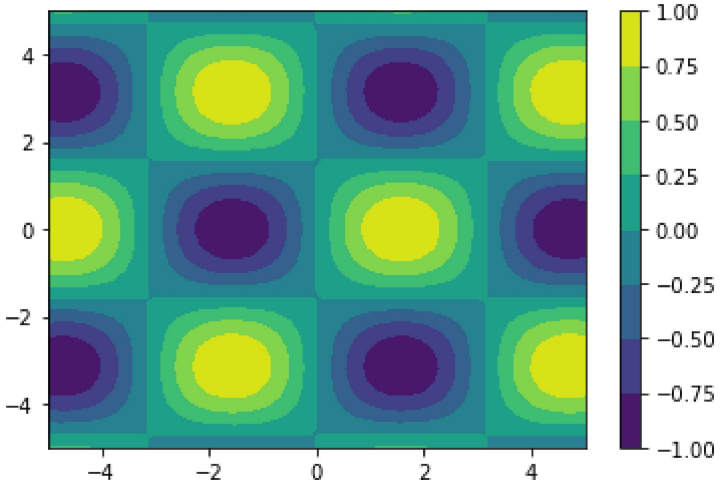
It takes three arguments: a grid of x values, a grid of y values, and a grid of z values

In [71]:
```
f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,50)
y = np.linspace(-5,5,40)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

plt.contour(X,Y,Z,cmap="copper")
None
```



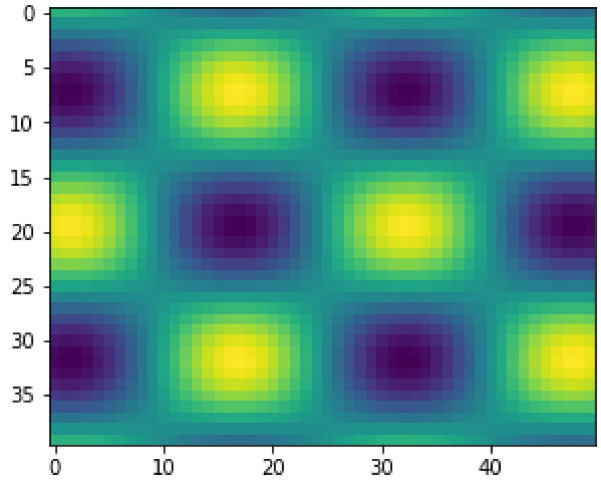
In [72]:
```
plt.contourf(X,Y,Z)
plt.colorbar()
```

