Introduction to Python Programming II

Theo Narh (Ph.D)

Graduate School of Nuclear & Allied Sciences (SNAS)
University of Ghana

Outline

Functions

Defining Your Own Functions I

- We can define our own functions in Python and reuse them throughout the program. The syntax for defining a function is as follows:

```
1 def functionName(parameters):
2     code detailing what the function should do
3     return [expression]
```

- There are two keywords here, def and return.
- **def**

  tells the program that the indented code from the next line onwards is part of the function.
- **return**

  is the keyword that we use to return an answer from the function. There can be more than one return statements in a function.

Defining Your Own Functions II

- However, once the function executes a return statement, the function will exit. If your function does not need to return any value, you can omit the return statement. Alternatively, you can write return or return None.
- Suppose we want to determine if a given number is a prime number. Here's how we can define the function using the modulus (%) operator

```
1    def checkIfPrime(numberToCheck):
2      for x in range(2, numberToCheck):
3        if (numberToCheck%x == 0):
4          return False
5      return True
```

- To use this function, we type

```
answer = checkIfPrime(13)
```

Variable Scope I

- An important concept to understand when defining a function is the concept of variable scope. Variables defined inside a function are treated differently from variables defined outside. There are two main differences.
- Firstly, any variable declared inside a function is only accessible within the function. These are known as local variables. Any variable declared outside a function is known as a global variable and is accessible anywhere in the program.

UNIVERSITY OF GHANA

Variable Scope II

- To understand this, try the code below:

```
1  def foo():
2      y = "local"
```

- then RUN this command::

```
1  foo()
2  print(y)
```

- Output::

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

Variable Scope III

- The output shows an error because we are trying to access a local variable y in a global scope whereas the local variable only works inside foo() or local scope.
- Create a variable outside of a function, and use it inside the function

```
1  x = "awesome"
2  def myfunc():
3    print("Python is " + x)
4
5  myfunc()
```

- Output::

```
Python is awesome
```

Variable Scope IV

- Create a variable inside a function, with the same name as the global variable

```
1  x = "awesome"
2  def myfunc():
3    x = "fantastic"
4    print("Python is " + x)
5
6  myfunc()
7  print("Python is " + x)
```

- Output::

```
Python is fantastic
Python is awesome
```

Python modules

UNIVERSITY OF GHANA

SymPy I

1. Basic operations
   - Evaluate $\cos(x) + 1$ at $x = 0$.

```
1  from sympy import *
2  x = symbols("x")
3
4  expr = cos(x) + 1
5  print(expr.subs(x, 0))
```

   - Ans: 2

UNIVERSITY OF GHANA

SymPy II

2. Sums and Products:: $\sum_{i=1}^{4} x + iy$, $\prod_{i=0}^{5} x + iy$

```
>>> from sympy import *
>>> init_printing(use_latex=True)

>>> x, y, i = symbols('x, y, i')
>>> summation(x + i*y, (i, 1, 4))  # Sum over i=1,2,3,4.
4*x + 10*y
>>> product(x + i*y, (i, 0, 5))  # Multiply over i=0,1,2,3
x*(x + y)*(x + 2*y)*(x + 3*y)*(x + 4*y)*(x + 5*y)
```

3. Linear Algebra

$$x + y + z = 5$$
$$2x + 4y + 3z = 2$$
$$5x + 10y + 2z = 4$$

SymPy IV

```
>>> x, y, z = symbols('x y z')

>>> # Define the augmented matrix M = [A|b].
>>> M = Matrix([ [1, 1, 1, 5],
... [2, 4, 3, 2],
... [5, 10, 2, 4] ])

>>> # Solve the system by::
>>> solve_linear_system(M, x, y, z)
    98      -45
{x: --, y: ----, z: 2/11}
    11       11
```

SymPy V

4. Trigonometry

```
>>> x = Symbol('x')

>>> expr = sin(2*x) + cos(2*x)

>>> expand_trig(expr)
                          2
2*sin(x)*cos(x) + 2*cos (x) - 1

>>> trigsimp(cos(x)**2 + sin(x)**2)
1
```

SymPy VI

5. Factorise

```
>>> x, y = symbols('x y')
>>> factor(x**2 -y**2)
(x - y)*(x + y)
```

SymPy VII

6. simplify

```
>>> x, y = symbols('x y')
>>> simplify((x**3 + x**2 - x -1) / (x**2 + 2*x +1))
x - 1
>>> simplify((x**4 -1) / (x-1))
 4
x  - 1
------
x - 1
```

SymPy VIII

7. expand

```
>>> x, y = symbols('x y')
>>> expand((x + 2*y)**3)
 3      2         2       3
x  + 6*x *y + 12*x*y  + 8*y
```

SymPy IX

8. Change of subject: $a = \frac{b+cd}{e}$

```
>>> a, b, c, d, e  = symbols('a b c d e')
>>> expr1 = Eq(a, (b + c*d)/e)
>>> expr1
     b + c*d
a = -------
       e

>>> # make d the subject
>>> expr2 = Eq(d, solve(expr1, d)[0])
>>> expr2
     a*e - b
d = -------
       c
```

Numpy I

- This package provides basic routines for manipulating large arrays and matrices of numeric data.
- There are several ways to import NumPy. The standard approach is to use a simple import statement:

```
import numpy
import numpy as np
from numpy import *
```

1. **Arrays**: Arrays are similar to lists in Python, except that every element of an array must be of the same type, typically a numeric type like float or int. Arrays make operations with large amounts of numeric data very fast and are generally much more efficient than lists.

