PYTHON TUTORIALS FOR BEGINNERS

1 How to install python

Step 1: Download the Anaconda installer. https://www.anaconda.com/download

Step 2: Go to your Downloads folder and double-click the installer to launch. To prevent permission errors, do not launch the installer from the set Favourite folder.

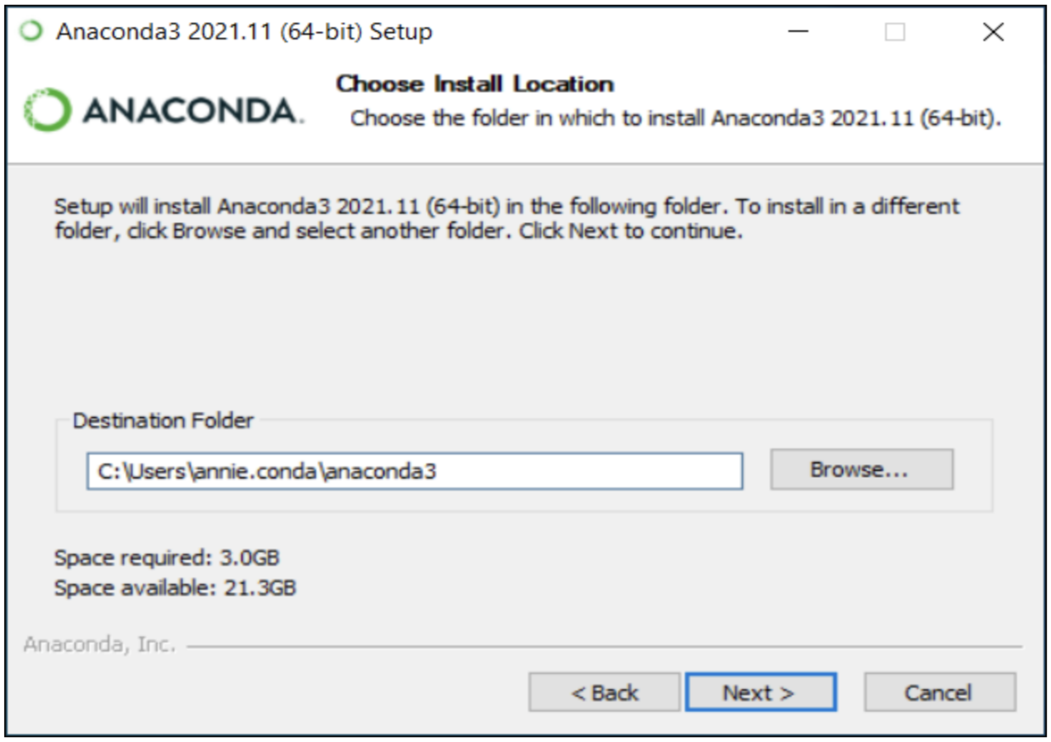
Step 3: Click Next.

Step 4: Read the licensing terms and click I agree.

Step 5: It is recommended that you install for Just Me, which will install Anaconda Distribution to just the current user account. Only select an install for All Users if you need to install for all users' accounts on the computer (which requires Windows Administrator privileges).

Step 6: Click Next.

Step 7: Select a destination folder to install Anaconda and click Next. Install Anaconda to a directory path that does not contain spaces or Unicode characters. For more information on destination folders, see the FAQ.


Anaconda3 2021.11 (64-bit) Setup
ANACONDA.
Choose Install Location
Choose the folder in which to install Anaconda3 2021.11 (64-bit).

Setup will install Anaconda3 2021.11 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Users\annie.conda\anaconda3

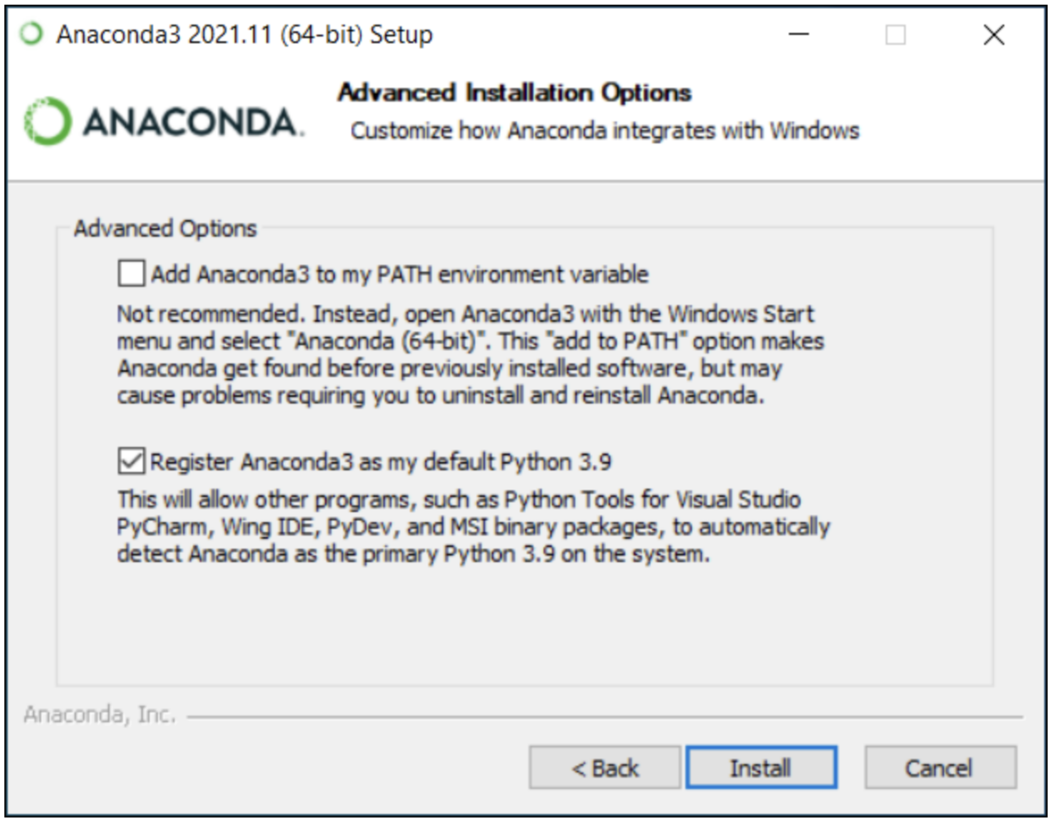
Space required: 3.0GB
Space available: 21.3GB

Browse...
Anaconda, Inc.
< Back
Next >
Cancel

Step 8: Choose whether to add Anaconda to your PATH environment variable or register Anaconda as your default Python. We don't recommend adding Anaconda to your PATH environment variable, since this can interfere with other software. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

Note:-

As of (Anaconda Distribution 2022.05), the option to add Anaconda to the PATH environment variable during an All user installation has been disabled. This was done to address a security exploit. You can still add Anaconda to the PATH environment variable during a Just Me installation.


Anaconda3 2021.11 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

- [ ] Add Anaconda3 to my PATH environment variable

  Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

- [✓] Register Anaconda3 as my default Python 3.9

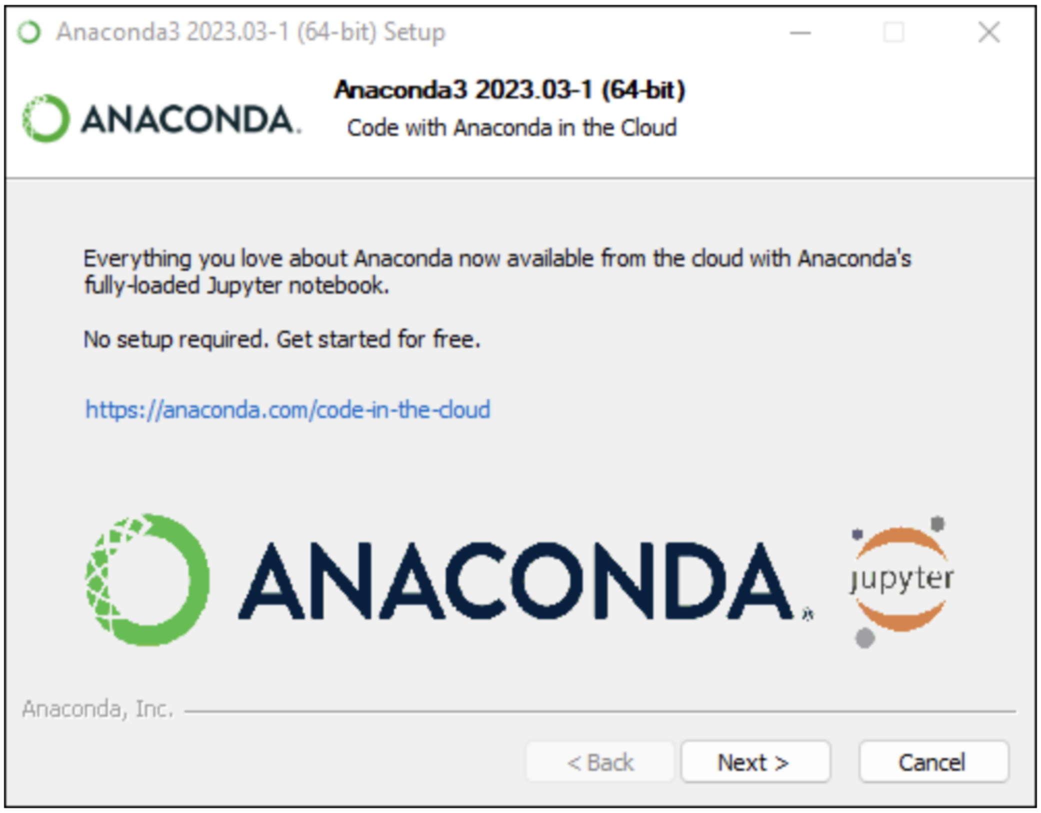
  This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.9 on the system.

Anaconda, Inc.

Step 9: Click Install. If you want to watch the packages Anaconda is installing, click Show Details.

Step 10: Click Next.

Step 11: Optional: To learn more about Anaconda's cloud notebook service, go to `https://www.anaconda.com/code-in-the-cloud.`


Anaconda3 2023.03-1 (64-bit) Setup

Anaconda3 2023.03-1 (64-bit)

Code with Anaconda in the Cloud

Everything you love about Anaconda now available from the cloud with Anaconda's fully-loaded Jupyter notebook.

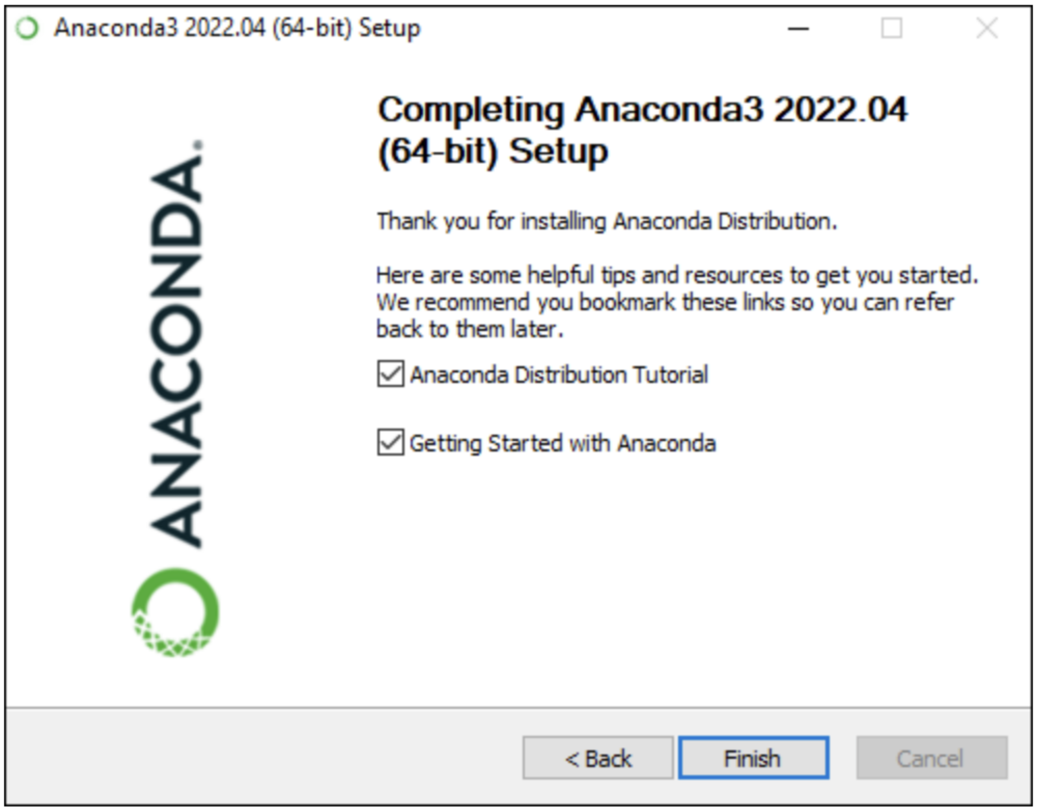
No setup required. Get started for free.

https://anaconda.com/code-in-the-cloud

Anaconda, Inc.

