Lecture 10: ES 102
Arrays and Scientific Computing

Introduction to Computing

IIT Gandhinagar, India

October 22, 2013

Arrays

Array data structure (or simply an array) is a data structure consisting of a *c*ollection of elements (values or variables), each identified by at least one *a*rray index or key.

- 1D array $x = [1, 3, 5, 7, 9]$,$x = [1.5, 2.5, 0.8, 0.88]$
  lst=[“Apple”, “Orange”,“Banana”]

Arrays

Array data structure (or simply an array) is a data structure consisting of a *c*ollection of elements (values or variables), each identified by at least one *a*rray index or key.

- 1D array $x = [1, 3, 5, 7, 9]$,$x = [1.5, 2.5, 0.8, 0.88]$
  lst=[“Apple”, “Orange”,“Banana”]

- Accessing them : $x[0] = 1, x[4] = 9$; `lst[1]='Orange'`

Arrays

Array data structure (or simply an array) is a data structure consisting of a *c*ollection of elements (values or variables), each identified by at least one *a*rray index or key.

- 1D array $x = [1, 3, 5, 7, 9]$,$x = [1.5, 2.5, 0.8, 0.88]$
  lst=[“Apple”, “Orange”,“Banana”]

- Accessing them : $x[0] = 1, x[4] = 9$; `lst[1]='Orange'`

- Two Dimensional Arrays

$$a = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

$a_{23} \rightarrow a[1][2] = 5$

Arrays in Python

```
"""
Creating 1D and 2D Arrays in Python without Numpy
"""
# List as Array: 1D
a=[1,2,3,4]
print "a[0]=",a[0], "a[1]=",a[1]
print "2*a=",2*a

#2D array
a=[]  # Empty lists
for i in xrange(3):
      a.append([])  # List
      for j in xrange(3):
            a[i].append(i+j) # List of lists!

print a
print "a[1][2] is ", a[1][2]

# Direct Specification
lst=[[1,2,3],[4,5,6],[7,8,9]]

print "lst[1][2] is " , lst[1][2]
```

L10Programs/ArrayEx.py

Python For Scientific Computing

1. `SciPy` (pronounced Sigh Pie) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. It includes: `Numpy`,`SymPy`, `Matplotlib` (`http://scipy.org/`)

Python For Scientific Computing

1. `SciPy` (pronounced Sigh Pie) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. It includes: `Numpy`,`SymPy`, `Matplotlib` (`http://scipy.org/`)

2. `NumPy` is the fundamental package for scientific computing with Python.
   - Multi-dimensional array object
   - Many useful math functions
   - Linear algebra, Fourier transform, and random number capabilities

   `http://www.numpy.org/use: import numpy as np from numpy import *`

Python For Scientific Computing

1. `SciPy` (pronounced Sigh Pie) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. It includes: `Numpy`,`SymPy`, `Matplotlib` (`http://scipy.org/`)

2. `NumPy` is the fundamental package for scientific computing with Python.
   - Multi-dimensional array object
   - Many useful math functions
   - Linear algebra, Fourier transform, and random number capabilities

   `http://www.numpy.org/use: import numpy as np from numpy import *`

3. `Matplotlib` :is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. (`http://matplotlib.org/`)

NumPy 1D Array

```
# 1D arrays
from numpy import *
a=array([2,3,4,5])

print a
print a.dtype
print a.ndim
print len(a)
```

L10Programs/Array1D.py

NumPy 1D Array

```
# 1D arrays
from numpy import *
a=array([2,3,4,5])

print a
print a.dtype
print a.ndim
print len(a)
```

L10Programs/Array1D.py

```
# use of arange() function to create linearly spaced arrays
from numpy import *

x=arange(10)
print "\n \n" ,x

y=arange(0,10,0.5)
print "\n \n" ,y

z=arange(10,1,-0.5)
print "\n \n" ,y
```