Numpy II

1. Create an array $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$.

```
>>> import numpy as np
>>> A = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> A
[[1. 2. 3.]
 [4. 5. 6.]]
```

2. Select the **first** element in A.

```
>>> A[0,0]
1.0
```

3. Select the **last** row in A.

```
>>> A[1,:]
[4. 5. 6.]
```

UNIVERSITY OF GHANA

Numpy III

4. Select the **second** column in A.

```
>>> A[:,1]
[2. 5.]
```

5. Select the **first two** columns in A.

```
>>> A[:,:2]
 [[1. 2.]
 [4. 5.]]
```

6. Transpose A.

```
>>> A.transpose()
 [[1. 4.]
  [2. 5.]
 [3. 6.]]
```

UNIVERSITY OF GHANA

Numpy IV

7. Convert A into binary string (i.e., not in human-readable form) and revert.

```
>>> s = A.tostring()
>>> s
b'\x00\x00\x00\x00\x00\x00\xf0?\x00\x00\x00\x00\x00\x00\x00
00\x00\x08@\x00\x00\x00\x00\x00\x00\x10@\x00\x00\x00\x00\x0
x00\x00\x00\x00\x18@'

>>> np.fromstring(s)
[1. 2. 3. 4. 5. 6.]
```

2. Concatenate $\boldsymbol{B}$ and $\boldsymbol{C}$:

$$B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad C = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Numpy V

```
>>> B = np.array([[1, 2], [3, 4]], float)
>>> C = np.array([[5, 6], [7,8]], float)

>>> np.concatenate((B,C))
 [[1. 2.]
  [3. 4.]
  [5. 6.]
 [7. 8.]]
>>>
>>> np.concatenate((B,C), axis=0)
 [[1. 2.]
  [3. 4.]
  [5. 6.]
 [7. 8.]]
```

```
>>> np.concatenate((B,C), axis=1)
 [[1. 2. 5. 6.]
 [3. 4. 7. 8.]]
```

3. Other ways to create arrays
   1. `arange` function

```
>>> np.arange(5, dtype=float)
[0. 1. 2. 3. 4.]

>>> np.arange(1, 6, 2, dtype=int)
[1 3 5]
```

Numpy VII

2. `newaxis` function

```
>>> a = np.array([1, 2, 3], float)
>>> a
[1. 2. 3.]
>>> a[:,np.newaxis]
[[1.]
 [2.]
 [3.]]
>>> a[:,np.newaxis].shape
(3, 1)
```

Numpy VIII

3. `zeros and ones like` functions

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a
[[1. 2. 3.]
 [4. 5. 6.]]
>>> np.zeros_like(a)
[[0. 0. 0.]
 [0. 0. 0.]]
>>> np.ones_like(a)
[[1. 1. 1.]
 [1. 1. 1.]]
```

Numpy IX

4. array mathematics function

```
>>> a = np.array([1,2,3], float)
>>> b = np.array([5,2,6], float)
>>> a + b
[6. 4. 9.]
>>> a - b
[-4.  0. -3.]
>>> a % b
[1. 0. 3.]
>>> b**a
[  5.   4. 216.]
```

Numpy X

5. array iteration

```
>>> a = np.array([1, 4, 5], int)
>>> for x in a:
...     print(x)
... pass
  File "<stdin>", line 3
    pass
    ^
SyntaxError: invalid syntax
```

Numpy XI

6. array basic operation

```
>>> import numpy as np
>>> v = np.array([2, 4, 3], float)
>>> v.sum()
9.0
>>> v.prod()
24.0
>>> v.mean()
3.0
>>> v.var()
0.6666666666666666
>>> v.min()
2.0
>>> v.max()
4.0
>>> v = np.array([[0, 2], [3, -1], [3, 5]], float)
>>> v.mean(axis=0)
```

UNIVERSITY OF GHANA

Numpy XII