Or click Continue to proceed.

Step 12: After the successful installation, you will see the “Thanks for installing anaconda” dialog box:



Anaconda3 2022.04 (64-bit) Setup

Completing Anaconda3 2022.04 (64-bit) Setup

Thank you for installing Anaconda Distribution.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

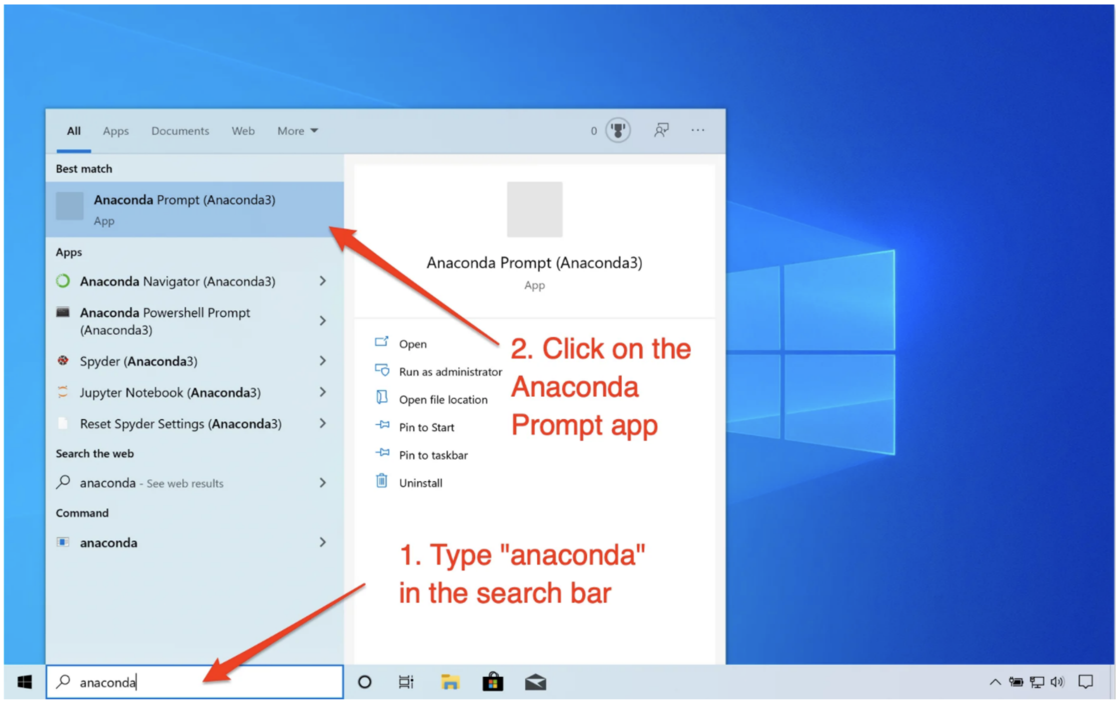
☑ Anaconda Distribution Tutorial

☑ Getting Started with Anaconda

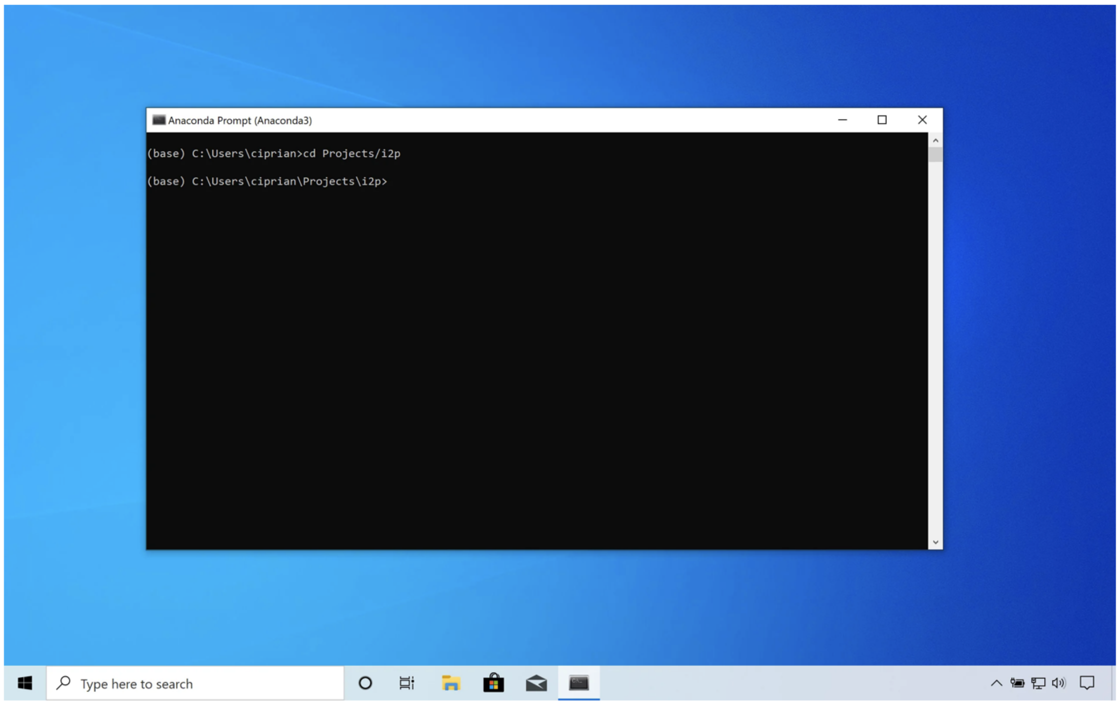
Step 13: If you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes "Anaconda Distribution Tutorial" and "Learn more about Anaconda". Click the Finish button.

2 How to open jupyter notebook in your system.

Step 1: Find and open the Anaconda Prompt app using the search bar. Alternatively, you can use the Anaconda Powershell Prompt. The Anaconda Powershell Prompt has more bells and whistles and it is generally nicer to work with.

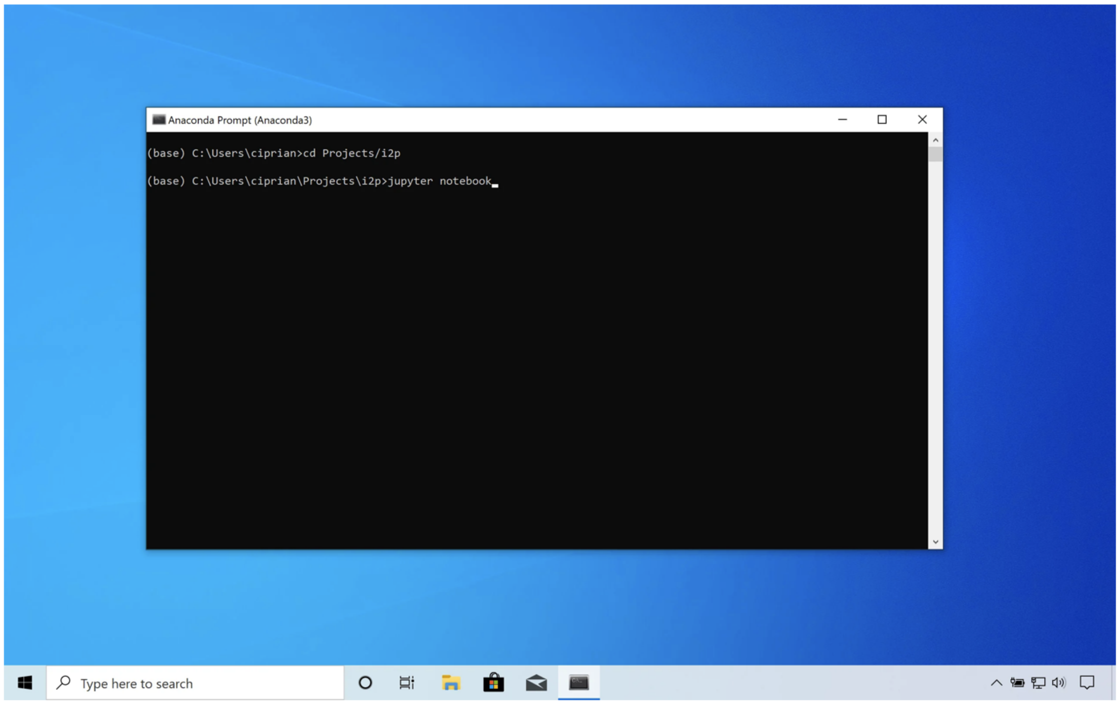

All
Apps
Documents
Web
More
Best match
Anaconda Prompt (Anaconda3)
App
Apps
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Anaconda3)
Spyder (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)
Search the web
anaconda - See web results
Command
anaconda
Anaconda Prompt (Anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
2. Click on the Anaconda Prompt app
1. Type "anaconda" in the search bar
anaconda

Step 2: Once the Anaconda Prompt (or Anaconda Powershell Prompt) app opens, navigate to the desired folder, using the cd command.



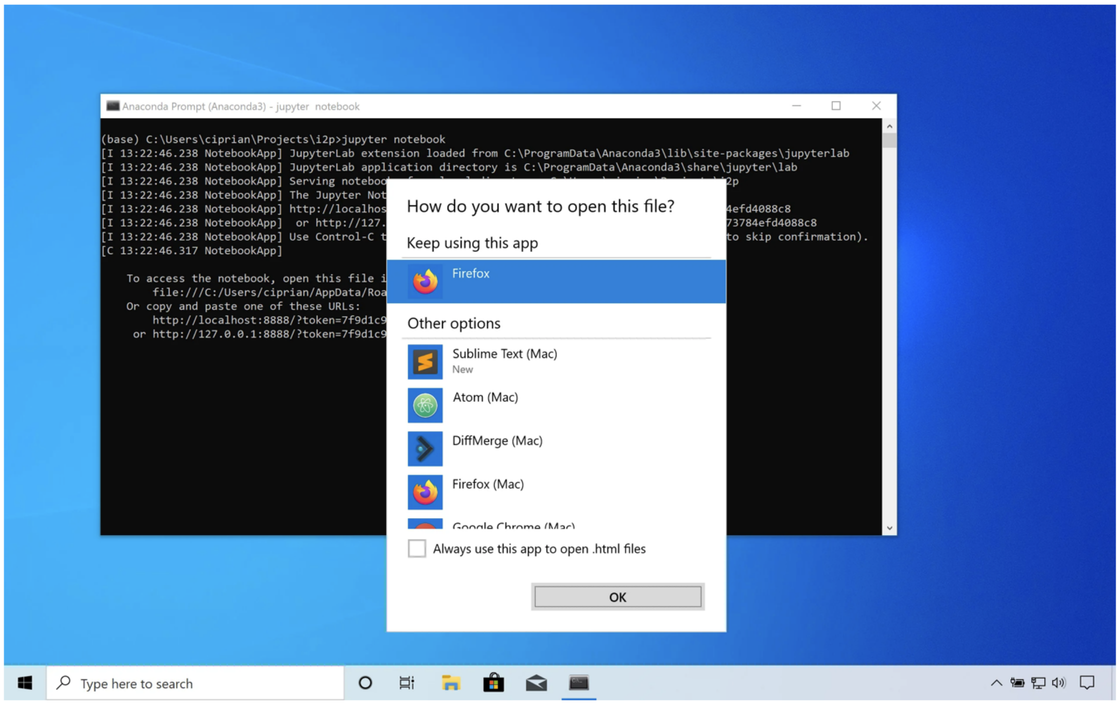
```
(base) C:\Users\ciprian>cd Projects/i2p

(base) C:\Users\ciprian\Projects\i2p>
```

Step 3: Once in the desired folder, type jupyter notebook followed by the Enter key.



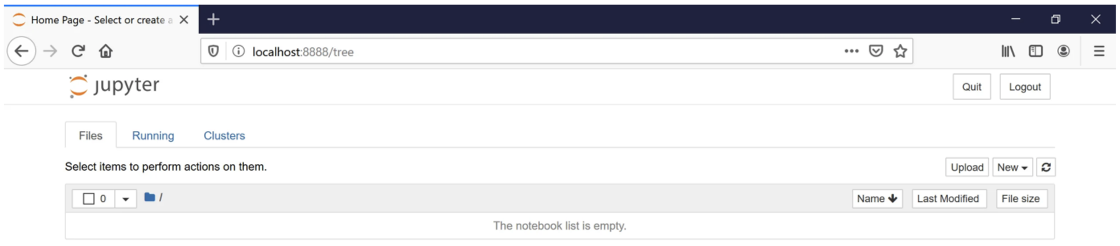
```
(base) C:\Users\ciprian>cd Projects/i2p

(base) C:\Users\ciprian\Projects\i2p>jupyter notebook
```

Step 4: The Jupyter server will start. You should see some server logs printed. You may be prompted to select an application to open Jupyter in. Firefox or Chrome are preferred.



```
(base) C:\Users\ciprian\Projects\i2p>jupyter notebook
[I 13:22:46.238 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 13:22:46.238 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 13:22:46.238 NotebookApp] Serving noteboo                                         2p
[I 13:22:46.238 NotebookApp] The Jupyter Not
[I 13:22:46.238 NotebookApp] http://localhos                                         4efd4088c8
[I 13:22:46.238 NotebookApp]  or http://127.                                         73784efd4088c8
[I 13:22:46.238 NotebookApp] Use Control-C t                                         to skip confirmation).
[C 13:22:46.317 NotebookApp]

    To access the notebook, open this file i
        file:///C:/Users/ciprian/AppData/Roa
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=7f9d1c9
     or http://127.0.0.1:8888/?token=7f9d1c9
```

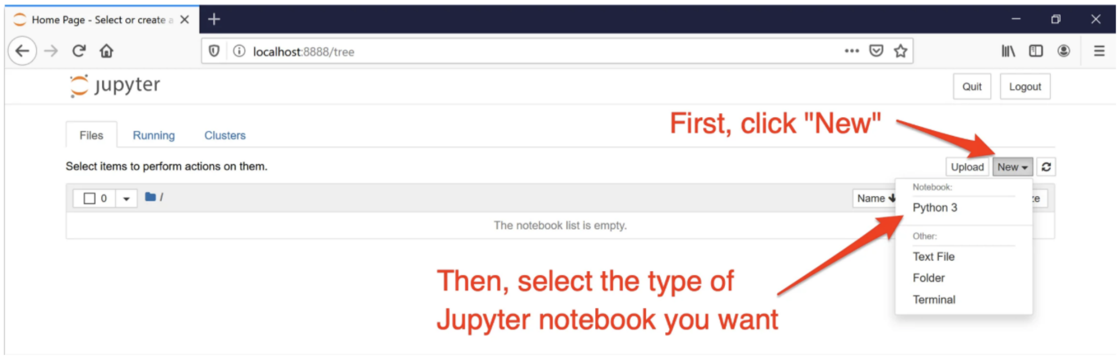
Step 5: Shortly after, a browser window should open, showing the files and folders located in the folder where you started the Jupyter server (in my case, this folder is {`C:\Users\ciprian\Projects\i2p`})



Select items to perform actions on them.

The notebook list is empty.

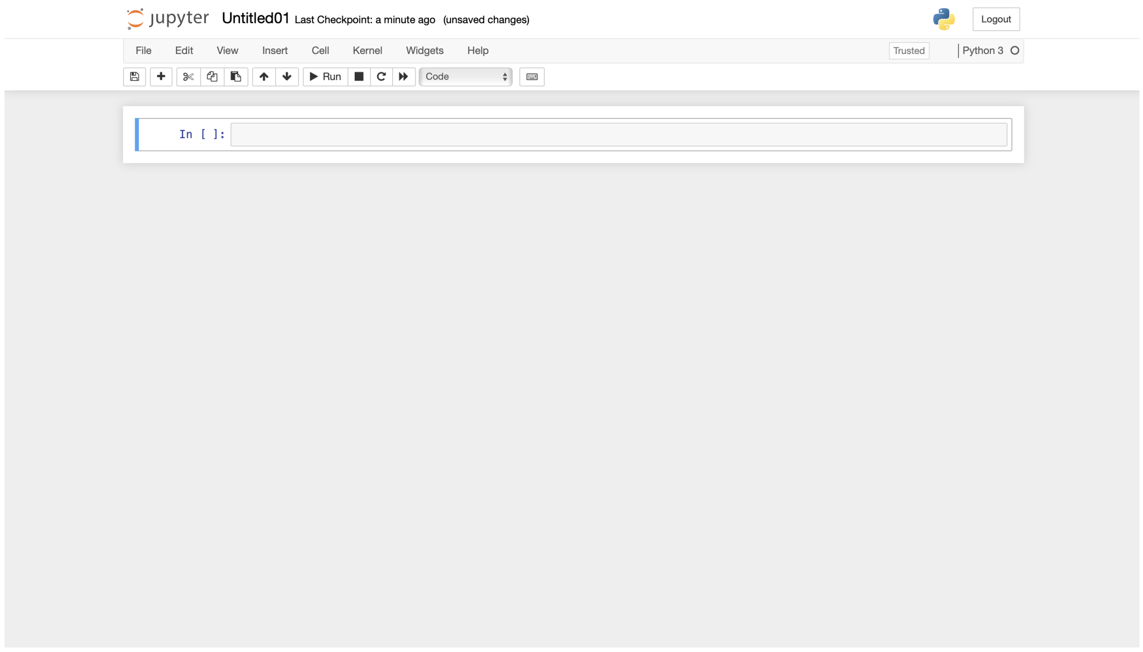
Step 6: Create a new Jupyter notebook.



Select items to perform actions on them.

The notebook list is empty.

Step 7: Your new Jupyter notebook is now ready and you can start learning more about data types, variables, numbers, and more!



Basic Commands for Python

2.1 Variables

```
[3]: # variables-(Stores data)
x=5
print(x) # here the print statment use for giving the output.
```

```
5
```

2.1.1 Types of operator

Arithmatic operators

Assignment operators

Comparison operators

Logical operators

```
[ ]: #Arithmatic operators
+,-,*,/
//(floor divison)
%(modulus)
**(exponential )
```

```
[15]: x=50
y=3
print(x+y)
print(x-y)
print(x*y)
print(x/y)
print(x//y)
print(x**y)
print(x%y)
```

```
53
47
150
16.666666666666668
16
125000
2
```

```
[ ]: #Assignment operators
=,+=,-=,*=,/=,//=,**=,%=
```

[16]:
```
v=5
v+=5 #means v=v+5
print(v)
v-=5 #means v=v-5
print(v)
v*=5 #means v=v*5
print(v)
v/=5 #means v=v/5
print(v)
v//=5 #means v=v//5
print(v)
v**=5 #means v=v**5
print(v)
v%=5 #means v=v%5
print(v)
```

```
10
5
25
5.0
1.0
1.0
1.0
```

[ ]:
```
#Comparison operators
<,>,==(equals to),>=,<=,!=(not equals to)
```

[17]:
```
# when we use comparison operator it will give us the output true or false.
a=20
b=30
print(a<b)
print(a>b)
print(a!=b)
print(a<=b)
print(a>=b)
print(a==b) #double equal used for check they are same or not.
```

```
True
False
True
True
False
False
```

[ ]:
```
#Logical operators
and, or, not
```

[18]:
```
a=45
b=79
print(a<b and b>a) # If both the conditions are true it will give true else␣
 ↪false.
print(a<b or b<a) #If one of the condition is true then it will give the output␣
 ↪true.
print(not a<b) # gives the negation of the statement.
print(not(a<b and b>a))
```

```
True
True
False
False
```

[ ]:
```
# Conditional statements.
if, elif,if-else
```

[19]:
```
#if

a=30
b=40
if a<b:
    print('a is less than b.') #It will give output if the statment is true.␣
 ↪And no output for the false one.
```

```
a is less than b.
```

[20]:
```
#if-else
a=30
b=40
if a<b:
    print('a is less than b.')
else:
    print('b is less than a.') #It will give output whether the statment is␣
 ↪true or false.
```

```
a is less than b.
```

[21]:
```
#elif
a=30
b=40
c=50
if a<b and b<c:
    print('a is smallest.')
elif b<a and a<c:
    print('b is smallest.')
```

```
```

```
a is smallest.
```

3 Lists

[22]:
```
l=[1,4,9,16,25] # This is a list of squares of 1st 5 natural numbers.
l2=[2.5,3.8,'hello','k']

#Indexing
print(l[4])

#Slicing
print(l[a:b]) #Always including the starting(a) but excludes the ending␣
 ↪index(b).

#reversing.
print(l[ : :-1])
print(l)
l.reverse()
print(l)

#Length of the list
print(len(l)) # Number of elements in the list


l=[1,4,9,25] # This is a list of squares of 1st 5 numbers.
#(l2=[2.5,3.8,'hello','k'])

#Adding elements
l.append(36)
print(l)

l.insert(3,16)
print(l)

#removing elements from list
l.remove(16) #inside paranthesses write the element which you want to remove
#or
l.pop(2) #inside paranthesses write the index
```

```
25
[]
[25, 16, 9, 4, 1]
```

```
[1, 4, 9, 16, 25]
[25, 16, 9, 4, 1]
5
[1, 4, 9, 25, 36]
[1, 4, 9, 16, 25, 36]
```

[22]:
```
9
```

4 Range function

[ ]:
```python
range(0,6) #always includes the staring but excludes the ending.

range(n) #This means collection of sequence of numbers from 0 to n-1.

range(a,b,s) # (start,end(excluded),steps).
```

5 For Loop

[23]:
```python
for i in range(6): #for (i=terating variable) in (range=sequence):
                            #print(iterating variable)
    print(i)
```

```
0
1
2
3
4
5
```

[24]:
```python
# for loop with break
for i in range(6):

    print(i)
    if i==3:
        break # We use this when we want to come out of the loop.
```

```
0
1
2
3
```

[25]:
```python
#Nested for loop by example.

for i in range(1,4):
    for j in range(1,11):
        print(i,'*',j,'=',i*j)
    print()
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

[26]:
```
# Factorial of any number.
n=int(input('I want to find factorial of '))
fac=1  #This is an flag variable.
for i in range(1,n+1):
    fac=fac*i
print(fac)
```

```
I want to find factorial of 5
120
```

6 While loop

[27]:
```
i=0
while (i<4):
    print(i)
    i+=1  #This is increment.

print(i)
print('end')
```

```
0
1
2
3
4
end
```

[28]:
```
#while loop with break

i=1
while i<5:
    print(i)
    if i==2:
        break

    i=i+1
```

```
1
2
```

7 Functions

It is the block of statements or block of codes which perform some specific task when it is called.

[ ]:
```
# Function_syntax
def function_name(parameters):
    function_body
    print expression/ return expression
```

[ ]:
```
function_name(a,b)
```

[78]:
```
def adds(a,b):
    c=a+b
    print(c)
```

[79]:
```
v=adds(1,2)
```

```
3
```

[80]:
```python
print(v)
```

```
None
```

[81]:
```python
def adds(a,b):
    c=a+b
    return c
```

[82]:
```python
w=adds(1,2)
```

[83]:
```python
print(w)
```

```
3
```

[84]:
```python
def amean(x,y):
    z=(x+y)/2
    return z
```

[85]:
```python
v=amean(2,3)
```

[86]:
```python
def gmean(g,h):
    i=(g*h)**(1/2)
    return i
```

[87]:
```python
w=gmean(2,3)
```

[88]:
```python
if v>w:
    print("A.M is greater")
else:
    print("A.M is not")
```

```
A.M is greater
```

7.1 TASK

Suppose in a certain exam, there are Paper-1 and Paper-2. In order to be qualify, one has to secure atleast 20 marks in each paper as well as sum of marks obtained in both the paper must be atleast 50 marks.If one fails to satisfy any one of the above conditions,he/she declares to fail. Write a function for this.

[40]:
```python
def passc(a,b):
    if a>=20 and b>=20 and (a+b)>=50:
        print("Qualified")
    else:
        print("Disqualified")
```

[41]:
```
passc(21,23)
```

```
Disqualified
```

8 NUMPY

It is a scientific package used for computing in Python. It provides a Python Library that is used for working with multidimensional arrays.

8.1 Array

An array is a grid of values and it contains information about the raw data.

[42]:
```
import numpy as np
```

[43]:
```
# 0-D Array(Scalar)
a=21
b=np.array([22])
print(b)
```

