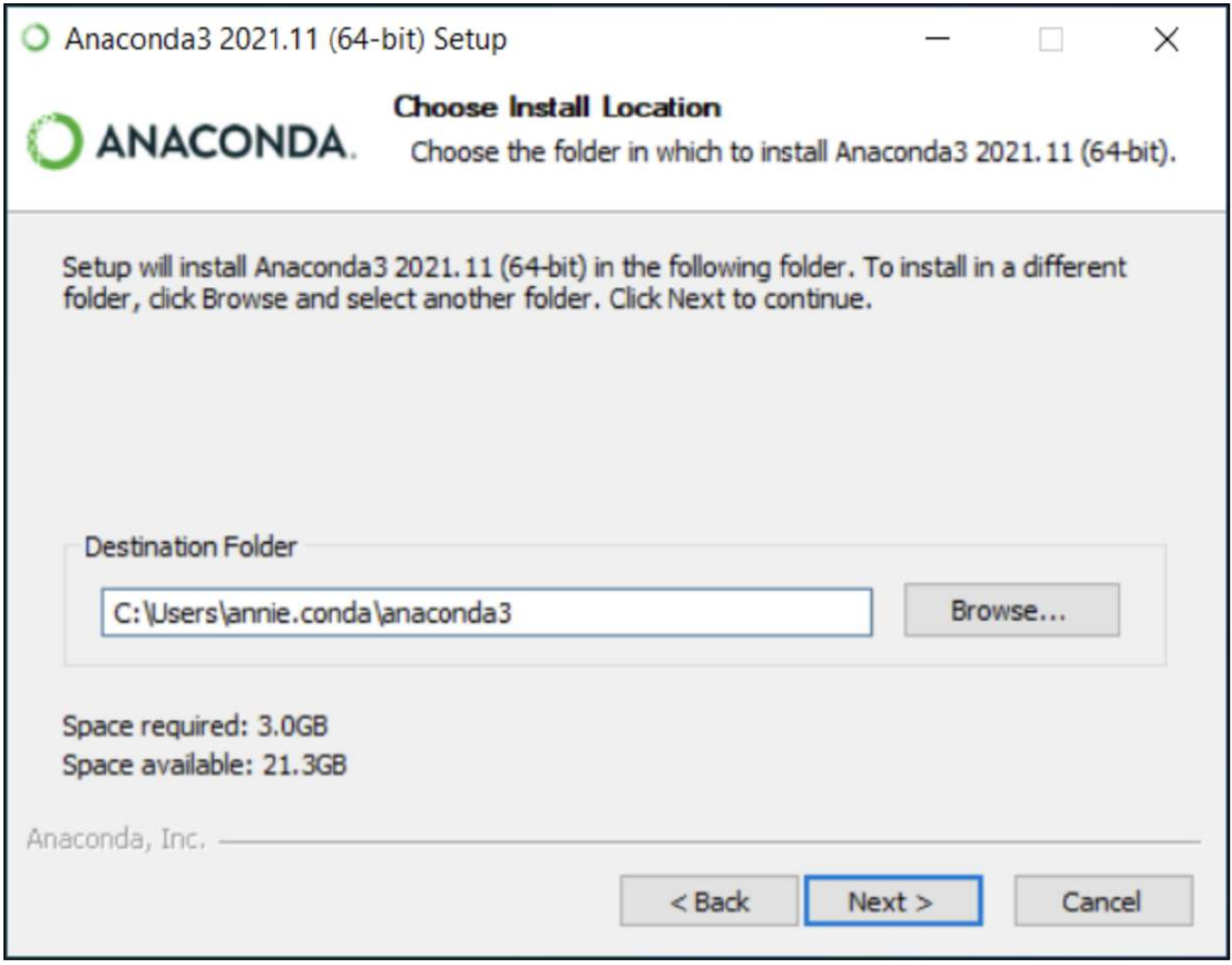
Installation

1. Download the Anaconda installer.
2. Go to your Downloads folder and double-click the installer to launch. To prevent permission errors, do not launch the installer from the Favorites folder.

> **Note**
>
> If you encounter issues during installation, temporarily disable your anti-virus software during install, then re-enable it after the installation concludes. If you installed for all users, uninstall Anaconda and re-install it for your user only.