```
17  array([2., 2.])
18  >>> v.mean(axis=1)
19  array([1., 1., 4.])
20  >>> v.min(axis=1)
21  array([ 0., -1.,  3.])
22  >>> v.max(axis=0)
23  array([3., 5.])
24  >>> v = np.array([6, 2, 5, -1, 0], float)
25  >>> sorted(v)
26  [-1.0, 0.0, 2.0, 5.0, 6.0]
27  >>> v.sort()
28  >>> v = np.array([1, 1, 4, 5, 5, 5, 7], float)
29  >>> np.unique(v)
30  array([1., 4., 5., 7.])
31  >>> v = np.array([[1, 2], [3, 4]], float)
32  >>> v.diagonal()
33  array([1., 4.])
```

Visualization with Matplotlib I

- Plot $\sin(x)$, $x \in [0, 10]$

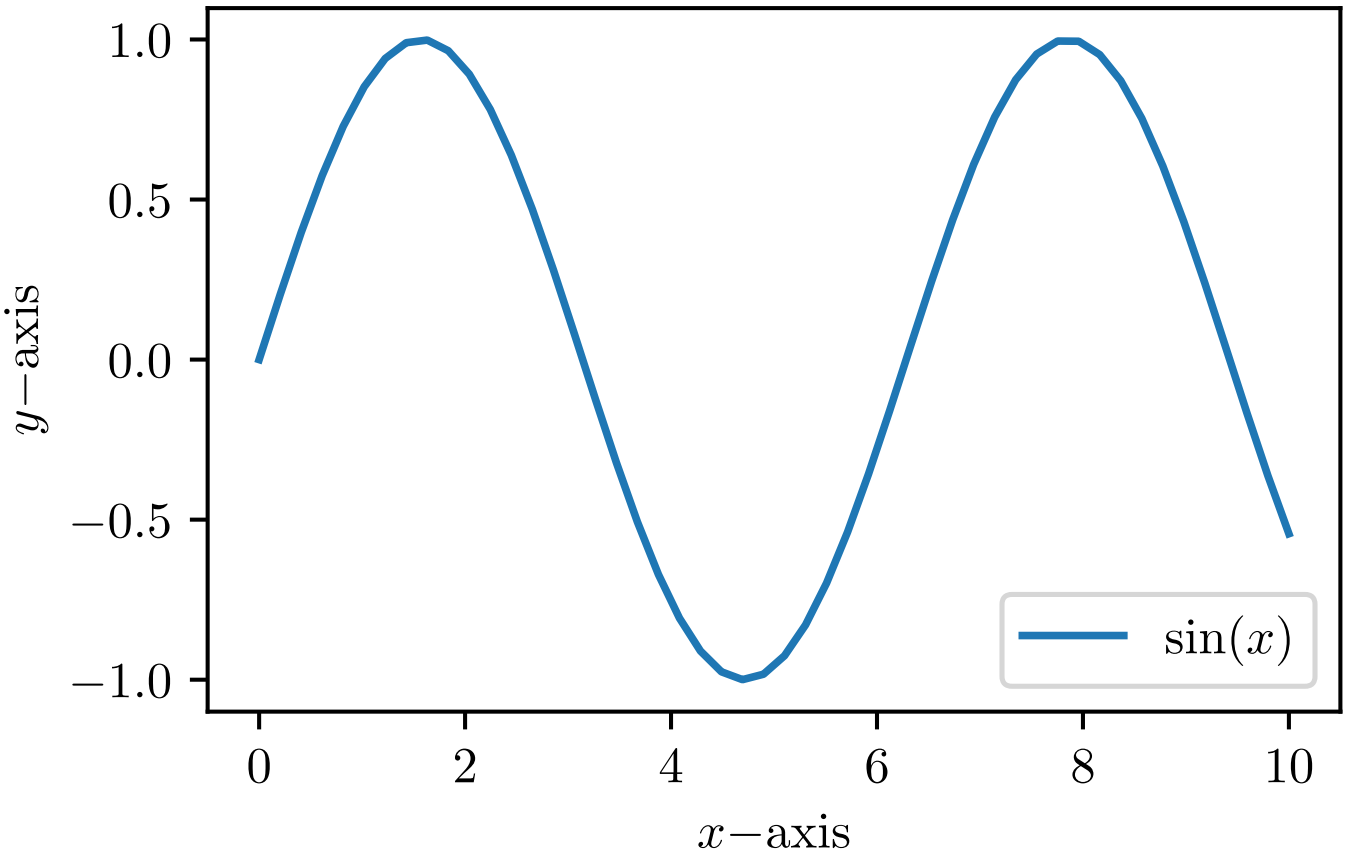
```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  plt.rc('text', usetex=True)
5  plt.rc('font', family='serif')
6  plt.rc('font', size=10.0)
7  plt.rc('legend', fontsize=10.0)
8  plt.rc('font', weight='normal')
9  x = np.linspace(0, 10)
10 plt.figure(figsize=(4, 2.5))
11 plt.plot(x, np.sin(x), label='$\sin(x)$')
12 plt.xlabel(r'$x\mathrm{-axis}$')
13 plt.ylabel(r'$y\mathrm{-axis}$')
14 plt.legend(loc='lower right')
```

UNIVERSITY OF GHANA

Visualization with Matplotlib II

