In [1]:
```
# Hi Welcome to Python!
# Technically, this is Jupyter
# Jupyter stands for Julia Python R
```

In [2]:
```
# At the simplest level you can use a notebook as a fancy calculator!

a = 2
b = 2
```

In [3]:
```
a+b
```

Out[3]: 4

In [4]:
```
b = 10
```

In [5]:
```
a+b
```

Out[5]: 12

In [7]:
```
# Cells come in different kinds, cells with (Python) code and cells with Markdow

# Markdown is a wrapper for html and Latex (and other things)
```

Hello! This is a header

This is a smaller header

This is a yet smaller header

This is another Markdown cell.

This is how you do italics

This is also a way to do italics

This is how you do bold

- Look! its a bullet point
- And another
- and another...
- Yes

and lists? easy too:

1. Item number one

2. Item number two
3. Item number three...

    A. Item number one
    B. Item number two
    C. Item number three...

Most amazingly, you can also do LaTeX code: $\int_{-\infty}^{\infty} e^{-\pi x^2}\ dx = 1$

Amazing!

$$x^2 + 1 = 0$$

To do a series of equations

$$\begin{aligned} x^2 + 1 \\ x^2 + 2x + 1 \\ \cos(x) + \sin(2x) \end{aligned}$$

You can also do links, rather simply:

Click here to go to Google

Here is a website that uses a lot of Markdown and has an interesting book

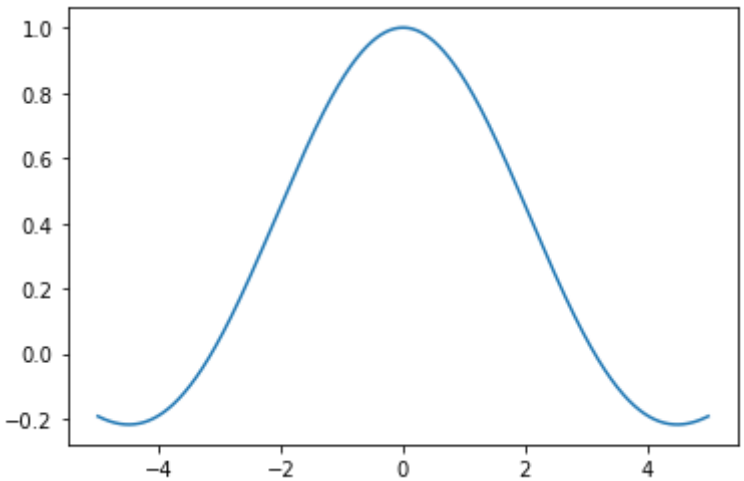
In [44]:
```
# You can use this as your sandbox for coding
# kind of like a super graphing calculator sitting on a browser tab

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5,5,1000)
y = np.sin(x) / x

plt.plot(x,y)
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x7fe3d79e5100>]
```



In [29]:
```
# You can define functions

def Sum(a,b):
    return a+b

def Multiplication(a,b):
    return a*b

def IsThisTheNumberOne(x):
    if x == 1:
        return True
    else:
        return False
```

In [30]:
```
HelloWorld()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-30-45c4279b041a> in <module>
----> 1 HelloWorld()

NameError: name 'HelloWorld' is not defined
```

In [33]:
```
def HelloWorld():
    print('Howdy!')

HelloWorld()
```

```
Howdy!
```

In [35]:
```
# You can also work with strings

String1 = "Night gathers..."
String2 = " and now my watch begins"
String3 = Sum(String1,String2)
```

In [38]:
```
print(String1)
print(String2)
```

```
Night gathers...
 and now my watch begins
```

In [39]:
```
print(String3)
```

```
Night gathers... and now my watch begins
```

In [40]:
```
# The way you import libraries is as follows

import numpy
import matplotlib
```

In [41]:
```
# You can also give nicknames to libraries
```

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

In [45]:
```
# In numpy we work with arrays,
# and there are a number of functions
# to create arrays and to operate on them

x = np.arange(0,1,0.01)
# Creates an array made out of equally spaced numbers
# betwen 0 and 1, with a step size of 0.01
```

In [46]:
```
x
```

Out[46]:
```
array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99])
```

In [49]:
```
x = np.linspace(0,99,100)
# Creates an array between 0 and 100
# containing 100 equally spaced elements
```

In [50]:
```
x
```

Out[50]:
```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
       13., 14., 15., 16., 17., 18., 19., 20., 21., 22., 23., 24., 25.,
       26., 27., 28., 29., 30., 31., 32., 33., 34., 35., 36., 37., 38.,
       39., 40., 41., 42., 43., 44., 45., 46., 47., 48., 49., 50., 51.,
       52., 53., 54., 55., 56., 57., 58., 59., 60., 61., 62., 63., 64.,
       65., 66., 67., 68., 69., 70., 71., 72., 73., 74., 75., 76., 77.,
       78., 79., 80., 81., 82., 83., 84., 85., 86., 87., 88., 89., 90.,
       91., 92., 93., 94., 95., 96., 97., 98., 99.])
```