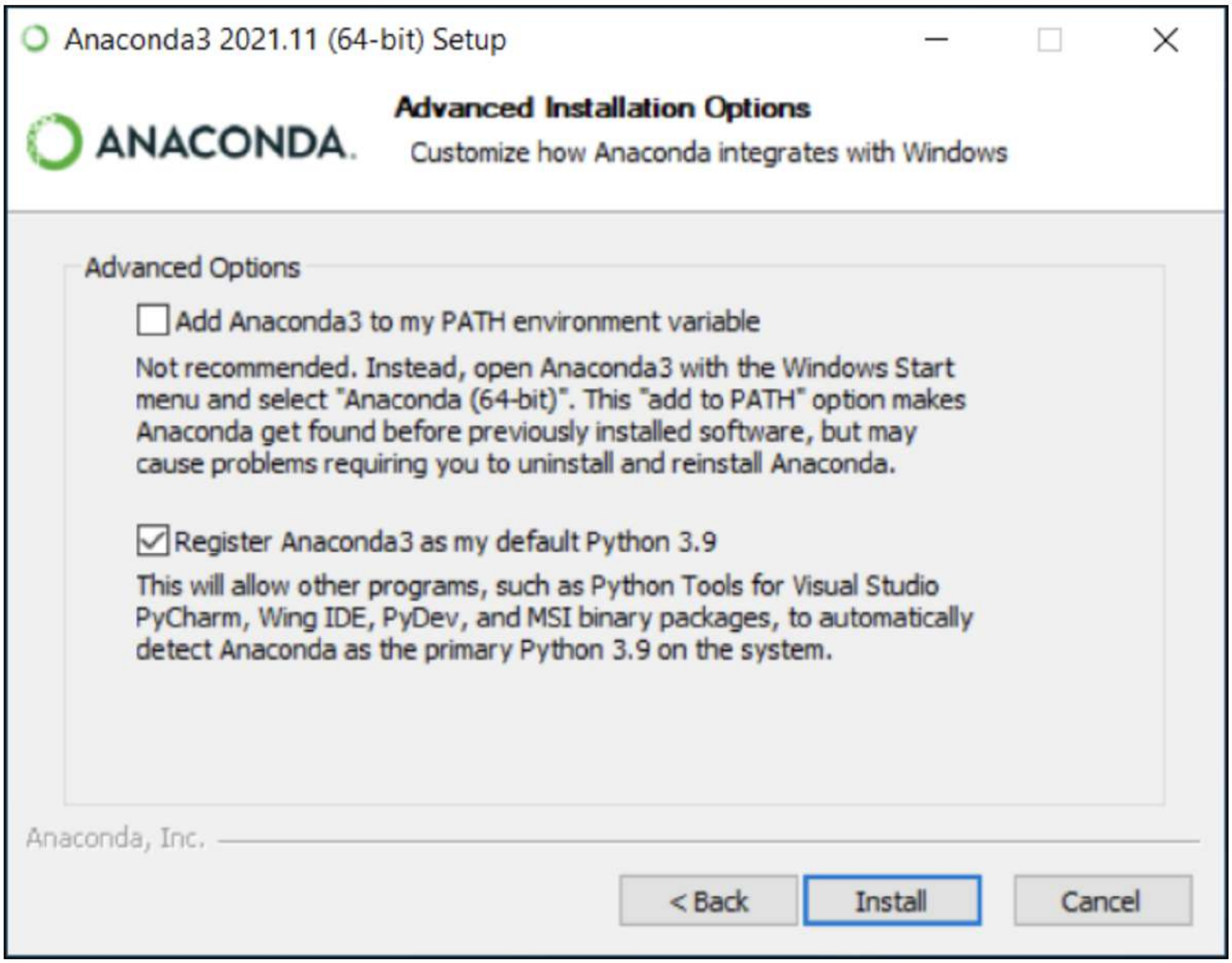
3. Click **Next**.
4. Read the licensing terms and click **I Agree**.
5. It is recommended that you install for **Just Me**, which will install Anaconda Distribution to just the current user account. Only select an install for **All Users** if you need to install for all users' accounts on the computer (which requires Windows Administrator privileges).
6. Click **Next**.
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.

> **Caution**
>
> Do not install as Administrator unless admin privileges are required.


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
Browse...
Space required: 3.0GB
Space available: 21.3GB
Anaconda, Inc.
< Back
Next >
Cancel

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.

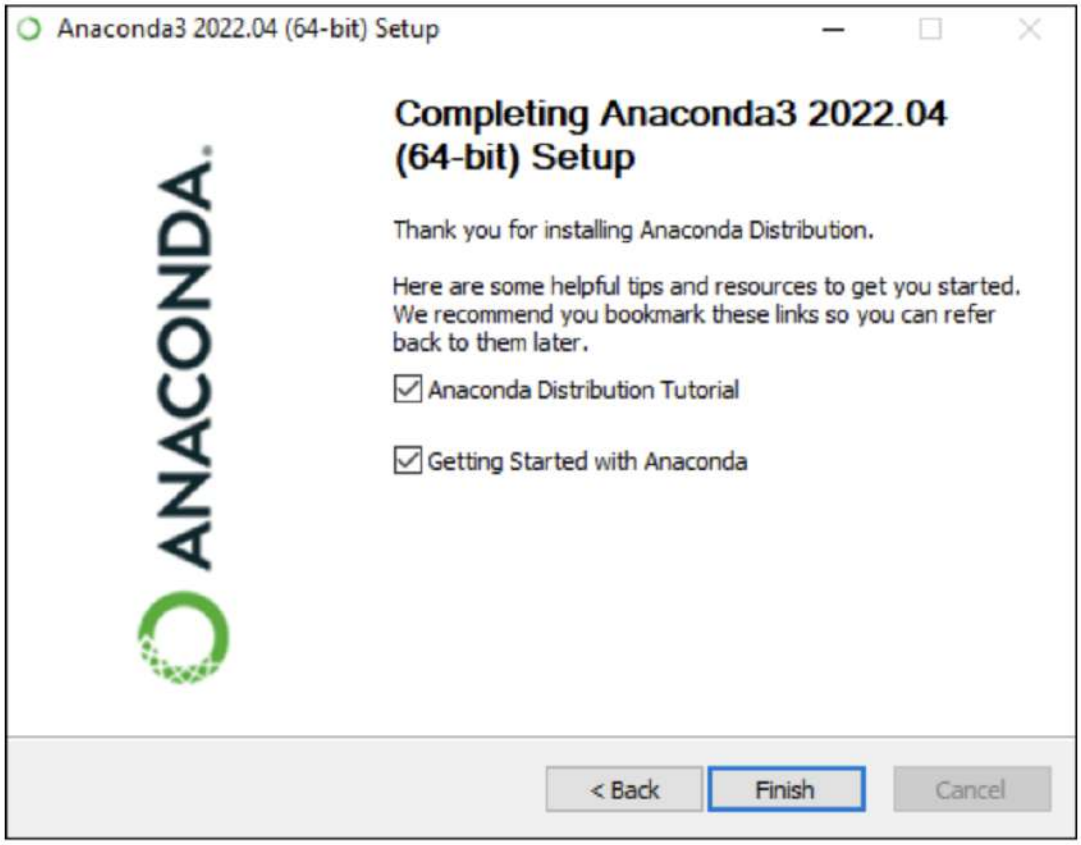
> **Note**
>
> As of `Anaconda Distribution 2022.05`, the option to add Anaconda to the PATH environment variable during an **All Users** installation has been disabled. This was done to address a security exploit. You can still add Anaconda to the PATH environment variable during a **Just Me** installation.


Anaconda3 2021.11 (64-bit) Setup
ANACONDA.
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.
< Back
Install
Cancel

9. Click **Install**. If you want to watch the packages Anaconda is installing, click Show Details.
10. Click **Next**.
11. Optional: To learn more about Anaconda's cloud notebook service, go to https://www.anaconda.com/code-in-the-cloud.

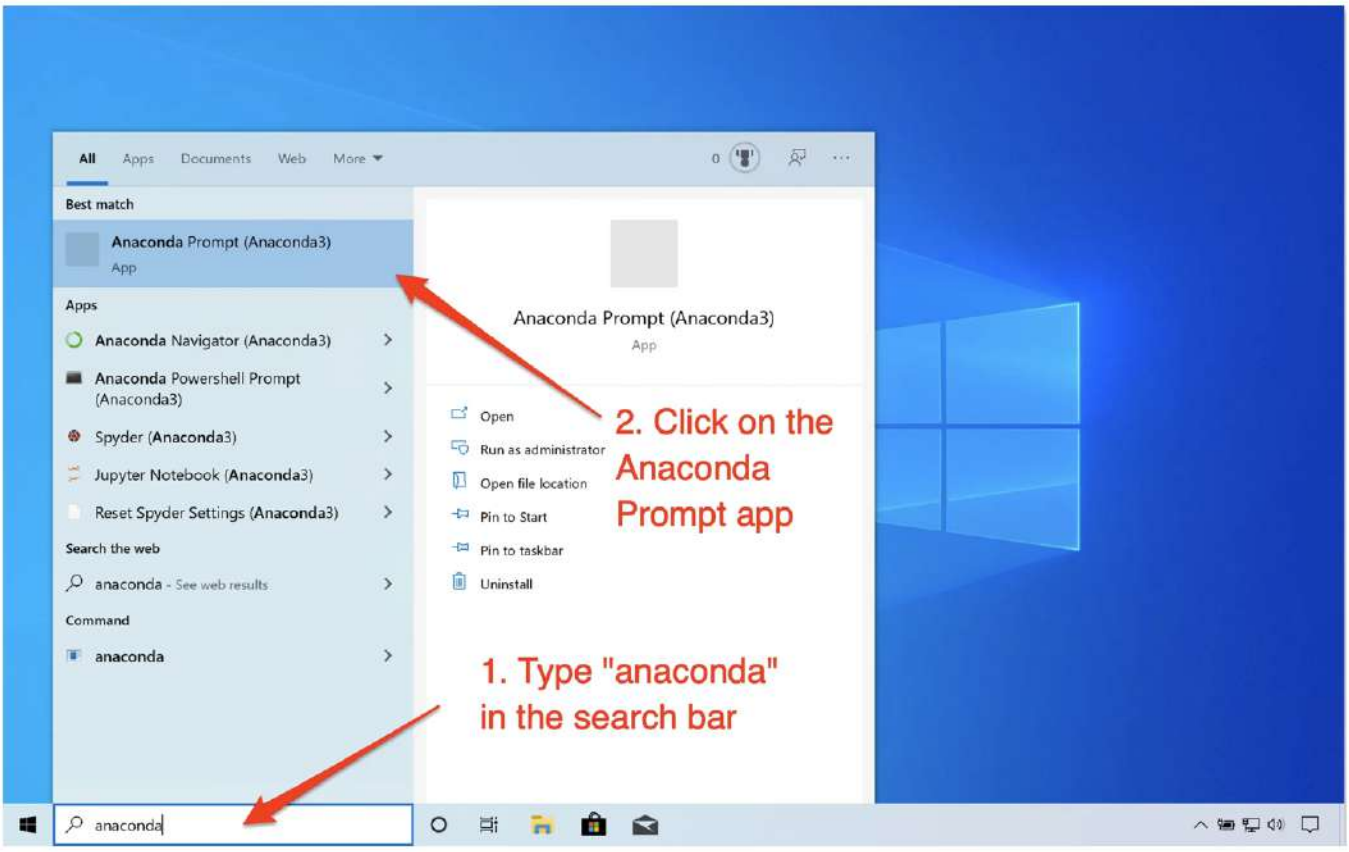
Or click **Continue** to proceed.