Out[72]: <matplotlib.colorbar.Colorbar at 0x1c94b5275e0>



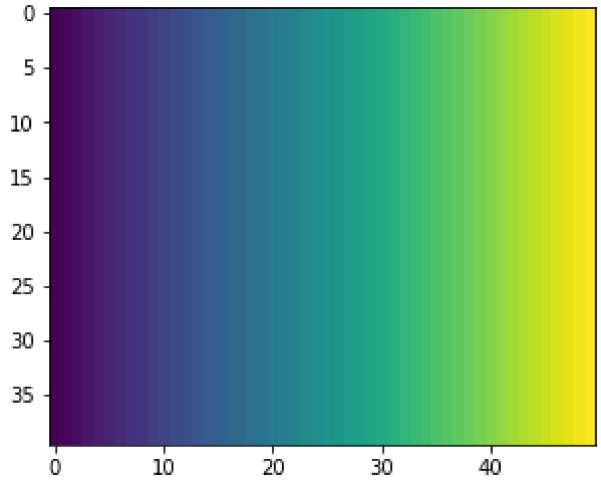
In [73]:
```
1 plt.imshow(Z)
```

Out[73]: <matplotlib.image.AxesImage at 0x1c951112940>



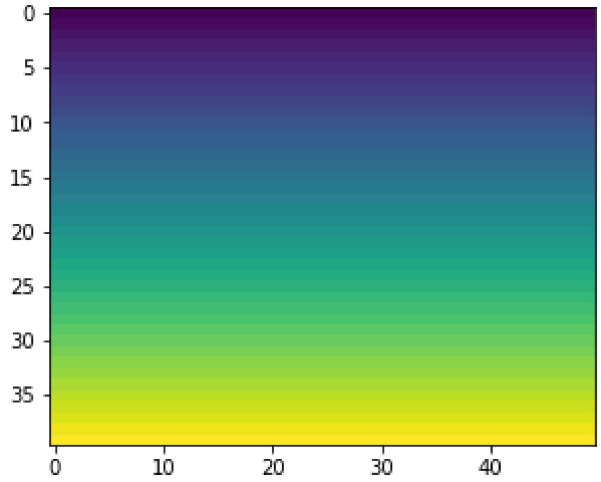
In [74]:
```
1 plt.imshow(X)
```

Out[74]: <matplotlib.image.AxesImage at 0x1c950f1f490>

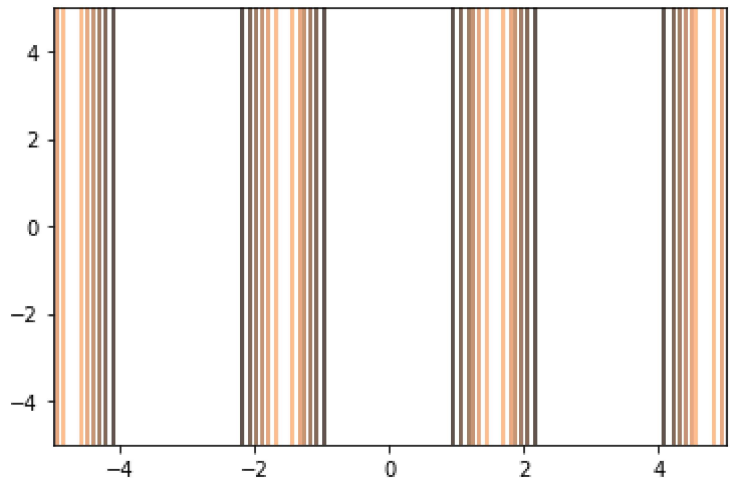


In [75]:
```
1 plt.imshow(Y)
```

Out[75]: <matplotlib.image.AxesImage at 0x1c9617a0340>



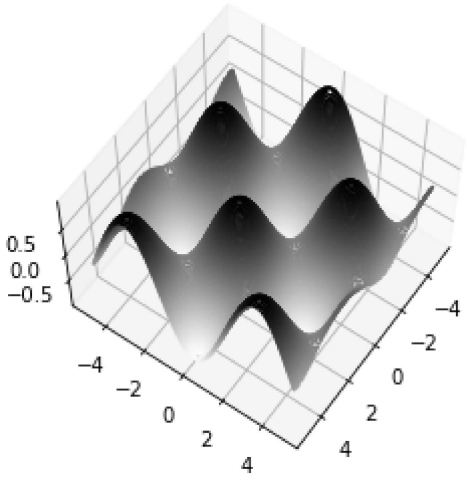
In [76]:
```
1 f = lambda x,y: np.sin(x)**10
2 x = np.linspace(-5,5,50)
3 y = np.linspace(-5,5,40)
4
5 X,Y = np.meshgrid(x,y)
6 Z = f(X,Y)
7
8 plt.contour(X,Y,Z,cmap="copper")
9 None
```



Three dimensional Contour Plots

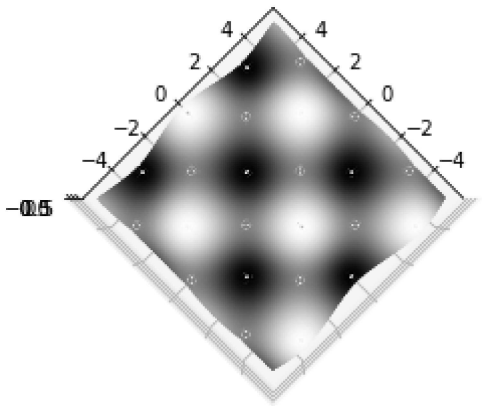
In [78]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")

f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,50)
y = np.linspace(-5,5,40)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.contour3D(X,Y,Z, 100, cmap = 'binary')

ax.view_init(60,35)
None
```



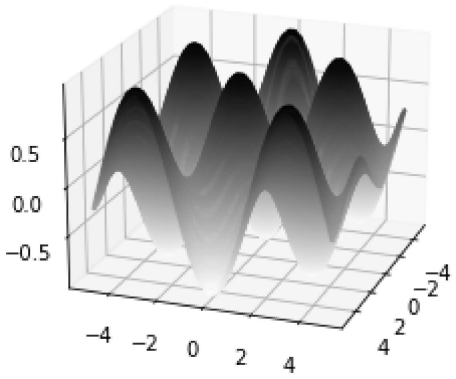
In [79]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")

f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,50)
y = np.linspace(-5,5,40)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.contour3D(X,Y,Z, 100, cmap = 'binary')

ax.view_init(-90,45)
None
```

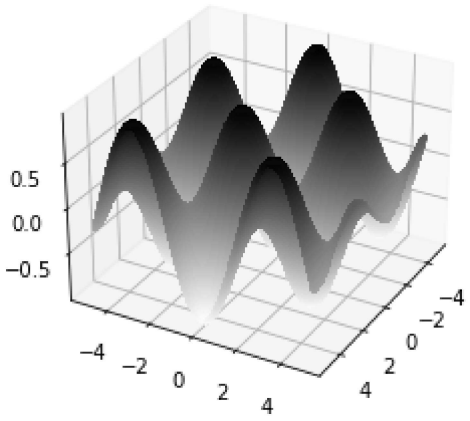


In [80]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")

f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,50)
y = np.linspace(-5,5,40)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.contour3D(X,Y,Z, 100, cmap = 'binary')

ax.view_init(20,20)
None
```



In [81]:
```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")
f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,50)
y = np.linspace(-5,5,40)

for i in range(-30,30):
    X,Y = np.meshgrid(x,y)
    Z = f(X,Y)
    ax.contour3D(X,Y,Z, 100, cmap = 'binary')

    ax.view_init(i,i)
    None
```



In [ ]:
```

```

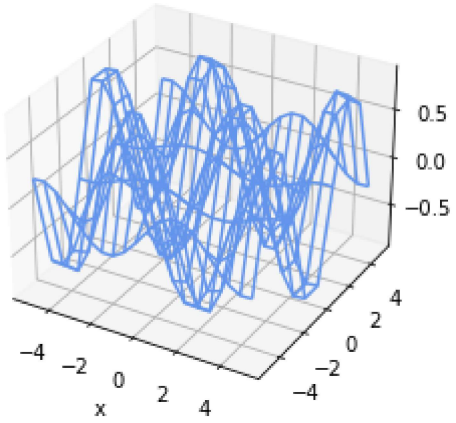
In [82]:
```
help(ax.view_init)
```

```
Help on method view_init in module mpl_toolkits.mplot3d.axes3d:

view_init(elev=None, azim=None) method of matplotlib.axes._subplots.Axes3DS
ubplot instance
    Set the elevation and azimuth of the axes in degrees (not radians).

    This can be used to rotate the axes programmatically.