```
[22]
```

[44]:
```
# 1-D Array (it contains 0-D array as elements)
d=np.array([1,2,3])
print(d)
```

```
[1 2 3]
```

[45]:
```
# 2-D Array(it contains 1-D Array as elements)
e=np.array([4,5,6])
print(np.array([d,e]))
```

```
[[1 2 3]
 [4 5 6]]
```

[46]:
```
# Create 3 by 3 matrix
print(np.array([[1,2,3],[4,5,6],[7,8,9]]))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[47]:
```
# Task (create a 2 by 4 matrix)
b=np.array([1,2,3,4])
c=np.array([0,1,2,3])
print(np.array([b,c]))
```

```
[[1 2 3 4]
 [0 1 2 3]]
```

[48]:
```
# Sum of matrices
A=np.array([[1,2],[1,5]])
B=np.array([[0,3],[3,2]])
print(A)
print(B)
print(A+B)
```

```
[[1 2]
 [1 5]]
[[0 3]
 [3 2]]
[[1 5]
 [4 7]]
```

[49]:
```
# Matrix Multiplication
print(np.matmul(A,B))
```

```
[[ 6  7]
 [15 13]]
```

[50]:
```
# Componentwise Multilpication(when order of matrices are same.)
x=A*B
print(x)
```

```
[[ 0  6]
 [ 3 10]]
```

[51]:
```
# Scalar multiplication
b=2*A
print(b)
```

```
[[ 2  4]
 [ 2 10]]
```

[52]:
```
# Order of Matrix
print(np.shape(b))
```

```
(2, 2)
```

[53]:
```
# Special matrices
```

[54]:
```
# Matrix of all entry 1
c=np.ones([2,3],dtype=int)
print(c)
```

```
[[1 1 1]
 [1 1 1]]
```

[55]:
```
# Matrix of all entry as 0
d=np.zeros([3,4],dtype=int)
print(d)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

[56]:
```
# Identity Matrix
i=np.identity((3),dtype=int)
print(i)
```

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

[57]:
```
# Diagonal Matrix
e=np.diag([1,2,3,4])
print(e)
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

[58]:
```
# Determinant of a matrix
print(np.linalg.det(e))
```

```
23.999999999999993
```

[59]:
```
# Transpose of a Matrix
print(np.transpose(d))
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]
```

[60]:
```
# Maximum entry in a matrix
print(np.max(e))
```

```
4
```

[61]:
```
# minimum entry in a matrix
print(np.min(e))
```

```
0
```

[62]:
```python
s=np.array([[1,2,3],[0,5,6],[3,0,2]])
print(s)
```

```
[[1 2 3]
 [0 5 6]
 [3 0 2]]
```

[63]:
```python
# Extraction of any entry from a matrix
s[1,2]
```

[63]:
```
6
```

[64]:
```python
# To get any column of a matrix
s[:,1]
```

[64]:
```
array([2, 5, 0])
```

[65]:
```python
# To get any row of a matrix
s[1,:]
```

[65]:
```
array([0, 5, 6])
```

[66]:
```python
import numpy as np
```

8.1.1 Creates an array with evenly spaced values between the specified start, end, and increment values.

[67]:
```python
x1=np.arange(1,10,1)
print(x1)
```

```
[1 2 3 4 5 6 7 8 9]
```

8.1.2 Creates an array with evenly spaced values between specified start and end values, using a specified number of elements

[68]:
```python
x2=np.linspace(1,10,12)
print(x2)
```

```
[ 1.          1.81818182  2.63636364  3.45454545  4.27272727  5.09090909
  5.90909091  6.72727273  7.54545455  8.36363636  9.18181818 10.        ]
```

8.2 Random Integers

[69]:
```python
np.random.rand(2,4)
```

```
[69]: array([[0.26995818, 0.23300628, 0.97685512, 0.81074431],
              [0.67427581, 0.30449314, 0.34963785, 0.10362583]])
```

```
[70]: np.random.randint(10,size=(2,4))
```

```
[70]: array([[5, 0, 7, 2],
              [2, 0, 9, 0]])
```

```
[6]: np.random.randint(7,10,size=(2,4))
```

```
[6]: array([[8, 9, 7, 7],
             [8, 7, 8, 7]])
```

8.3 Meshgrid

```
[72]: x3=np.array([1,2,3])
      x4=np.array([-1,0,1])
      X,Y=np.meshgrid(x3,x4)
      print(x3,x4,X,Y)
```

```
[1 2 3] [-1  0  1] [[1 2 3]
 [1 2 3]
 [1 2 3]] [[-1 -1 -1]
 [ 0  0  0]
 [ 1  1  1]]
```

```
[74]: Z=X**3+Y**2+2*X*Y+8
      print(Z)
```

```
[[ 8 13 30]
 [ 9 16 35]
 [12 21 42]]
```

8.4 Now find the value of function $Z = X + 10Y$ over $X \in [0, 5]$ and $Y \in [0, 5]$ with increment 1

```
[9]: x=np.arange(0,6,1)
     y=np.arange(0,6,1)
     m,n=np.meshgrid(x,y)
     Z=n+10*m
     print(Z)
```

```
[[ 0 10 20 30 40 50]
 [ 1 11 21 31 41 51]
 [ 2 12 22 32 42 52]
 [ 3 13 23 33 43 53]
 [ 4 14 24 34 44 54]
 [ 5 15 25 35 45 55]]
```

8.5 Slicing sub arrays from a Matrix

```
[10]: Z[:3,:3]
```

```
[10]: array([[ 0, 10, 20],
             [ 1, 11, 21],
             [ 2, 12, 22]])
```

```
[11]: Z[1:3,2:5]
```

```
[11]: array([[21, 31, 41],
             [22, 32, 42]])
```

```
[12]: Z[::2,::2]
```

```
[12]: array([[ 0, 20, 40],
             [ 2, 22, 42],
             [ 4, 24, 44]])
```

```
[13]: Z[1:5:2,2:6:2]
```

```
[13]: array([[21, 41],
             [23, 43]])
```

8.6 Constructing matrices with 1's on subdiagonal or superdiagonal

```
[14]: A=np.eye(3,dtype=int)
      B=np.eye(3,k=1,dtype=int)
      C=np.eye(3,k=-1,dtype=int)
      A,B,C
```

```
[14]: (array([[1, 0, 0],
              [0, 1, 0],
              [0, 0, 1]]),
       array([[0, 1, 0],
              [0, 0, 1],
              [0, 0, 0]]),
       array([[0, 0, 0],
              [1, 0, 0],
              [0, 1, 0]]))
```

8.7 Trigonometric Functions

```
[15]: np.round(np.sin(np.pi*5.4),decimals=4)
```

```
[15]: -0.9511
```

```
[16]: np.log(2)
```

```
[16]: 0.6931471805599453
```

```
[17]: np.log2(2)
```

```
[17]: 1.0
```

9 Symbolic Computing

```
[18]: import sympy as sp
      sp.init_printing()
```

```
[19]: A=np.pi
      A
```

[19]: 3.14159265358979

```
[20]: A=sp.pi
      A
```

[20]: π

```
[21]: B=np.sin(np.pi)
      B
```

[21]: $1.22464679914735 \cdot 10^{-16}$

```
[22]: B=sp.sin(sp.Symbol("x")*sp.pi)
      B
```

[22]: $\sin(\pi x)$

```
[23]: B=sp.sin(sp.Symbol("x",integer=True)*sp.pi)
      B
```

[23]: 0

```
[24]: x=sp.Symbol("x")
      sp.sqrt(x**2)
```

[24]: $\sqrt{x^2}$

[25]:
```
x=sp.Symbol("x",positive=True)
sp.sqrt(x**2)
```
[25]: x

9.0.1 Expressions

[26]:
```
x=sp.Symbol("x")
```
[27]:
```
expr=1+2*x**2+3*x**3
expr
```
[27]: $3x^3 + 2x^2 + 1$

[28]:
```
expr.args
```
[28]: $\left(1,\ 2x^2,\ 3x^3\right)$

[29]:
```
expr.args[2]
```
[29]: $3x^3$

9.0.2 Simplification

[30]:
```
expr=2*(x**2-x)-x*(x+1)
expr
```
[30]: $2x^2 - x(x+1) - 2x$

[31]:
```
sp.simplify(expr)
```
[31]: $x(x-3)$

[32]:
```
expr.simplify()
```
[32]: $x(x-3)$

9.0.3 Expand

[33]:
```
expr=(x+1)*(x+2)
sp.expand(expr)
```
[33]: $x^2 + 3x + 2$

[34]:
```
expr.expand()
```
[34]: $x^2 + 3x + 2$

[35]:
```
expr=sp.sin(x+sp.Symbol("y"))
expr.expand(trig=True)
```

[35]:

$\sin(x)\cos(y) + \sin(y)\cos(x)$

9.0.4 Factor, Collect and Combine

[36]:
```
expr=x**2-1
expr.factor()
```

[36]:

$(x-1)(x+1)$

[37]:
```
x=sp.Symbol("x",positive=True)
y=sp.Symbol("y",positive=True)
sp.logcombine(sp.log(x) - sp.log(y))
```

[37]:

$\log\left(\frac{x}{y}\right)$

[38]:
```
z=sp.Symbol("z")
expr=x+y+z+x*y*z
expr.collect(x)
```

[38]:

$x(yz+1)+y+z$

[39]:
```
expr.collect(y)
```

[39]:

$x+y(xz+1)+z$

9.0.5 Apart and Together

[40]:
```
sp.apart(1/(x**2+3*x+2),x)
```

[40]:

$-\frac{1}{x+2}+\frac{1}{x+1}$

[41]:
```
sp.together(1/(y*x+y)+1/(1+x))
```

[41]:

$\frac{y+1}{y(x+1)}$

9.0.6 Substitutions

[42]:
```
expr=x*y+z**2*x
values={x:1.25,y:0.4,z:3.2}
expr.subs(values)
```

[42]:

13.3

9.0.7 Numerical Evaluation

[43]:
```
sp.N(sp.pi)
```

[43]:

3.14159265358979

[44]:
```
from sympy import I,pi,oo
```

[45]:
```
sp.N(pi)
```

[45]:

3.14159265358979

[46]:
```
sp.N(pi,50)
```

[46]:

3.1415926535897932384626433832795028841971693993751

10 Matplotlib

10.0.1 Installation of Matplotlib

[54]:
```
!pip install matplotlib
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-
packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

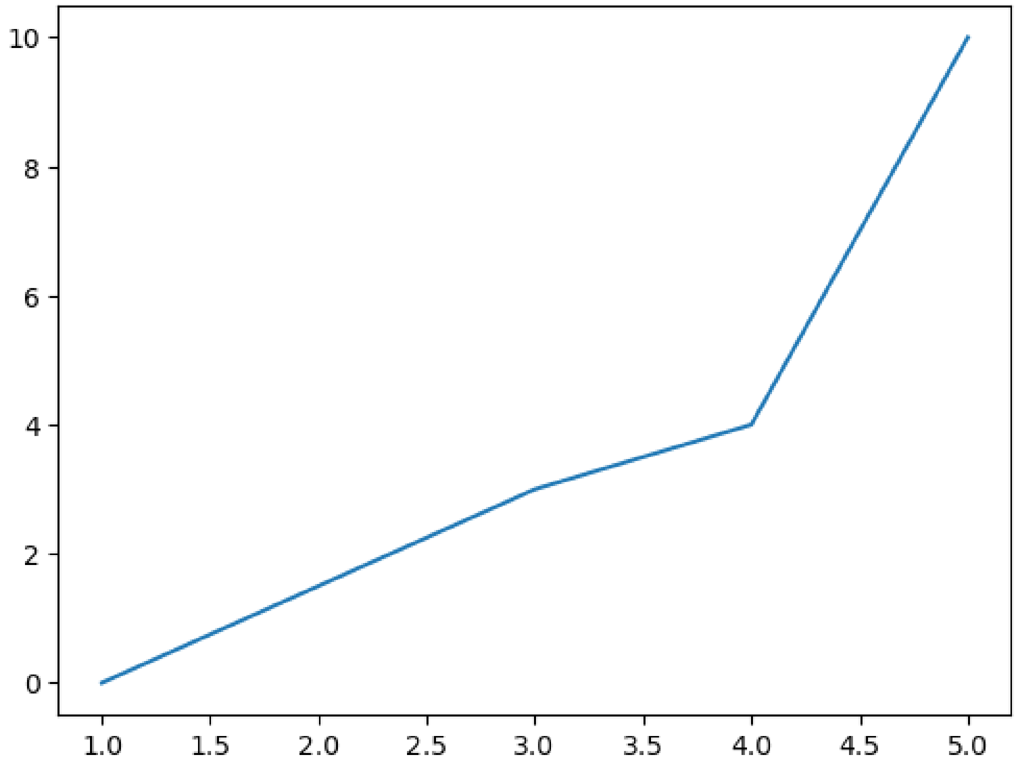
10.0.2 Import matplotlib

[2]:
```python
import matplotlib as mpl
import matplotlib.pyplot as plt
```

10.0.3 Start your first code using plt.plot() function

It has two mandatory arguments. These are numpy arrays with numerical data for the x and y values of the graphs.

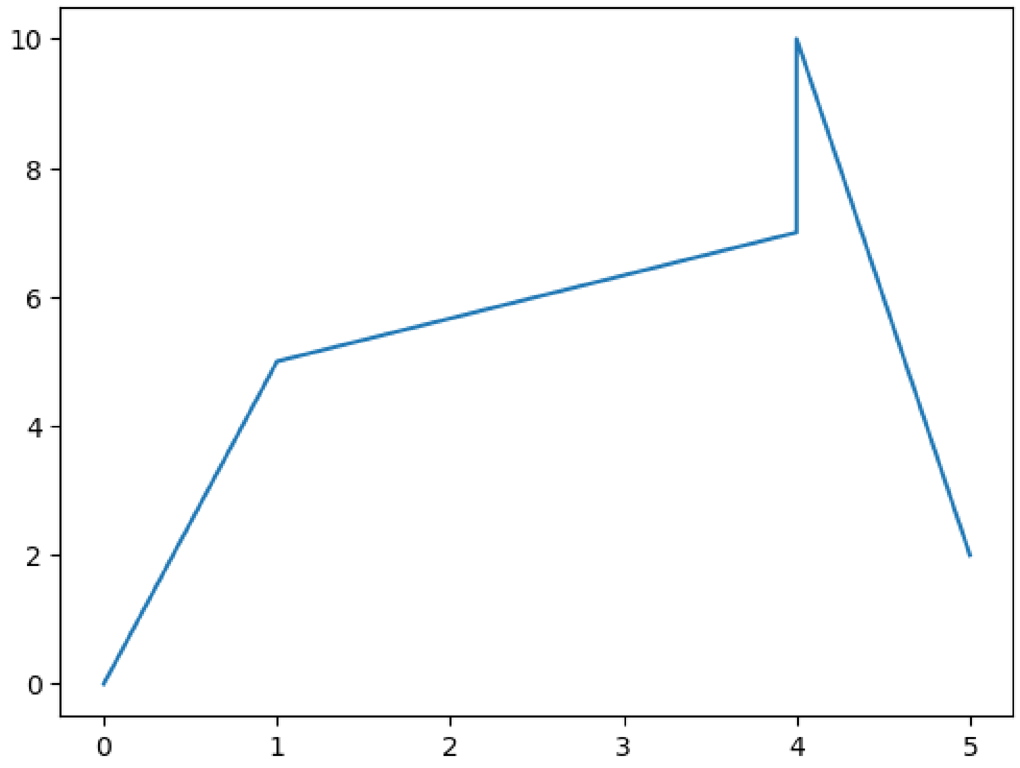
[6]:
```python
import numpy as np
x=[1,3,4,5]
y=[0,3,4,10]
plt.plot(x,y)
plt.show()
```



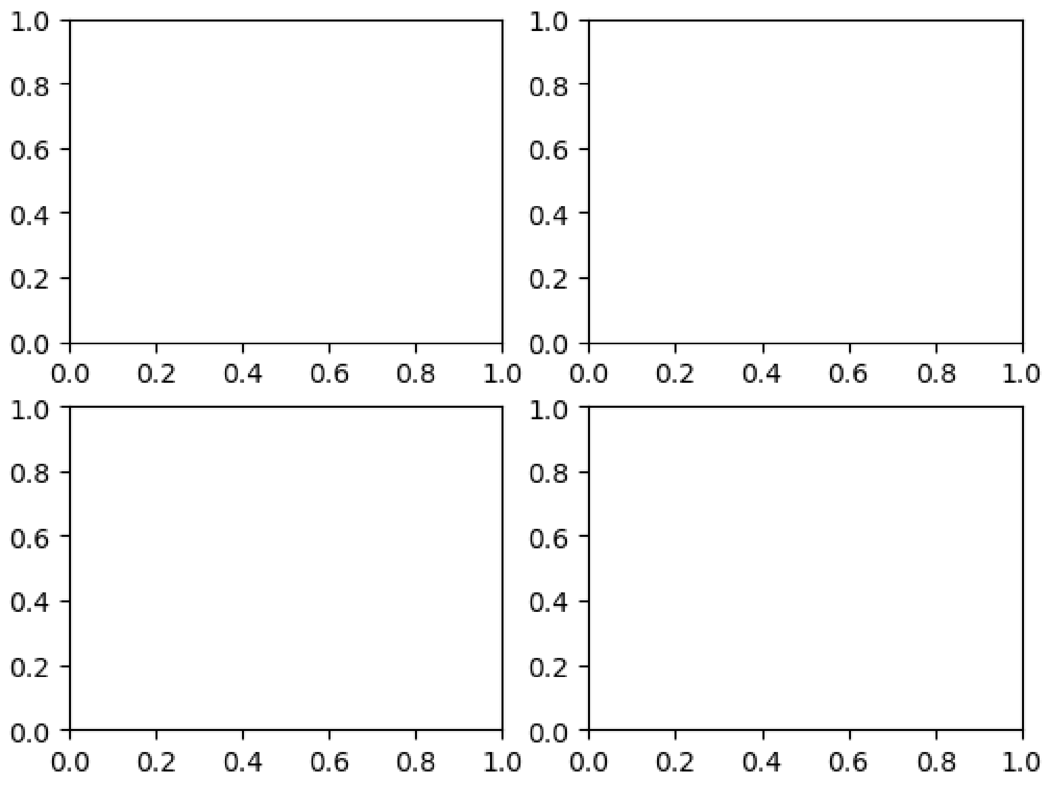
10.0.4 Figure and Axes instances

generate figure and axes instances using function plt.subplots(number__of__figure, number__of__axes__in__each__figure)

[61]:
```
x=[0,1,4,4,5]
y=[0,5,7,10,2]
fig,ax=plt.subplots()
ax.plot(x,y)
fig.savefig("GRAPH.png",dpi=100)
```

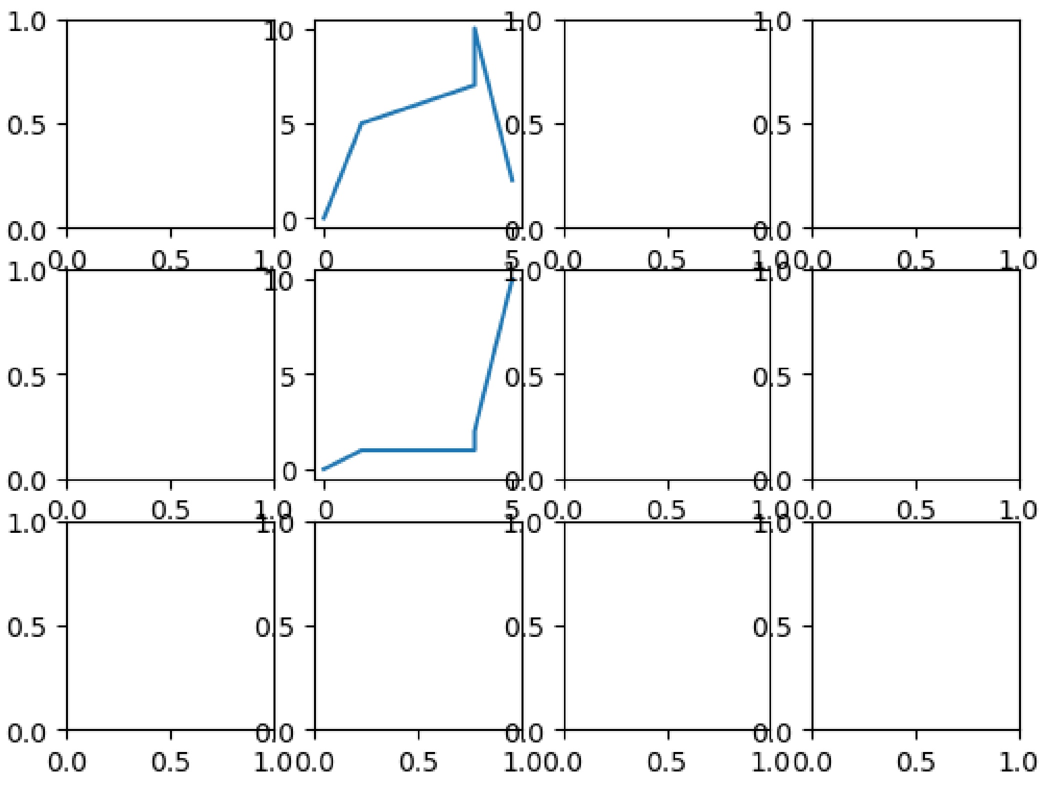


[62]:
```
x=[0,1,4,4,5]
y=[0,5,7,10,2]
fig,ax=plt.subplots(2,2)
#ax.plot(x,y)
```



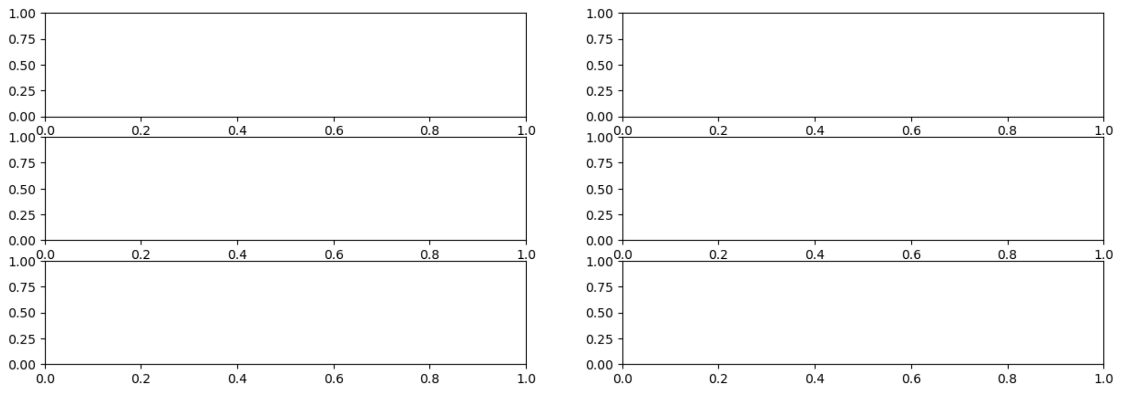
```
[65]: x=[0,1,4,4,5]
      y=[0,5,7,10,2]
      z=[0,1,1,2,10]
      fig,ax=plt.subplots(3,4)
      ax[0,1].plot(x,y)
      ax[1,1].plot(x,z)
```

```
[65]: [<matplotlib.lines.Line2D at 0x20ee367c110>]
```



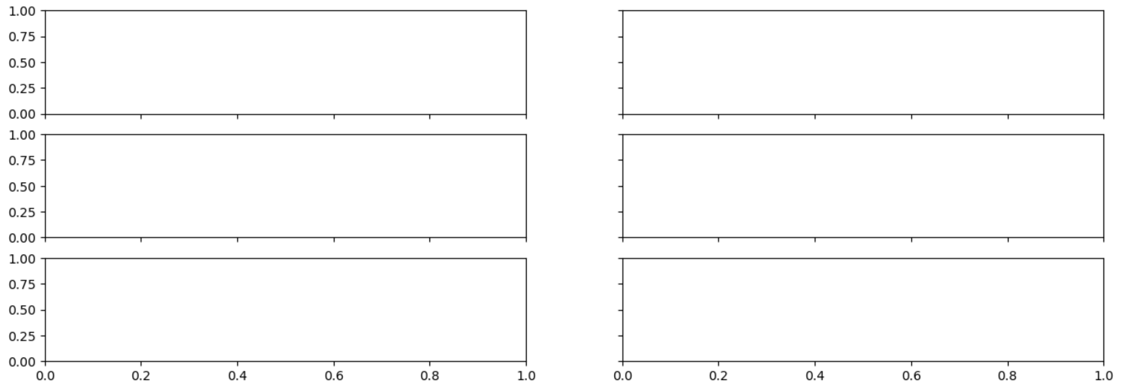
figsize=(width,height)