In [56]:
```
# In Python there are arrays, lists, dictionaries

# If you write with brackets, you create lists

This_is_a_list = ['One', 'Two','Six']

print(This_is_a_list)

Another_list = [1,2,3,4,5]

print(Another_list)

Yet_another_list = [1.0,2.0,3.0,4.0,5.0]

Yet_another_list
```

```
['One', 'Two', 'Six']
[1, 2, 3, 4, 5]
```

Out[56]:
```
[1.0, 2.0, 3.0, 4.0, 5.0]
```

In [57]:
```
# To call elements from a list you use brackets

Another_list[0]
```

Out[57]:
```
1
```

In [59]:
```
for i in range(len(This_is_a_list)):
    print(This_is_a_list[i])

for i in range(len(Yet_another_list)):
    print(Yet_another_list[i])
```

```
One
Two
Six
1.0
2.0
3.0
4.0
5.0
```

In [63]:
```
# To check a variable type we use the 'type' function
# which returns a string naming the variable type

print(type(1.0))
print(type(1))
print(type(Another_list))
print(type(Another_list[0]))

# The en
```

```
<class 'float'>
<class 'int'>
<class 'list'>
<class 'int'>
```

In [70]:
```
# Now you can use a list whose entries are ints or floats
# and pass it as an argument on the function numpy.array
# to turn it into a numerical array

array1 = np.array(Another_list)
print(type(array1))
print(array1)
print(type(array1[0]))
array2 = np.array(Yet_another_list)
print(array2)
print(type(array2))
print(type(array2[0]))
```

```
<class 'numpy.ndarray'>
[1 2 3 4 5]
<class 'numpy.int64'>
[1. 2. 3. 4. 5.]
```

```
<class 'numpy.ndarray'>
<class 'numpy.float64'>
```

In [72]:
```
array3 = array1+array2
print(array3)
print(type(array3[0]))
```

```
[ 2.  4.  6.  8. 10.]
<class 'numpy.float64'>
```

In [82]:
```
# Matrices are basically arrays of arrays

A = np.array([[1.0,1.0],[0.0,1.0]])
A

v = np.array([2.0,3.0])

# This is one way of doing vector matrix multiplication
print(A @ v)

# Also from the right (if the shape of the matrix vector is right)

print(v @ A)

# Also A.T gives you the transpose

print(A.T @ v)
```

```
[5. 3.]
[2. 5.]
[2. 5.]
```

In [85]:
```
# There are many functions to create matrices
# For example

D = np.diag([2,3,5])
D

# For more on the *many* ways of creating matrices
# check out the SciPy lectures
```

Out[85]:
```
array([[2, 0, 0],
       [0, 3, 0],
       [0, 0, 5]])
```

In [86]:
```
# To call the elements of a 2D array you can use two brackets

A[0][0]
```

Out[86]:
```
1.0
```

In [93]:
```
# or commas

print(A[0,0])
print(D[1,1])
```

```
# If you use the operator * with two arrays
# of equal 'shape' you get multiplication component wise:

x = np.arange(0,10,1)
y = np.arange(10,20,1)
print(x)
print(y)
print(x*y)

# Compare with

print(x@y)
# (this returns the inner or dot product between the two vectors)
```

```
1.0
3
[0 1 2 3 4 5 6 7 8 9]
[10 11 12 13 14 15 16 17 18 19]
[  0  11  24  39  56  75  96 119 144 171]
735
```

In [95]:

```
# A few more facts about arrays

# They have a length, called by the function len

len(x)

# They also have a shape, called by the numpy function np.shape

print(np.shape(x))
print(np.shape(D))
```

```
(10,)
(3, 3)
```

In [101…

```
# Let's talk about other libraries
# The most important one for us besides numpy and matplotlib
# is the SciPy library, it has several submodules
# that solve differential equations, generate random numbers,
# solve nonlinear equations, optimize functions, and solve linear systems

import scipy as scipy
```

In [96]:

```
# There is a quick reference you can use here on Python,
# simply write down the name of a library, or of a function followed by a '?'

# Example:

np?
```

In [99]:

```
# You can also do it for functions

np.arange?
np.linspace?

# Great way to refresh details about syntax
```

In [102…
```
# Let's explore the Scipy module

scipy?
```

In [103…
```
import scipy.stats as sts
```

In [107…
```
sts?
```

In [125…
```
# Having seen the list of functions in sts
# We note there are many functions that can generate random variables
# Let us learn aboutnthe stats.uniform function

sts.uniform?
```

In [124…
```
unif = sts.uniform(0,1)
x1 = unif.rvs(size=5)
x2 = unif.rvs(size=5)
x3 = unif.rvs(size=5)
print(x1)
print(x2)
print(x3)
```

```
[0.62565905 0.4886034  0.76027947 0.34072465 0.77202621]
[0.45425626 0.82224203 0.05620318 0.80289511 0.28711439]
[0.83379528 0.74920448 0.28920344 0.03465239 0.26951783]
```

In [ ]:

In [ ]: