Module 2:
Arrays

By Ninad Gaikwad

Reference - “Core Python Programming”
Dr. R. Nageshwara Rao
Dreamtech Press

Overview:

- Array is an object that stores a group of elements of the same datatype.
- They can increase or decrease their size dynamically
- Advantages of using arrays:
  - Can store only one type of elements
  - Use less memory than lists
  - Size is not fixed
  - Methods useful to process elements of array are available in array module

Creating an array

- Arrayname = array(type code, [elements])
- Eg
  - a = array('i', [4, 6, 2, 9])
  - arr = array('u', ['a','b','c','d'])
- Importing array module
  - import array
  - import array as ar
  - from array import *
- Typecode to create arrays:

Typecode	C Type	Minimum size in bytes
b	Signed integer	1
B	Unsigned Integer	1
i	Signed integer	2
I	Unsigned integer	2
l	Signed integer	4
L	Unsigned integer	4
f	Floating point	4
d	Double precision floating point	8
u	Unicode character	2

Indexing, Slicing and Functions of arrays

- Indexing and slicing of arrays
  - len(x) gives the size of array x
  - x[i] gives a i^{th} element of array
  - Arryname[start, stop, stride] useful to retrieve a range of elements

- Functions to process arrays
  - Methods: a.append(x), a.count(x), a.extend(x), a.index(x), a.fromfile(f,n), a.fromlist(list), a.fromstring(s), a.insert(i,x), a.pop(x), a.pop(), a.remove(x), a.reverse(), a.tofile(f), a.tolist(), a.tostring
  - Variables: typecode, itemsize
  - marks = [int(num) for num in str] # will create a numerical array from a string
  - Sort an array using bubble sort

Types of arrays

1. Single dimensional
   - Python supports only single dimensional arrays
   - Array contains only single row or single column
2. Multidimensional
   - Working with arrays using numpy
   - numpy is a package that contains several classes, functions, variables etc to deal with scientific calculations
   - Importing numpy:
     i.
```
import numpy
arr = numpy.array([10, 20, 30, 40])
print(arr)
```
     ii.
```
import numpy as np
arr = np.array([10, 20, 30, 40])
print(arr)
```
     iii.
```
from numpy import *
arr = array([10, 20, 30, 40])
print(arr)
```

Ways of creating arrays in numpy

1. Using array() function
   - Eg
     arr = array([10, 20, 30, 40])
     arr = array([‘a’, ‘b’, ‘c’, ‘d’])
     arr = array([‘a’, ‘b’, ‘c’, ‘d’], typedef= str)
2. Using Linspace() function
   - Linespace function produces equally spaced points
   - Eg linespace(start, stop, n)
     linespace(0, 10, 5)
   - Start, stop represent starting and end elements. ‘n’ represents the number of parts the elements should be divided, if omitted it is taken as 50.
3. Using Logspace() function
   - Logspace produces equally spaced points on a logarithmically spaced scale
   - Eg logspace(start, stop, n)
     b = logspace(1, 4, 5)
   - The above function starts with 10^1 to 10^4. These values are divided into five equal points and stored in b ie. b= [10.0 56.2 316.2 1778.3 10000.0]
   - Logspace starts with a value which is 10 to the power of start and ends in 10 to the power of stop, the values in between are divided into 5 equal parts

Ways of creating arrays in numpy

4. Using arrange() function
    - Creates an array with a group of elements from start to one element prior to stop in the steps of stepsize
    - Eg, arrange(1, 10, 3) creates array [1 4 7]
5. Using zeros() and ones() functions
    - zeros(n, datatype) function creates an array with n number of zeros and specified datatype
    - ones(n, datatype) function creates an array with n number of ones and specified datatype
    - datatype can be integer or float

Operations and Operators for arrays

- Mathematical operations on arrays
  - sin(arr), cos(arr), tan(arr), arcsin(arr), arccos(arr), arctan(arr), log(arr), abs(arr), sqrt(arr), power(arr, n), exp(arr), sum(arr), prod(arr), min(arr), max(arr), mean(arr), median(arr), var(arr), cov(arr), std(arr), argmin(arr), unique(arr), sort(arr), concatenated([a, b])