L10Programs/arange_ex.py

NumPy 2D Array

```
# 2D Arrays in NumPy
from numpy import *

a=array([[2.5,3,8.5],[9,7,6]])
print a

print a.ndim
print a.shape

"""
Output:
[[ 2.5  3.   8.5]
 [ 9.   7.   6. ]]
2
(2, 3)
"""
```

L10Programs/Array2D.py

2

[2] http://wiki.scipy.org/Tentative_NumPy_Tutorial

Array Specification

```
#Specifying Type
from numpy import *

c = array([1, 2, 3], dtype=float)

c = array( [ [1,2], [3,4] ], dtype=complex )
print c
""" Output :
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
"""
```

L10Programs/ArraySpecs.py

Some common arrays

```
from numpy import *

a=zeros((2,2)) # 2x2 zeero matrix
print a

b=ones((3,2))  # 3x2 Ones matrox
print b

c=eye(3)   # 3x3 Identity matrix
print c
```

L10Programs/commonarrays.py

Random Numbers

```
#random numbers
from numpy import *

a=random.rand(5) # Creates 5 random numbers between [0.1] uniform
print a

b=random.randn(5) # 5 normally (Gaussian) with mean 0 and variance
print b
```

L10Programs/random_nos.py

Basic math

```
#Basic Arithmetic operations
#Arithmetic operators on arrays apply elementwise. A new array is c
from numpy import *
a=array([8,9,10,11])

b=array([3,5,7,9])

c=a-b
print "a=",a,"\nb=",b ,"\nc=a-b=",c

print "a*2", a*2  # Scalar multiplication
print "a**2", a**2  # Exponentiation on each element
print "\n Dot product a.b=", dot(a,b) # Dot product

print "sin(b)=", sin(b)  # math operation on each element
print "exp(b)=", exp(b)  # math operation on each element

print "elements where a<10=",a<10
```

L10Programs/ArithMath.py

Matrix operations

```
#Matrix operations
from numpy import *

A=array([[1,1],[2,3]])
print "A=", A

B=array([[3,4],[5,6]])
print "B=", B

print "A*B=", A*B  # This is not matrix multiplication ... each elem

#-----------------------------------------------------------------
#Accessing elements
a=([1,3,5,7,9,11])

print a
print "a[0]=", a[1], "a[1]=", a[1]

print "a[2:5]=", a[2:5]
#-----------------------------------------------------------------
```

L10Programs/matrixops.py

Matrix Addition

```
"""
Basic program for matrix addintion
"""
from numpy import *

def matadd(A,B):

    s1=A.shape
    s2=B.shape

    if (s1!=s2):
      print "Error! Dimensions of A and B do not match"
      return 0
    else :
      m=s1[0]
      n=s1[1]
      C=zeros((m,n))

      for i in range(m):
        for j in range(n):
          C[i,j]=A[i,j]+B[i,j]

      return C
```

Matrix Subtraction

```
"""
Basic program for matrix  subtraction
"""
from numpy import *

def matsub(A,B):

    s1=A.shape
    s2=B.shape

    if (s1!=s2):
      print "Error! Dimensions of A and B do not match"
      return 0
    else :
      m=s1[0]
      n=s1[1]
      C=zeros((m,n))

      for i in range(m):
        for j in range(n):
           C[i,j]=A[i,j]-B[i,j]

      return C

x= array([[1, -2, 2],[0, 3, 1],[-1, -1, 0]], dtype = float)
```

Matrix Multiplication

```
"""
Basic program for matrix multiplication
"""
from numpy import *

def matmul(A,B):

    s1=A.shape
    s2=B.shape
    m,n,p,q=s1[0],s1[1],s2[0],s2[1]

    if (n!=p):
      print "Error! Dimensions of A and B not compatible for multipl
      return 0
    else :
      C=zeros((m,q))

      for i in range(m):
        for j in range(q):
          C[i,j]=0
          for k in range(n):
            C[i,j]=C[i,j]+A[i,k]*B[k,j]

      return C
```

Problems set

see lecture 10 problem set