```
[66]: fig, ax= plt.subplots(3,2,figsize=(15,5))
      fig.savefig('graph.png',dpi=100)
```



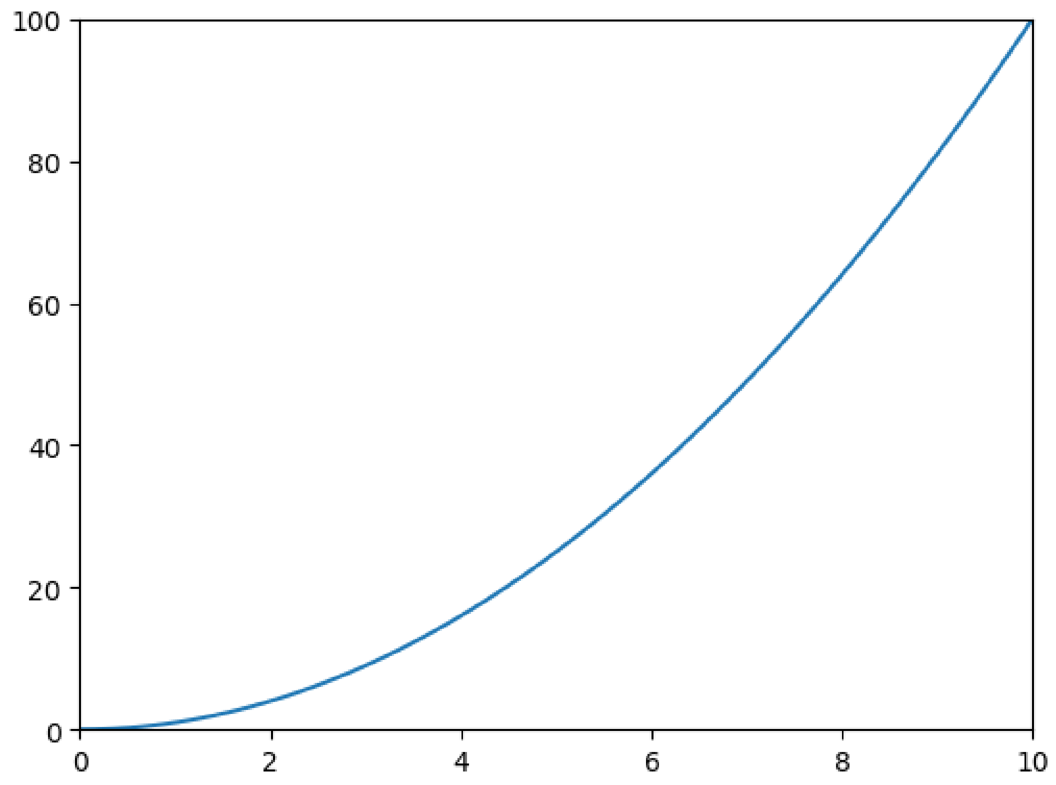
Sharing X and Y axis using sharex and sharey

```
[3]: fig, ax= plt.subplots(3,2,figsize=(15,5),sharex=True, sharey=True)
```



```
[8]: x=np.linspace(0,10,2000)
     y= x**2
     fig,ax=plt.subplots()
     ax.set_ylim(0,100)
     ax.set_xlim(0,10)
     ax.plot(x,y)
```

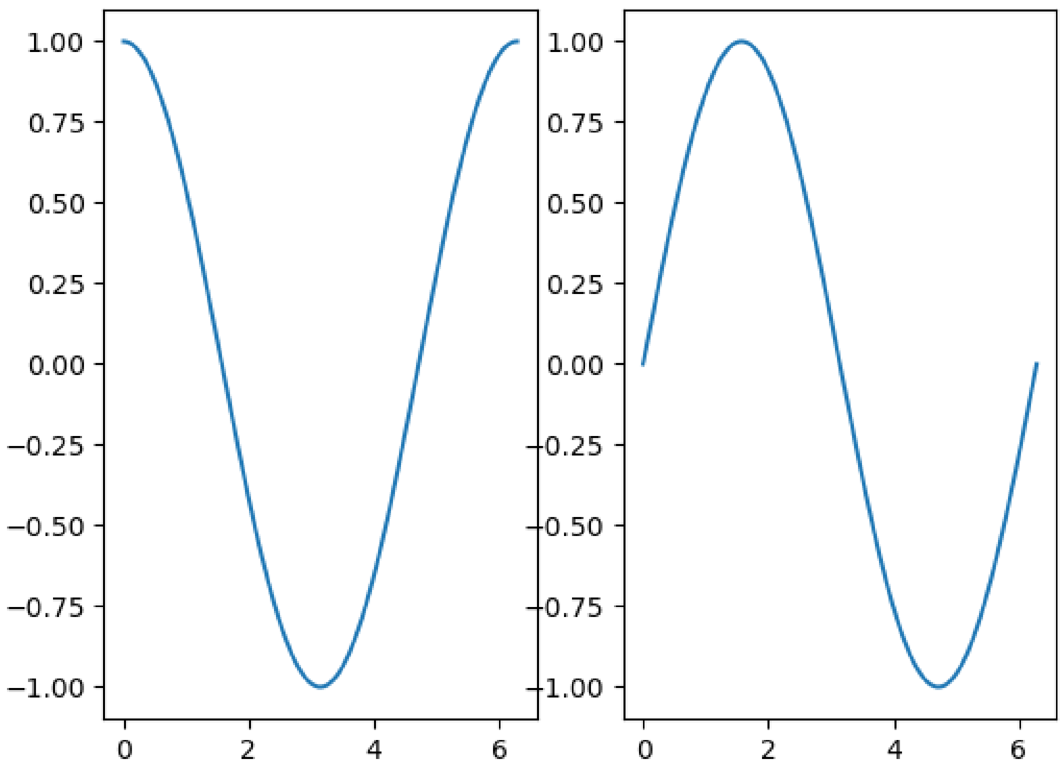
```
[8]: [<matplotlib.lines.Line2D at 0x1ab53332dd0>]
```



10.1 Task : In a figure having 2 axes, plot $\sin(x)$ in second axis and $\cos(x)$ in first axis from 0 to 2 π

[21]:
```
x=np.linspace(0,2*np.pi,1000)
y=np.sin(x)
z=np.cos(x)
fig,ax=plt.subplots(1,2)
ax[1].plot(x,y)
ax[0].plot(x,z)
```

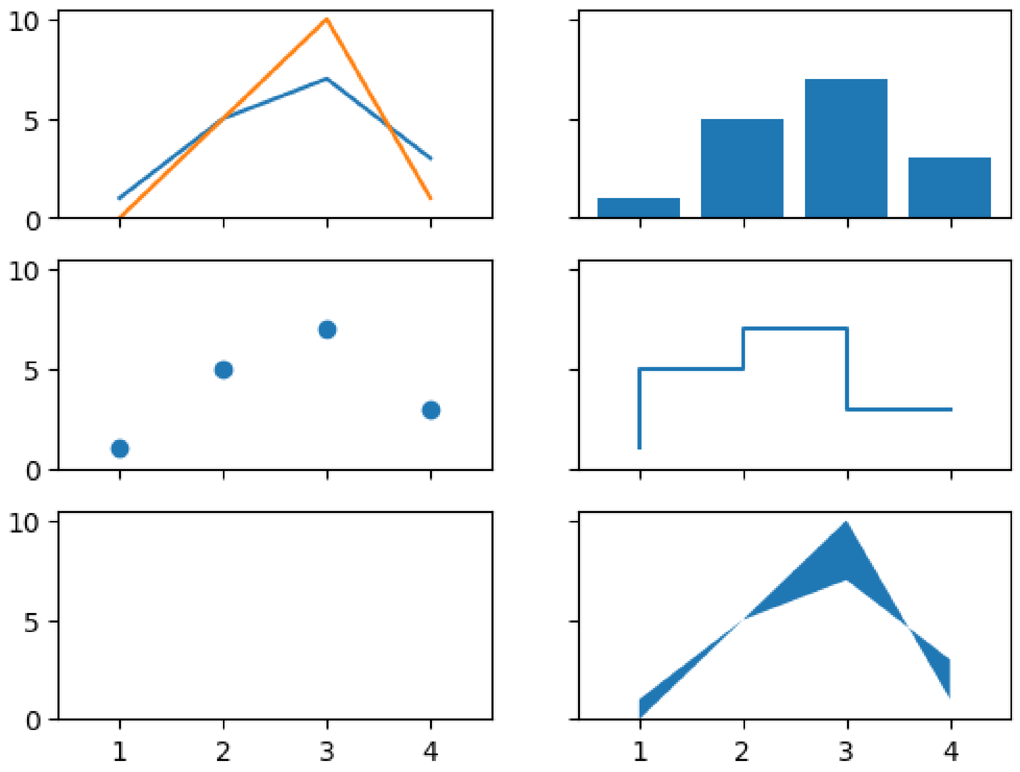
[21]:
```
[<matplotlib.lines.Line2D at 0x1ab569b0450>]
```



10.2 Types of graphs

```
[86]: fig,ax=plt.subplots(3,2,sharex=True, sharey=True)
      x=[1,2,3,4]
      y=[0,5,10,1]
      z=[1,5,7,3]
      ax[0,0].plot(x,z)
      ax[1,0].scatter(x,z)
      ax[1,1].step(x,z)
      ax[0,1].bar(x,z)
      ax[0,0].plot(x,y)
      ax[2,].fill_between(x,y,z)
```

```
[86]: <matplotlib.collections.PolyCollection at 0x20ee3bac990>
```



10.3 Line properties

color =‘red’ , ‘blue’ , ‘green’ , ‘yellow’ etc.

linewidth = lw= float number

linestyle=ls= “-” (solid),“–” (dashed) ,“:” (dotted) ,“-.” (dash-dotted)

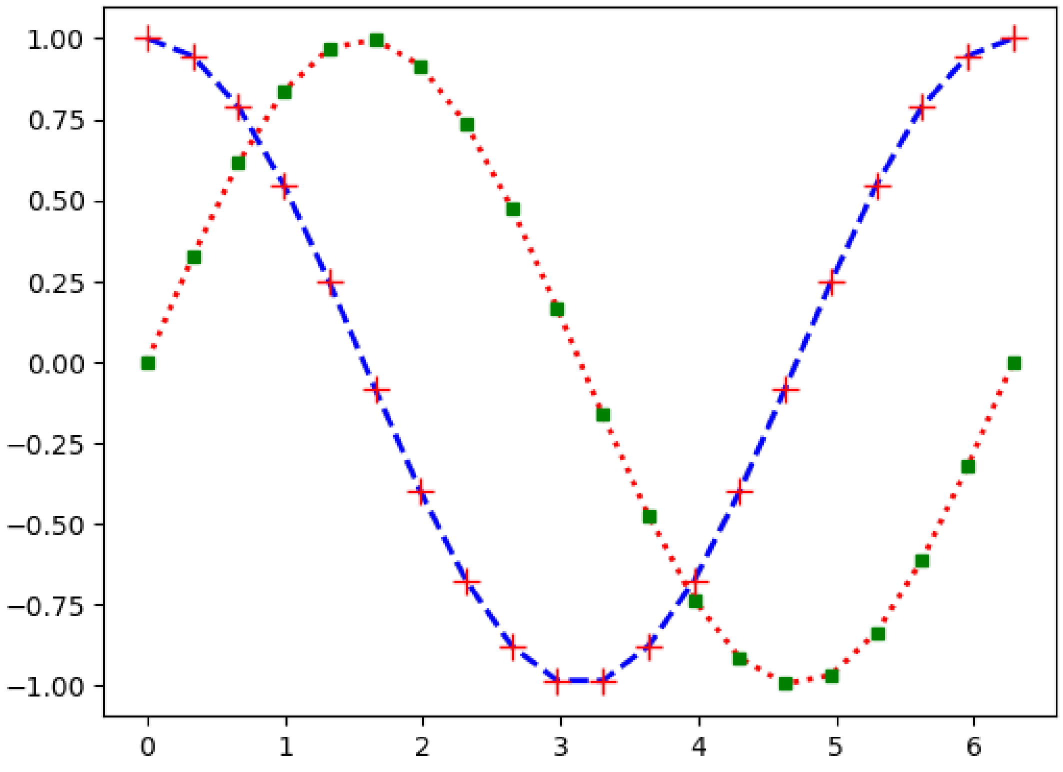
marker = “+” , “o” , “s” (square), “.”

markersize= float number

markerfacecolor= (same colors as above)

```
[16]: x = np.linspace(0, 2 * np.pi, 20)
      y = np.sin(x)
      z=np.cos(x)
      fig, ax = plt.subplots()
      ax.plot(x,y,color="red",lw=2,ls=":
       ↪",marker="s",markersize=4,markerfacecolor="green",markeredgecolor="green")
```

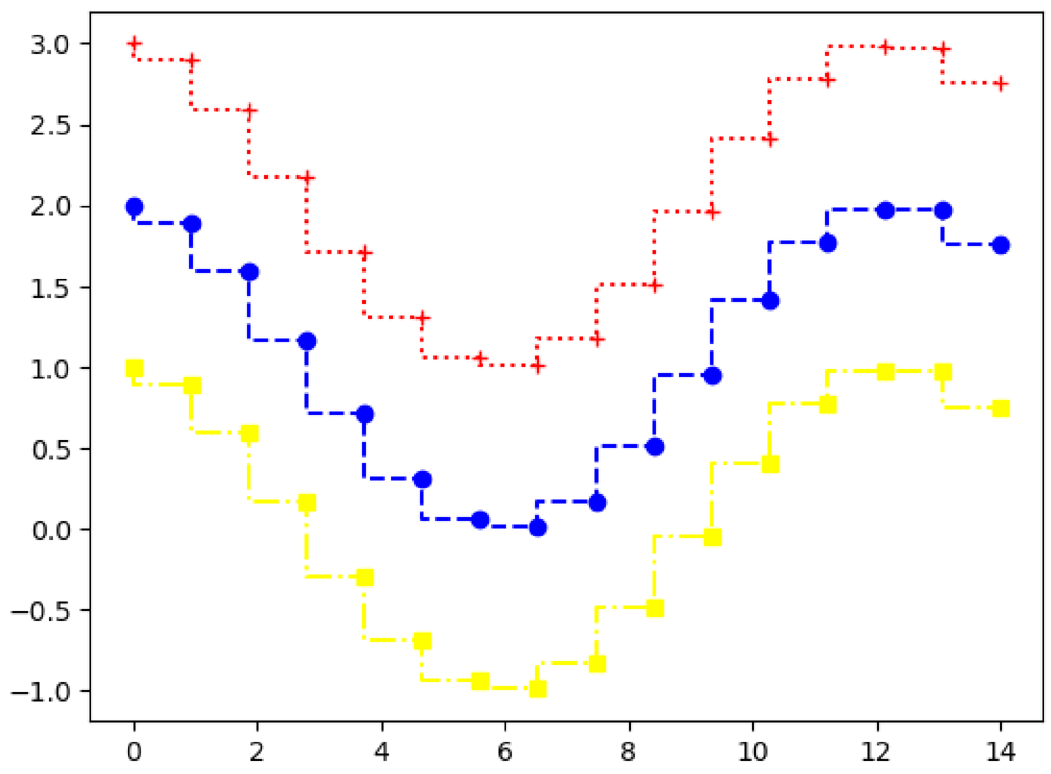
```
ax.
 ↪plot(x,z,color="blue",lw=2,ls="--",marker="+",markersize=10,markeredgecolor="red")
plt.show()
```



10.3.1 Task: Create a step graph of cos(x/2), cos(x/2)+1,cos(x/2)+2 over a range of integers from 1 to 14 with different marker and different colors in one axes instance.

```
[98]: x=np.linspace(0,14,16)
      y=np.cos(x/2)
      fig,ax=plt.subplots()
      ax.step(x,y+1,marker='o',color='blue',ls='--')
      ax.step(x,y+2,marker='+',color='red',ls=':')
      ax.step(x,y,marker='s',color='yellow',ls='-.')
```

```
[98]: [<matplotlib.lines.Line2D at 0x20ee6f73050>]
```



10.3.2 Axes properties

Axis Label : set_xlabel("string_type_arguement") and set_ylabel("string_type_argument").

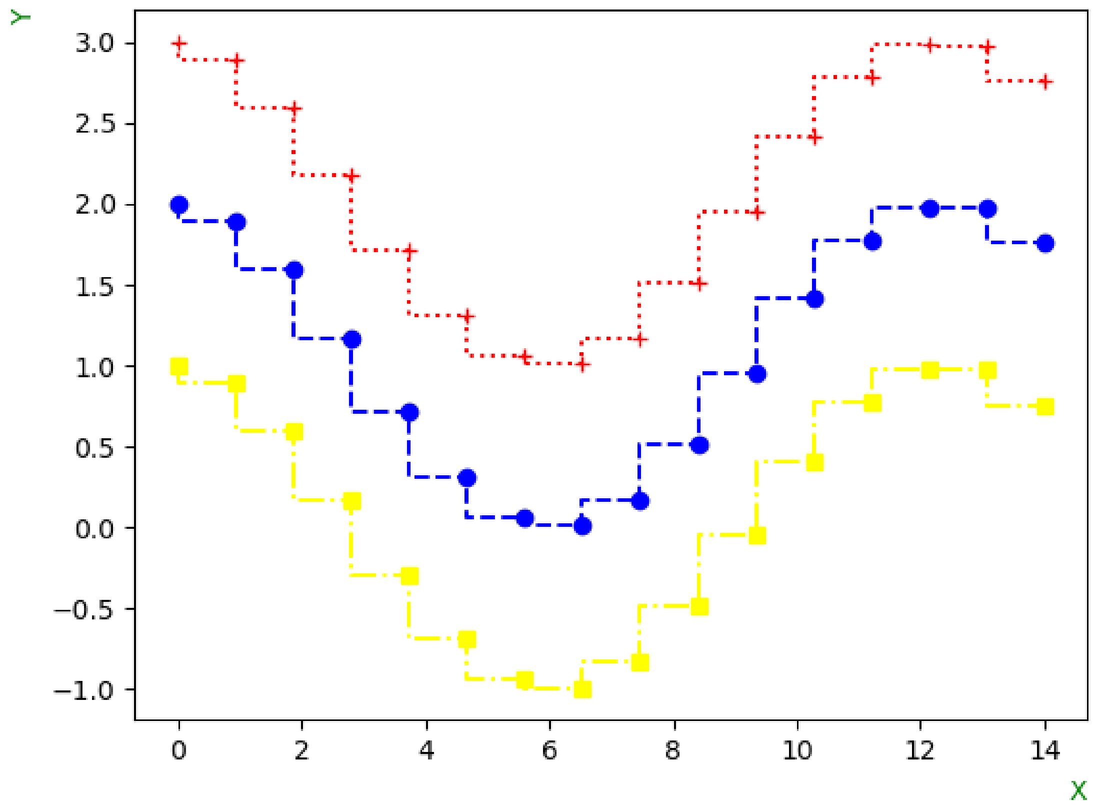
We can use color property in the same way.

labelpad (shows spacing in units of points from the axis to label)

loc ='right', 'left' , 'center' (location of the x axis label) and 'bottom', 'top', 'center' for the y axis

[101]:
```
x=np.linspace(0,14,16)
y=np.cos(x/2)
fig,ax=plt.subplots()
ax.step(x,y+1,marker='o',color='blue',ls='--')
ax.step(x,y+2,marker='+',color='red',ls=':')
ax.step(x,y,marker='s',color='yellow',ls='-.')
ax.set_xlabel('X',labelpad=5,color='green',loc='right')
ax.set_ylabel('Y',labelpad=5,color='green',loc='top')
```

```
[101]: Text(0, 1, 'Y')
```



10.4 Legends and Title

10.4.1 plot(x,y,label=‘string_type’)

10.4.2 legend(loc=1 or 2 or 3 or 4,)

10.4.3 set_title(’string_type’)

loc =1 (upper right corner)

loc =2 (upper left corner)

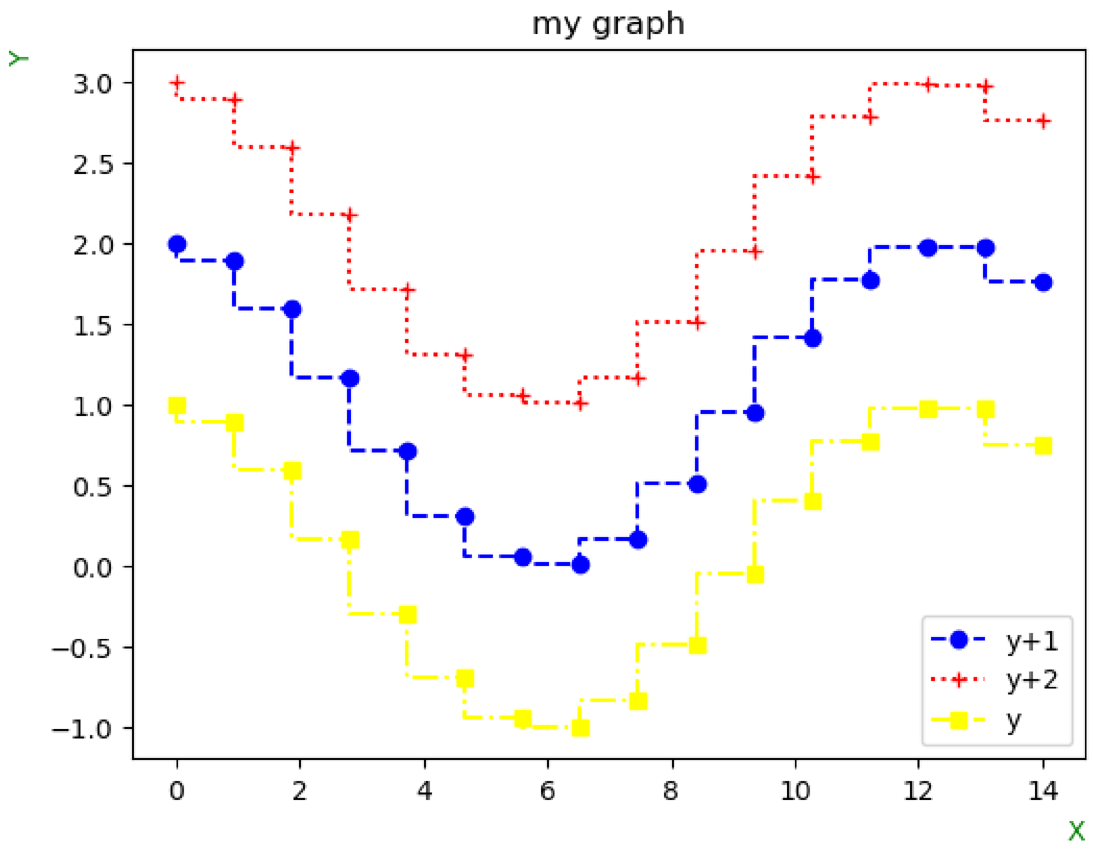
loc =3 (lower left corner)

loc =4 (lower right corner)

```
[19]: x=np.linspace(0,14,16)
      y=np.cos(x/2)
      fig,ax=plt.subplots()
```