```
15  plt.savefig('myplot1.pdf', bbox_inches='tight')
16
17  # Include the plot in the current LaTeX document
18  print(r"\begin{center}")
19  print(r"\includegraphics[width=0.75\textwidth]{myplot1.pdf
20  print(r"\end{center}")
```

Visualization with Matplotlib III



Visualization with Matplotlib IV

- plot $\cos(2\pi t)e^{-t}$, $t \in [0, 5]$

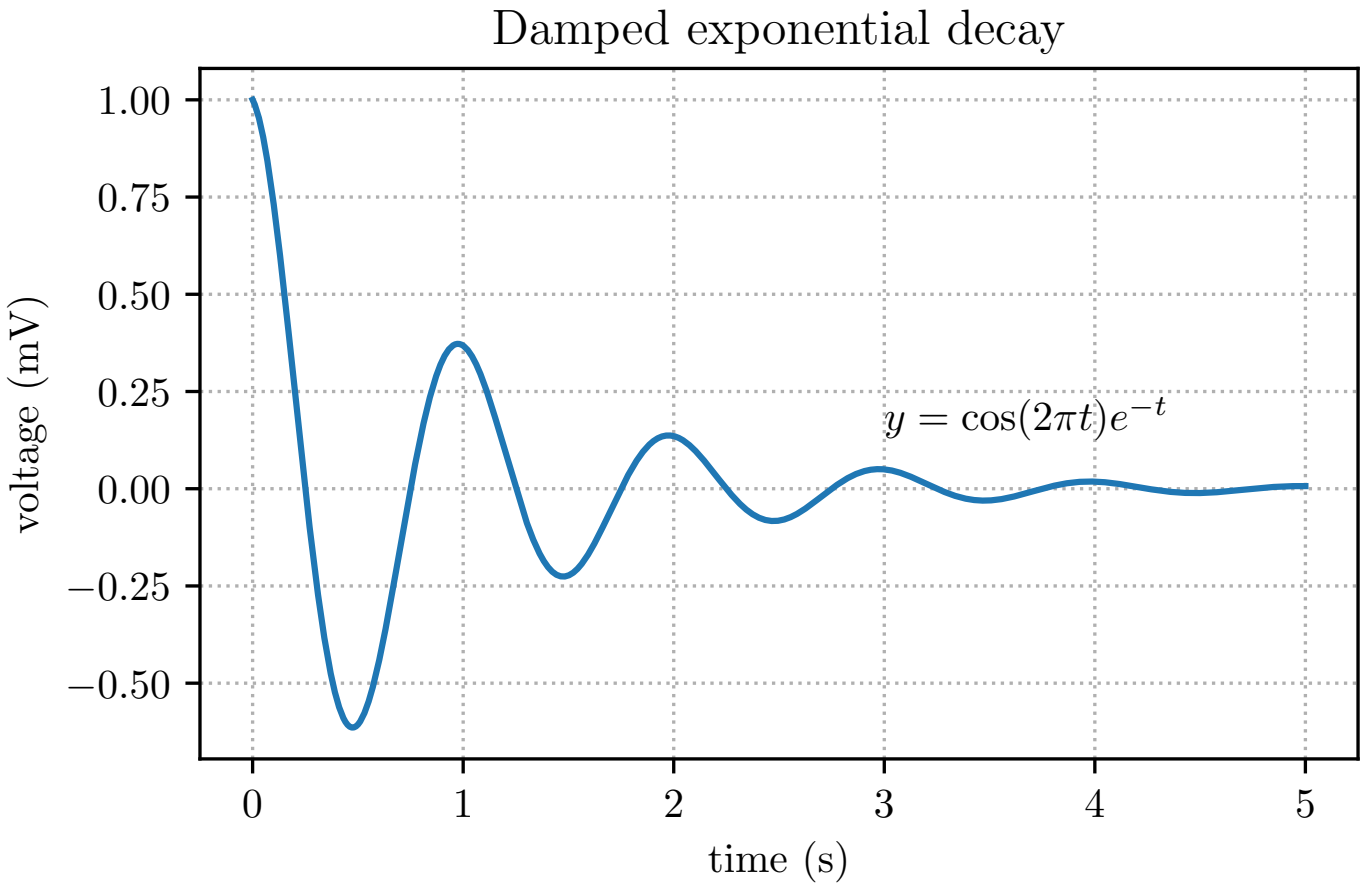
```
1  import numpy as np
2  from matplotlib.pylab import plt
3  # Define f(t), the desired function to plot
4  def f(t):
5      return np.cos(2 * np.pi * t) * np.exp(-t)
6  # Generate the points (t_i, y_i) to plot
7  t = np.linspace(0, 5, 500)
8  y = f(t)
9  # Begin with an empty plot, 5 x 3 inches
10 plt.clf()
11 plt.figure(figsize=(5, 3))
12 # Use TeX fonts
13 plt.rc("text", usetex=True)
14
```

UNIVERSITY OF GHANA

Visualization with Matplotlib V

```
15  # Generate the plot with annotations
16  plt.plot(t, y)
17  plt.title("Damped exponential decay")
18  plt.text(3, 0.15, r"$y = \cos(2 \pi t) e^{-t}$")
19  plt.xlabel("time (s)")
20  plt.ylabel("voltage (mV)")
21  plt.grid(linestyle='dotted')
22  # Save the plot as a PDF file
23  plt.savefig("myplot.pdf", bbox_inches="tight")
24  # Include the plot in the current LaTeX document
25  print(r"\begin{center}")
26  print(r"\includegraphics[width=0.75\textwidth]{myplot.pdf}")
27  print(r"\end{center}")
```

Visualization with Matplotlib VI



UNIVERSITY OF GHANA

Visualization with Matplotlib VII

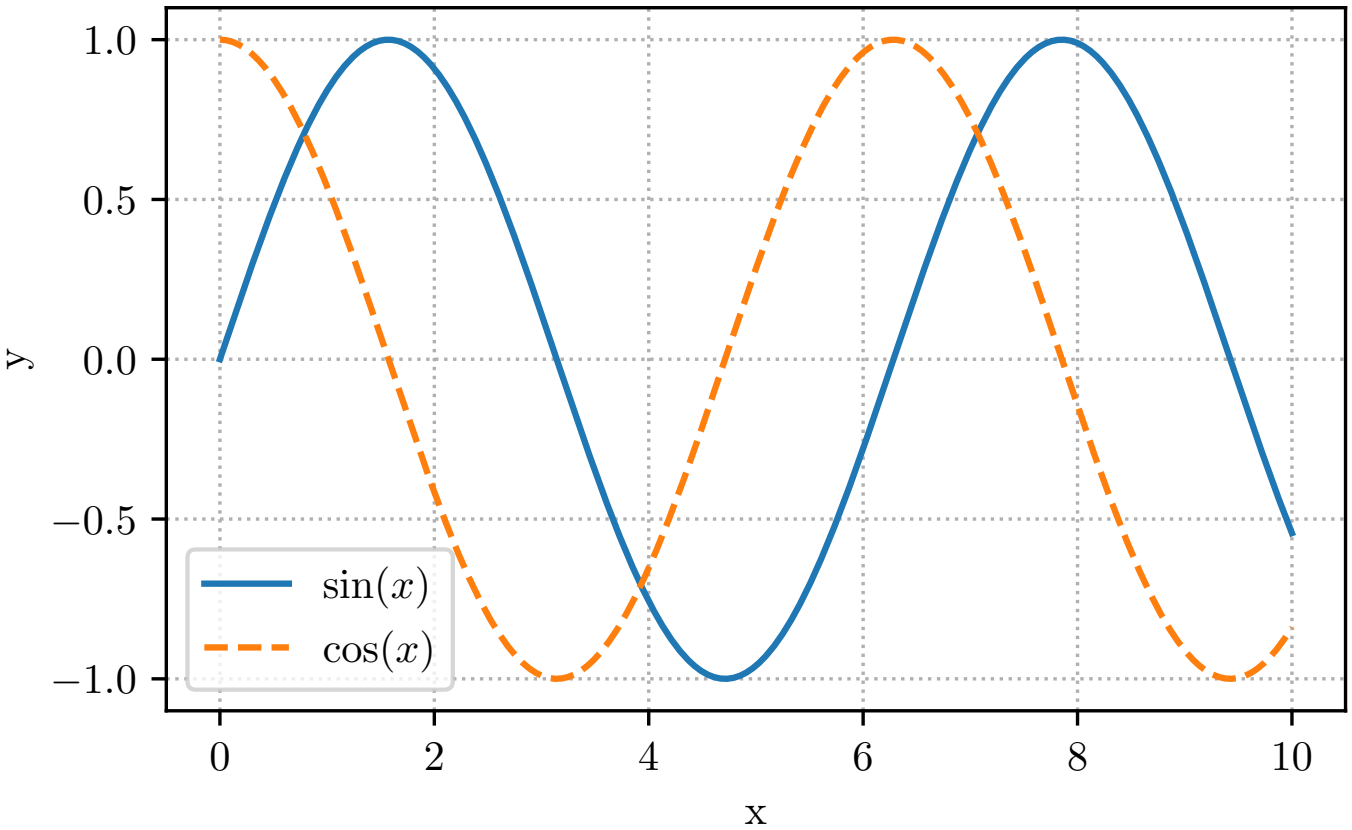
- Making multiple plots on a single figure.

