SCIENTIFIC PYTHON

Dr. Ram Prasad K
VisionCog R&D

ram.krish@visioncog.com
https://www.visioncog.com

Scientific Python

- *Numpy*
- *Matplotlib*

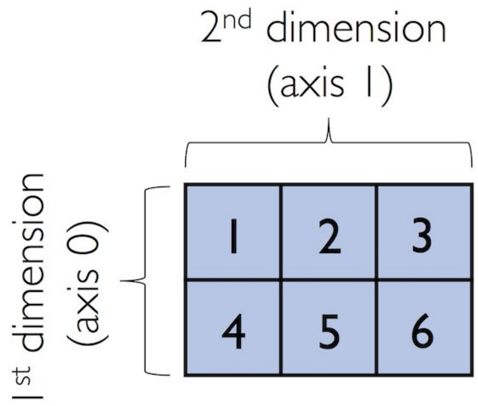
Scientific Python

N-dimensional array

```python
import numpy as np

a = np.array([1, 2, 3])    # Create a rank 1 array

print(type(a))             # Prints "<class 'numpy.ndarray'>"
print(a.shape)             # Prints "(3,)"

print(a[0], a[1], a[2])    # Prints "1 2 3"

a[0] = 5                   # Change an element of the array
print(a)                   # Prints "[5, 2, 3]"


b = np.array([[1,2,3],[4,5,6]])     # Create a rank 2 array

print(b.shape)                      # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])    # Prints "1 2 4"
```

High performance multi-dimensional array data structures

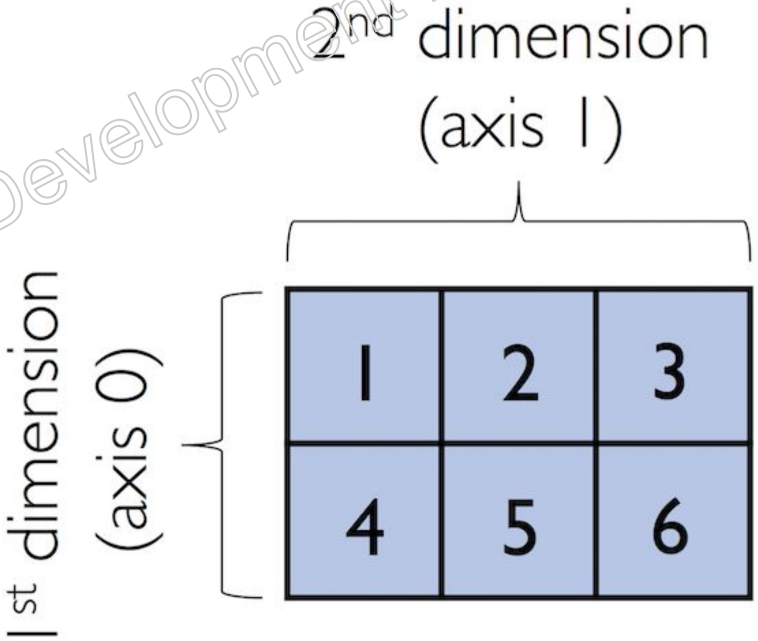
Mostly implemented in C language



Scientific Python

N-dimensional array

```
l = [[1, 2, 3], [4, 5, 6]]  # list of lists
ary2d = np.array(l)         # Convering list to array

print(ary2d)
# [[1 2 3]
#  [4 5 6]]

print(ary2d.dtype)
# Prints "int64"

float32_ary = ary2d.astype(np.float32)
# Converting the type of array

print(float32_ary)
# Prints "[[1. 2. 3.]
#          [4. 5. 6.]]"

print(ary2d.shape)
# Prints "(2,3)"
```



Scientific Python

Array construction routines

```
import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)         # Create a 2x2 identity matrix
print(d)              # Prints "[[ 1.  0.]
                      #          [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]
                             #               [ 0.68744134  0.87236687]]"
```

Useful as placeholders

We also get an initialized array

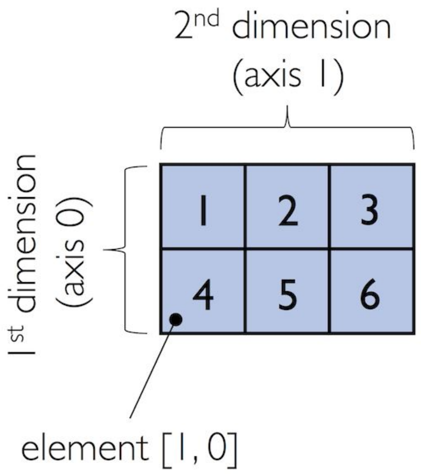
Scientific Python

Array slicing

```python
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"

b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]

print(a[0, 1])   # Prints "77"
```

Sliced result is actually a pointer to the original array.