12. After a successful installation you will see the "Thanks for installing Anaconda" dialog box:

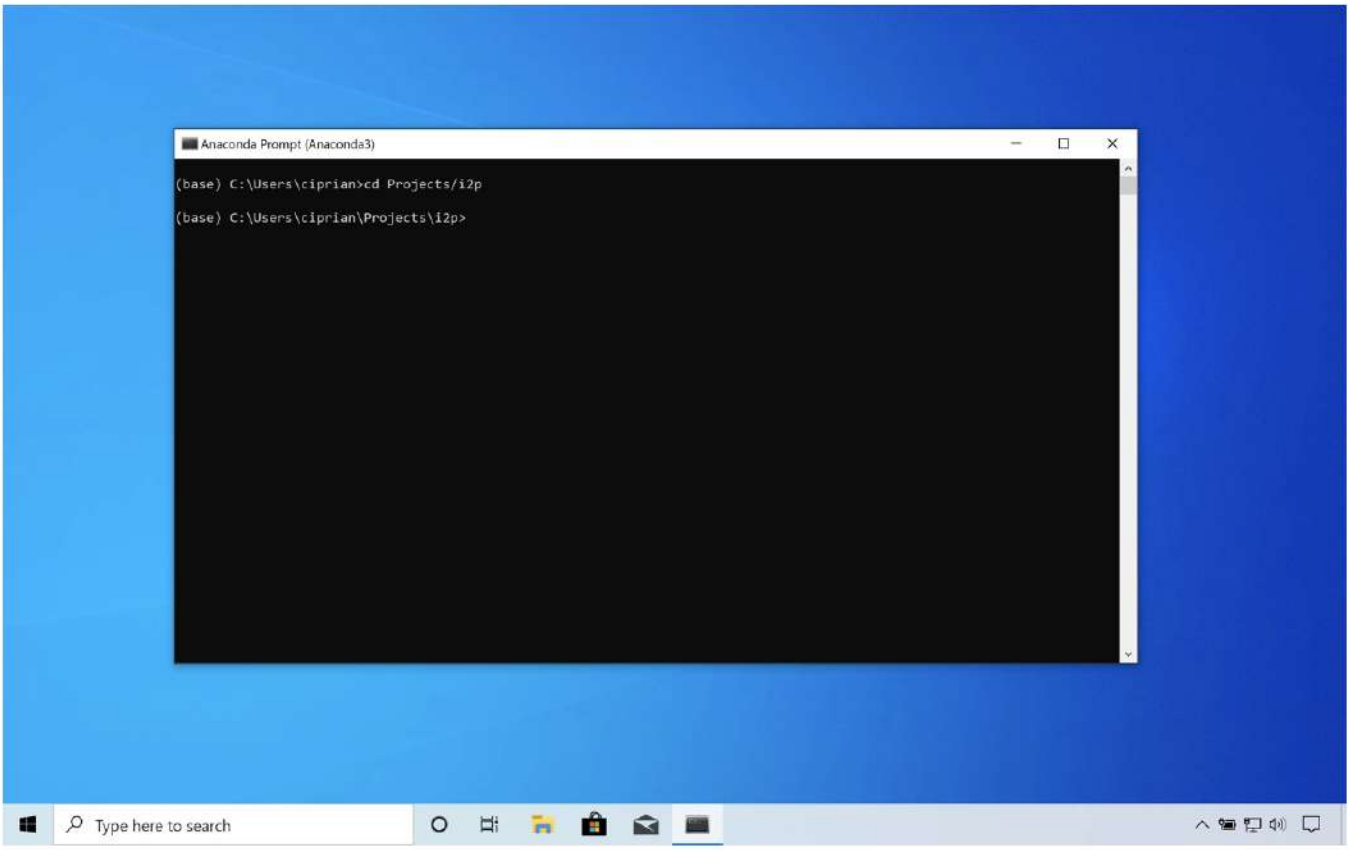


13. If you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes "Anaconda Distribution Tutorial" and "Learn more about Anaconda". Click the **Finish** button.

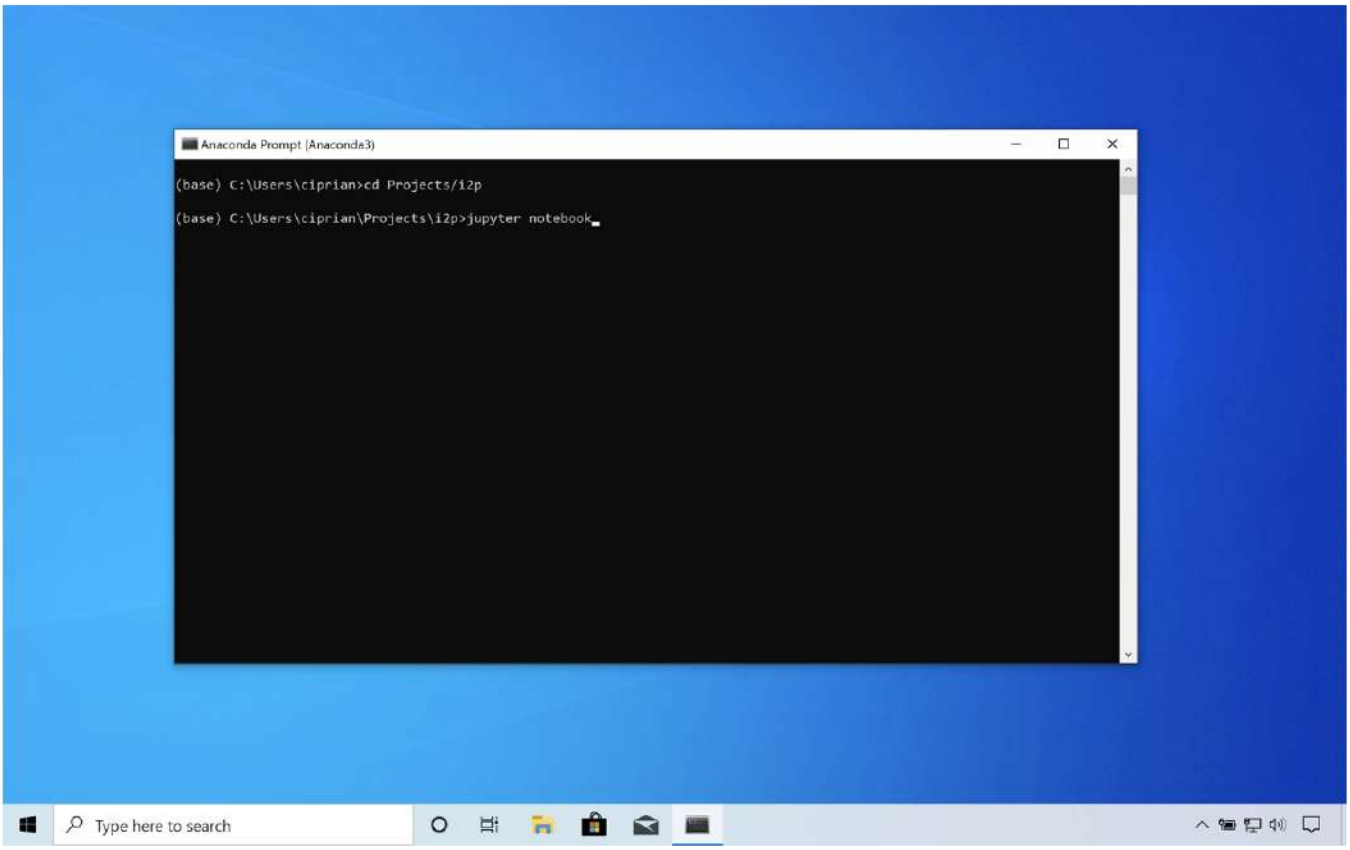
Step 1. Find and open the **Anaconda Prompt** app using the search bar. Alternatively, you can use the **Anaconda Powershell Prompt**. The **Anaconda Powershell Prompt** has more bells and whistles and it is generally nicer to work with.



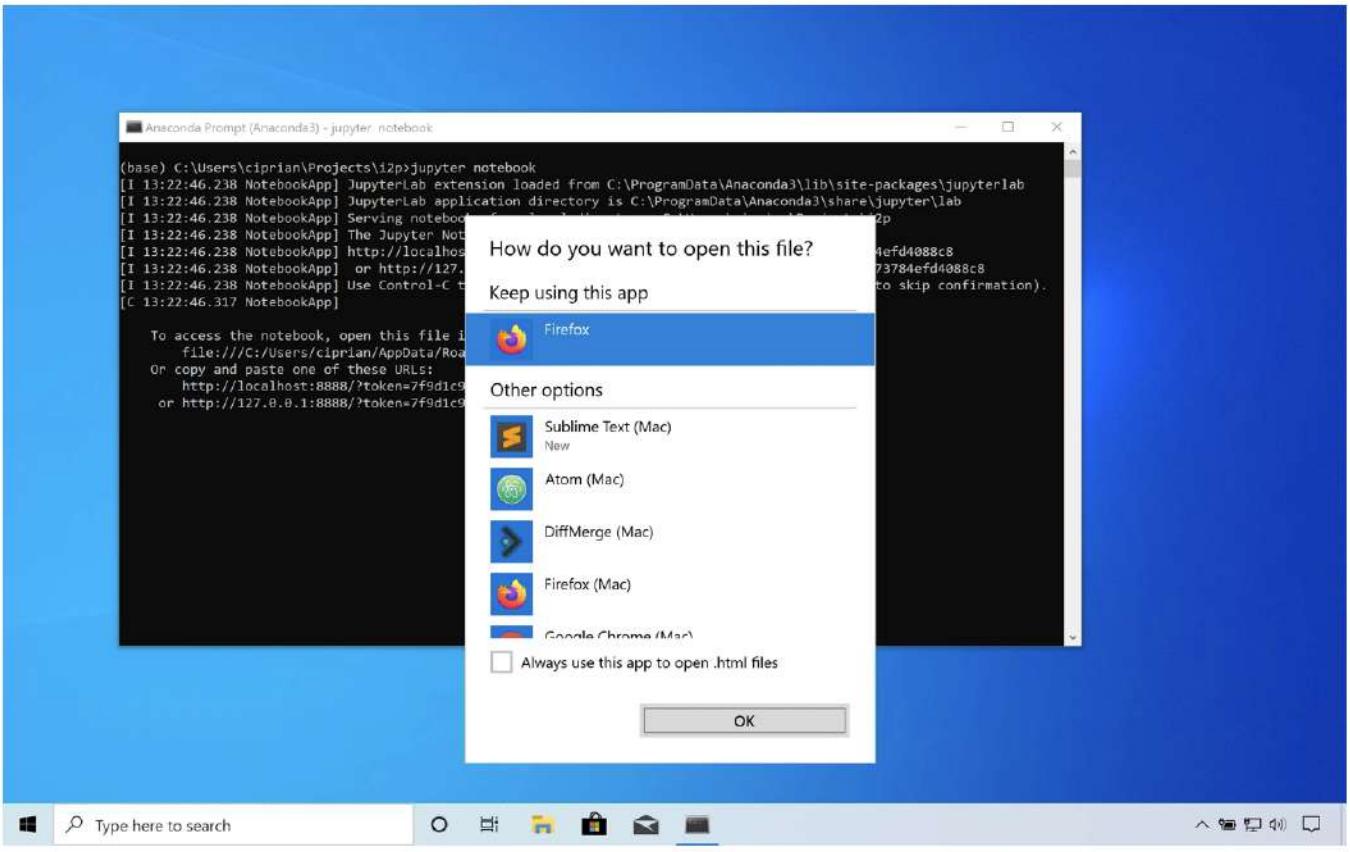
Step 2. Once the **Anaconda Prompt** (or **Anaconda Powershell Prompt**) app opens, navigate to the desired folder, using the `cd` command.