```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  x = np.linspace(0, 10, 500)
5  plt.clf()
6  plt.figure(figsize=(5, 3))
7  # Use TeX fonts
8  plt.rc("text", usetex=True)
9
10 plt.plot(x, np.sin(x), '-', label=r'$\sin(x)$')
11 plt.plot(x, np.cos(x), '--', label=r'$\cos(x)$');
12 plt.xlabel("x")
13 plt.ylabel("y")
14 plt.grid(linestyle='dotted')
```

Visualization with Matplotlib VIII

```
15  plt.legend(loc='best')
16
17  # Save the plot as a PDF file
18  plt.savefig("myplot2.pdf", bbox_inches="tight")
19  # Include the plot in the current LaTeX document
20  print(r"\begin{center}")
21  print(r"\includegraphics[width=0.75\textwidth]{myplot2.pdf
22  print(r"\end{center}")
```

Visualization with Matplotlib IX



Visualization with Matplotlib X

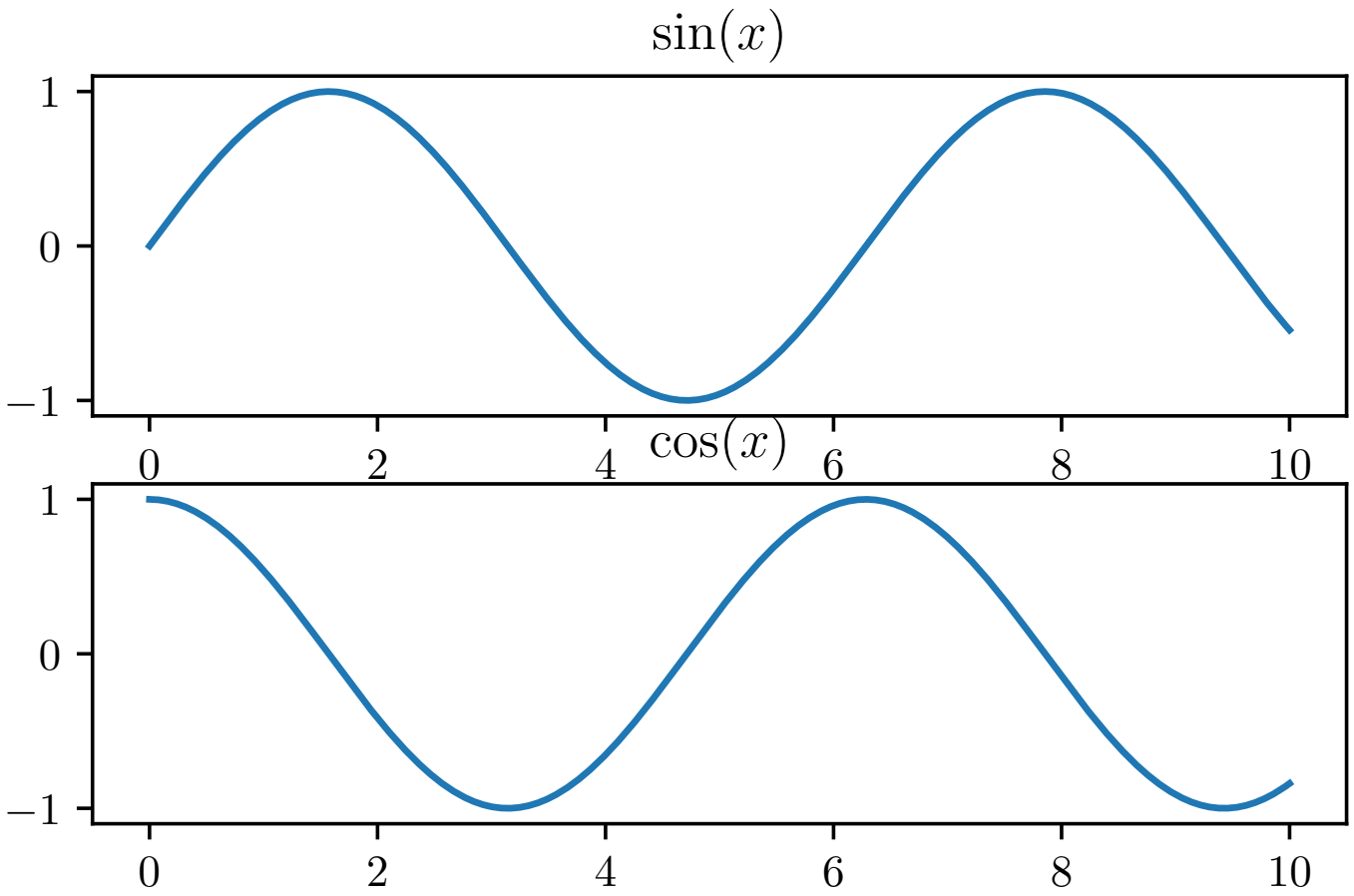
- Making multiple plots on different figures.