```
ax.step(x,y+1,marker='o',color='blue',ls='--',label='y+1')
ax.step(x,y+2,marker='+',color='red',ls=':',label='y+2')
ax.step(x,y,marker='s',color='yellow',ls='-.',label='y')
ax.set_xlabel('X',labelpad=5,color='green',loc='right')
ax.set_ylabel('Y',labelpad=5,color='green',loc='top')
ax.legend(loc=4)
ax.set_title('my graph')
```

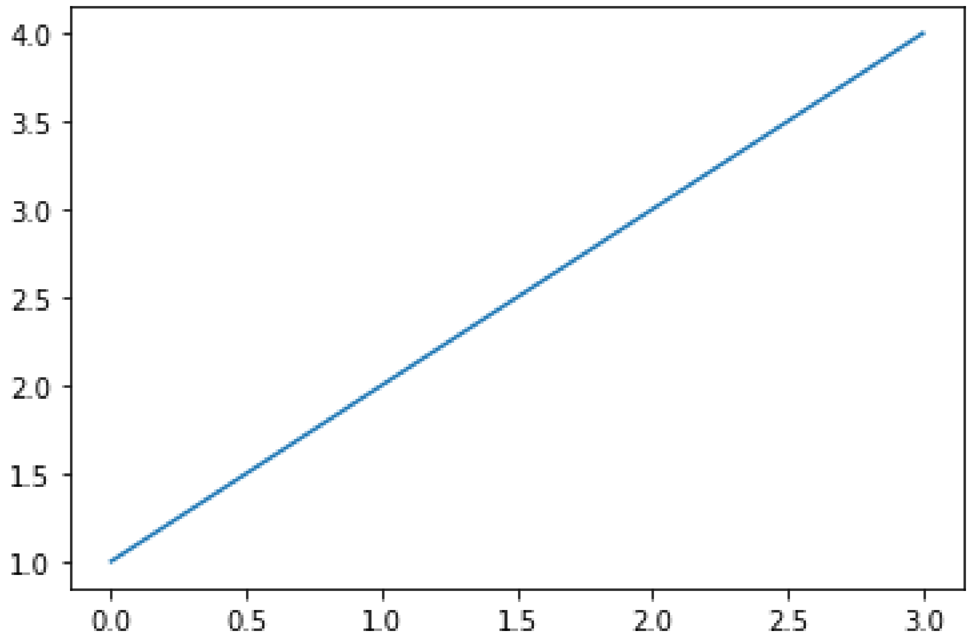
[19]:
```
Text(0.5, 1.0, 'my graph')
```



10.4.4 In a Jupyter notebook, if you want to display a figure in a separate window rather than inline

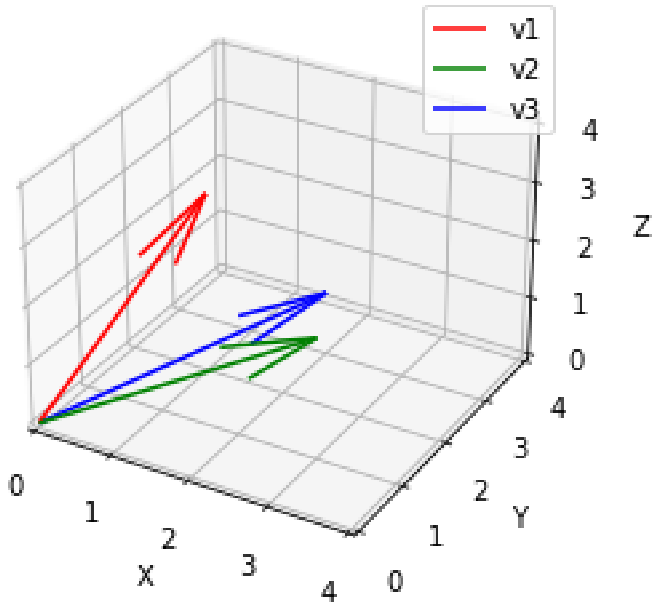
[2]:
```
%matplotlib inline
#This will show the figure in this inline within the notebook(it is by default␣
↪also.)
```

```
#%matplotlib qt
#if instead of %matplotlib inline if you use %matplotlib qt,
#it will open the figure in a separate window.

import matplotlib.pyplot as plt

x=[0,1,2,3]
y=[1, 2, 3, 4]
# Create a figure
plt.figure()
#This function is used to create a new figure (or window) for your plot.
# It initializes a blank canvas for you to draw your plots on.
# By default, it creates a figure in the current notebook cell or,
# if you've executed %matplotlib qt, it will create a figure in a separate GUI␣
 ↪window.

plt.plot(x,y)

# Show the figure
plt.show()
```



10.5 Plotting 3D vectors in space

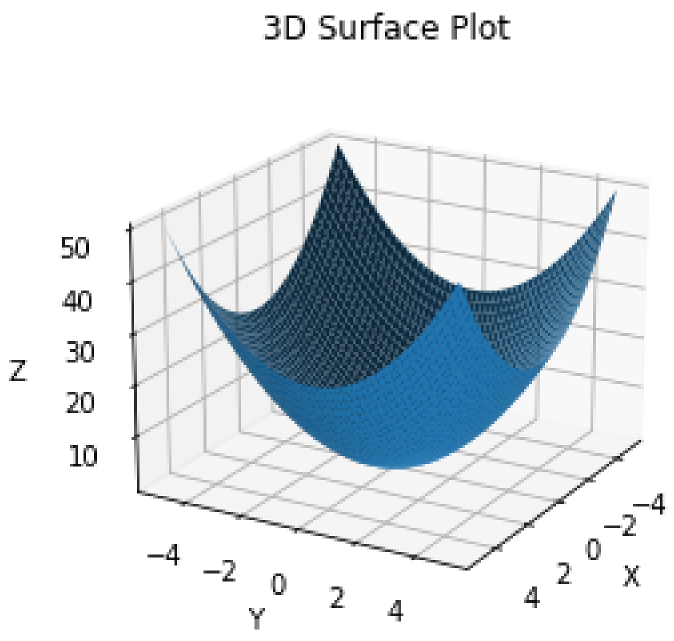
[4]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
#(If you are using different platforms other than
#jupyter notebook then you will need this command to get the access for 3D␣
 ↪plot)
import numpy as np


# Define three 3D vectors
v1 = np.array([1, 2, 3])
v2 = np.array([3, 1, 2])
v3 = np.array([2, 3, 1])

# Create a figure
fig = plt.figure()
ax = fig.add_subplot(projection='3d')

# Plot the vectors
ax.quiver(0, 0, 0, v1[0], v1[1], v1[2], color='r', label='v1')
ax.quiver(0, 0, 0, v2[0], v2[1], v2[2], color='g', label='v2')
ax.quiver(0, 0, 0, v3[0], v3[1], v3[2], color='b', label='v3')
# quiver command takes 6 entries, where first 3 entries will be the origin␣
 ↪point of the vector in the
                                         #plotting, and last 3 entries will be␣
 ↪the coordinates of the vector.
#also after 6 entries i have included the color and labelling.

# Set axis limits
ax.set_xlim([0, 4])
ax.set_ylim([0, 4])
ax.set_zlim([0, 4])

# Add labels
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Add legend
plt.legend()

# Show the plot
plt.show()
```



10.6 3D Surface plotting

10.6.1 Plot the function $(Z = x^2 + y^2)$ *over* $([-5, 5] \times [-5, 5])$

[5]:
```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
#%matplotlib qt

# Generate data
x = np.linspace(-5, 5, 200)
y = np.linspace(-5, 5, 200)
X,Y=np.meshgrid(x, y) # This will form the meshgrid.
Z=(X**2 + Y**2)


# Create a 3D surface plot
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.plot_surface(X, Y, Z)

# Add labels and title
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D Surface Plot')

# Customize viewing angle
ax.view_init(elev=20, azim=30) # This line customizes the viewing angle of the␣
 ↪plot.
                                            # 'elev' controls the elevation angle (tilting␣
 ↪the plot),and
                                            # 'azim' controls the azimuthal angle (rotating␣
 ↪the plot).
# Show the plot
plt.show()
```



11 Calculus continue

11.1 Evaluating the value of expressions over array

[1]:
```
import numpy as np
import sympy as sp
sp.init_printing()
```

[116]:
```
x=sp.Symbol("x")
expr0=x**3+x**2+1
expr0.subs({x:1})
a=np.linspace(1,10,10)
print(a)
expr0.subs({x:a})
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

[116]: $x^3 + x^2 + 1$

[122]:
```
x=sp.Symbol("x")
expr0=x**3+x**2+1
expr0.subs({x:1})
a=np.linspace(1,10,10)
expr0=sp.lambdify(x,expr0,modules="numpy")
expr0(a)
```

[122]:
```
array([   3.,   13.,   37.,   81.,  151.,  253.,  393.,  577.,  811.,
       1101.])
```

11.2 Calculus

11.2.1 Derivatives

[2]:
```
# First order derivatives
x=sp.Symbol("x")
expr=x**4+x**3+x**2+x+1
expr.diff(x)
```

[2]: $4x^3 + 3x^2 + 2x + 1$

[5]:
```
# Multi order derivatives
expr.diff(x,2)
```

[5]: $2 \cdot \left(6x^2 + 3x + 1\right)$

[6]:
```
# Partial derivatives
y=sp.Symbol("y")
expr1=sp.sin(x*y)*sp.cos(x/2)
expr1.diff(y)
```

[6]: $x \cos\left(\frac{x}{2}\right) \cos\left(xy\right)$

[7]:
```
# Multi order partial derivatives
expr1.diff(x,2,y,2)
```

[7]:

$$x^2y^2\sin(xy)\cos\left(\frac{x}{2}\right) + x^2y\sin\left(\frac{x}{2}\right)\cos(xy) + \frac{x^2\sin(xy)\cos\left(\frac{x}{2}\right)}{4} - 4xy\cos\left(\frac{x}{2}\right)\cos(xy) + 2x\sin\left(\frac{x}{2}\right)\sin(xy) - 2\sin(xy)\cos\left(\frac{x}{2}\right)$$

[10]:
```
# Derivative in symbolic form
d=sp.Derivative(sp.exp(sp.cos(x)),x)
d
```

[10]: $$\frac{d}{dx}e^{\cos(x)}$$

[13]:
```
d.doit()
```

[13]: $$-e^{\cos(x)}\sin(x)$$

11.2.2 Integration

[16]:
```
# Integration without limits
expr.integrate(x)
sp.integrate(expr,x)
```

[16]: $$\frac{x^5}{5} + \frac{x^4}{4} + \frac{x^3}{3} + \frac{x^2}{2} + x$$

[19]:
```
# Integration with limits
sp.integrate(expr,(x,1,2))
```

[19]: $$\frac{887}{60}$$

[20]:
```
sp.integrate(sp.exp(-x**2),(x,0,sp.oo))
```

[20]: $$\frac{\sqrt{\pi}}{2}$$

11.2.3 Task: Integrate $\left(f(x) = ae^{(-(x-b)/c))^2}\right)$ $from\,((-\infty,\infty))$

[22]:
```
# Integration with more than one variable
expr2=(x+y)**2
sp.integrate(expr2,x)
```

[22]: $$\frac{x^3}{3} + x^2y + xy^2$$

[23]:
```
sp.integrate(expr2,x,y)
```

[23]: $$\frac{x^3y}{3} + \frac{x^2y^2}{2} + \frac{xy^3}{3}$$

[24]:
```
# Definite Integration with more than one variable
sp.integrate(expr2,(x,0,1),(y,0,1))
```

[24]: $\frac{7}{6}$

11.2.4 Sums and Products

[29]:
```
# Sum of series
n=sp.Symbol("n",integer=True)
a=sp.Sum(1/(n**2),(n,1,sp.oo))
a
```

[29]: $\sum_{n=1}^{\infty} \frac{1}{n^2}$

[30]:
```
a.doit()
```

[30]: $\frac{\pi^2}{6}$

[39]:
```
sp.N(a,50)
```

[39]: 1.6449340668482264364724151666460251892189499012068

[31]:
```
# Product of Series
b=sp.Product(n,(n,1,7))
b
```

[31]: $\prod_{n=1}^{7} n$

[32]:
```
b.doit()
```

[32]: 5040

[35]:
```
sp.N(b)
```

[35]: 5040.0

11.3 Equations

11.3.1 Solve for one variable

[52]:
```
x=sp.Symbol("x")
expr3=x**2+2*x-3
c,d=sp.solve(expr3,x)
c,d
```

[52]: $(-3, \ 1)$

[63]:
```
y=sp.Symbol("y")
expr4=x**2+y**2
sp.solve(expr4,y)
```

[63]: $[-ix,\ ix]$

11.3.2 Solving simultaneous equations

[80]:
```
eq1=x+2*y-1
eq2=x-y+1
sol=sp.solve([eq1,eq2],[x,y])
sol
```

[80]: $\left\{x : -\frac{1}{3},\ y : \frac{2}{3}\right\}$

[84]:
```
eq3=x**2-y
eq4=y**2-x
sol1=sp.solve([eq3,eq4],[x,y])
sol1
```

[84]:
$$\left[(0,\ 0),\ (1,\ 1),\ \left(\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)^2,\ -\frac{1}{2}-\frac{\sqrt{3}i}{2}\right),\ \left(\left(-\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)^2,\ -\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)\right]$$

[107]:
```
eq3=x**2-y
eq4=y**2-x
sol2=sp.solve([eq3,eq4],[x,y],dict=True)
sol2
```

[107]:
$$\left[\{x : 0,\ y : 0\},\ \{x : 1,\ y : 1\},\ \left\{x : \left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)^2,\ y : -\frac{1}{2}-\frac{\sqrt{3}i}{2}\right\},\ \left\{x : \left(-\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)^2,\ y : -\frac{1}{2}+\frac{\sqrt{3}i}{2}\right\}\right]$$

11.4 Matrices in Sympy

[93]:
```
sp.Matrix([[1,2],[3,4]])
```

[93]:
$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

[108]:
```
m,n=sp.symbols("m,n")
#m=sp.Symbol("m")
#n=sp.Symbol("n")
sp.Matrix([[m,m+n],[m**2+n**3,m**2]])
```

[108]:
$$\begin{bmatrix}m & m+n\\ m^2+n^3 & m^2\end{bmatrix}$$