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

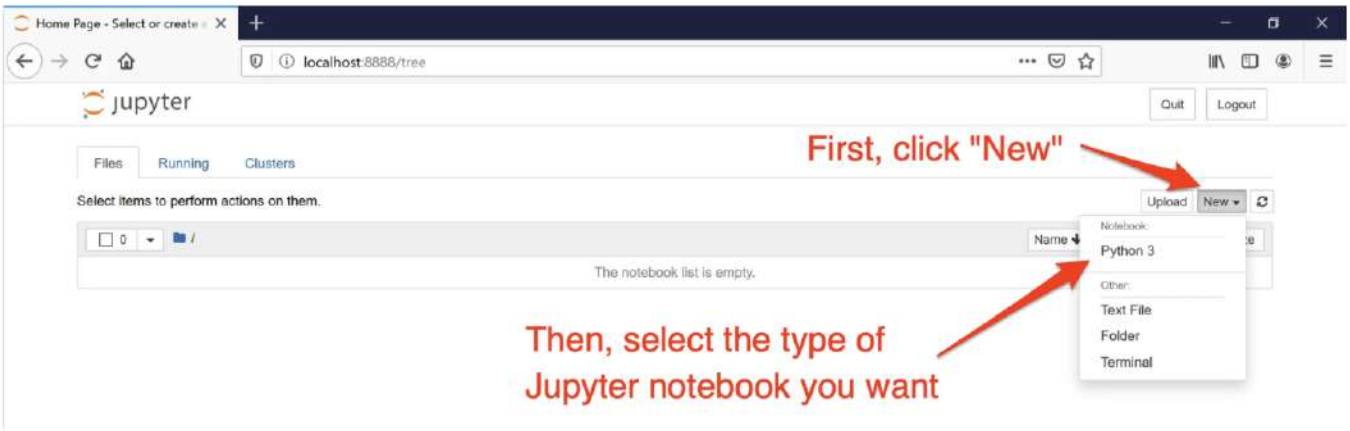


Step 4. The Jupyter server will start. You should see some server logs printed. You may be prompted to select an application to open Jupyter in. **Firefox** or **Chrome** are preferred.

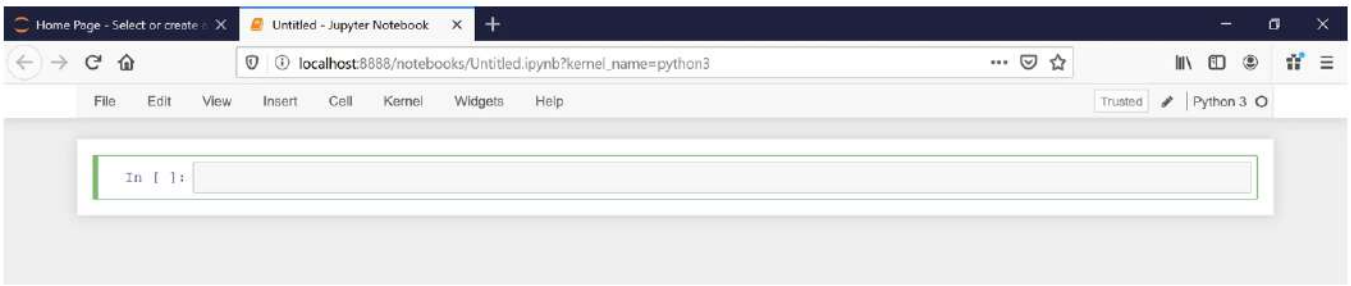


Step 5. Shortly after, a browser window should open, showing the files and folders located in the folder where you started the Jupyter server (in my case, this folder is `C:\Users\ciprian\Projects\i2p`).

Step 6. Create a new Jupyter notebook.



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



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

```
5
```

Types of operator

Arithmatic operators

Assignment operators

Comparison operators

Logical operators

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

In [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
```

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

In [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
```

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

In [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
```

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

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

```
True
True
False
False
```

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

In [19]:
```
#if

a=30
b=40
if a<b:
    print('a is less than b.') #It will give output if the statment is tru
```

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

In [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 i
```

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

In [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.
```

Lists

In [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 in

#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 remo
#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]
```

Out[22]: 9

Range function

In [ ]:
```
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).
```

For loop

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

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

In [24]:
```
# 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
```

In [25]:
```
#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
```

In [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
```

While loop

In [27]:

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

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

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

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

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

    i=i+1
```

```
1
2
```

Functions

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

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

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

In [29]:
```
def adds(a,b):
    c=a+b
    print(c)
```

In [30]:
```
v=adds(1,2)
```

```
3
```

In [31]:
```
print(v)
```

```
None
```

In [32]:
```
def adds(a,b):
    c=a+b
    return c
```

In [33]:
```
w=adds(1,2)
```

In [34]:
```
print(w)
```

```
3
```

In [35]:
```
def amean(x,y):
    z=(x+y)/2
    return z
```

In [36]:
```
v=amean(2,3)
```

In [37]:
```
def gmean(g,h):
    i=(g*h)**(1/2)
    return i
```

In [38]:
```
w=gmean(2,3)
```

In [39]:
```
if v>w:
    print("A.M is greater")
else:
    print("A.M is not")
```

```
A.M is greater
```

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.

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

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

```
Disqualified
```

NUMPY

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

Array

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

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

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

```
[22]
```

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

```
[1 2 3]
```

In [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]]
```

In [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]]
```

In [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]]
```

In [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]]
```

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

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

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

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

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

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

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

```
(2, 2)
```

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

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

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

In [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]]
```

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

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

In [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]]
```

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

```
23.999999999999993
```

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

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

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

```
4
```

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

```
0
```

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

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

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

Out[63]: 6

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

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

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

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

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

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

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

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

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

In [68]:
```
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.        ]
```

Random Integers

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

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

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

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

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

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

Meshgrid

In [7]:
```
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]]
```

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

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

Now find the value of function Z=X+10*Y over X∈ [0,5] and Y ∈ [0,5] with increment 1

In [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]]
```

Slicing sub arrays from a Matrix

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

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

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

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

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

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

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

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

Constructing matrices with 1's on subdiagonal or superdiagonal

In [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
```

Out[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]]))
```

Trigonometric Functions

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

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

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

Out[16]:
```
0.6931471805599453
```

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

Out[17]:
```
1.0
```

Symbolic Computing

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

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

Out[19]: 3.14159265358979

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

Out[20]: π

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

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

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

Out[22]: $\sin\left(\pi x\right)$

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

Out[23]: 0

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

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

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

Out[25]: x

Expressions

In [26]:
```
x=sp.Symbol("x")
```

In [27]:
```
expr=1+2*x**2+3*x**3
expr
```

Out[27]: $3x^3 + 2x^2 + 1$

In [28]:
```
expr.args
```

Out[28]: $\left(1,\ 2x^2,\ 3x^3\right)$

In [29]:
```
expr.args[2]
```

Out[29]: $3x^3$

Simplification

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

Out[30]: $2x^2 - x\left(x + 1\right) - 2x$

In [31]:
```
sp.simplify(expr)
```

Out[31]: $x\left(x - 3\right)$

In [32]:
```
expr.simplify()
```

Out[32]: $x\left(x - 3\right)$

Expand

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

Out[33]: $x^2 + 3x + 2$

In [34]:
```
expr.expand()
```

Out[34]: $x^2 + 3x + 2$

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

Out[35]: $\sin(x)\cos(y) + \sin(y)\cos(x)$

Factor, Collect and Combine

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

Out[36]: $(x-1)(x+1)$

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

Out[37]: $\log\left(\frac{x}{y}\right)$

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

Out[38]: $x(yz+1) + y + z$

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

Out[39]: $x + y(xz+1) + z$

Apart and Together

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

Out[40]: $-\frac{1}{x+2} + \frac{1}{x+1}$

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

Out[41]: $\frac{y+1}{y(x+1)}$

Substitutions

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

Out[42]: 13.3

Numerical Evaluation

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

Out[43]: 3.14159265358979

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

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

Out[45]: 3.14159265358979

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

Out[46]: $3.1415926535897932384626433832795028841971693993751$

Calculus

Derivatives

In [47]:
```
expr=x**4+x**3+x**2+x+1
expr.diff(x)
```

Out[47]: $4x^3 + 3x^2 + 2x + 1$

In [48]:
```
expr.diff(x,2)
```

Out[48]: $2 \cdot \left(6x^2 + 3x + 1\right)$

In [49]:
```
expr.diff(x,3)
```

Out[49]: $6 \cdot \left(4x + 1\right)$

In [50]:
```
expr1=(x+1)**3+y**2*(z-1)
```

In [51]:
```
expr1.diff(x)
```

Out[51]: $3(x+1)^2$

In [52]:
```
expr1.diff(y)
```

Out[52]: $2y\,(z-1)$

In [53]:
```
expr1.diff(x,y)
```

Out[53]: 0

Integration

In [54]:
```
expr2=sp.sin(x)
expr2
```

Out[54]: $\sin(x)$

In [55]:
```
sp.integrate(expr2)
```

Out[55]: $-\cos(x)$

In [56]:
```
sp.integrate(expr2,x)
```

Out[56]: $-\cos(x)$

In [57]:
```
a=sp.Symbol("a")
b=sp.Symbol("b")
sp.integrate(expr2,(x,a,b))
```

Out[57]: $\cos(a)-\cos(b)$

Sums and Products

In [58]:
```
n=sp.Symbol("n",integer=True)
x=sp.Sum(1/(n**2),(n,1,oo))
x
```

Out[58]: $\sum_{n=1}^{\infty} \frac{1}{n^2}$

In [59]:
```
x.doit()
```

Out[59]: $\frac{\pi^2}{6}$

In [60]:
```
y=sp.Product(n,(n,1,7))
y
```

Out[60]: $\prod_{n=1}^{7} n$

In [61]:
```
y.doit()
```

Out[61]: 5040

Equations

In [62]:
```
u=sp.Symbol("u")
expr4=u**2+2*u-3
expr4
```

Out[62]: $u^2 + 2u - 3$

In [63]:
```
sp.solve(expr4,u)
```

Out[63]: $[-3,\ 1]$

In [64]:
```
m=sp.Symbol("m")
n=sp.Symbol("n")
eq1=m+2*n-1
eq2=m-n+1
sol=sp.solve([eq1,eq2],[m,n])
sol
```

Out[64]: $\left\{m : -\frac{1}{3},\ n : \frac{2}{3}\right\}$

In [65]:
```
sol(1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[65], line 1
----> 1 sol(1)

TypeError: 'dict' object is not callable
```

In [66]:
```
sol1=sp.solve([eq1,eq2],[m,n],dict=True)
sol1
```

Out[66]:
$$\left[\left\{m : -\frac{1}{3},\ n : \frac{2}{3}\right\}\right]$$

In [67]:
```
sol1
```

Out[67]:
$$\left[\left\{m : -\frac{1}{3},\ n : \frac{2}{3}\right\}\right]$$

In [ ]:
```

```