```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  x = np.linspace(0, 10, 500)
5  plt.clf()
6  plt.figure(figsize=(5, 3))
7  # Use TeX fonts
8  plt.rc("text", usetex=True)
9
10 # create the first of two panels and set current axis
11 plt.subplot(2, 1, 1) # (rows, columns, panel number)
12 plt.plot(x, np.sin(x))
13 plt.title(r"$\sin (x)$")
14
```

Visualization with Matplotlib XI

```
15  # create the second panel and set current axis
16  plt.subplot(2, 1, 2)
17  plt.plot(x, np.cos(x));
18  plt.title(r"$\cos (x)$")
19
20  # Save the plot as a PDF file
21  plt.savefig("myplot3.pdf", bbox_inches="tight")
22  # Include the plot in the current LaTeX document
23  print(r"\begin{center}")
24  print(r"\includegraphics[width=0.75\textwidth]{myplot3.pdf
25  print(r"\end{center}")
26
```

Visualization with Matplotlib XII



Visualization with Matplotlib XIII

- Create a scatter plot.

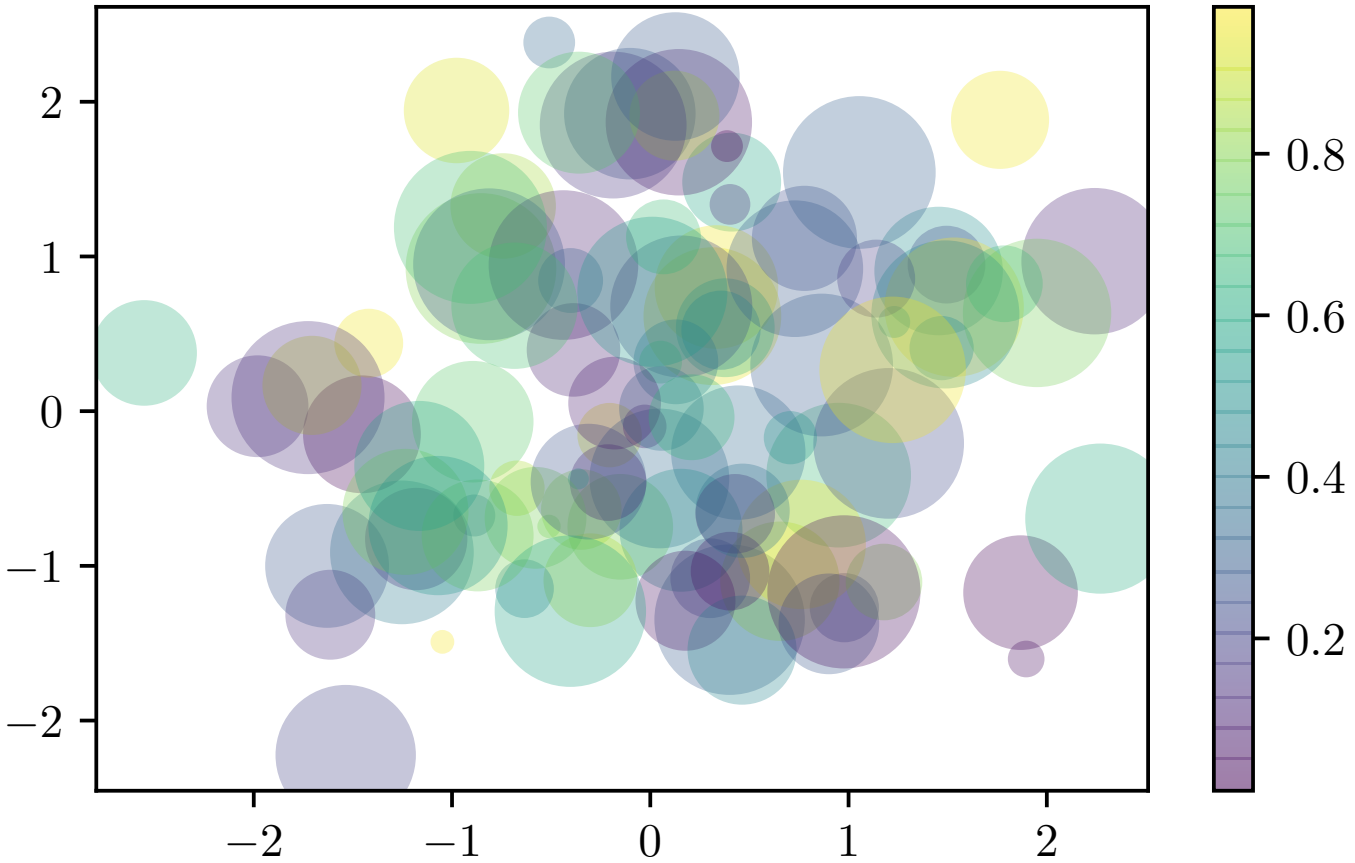
```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  plt.clf()
5  plt.figure(figsize=(5, 3))
6  # Use TeX fonts
7  plt.rc("text", usetex=True)
8
9  # create the scatter plot
10
11 rng = np.random.RandomState(0)
12 x = rng.randn(100)
13 y = rng.randn(100)
14 colors = rng.rand(100)
```

UNIVERSITY OF GHANA

Visualization with Matplotlib XIV