Modifying the sliced result will modify the original array.



Scientific Python

Boolean array indexing

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2)   # Find the elements of a that are bigger than 2;
                     # this returns a numpy array of Booleans of the same
                     # shape as a, where each slot of bool_idx tells
                     # whether that element of a is > 2.

print(bool_idx)      # Prints "[[False False]
                     #          [ True  True]
                     #          [ True  True]]"


# We use boolean array indexing to construct a rank 1 array
# consisting of the elements of a corresponding to the True values
# of bool_idx
print(a[bool_idx])  # Prints "[3 4 5 6]"

# We can do all of the above in a single concise statement:
print(a[a > 2])     # Prints "[3 4 5 6]"
```

Scientific Python

Array maths

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))
```

These operations are known as vectorized operations.

Scientific Python

Array maths

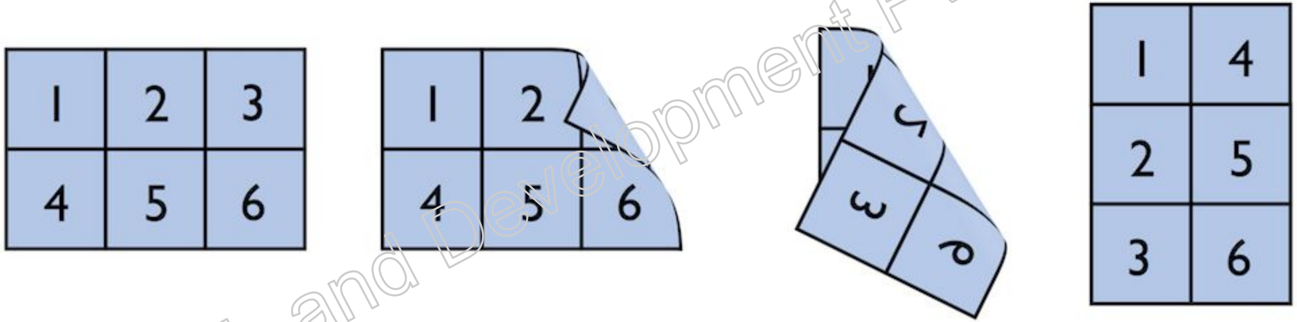
```
# Elementwise product; both produce the array
# [[ 5.0 12.0]
#  [21.0 32.0]]
print(x * y)
print(np.multiply(x, y))

# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print(x / y)
print(np.divide(x, y))

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print(np.sqrt(x))
```

These operations are known as vectorized operations.

Scientific Python

Array transpose

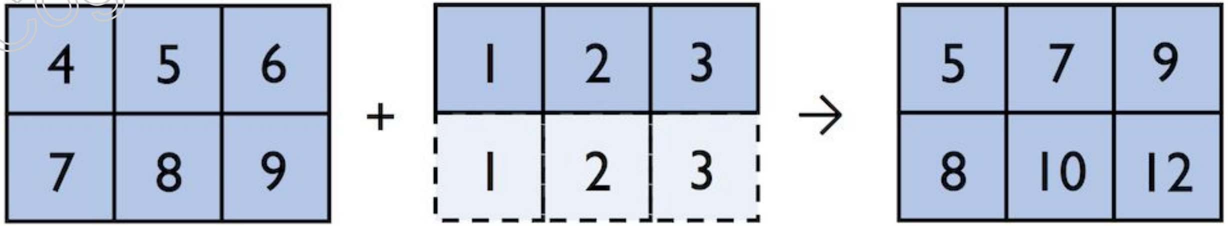
```python
import numpy as np

x = np.array([[1,2,3], [4,5,6]])

print(x)     # Prints "[[1 2 3]
             #          [4 5 6]]"

print(x.T)   # Prints "[[1 4]
             #          [2 5]
             #          [3 6]]"

# Note that taking the transpose of a rank 1 array does nothing:
v = np.array([1,2,3])

print(v)     # Prints "[1 2 3]"

print(v.T)   # Prints "[1 2 3]"
```



Scientific Python

Array broadcasting

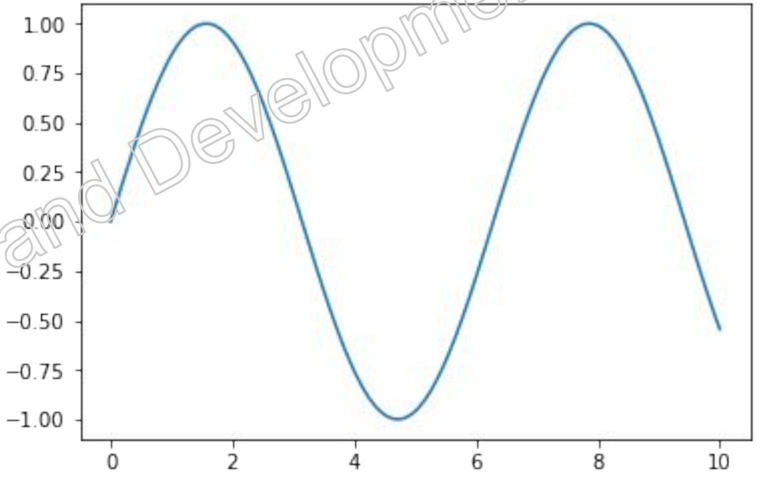
```
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[4,5,6], [7,8,9]])

v = np.array([1, 2, 3])

y = x + v  # Add v to each row of x using broadcasting

print(y)  # Prints "[[ 5  7  9]
          #          [ 8 10 12]]"
```



Scientific Python

Matplotlib

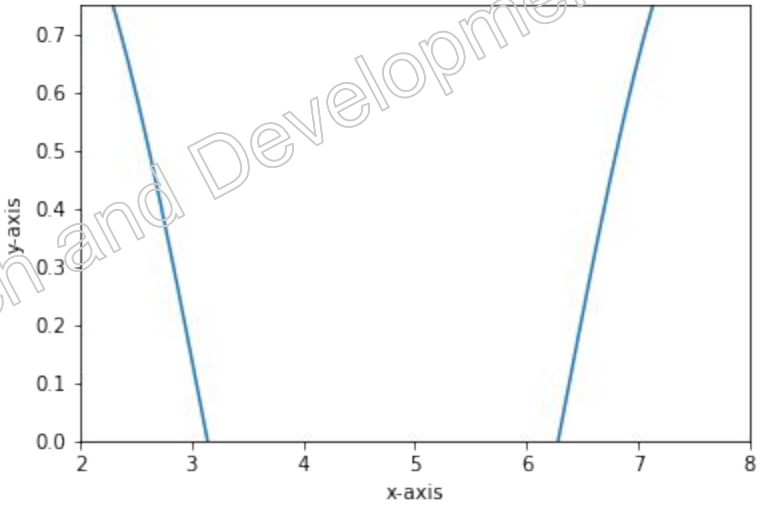
```
%matplotlib inline
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.show()
```



Scientific Python

Matplotlib

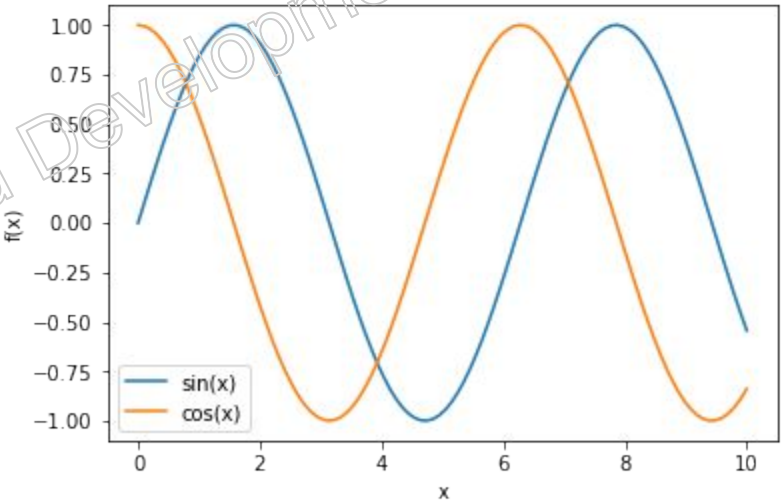
```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))

plt.xlim([2, 8])
plt.ylim([0, 0.75])

plt.xlabel('x-axis')
plt.ylabel('y-axis')

plt.show()
```



Scientific Python

Matplotlib

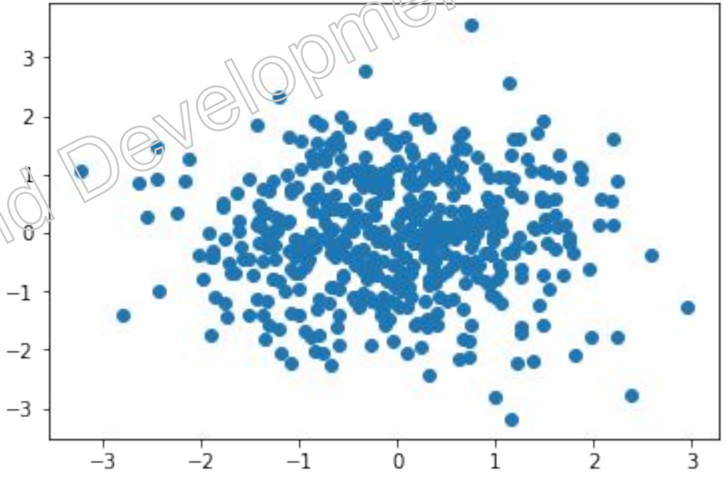
```
x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x), label=('sin(x)'))
plt.plot(x, np.cos(x), label=('cos(x)'))

plt.ylabel('f(x)')
plt.xlabel('x')

plt.legend(loc='lower left')
plt.show()
```



Scientific Python

Matplotlib

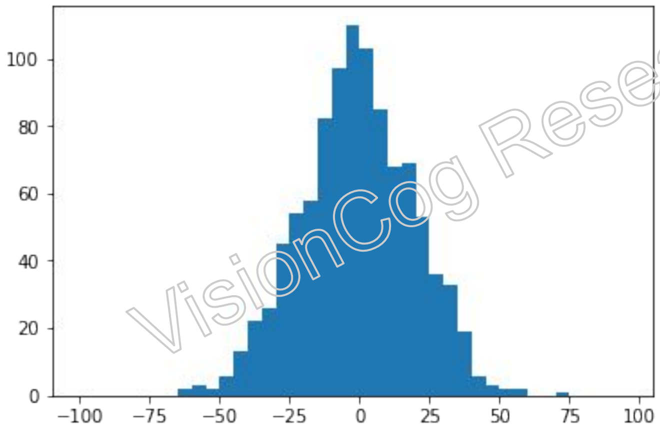
```
rng = np.random.RandomState(123)
x = rng.normal(size=500)
y = rng.normal(size=500)


plt.scatter(x, y)
plt.show()
```



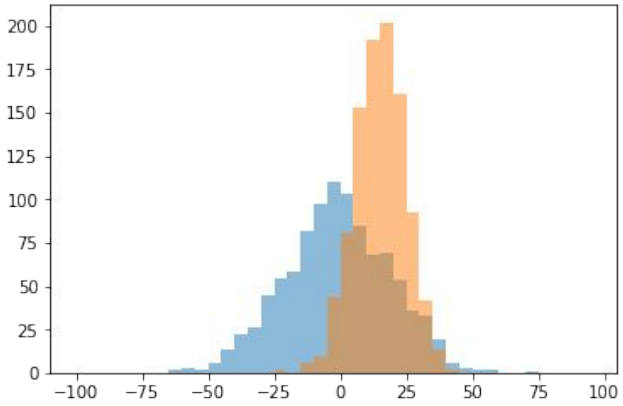
Scientific Python

Matplotlib

```
rng = np.random.RandomState(123)
x = rng.normal(0, 20, 1000)

# fixed bin size
bins = np.arange(-100, 100, 5) # fixed bin size

plt.hist(x, bins=bins)
plt.show()
```



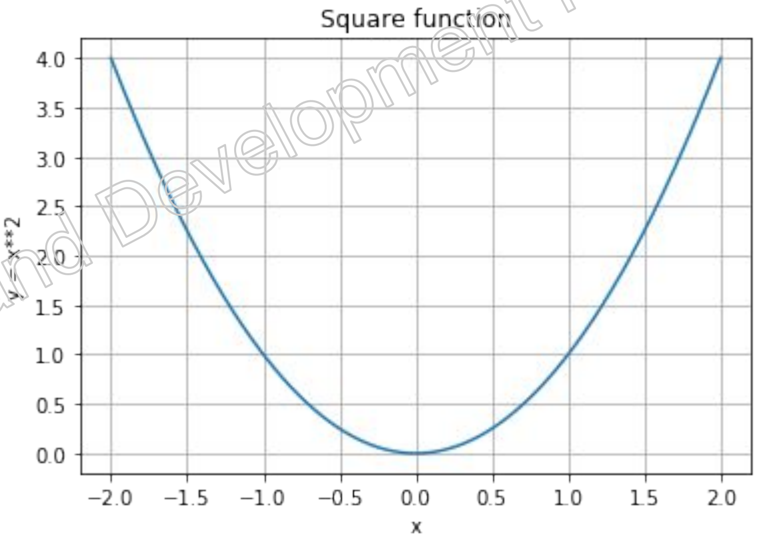

```
rng = np.random.RandomState(123)
x1 = rng.normal(0, 20, 1000)
x2 = rng.normal(15, 10, 1000)

# fixed bin size
bins = np.arange(-100, 100, 5) # fixed bin size

plt.hist(x1, bins=bins, alpha=0.5)
plt.hist(x2, bins=bins, alpha=0.5)
plt.show()
```



Scientific Python

Matplotlib

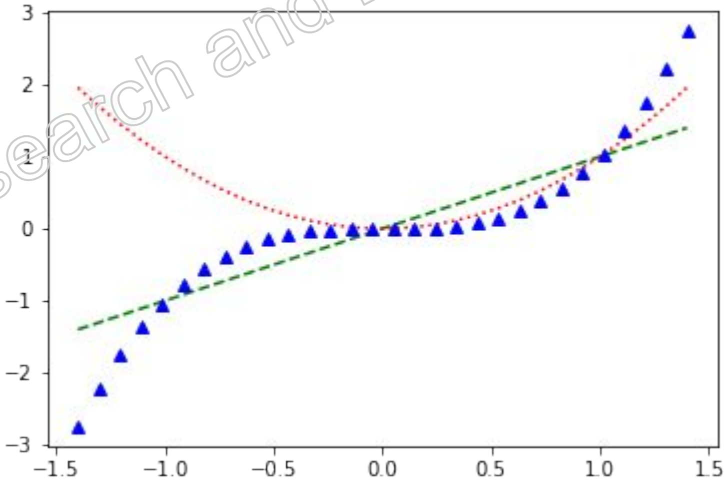
```
x = np.linspace(-2, 2, 500)
y = x**2

plt.plot(x, y)
plt.title("Square function")
plt.xlabel("x")
plt.ylabel("y = x**2")
plt.grid(True)
plt.show()
```



Scientific Python

Matplotlib

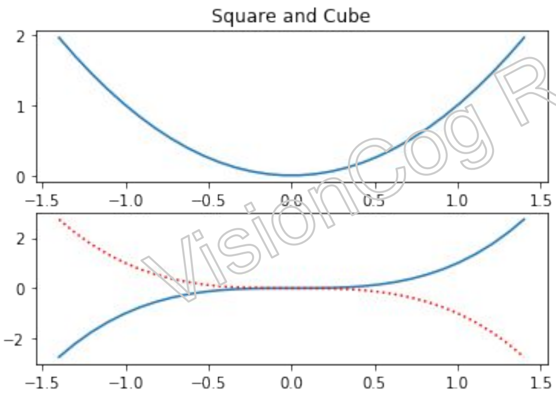
```
x = np.linspace(-1.4, 1.4, 30)
plt.plot(x, x, 'g--', x, x**2, 'r:', x, x**3, 'b^')
plt.show()
```



