string, lists, array

Strings

Strings are lists of characters.

Strings can be assigned variable names like:

```
a = "My dog's name is"

b = 'Alpha'
```

Strings can be concatenated using the “+” operator:

```
c = a +' '+ b

Output of c is "My dog's name is Alpha"
```

Strings are used for for different purposes: labeling data in data files, labeling axes in plots, formatting numerical output, requesting input for your programs, as arguments in functions, etc.

Lists

In python we use square brackets '[ ]'to represent lists and commas ',' to separate the elements.
A example is :

```
bicycles = ['flower', 'rideon','dance']
```

Remember that the index of the list starts at 0 instead of 1

```
bicycles[0] = flower
bicycles[1] = rideon
```

When the index is specified as -1, we can access the last element of the list:

```
bicycles[-1] = dance
```

There can be different types of elements in the list

```
L = [1, 'a', max, 3 + 4j, 'Hello, world!']
```

There are various functions that you can use to manipulate elements in a list, which I've described in detail here.

Arrays

Array is the most convenient Python data structure.

Arrays can be one-dimensional or two-dimensional.

To creat a array, we have different methods :

```
Creat one-dimensional arrays

    import numpy as np
    a1 = np.ones(4)
    a2 = np.zeros(5)

    There are many other ways to do this, but these are just two examples.

Creat two-dimentional arrays:

    t = (2,3)
    a = np.ones( t )
    e = np.eye( t[1] )
    b = np.array( [ [2, 3, 5], [7, 11, 13] ] )

    Here are another way:

        min_value, max_value, step_size = 0, 10, 2
        i = np.arange(min_value, max_value + step_size, step_size)

        x_value = np.linspace(0, 1, 10)
        y_value = np.logspace(0, 1, 10)
        xy_value = np.array( [x_value, y_value] )

        x = np.array([0, 1, 2])
        y = np.array([0, 1])
        X, Y = np.meshgrid(x, y)

        r = np.random.random(t)
```

Array index is like list that strats from 0.

We can use a colon to iterate over the elements in a group:

```
r[0,-1]
r[::2,:]
r[::-1,1::2]
```

We can transpose the array:

```
Suppose a is a array, then a.T is its transpose.
```

Beyond that, there's a lot more to learn about arrays, which I won't go into here.

plot

To plot in python, we need to call a library: 'matplotlib.pyplot'
Then I`d like to show you a few uses of the library below:

```
import matplotlib.pyplot as plt

plt.plot(x, y, color = 'red')

    In this function, 'x' is the data on the X-axis,
    'y' is the data on the Y-axis, and 'color' can be used to set the color.

plt.title('It's a test plotting')

    This function is used to add a title for our plotting.

plt.xlable('x-axis')
plt.xlable('y-axis')

    These two functions are used to label the axes.

plt.xlim(-4,4)
plt.ylim(-7,7)

    These two functions define the range of the axes.

plt.axhline(y = 0, color = 'blue')

    Add horizontal lines in data coordinates.

plt.scatter(x, y, alpha=0.6, color='black')

    This function is used to plot scatter figure.

plt.contour(u, v, a*u**2 + b*u*v + c*v**2 + d*u + e*v + 1, [0])

    This is a function that plots contour lines,
    but it can also be used to plot conic curves.

plt,legend('points','contour')

    Add legends to the figure we plotted.

plt.show()

    Output the image we plotted.
```

numerical routines - Numpy

Numpy has a variety of uses, and in our codes we always import numpy as np.

Here, I`d like to use some examples to explain its USES.

```
import numpy as np
```

Before using numpy, we should import it.

Numpy allows us to do some math:

```
x0 = np.sin(1)
x1 = np.exp(1)
x2 = np.sqrt(5)
x3 = np.mean(5,7,9)
```

There are other uses here, which I won't go into here, that you might use in your programming.

We've already mentioned using numpy to build arrays in our discussion of arrays above, and I'll show you some other ways to build arrays here.

```
Suppose a is a array, we want to build a array named b
that has the same size as a but all elements of it are zero (one).

b = np.zeros_like(a)

b = np.ones_like(a)
```

Numpy is essential when using python for matrix operations.I'm going to introduce some of these functions here.

Suppose A and B are two matrices.

C = np.dot(A,B)

    Using np.dot() function we get a new matrix C, where C = AB.

A_diag = np.diag(A)

    When A is a 1-dimensional array, the result A_diag is a matrix
    with the elements of a 1-dimensional array as diagonal elements.

    When A is A two-dimensional matrix, the result A_diag is A
    one-dimensional array of diagonal elements of the matrix.

A_inv = np.linalg.inv(A)

    If the matrix A has an inverse, then you can use this function
    to get the inverse of the matrix A.

q,r = np.linalg.qr(A)

    Do the QR decomposition of matrix A.

np.linalg.norm(A)

    Find the norm of A.

np.allclose(A, B, rtol=epsilon)

    Compare each of the corresponding elements of the two arrays to
    see if they are identical within the error range.If it is the
    same, the result is TRUE and False.