```
15  sizes = 1000 * rng.rand(100)
16
17  plt.scatter(x, y, c=colors, s=sizes, alpha=0.3,
18  cmap='viridis')
19  plt.colorbar();  # show color scale
20
21  # Save the plot as a PDF file
22  plt.savefig("myplot4.pdf", bbox_inches="tight")
23  # Include the plot in the current LaTeX document
24  print(r"\begin{center}")
25  print(r"\includegraphics[width=0.75\textwidth]{myplot4.pdf
26  print(r"\end{center}")
27
```

Visualization with Matplotlib XV



Visualization with Matplotlib XVI

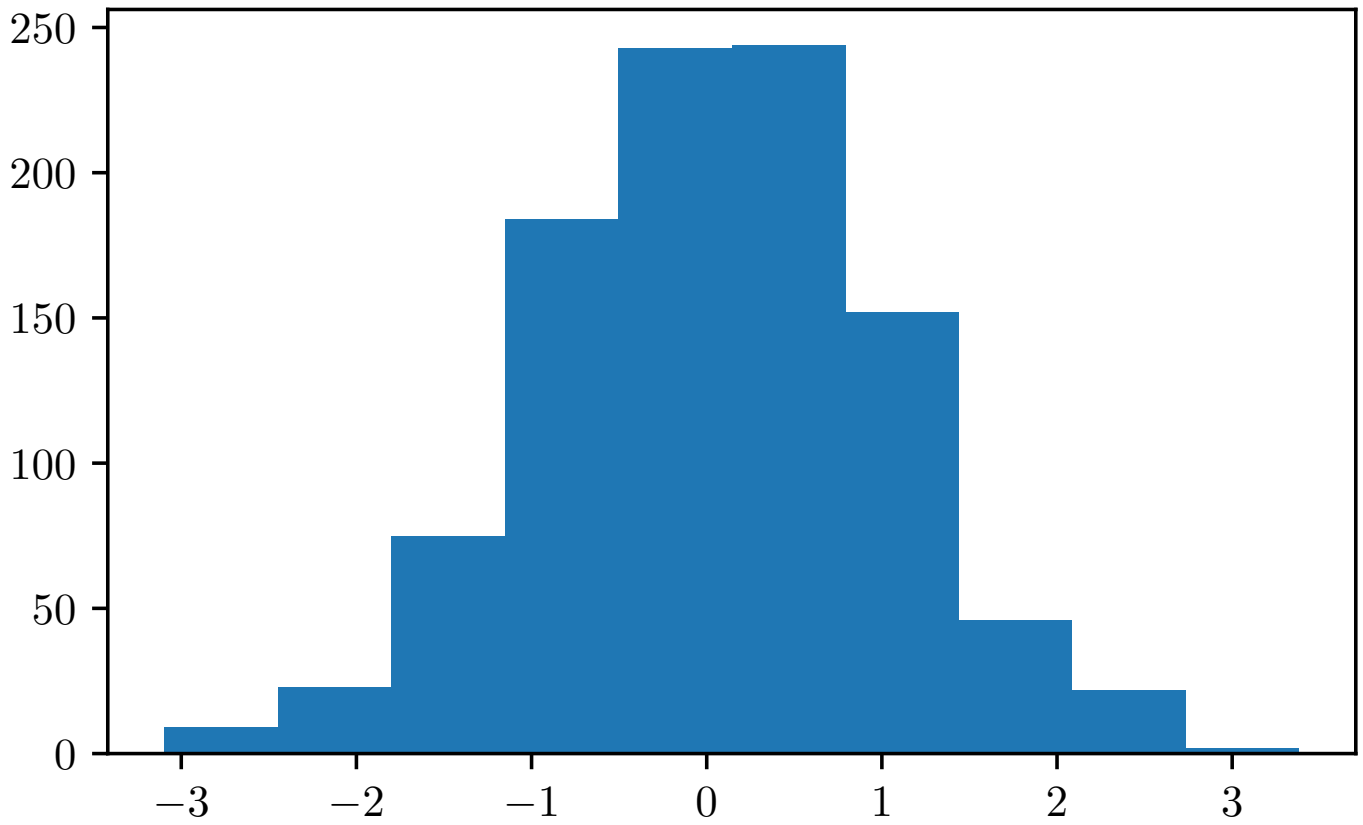
- Create histograms.