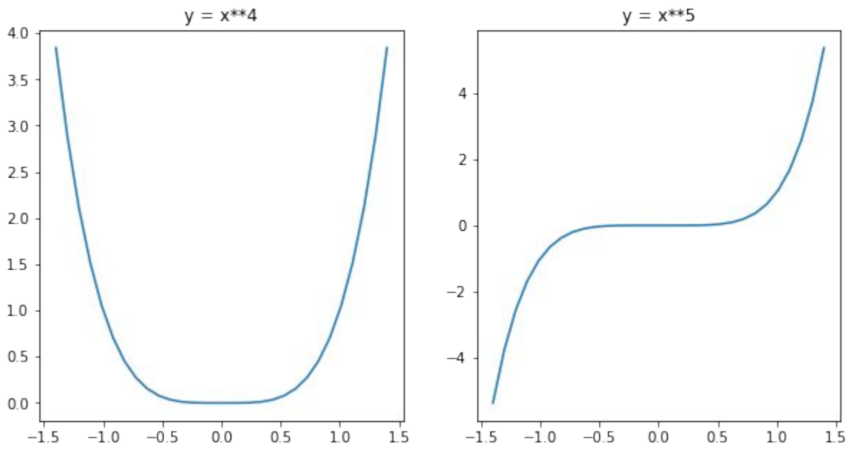
Scientific Python

Matplotlib

```
x = np.linspace(-1.4, 1.4, 30)

plt.figure(1)
plt.subplot(211)
plt.plot(x, x**2)
plt.title("Square and Cube")
plt.subplot(212)
plt.plot(x, x**3)
```



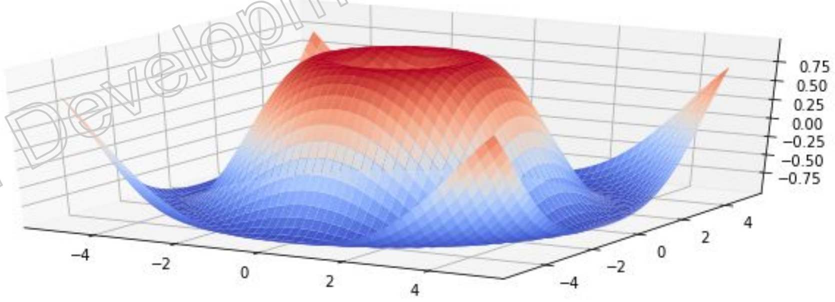

```
plt.figure(2, figsize=(10, 5))
plt.subplot(121)
plt.plot(x, x**4)
plt.title("y = x**4")
plt.subplot(122)
plt.plot(x, x**5)
plt.title("y = x**5")

plt.figure(1)      # back to figure 1, current subplot is 212 (bottom)
plt.plot(x, -x**3, "r:")

plt.show()
```



SCIENTIFIC PYTHON

Matplotlib

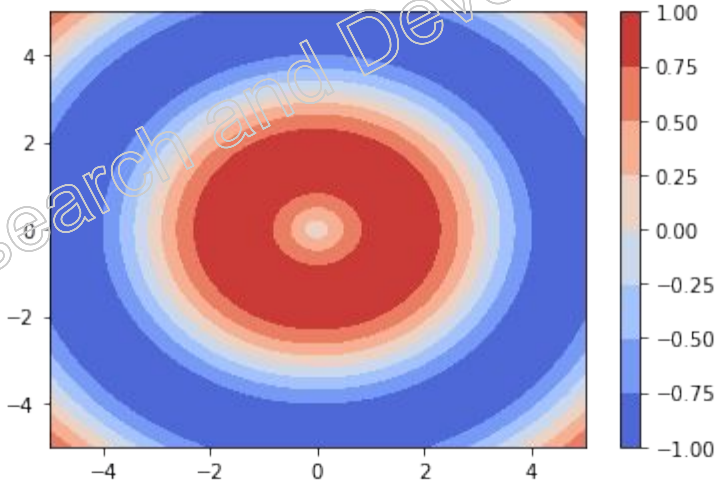
```
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

figure = plt.figure(1, figsize = (12, 4))
subplot3d = plt.subplot(111, projection='3d')
surface = subplot3d.plot_surface(X, Y, Z, rstride=1, cstride=1,
                                 cmap=matplotlib.cm.coolwarm, linewidth=0.1)
plt.show()
```



Scientific Python

Matplotlib

```
plt.contourf(X, Y, Z, cmap=matplotlib.cm.coolwarm)
plt.colorbar()
plt.show()
```



Open Exercise_02_ScientificPython.ipynb

Write appropriate code snippets to obtain the desired output shown as comments.