    'elev' stores the elevation angle in the z plane (in degrees).
    'azim' stores the azimuth angle in the (x, y) plane (in degrees).

    if 'elev' or 'azim' are None (default), then the initial value
    is used which was specified in the :class:`Axes3D` constructor.
```

Wireframes and surfaceplots

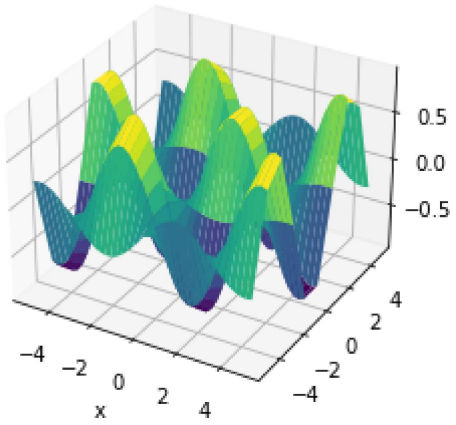
In [83]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")

f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,20)
y = np.linspace(-5,5,10)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.plot_wireframe(X,Y,Z,color = 'cornflowerblue')
ax.set_xlabel("x")

None
```



In [85]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

ax = plt.axes(projection = "3d")

f = lambda x,y: np.sin(x)*np.cos(y)
x = np.linspace(-5,5,40)
y = np.linspace(-5,5,10)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.plot_surface(X,Y,Z,rstride = 1, cstride = 1, cmap = "viridis",edgecol
ax.set_xlabel("x")

None
```

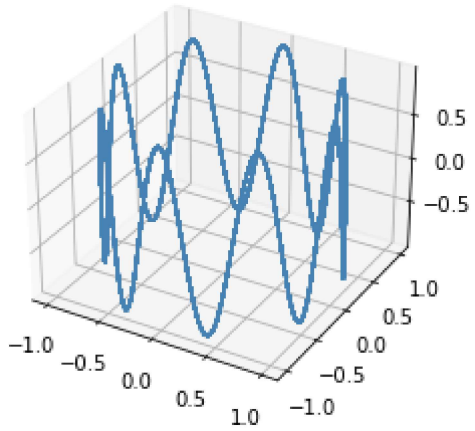


Plotting the following curves:

- (sint,cost,cos8t)
- (tcost,tsint,t)
- (t,t^2,t^3)

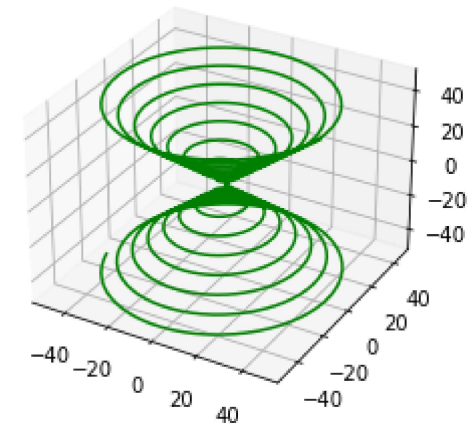
In [86]:

```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
import numpy as np

ax = plt.axes(projection='3d')
t = np.linspace(-100,100,10000)
# Data for a three-dimensional line
zline = np.cos(8*t)
xline = np.sin(t)
yline = np.cos(t)
ax.plot3D(xline, yline, zline, 'steelblue')
None
```



In [87]:

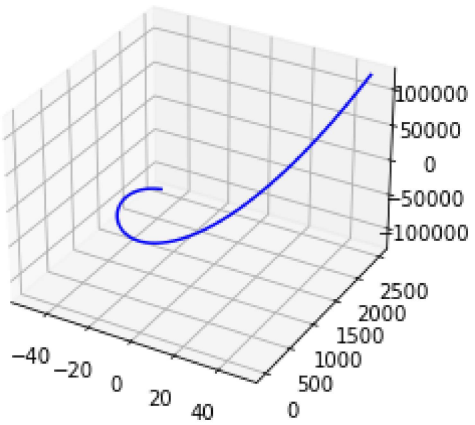
```
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
import numpy as np

ax = plt.axes(projection='3d')

# Data for a three-dimensional line
zline = np.linspace(-50, 50, 1000)
xline = zline*np.cos(zline)
yline = zline*np.sin(zline)
ax.plot3D(xline, yline, zline, 'green')
None
```



In [88]:
```
1 import numpy as np
2 a = np.linspace(1,10,3)
3 a
```

Out[88]:
```
array([ 1. ,  5.5, 10. ])
```

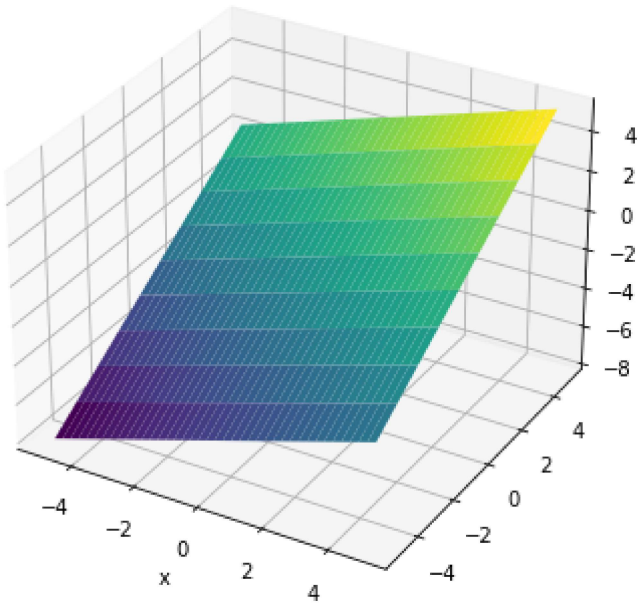
In [89]:
```
1  from mpl_toolkits import mplot3d
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  ax = plt.axes(projection='3d')
6
7  # Data for a three-dimensional line
8  xline = np.linspace(-50, 50, 1000)
9  yline = xline**2
10 zline = xline**3
11 ax.plot3D(xline, yline, zline, 'blue')
12 None
13
```



In [90]:

```
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

fig = plt.figure(figsize = [10, 6])
ax = fig.gca(projection = "3d")

a = int(input("a?"))
b = int(input("b?"))
c = int(input("c?"))
d = int(input("d?"))


f = lambda x,y: (a*x + b*y -d)/c
x = np.linspace(-5,5,40)
y = np.linspace(-5,5,10)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)

ax.plot_surface(X,Y,Z,rstride = 1, cstride = 1, cmap = "viridis",edgecol
ax.set_xlabel("x")

None
```

```
a?3
b?5
c?6
d?8
```



Topic 5: Differentiation and Integration

Date: 10.09.2022

Solve:

1) 3x^2 -2x+1=0

2) 3x-2y = 1 x+2y = 2

3) d/dx(3x^3 -sqrt(x)+e^x)

4) Intg(sin(x)^2 + e^2x)dx

5) [Intg(12x^4/7 + x^3 - 2x^2/9 -57)dx] 0 to

In [106]:
```
import sympy as sp
x,y = sp.symbols("x, y")

```

In [108]:
```
sp.solve(3*x**2-2*x+1)
```

Out[108]:
```
[1/3 - sqrt(2)*I/3, 1/3 + sqrt(2)*I/3]
```

In [109]:
```
sp.solve([3*x-2*y -1, x+2*y -2])

```

Out[109]:
```
{x: 3/4, y: 5/8}
```

In [110]:
```
sp.diff(3*x**3 - sp.sqrt(x) + sp.exp(x), x)
```

Out[110]:
$$9x^2 + e^x - \frac{1}{2\sqrt{x}}$$

In [111]:
```
sp.integrate(sp.sin(x)**2 + sp.exp(x)**2)
```

Out[111]:
$$\frac{x}{2} + \frac{e^{2x}}{2} - \frac{\sin(x)\cos(x)}{2}$$

In [112]:
```
sp.integrate(12*x**4/7 + x**3 -2*x*x/9 - 57, (x,0,1))
```

Out[112]:
$$-\frac{213499}{3780}$$

Arc Length

Arc length is the distance between the two points along a section of a curve. Determining the length of an irregular arc segment is also called rctification of a curve. The advent of the inifinitestimal calculus led to a general formula that provides a closed form solutions in some cases. To compute the length of a curve on the interval [a, b] we compute the integral

1. Find the arc length of f(x) = (x^3)/2 on [0, 2]
2. Find the arc length of $f(x) = x\hat{}2/8 - ln(x)$ on [1, 2]
3. Find the arc length of $f(x) = 13(x2 + 2)3/2$ on [2,3]

In [113]:
```
float(sp.integrate(sp.sqrt(1+ (sp.diff((x**3)/2))**2), (x,0,2)))
```

Out[113]: 4.842793473046998

In [114]:
```
float(sp.integrate(sp.sqrt(1+ (sp.diff((x**2)/8- sp.ln(x))**2)), (x,1,2)
```

Out[114]: 1.0681471805599454

In [115]:
```
float(sp.integrate(sp.sqrt(1+ (sp.diff(1/(3*((x**2+2)**3/2))))), (x,2,3)
```

Out[115]: 0.9987061153261502

Topic 6 : Area of a region

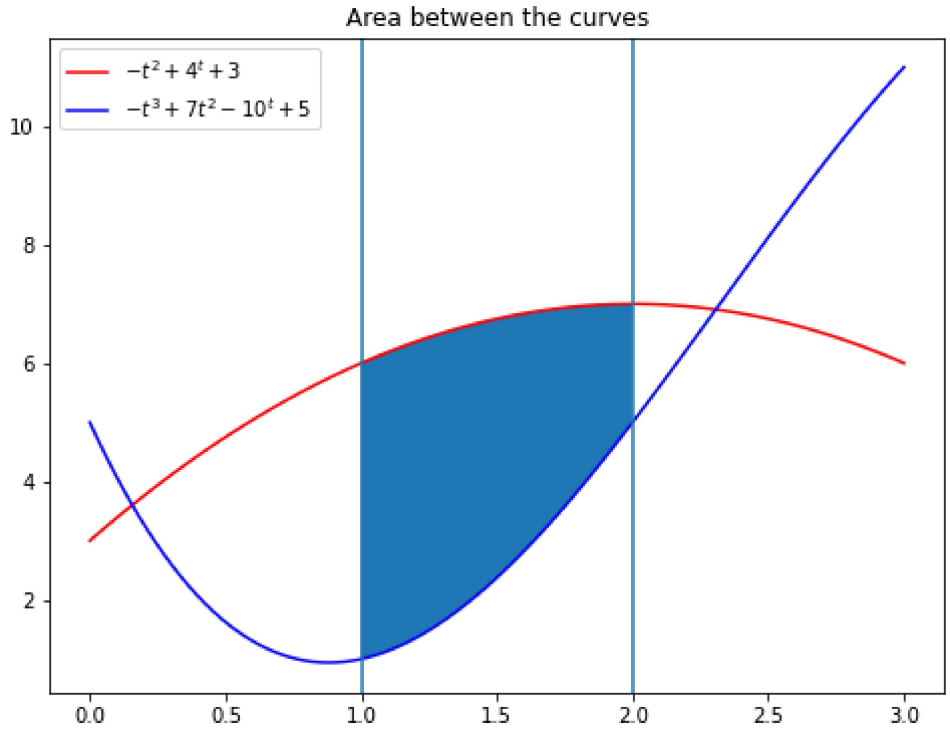
Find the area below $f(x) = -x^2 + 4x + 3$ and above $g(x) = -x^3 + 7x^2 - 10x + 5$ over the interval $1 < x < 2$

In [116]:
```
from sympy import *

def f(t):
    return -t**2 + 4*t + 3
def g(t):
    return -t**3 + 7*t**2 - 10*t + 5

t = Symbol("t")
def find_area(f,g,t,a,b):
    a = Integral(f-g,(t,a,b)).doit()
    return a

find_area(f(t),g(t),t,1,2)
```

Out[116]: $\frac{49}{12}$

In [117]:

```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

t = np.linspace(0, 3, 100)
plt.figure(figsize = [8, 6])
plt.plot(t, -t**2 + 4*t + 3, 'r-')
plt.plot(t, -t**3+7*t**2 - 10*t + 5, 'b-')
plt.axvline(x = 1)
plt.axvline(x = 2)
plt.title("Area between the curves")
plt.legend(['$-t^2 + 4^t + 3$','$-t^3+7t^2 - 10^t + 5$'])

p = np.linspace(1, 2, 50)
plt.fill_between(p, -p**2 + 4*p + 3, -p**3+7*p**2 - 10*p + 5)
plt.show()
```



Question 1)

Find the areas of the region enclosed by the parabola

$y = 2 - x^2$ and the line $y = -x$

In [118]:
```
#Finding point of Intersections
x = Symbol("x")
eq1 = 2- x**2
eq2 = -x
solve(eq1-eq2,x)


```

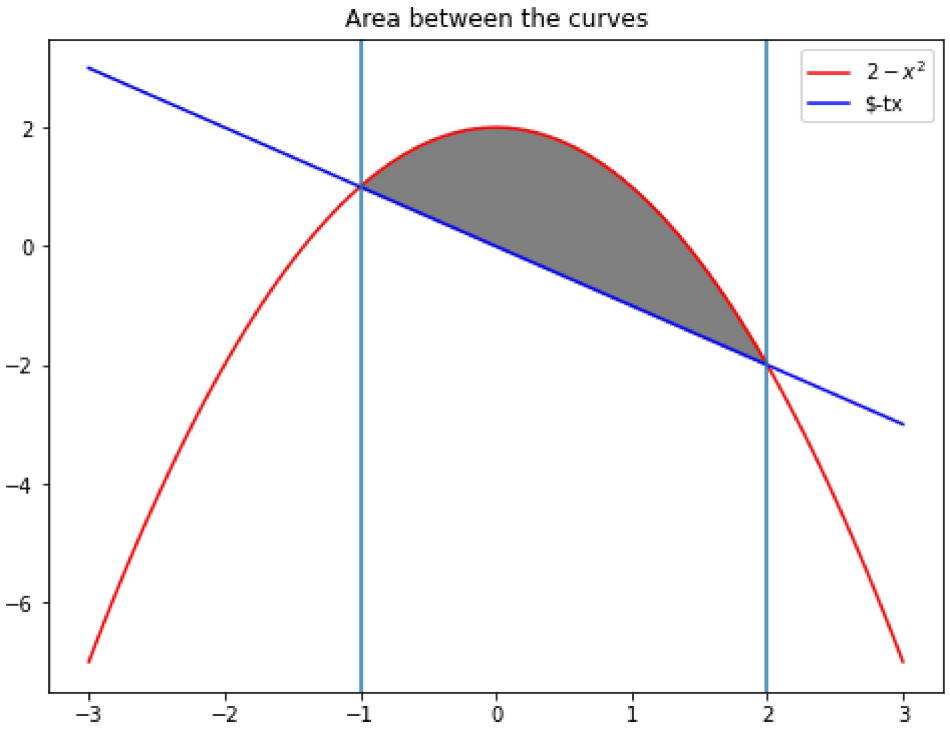
Out[118]: [-1, 2]

In [119]:
```
from sympy import *

def f(t):
    return 2-t**2
def g(t):
    return -t

t = Symbol("t")
def find_area(f,g,t,a,b):
    a = Integral(f-g,(t,a,b)).doit()
    return a

find_area(f(t),g(t),t,-1,2)
```

Out[119]: $\frac{9}{2}$

In [120]:

```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

t = np.linspace(-3, 3, 100)
plt.figure(figsize = [8, 6])
plt.plot(t, f(t), 'r-')
plt.plot(t, g(t), 'b-')
plt.axvline(x = -1)
plt.axvline(x = 2)
plt.title("Area between the curves")
plt.legend(['$2 - x^2$','$-tx'])