```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  plt.clf()
5  plt.figure(figsize=(5, 3))
6  # Use TeX fonts
7  plt.rc("text", usetex=True)
8
9  # create the histogram
10
11 data = np.random.randn(1000)
12 plt.hist(data)
13
14 # Save the plot as a PDF file
```

Visualization with Matplotlib XVII

```
15 plt.savefig("myplot5.pdf", bbox_inches="tight")
16 # Include the plot in the current LaTeX document
17 print(r"\begin{center}")
18 print(r"\includegraphics[width=0.75\textwidth]{myplot5.pdf
19 print(r"\end{center}")
20
```

Visualization with Matplotlib XVIII



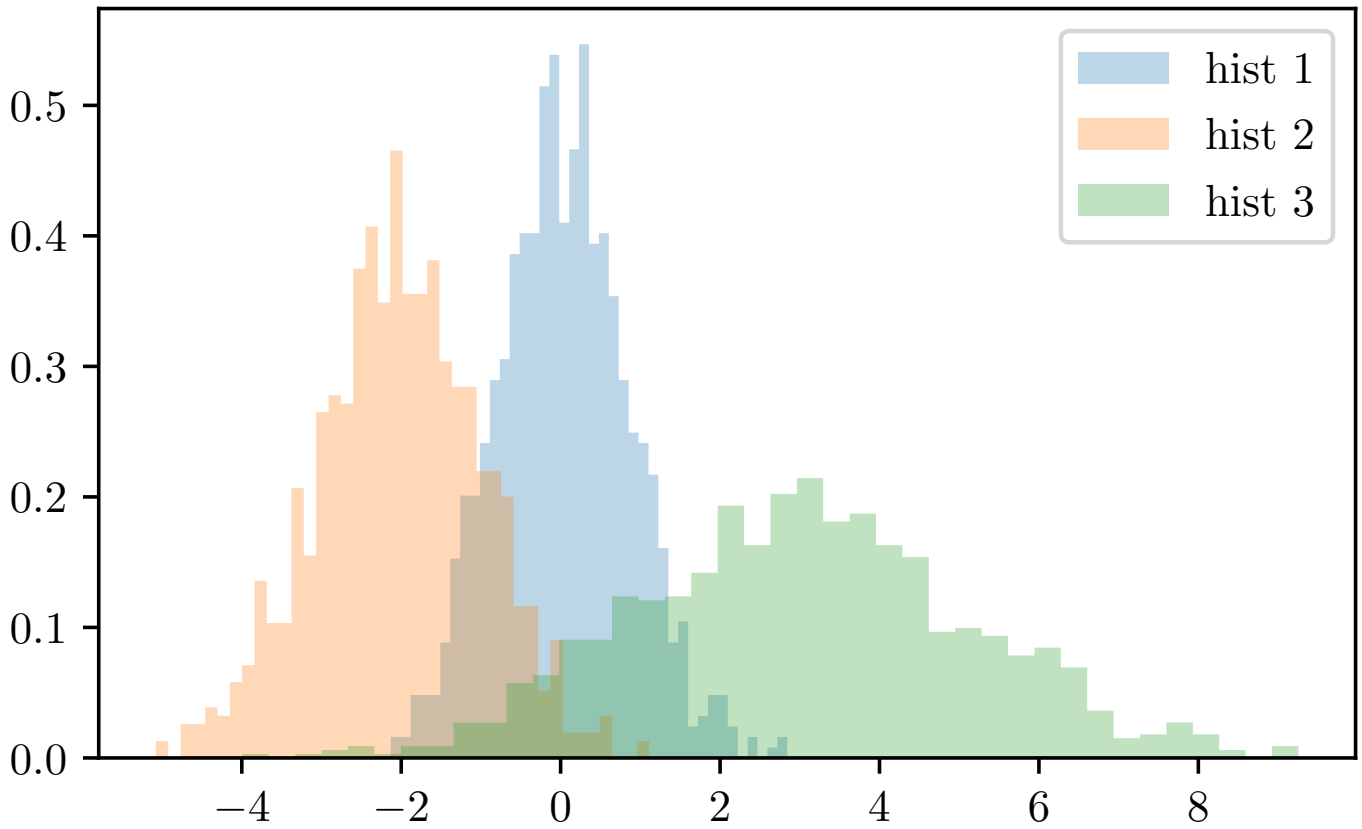
UNIVERSITY OF GHANA

Visualization with Matplotlib XIX

```
1  import numpy as np
2  from matplotlib.pylab import plt
3
4  plt.clf()
5  plt.figure(figsize=(5, 3))
6  # Use TeX fonts
7  plt.rc("text", usetex=True)
8
9  # create the histogram
10
11 x1 = np.random.normal(0, 0.8, 1000)
12 x2 = np.random.normal(-2, 1, 1000)
13 x3 = np.random.normal(3, 2, 1000)
14
15 kwargs = dict(histtype='stepfilled', alpha=0.3, density=Tr
```

```
16
17  plt.hist(x1, **kwargs, label=r"hist 1")
18  plt.hist(x2, **kwargs, label=r"hist 2")
19  plt.hist(x3, **kwargs, label=r"hist 3");
20  plt.legend(loc=0)
21  # Save the plot as a PDF file
22  plt.savefig("myplot7.pdf", bbox_inches="tight")
23  # Include the plot in the current LaTeX document
24  print(r"\begin{center}")
25  print(r"\includegraphics[width=0.75\textwidth]{myplot7.pdf
26  print(r"\end{center}")
```

Visualization with Matplotlib XXI



la fin! la fin! la fin!

UNIVERSITY OF GHANA