- Comparison and logical operators (return boolean values)
  - Comparing arrays: a==b, a>b, a<b # all give arrays with boolean type values
  - any(a) returns true if any of the elements in the array is true
  - all(a) returns true only if all the elements in the array are true
  - c = logical_and(a>0, a<4)
  - c = logical_or(a>0, a<4)
  - c = logical_not(a)
  - where(condition, expression1, expression2)
    Eg a = where(a%2==0,a,0)
    will write a values where the numbers are even else will write 0
  - nonzero(a) # gives the positions of elements with nonzero values

Aliasing, viewing and copying

- Aliasing
  - Giving another name to the same object
  - Eg. a = arrange(1, 6)
    b=a
- Viewing
  - Creating another array same as the existing for viewing purpose
  - Replicated copy will be dependent on the original
  - Also called as shallow copying
  - Eg b = a.view()
- Copying
  - Creates a new copy of an array which is independent of the original copy
  - Modifications in one does not trigger modifications in other copy
  - Also called as deep copying
  - Eg b = a.copy()

Slicing, Attributes and Methods

- Slicing and indexing in numpyArrays
  - arrayname [start: stop: stepsize]
  - arrayname[index]
- Attributes of an array
  - ndim: Gives the dimensions or the rank of an array
  - shape: Shape is a tuple listing the number of elements along each dimension
  - size: Gives the total number of elements in an array
  - itemsize: Memory size of array element in bytes.
  - dtype: Datatype of elements in an array
  - nbytes: Total number of bytes occupied by an array
- Other methods
  1. reshape()
     - Changes the shape of the array
     - Eg arr1= arrange(10)
       arr1 = arr1.reshape(2,5) # change shape to 2 rows and 5 cols
  2. flatten()
     - Returns a copy of array collapsed into one dimension
     - Eg arr1 = array([1, 2, 3], [4, 5, 6])
       arr1 = arr1.flatten()

Creating Multi-Dimensional arrays

- Using array() function
  - Numpy's array function can be used for creating multidimensional arrays
  - Eg arr1 = array([1, 2, 3], [4, 5, 6])
- Using ones() and zeros() function
  - Default datatype is float for the ones and zeros functions
  - Eg ones((3,4), int)
    zeros((3,4), float)
- Using eye() function
  - Creates an identity matrix (square matrix with diagonal elements one and others zero) of specified size
  - Eg eye(3)
    [1 0 0
     0 1 0
     0 0 1]
- Using reshape() function
  - Used to convert one dimensional array to multidimensional array
  - reshape(arrayname, (n, r, c))
    n = number of arrays
    r, c = number of rows and columns in each array
  - Eg a = arange(12)
    b = reshape(a, (2, 3, 2))

Indexing, Slicing in Multi-dimensional array

- Indexing arrayname(r, c) for two dimensional array, arrayname(n, r, c) for three dimensional arrays
- Slicing arrayname(rowrange, columnrange) for two dimensions
- Eg a = reshape(arrange(11, 36, 1),(5, 5))
  print(a)
  [[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]]

  print(a[2:4, 3:]) # will give
  [[24 25
  29 30]]

Matrix

- Matrix represents a rectangular array of elements arranged in rows and columns
- Eg a = matrix([1, 2, 3], [4, 5, 6]) # Numerical matrix
  Or matrix(“1 2 3; 4 5 6”) # String matrix
- a=diagonal(matrix) # Gives the diagonal elements of matrix
- big = a.max() # gives the biggest element of matrix
- small = a.min() # gives the smallest element of the matrix
- a.sum() # sum of all the elements
- a.mean() # average of all the elements
- a = m.prod(0) # Performs column wise multiplication fives one row as result
- a = m.prod(1) # Performs row wise multiplication fives one column as result
- a = sort(a, 0) # Each row Sorted
- a = sort(a, 1) # Each column Sorted, default axis
- m.transpose() OR m.getT() # Gives transpose of a matrix
- Matrix addition subtraction and division is performed on individual elements like m=a + b

Matrix

- Matrix multiplication (a*b) is performed scientifically, ie only if the number of columns in a is equal to the number of rows in b
- Random numbers
  - rand() function included in random module
  - Generates values from 0 to 1
  - Eg a = random.rand(5) will generate an array of 5 random numbers between 0 and 1