p = np.linspace(-1, 2, 50)
plt.fill_between(p, f(p), g(p), color = 'grey')
plt.show()
None
```



Question 2)

Find the area enclosed between $x = 3y - y^2$ and $x = 0.5y^2$ between $y = 0$ and $y = 2$

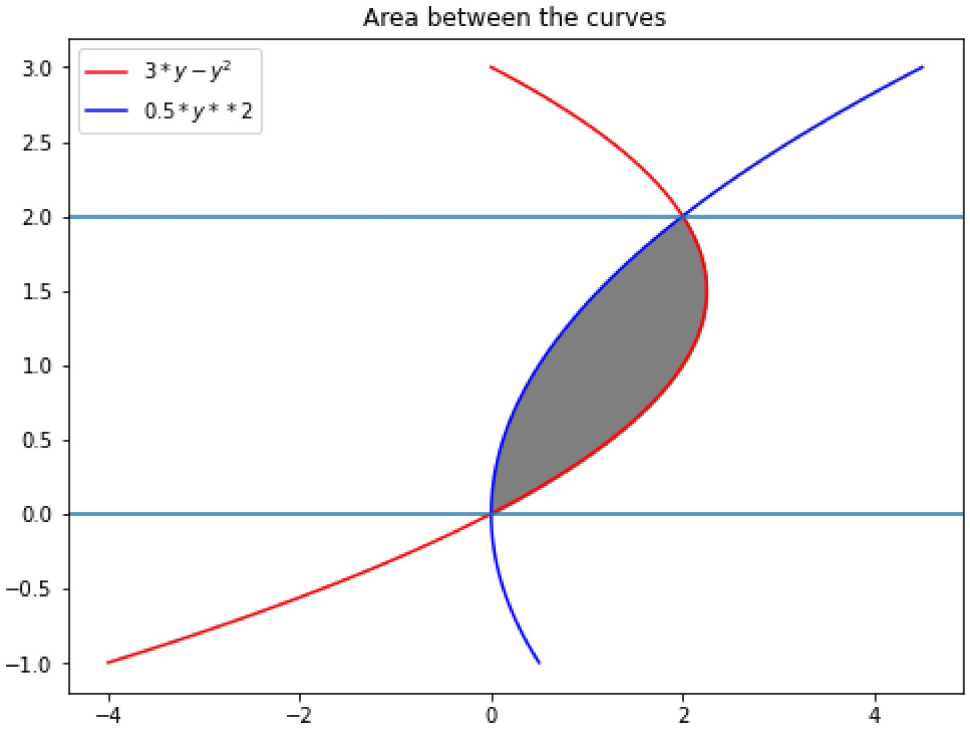
In [121]:
```
#Finding point of Intersections
x = Symbol("x")
y = Symbol("y")
eq1 = 3*y - y**2
eq2 = 0.5*y**2
solve(eq1-eq2,y)


```

Out[121]: [0.0, 2.00000000000000]

In [122]:
```
#Finding the area
x = Symbol("x")
y = Symbol("y")

eq1 = 3*y - y**2
eq2 = 0.5*y**2
integrate(eq1-eq2, (y,0,2))
```

Out[122]: 2.0

In [123]:
```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

y = np.linspace(-1, 3, 100)

plt.figure(figsize = [8, 6])
plt.plot(3*y - y**2 ,y, 'r-')
plt.plot(0.5*y**2 ,y, 'b-')
plt.axhline(y = 0)
plt.axhline(y = 2)
plt.title("Area between the curves")
plt.legend(['$3*y - y^2$','$0.5*y**2$'])

p = np.linspace(0, 2)
plt.fill_betweenx(p,3*p - p**2 ,0.5*p**2,where= (p>0) &(p<2),interpolate
plt.show()
None
```



Question 3)

Find the area enclosed between $f(x) = x^2 - 3x + 2$ and $g(x) = 2x - 2$.

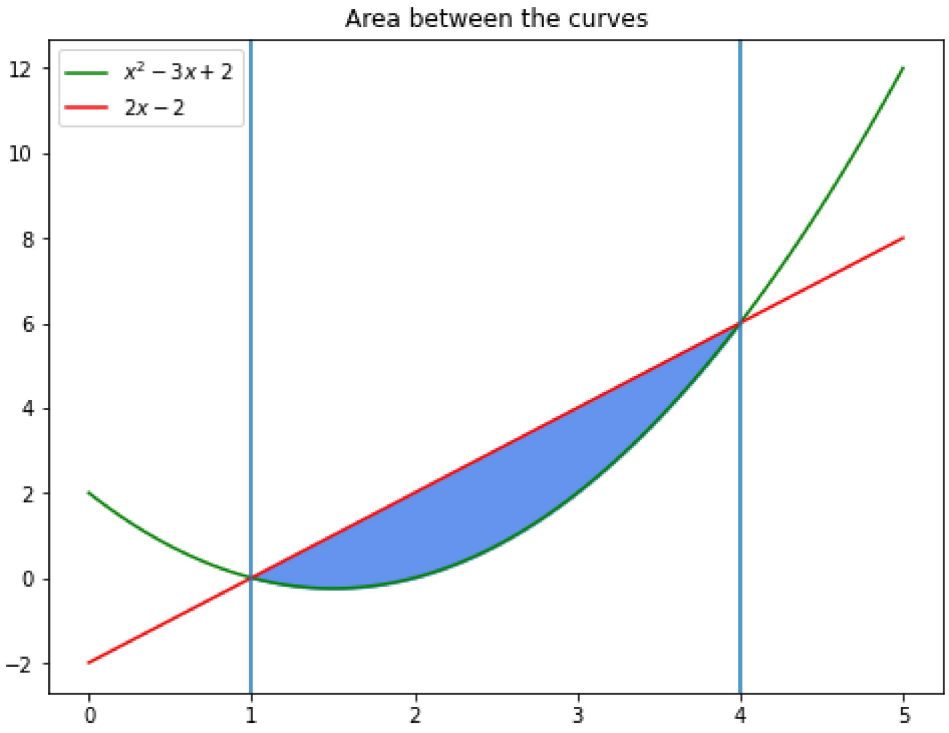
In [124]:
```
f = lambda x: x**2 - 3*x + 2
g = lambda x: 2*x - 2
solve(f(x) - g(x), x)
```

Out[124]: [1, 4]

In [125]:
```
a = integrate(g(x) - f(x), (x, 1, 4))
a
```

Out[125]: $\frac{9}{2}$

In [126]:
```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

t = np.linspace(0, 5, 100)
plt.figure(figsize = [8, 6])
plt.plot(t, f(t), 'g-')
plt.plot(t, g(t), 'r-')
plt.axvline(x = 1)
plt.axvline(x = 4)
plt.title("Area between the curves")
plt.legend(['$x^2 - 3x + 2$','$2x - 2$'])

p = np.linspace(1, 4, 50)
plt.fill_between(p, f(p), g(p), color = 'cornflowerblue')
plt.show()
None
```



Question 4)

Find the area enclosed between $f(x) = x^3 - 6x + 3$ and $g(x) = x^2 + 3$.

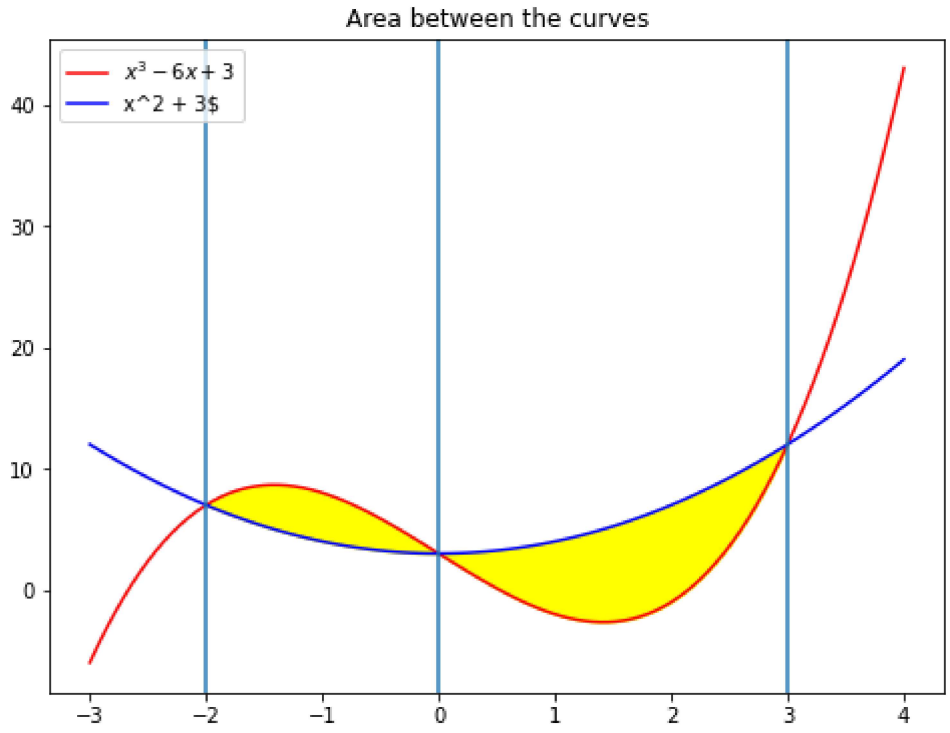
In [127]:
```
f = lambda x: x**3 - 6*x + 3
g = lambda x: x**2 + 3
solve(f(x) - g(x), x)
```

Out[127]: [-2, 0, 3]

In [128]:
```
a = integrate(g(x) - f(x), (x, -2, 0)) + integrate(g(x) - f(x), (x, 0, 3
a
```

Out[128]: $\frac{125}{12}$

In [129]:
```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np

t = np.linspace(-3, 4, 100)
plt.figure(figsize = [8, 6])
plt.plot(t, f(t), 'r-')
plt.plot(t, g(t), 'b-')
plt.axvline(x = -2)
plt.axvline(x = 0)
plt.axvline(x = 3)
plt.title("Area between the curves")
plt.legend(['$x^3 - 6x + 3$','x^2 + 3$'])

p = np.linspace(-2, 3, 50)
plt.fill_between(p, f(p), g(p), color = 'yellow')
plt.show()
None
```

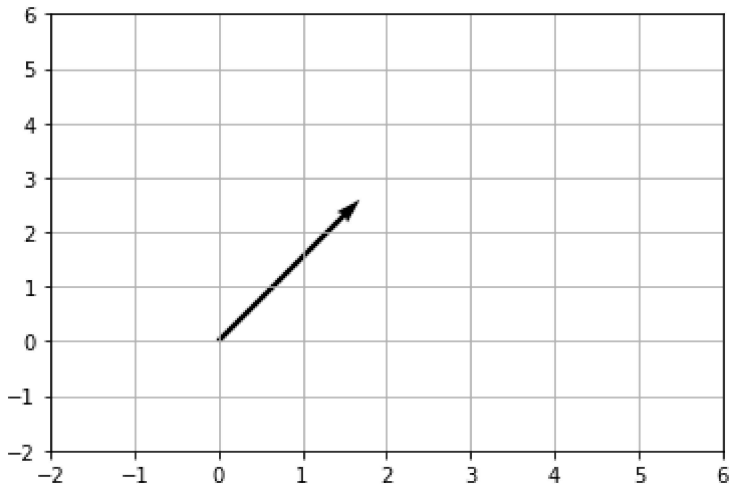


Topic 7: Plotting Vector Fields

Date: 05.11.2022

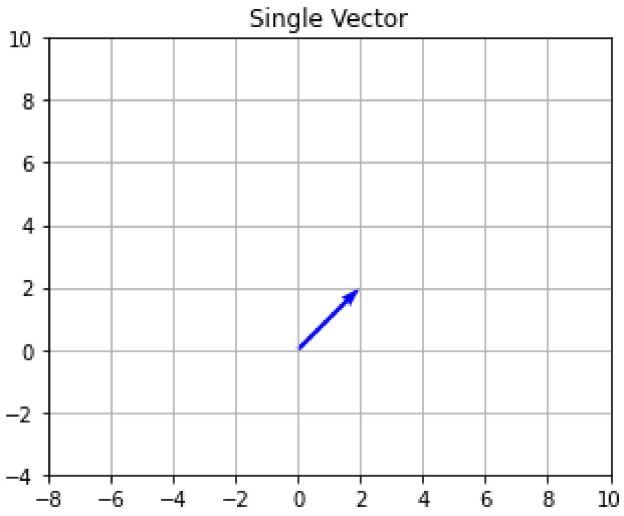
In [130]:
```
1 import numpy as np
2 from matplotlib import pyplot as plt
```

In [133]:
```
1  #Ploting a single vector
2
3  #Vector origin
4  x = 2
5  y = 2
6
7  X,Y = np.meshgrid(x,y)
8
9
10
11 plt.quiver(X,Y,units = "xy",scale =1)
12 plt.xlim(-2,6)
13 plt.ylim(-2,6)
14 plt.grid(True)
```

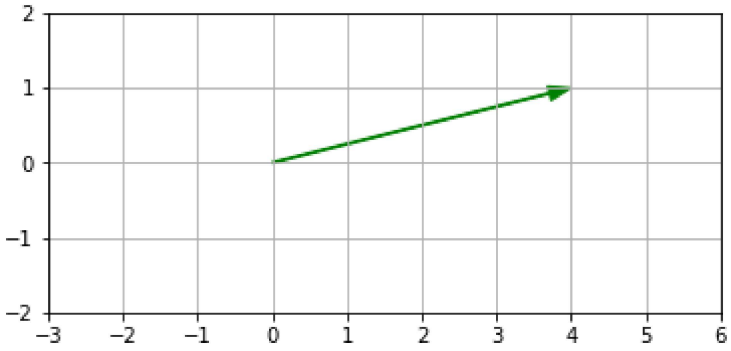


In [135]:

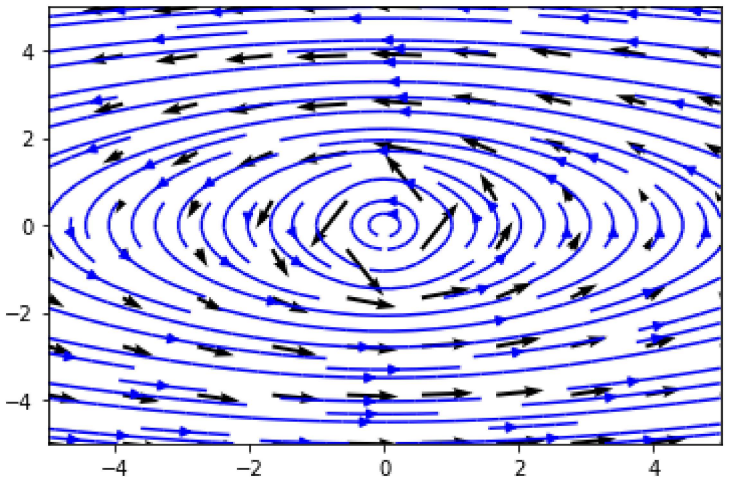
```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt

# Vector origin location
X = [0]
Y = [0]

# Directional vectors
U = [2]
V = [2]

# Creating plot
plt.quiver(X, Y, U, V, color='b', units='xy', scale=1)
plt.title('Single Vector')

# x-lim and y-lim


# Show plot with grid
plt.axis([-8, 10, -4, 10])
plt.grid()
plt.gca().set_aspect("equal")
plt.show()


```



In [137]:
```
def plot_vector2d(vector2d, origin=[0, 0], **options):
    return plt.arrow(origin[0], origin[1], vector2d[0], vector2d[1],
          head_width=0.2, head_length=0.3, length_includes_head=True,
          width=0.02,
          **options)

plot_vector2d([4,1], color='g')

plt.axis([-3, 6, -2, 2])
plt.grid()
plt.gca().set_aspect("equal")

plt.show()
```



In [138]:
```
X,Y = np.meshgrid(np.linspace(-5,5,10),np.linspace(-5,5,10))
U = -Y/np.sqrt(X**2+Y**2)
V = X/(X**2+Y**2)
plt.streamplot(X,Y,U,V,color="b")
plt.quiver(X,Y,U,V)
plt.show()
```



Topic 8: Multiple integral

In [139]:
```
import numpy as np
```

In [140]:
```
import matplotlib.pyplot as plt
```

In [141]:
```
from sympy.abc import *
```

In [142]:
```
from sympy import *
```

Evaluate the line integral yds over the curve C along the parabola y = 2(x)^1/2 from point A (3,2 root 3) to point B $(24, 46^{1/2})$

In [143]:
```

x = t
y = 2*sqrt(t)
func = y
ds = sqrt((diff(x,t)**2) + (diff(y,t)**2))

integrate(func*ds,(t,3,24))
```

Out[143]: 156

Find the work done by the vector field F = r^2 i in moving a particle along the helix given by x = cost, y = sint , z = t, from the point (1,0,0) to (1,0,4pi)

r2 = x2 + y2 + z2

In [144]:
```
x = cos(t)
y = sin(t)
z = t

Func = (x**2 + y**2 + z**2  , 0   , 0)

integrate(Func[0]*diff(x,t) + Func[1]*diff(y,t) + Func[2]*diff(z,t),[t,0
```

Out[144]: $16\pi^2$

Conclusion of the Lab Record

We used python to showcase multiple theories related to Multivariable Calculus. Topics such as Vectors, Vector Operations, Area Under the curve, Plotting of Vectors, Intergration, Differentiation etc were shown and sums related to these topics were solved. This helped in furthering our knowledge of Multivariable Calculus by solving difficult problems and visualizing the various types of graphs of various functions.