CSB303: Artificial Intelligence

Instructor : [Dr. Chandra Prakash]

- For more information visit the class website (https://cprakash86.wordpress.com/csl-303-artificial-intelligence/).

Tutorial 3 : PYTHON PROGRAMMING

PART 3.1: Introduction to Python Programming

- Welcome to your second lab session on Python!
- This tutorial contains some basic functions and syntax of Python, at the end of the tutorial some programming exercises are given to help you test your understanding on Python programming.
- Here we give you a very brief overview of Python programming with concepts and functions that will be used later in lab sessions and assignments.
- For detailed description of each concepts covered here, you may refer the official websites of Python (https://www.python.org/ (https://www.python.org/)) and NumPy (http://www.numpy.org/ (http://www.numpy.org/)).

Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python has become a common language for machine learning research and is the primary language for TensorFlow.

Python 3.0, released in 2008, was a significant revision of the language that is not entirely backward-compatible, and much Python 2 code does not run unmodified on Python 3. This course makes use of Python 3. Furthermore, TensorFlow is not compatible with versions of Python earlier than 3. A non-profit organization, the Python Software Foundation (PSF), manages and directs resources for Python development. On January 1, 2020, the PSF discontinued the Python 2 language and no longer provides security patches and other improvements. Python interpreters are available for many operating systems.

Python 3.x is the programming language that will be used for this class. Python, as a programming language, has the widest support for deep learning. The three most popular frameworks for deep learning in Python are:

- TensorFlow (https://www.tensorflow.org/) (Google)
- MXNet (https://github.com/dmlc/mxnet) (Amazon)
- CNTK (https://cntk.ai/) (Microsoft)

Some references on popular programming languages for AI/Data Science:

- Popular Programming Languages for AI (https://en.wikipedia.org/wiki/List_of_programming_languages_for_artificial_intelligence)
- Popular Programming Languages for Data Science (http://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html)

Python Resources

- Google CoLab (https://colab.research.google.com/) - Free web-based platform that includes Python, Juypter Notebooks, and TensorFlow [Cite:GoogleTensorFlow] (http://download.tensorflow.org/paper/whitepaper2015.pdf). No setup needed.
- Python Anaconda (https://www.continuum.io/downloads) - Python distribution that includes many data science packages, such as Numpy, Scipy, Scikit-Learn, Pandas, and much more.
- Juypter Notebooks (http://jupyter.org/) - Easy to use environment that combines Python, Graphics and Text.
- TensorFlow (https://www.tensorflow.org/) - Google's mathematics package for deep learning. .

Software Installation

This class is technically oriented. You needs to be compile and execute Python code for Machine learning Techniques. There are two options for you to accomplish this:

- Install Python, TensorFlow and some IDE (Jupyter, TensorFlow, and others)
- Use Google CoLab

The first module of this course provide an introduction to some aspects of the Python programming language. However, entire course focus on Python. This module one will not cover every detail of this language. The reader is encouraged to consult additional sources on the Python language.

Let's begin by printing Hello World.

Google CoLab Instructions

The following code ensures that Google CoLab is running the correct version of TensorFlow.

In [53]:

```
try:
    from google.colab import drive
    %tensorflow_version 2.x
    COLAB = True
    print("Hello World")
    print("Note: using Google CoLab")
except:
    print("Hello World")
    print("Note: not using Google CoLab")
    COLAB = False
```

```
Hello World
Note: using Google CoLab
```

In [54]:

```
# Single line comment (this has no effect on your program)
print("Hello World") # Say hello
```

```
Hello World
```

Python is a programming language that lets you work quickly and integrate systems more effectively. Some features of python programming are:

- A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python supports object-oriented programming with classes and multiple inheritances.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
- Python contains advanced programming features such as generators and list comprehensions.
- Python's automatic memory management frees you from having to manually allocate and free memory in your code.

1.1 Your first Python Code: Print a string - print() function displays the output on the console.

In [3]:

```
print ('Welcome to the NIT Delhi!')
```

```
Welcome to the NIT Delhi!
```

In [4]:

```
a=5
```

In [5]:

```
a
```

Out[5]:

```
5
```

In [6]:

```
a=5*4
```

In [7]:

```
a
```

Out[7]:

```
20
```

In [8]:

```
print("""Print
Multiple
Lines
""")
```

```
Print
Multiple
Lines
```

1.2 Input and assignment - input() function takes user's input.

Input Function

- You can use an the input function, just as you might have used 'scanf' in C.
- When the input function is called, the program flow will be stopped until you give an input and end it using the return key.
- The text of the optional parameter, i.e., the prompt, will be printed on the screen.
- The input of the user will be interpreted. If you input an integer value, the input function will return this integer value. If you input a list, the function will return the list.

Let's have a look at the following example:

In [9]:

```
name = input('What is your name?\n')
print ('Welcome ' + name + ' to the PYTHON PROGRAMMING @ AI Course!')
```

```
What is your name?
Bhavyesh
Welcome Bhavyesh to the PYTHON PROGRAMMING @ AI Course!
```

Important note: Python is indent sensitive.

- Python is case sensitive.
- Python is also indent sensitive.
- If you observe the 'for loop' and the 'if-else' statement, there is indent within the loop.
- For example if you run the following code, it will show "IndentationError".
- Run the cell to see the error and correct the code by putting indent.

In [13]:

```
for number in [1,10,20,30,40,50]:
print(number,"\n")

# Identify the Error and resolve
```

```
  File "<ipython-input-13-ebc0cd6996f1>", line 2
    print(number,"\n")
        ^
IndentationError: expected an indented block
```

- As you can see above it is giving IndentationError
- We can overcome this error by providing a single/multiple space or tab seperation

In [12]:

```
for number in [1,10,20,30,40,50]:
    print(number,"\n")
```

```
1

10

20

30

40

50
```

1.3 If Else Statement -

An if-else statement has the following syntax.

Syntax

if expression:

> statement(s)

else:

> statement(s)

In [14]:

```
num = input("number :")
if int(num) > 0:
    print("Positive number")
elif int(num) == 0:
    print("Zero")
else:
    print("Negative number")
```

```
number :10
Positive number
```

1.4 Loops in Python - We can iterate loop using various types. Here we are using built-in enumerate function.

In [15]:

```
names = ['NIT Delhi', 'JMI Delhi', 'IIT Delhi', 'IIITM Gwalior ']
print('Centrally Funded Technical Institute:')
for i, name in enumerate(names):
    print ("{iteration} : {name}".format(iteration=i+1, name=name))
```

```
Centrally Funded Technical Institute:
1 : NIT Delhi
2 : JMI Delhi
3 : IIT Delhi
4 : IIITM Gwalior
```

1.5 While Loop - Same as other programming languages

In [16]:

```
names = ['NIT Delhi', 'JMI Delhi', 'IIT Delhi', 'IIITM Gwalior ']
value=0
print('Centrally Funded Technical Institute:')
while value<len(names):
    print("{iteration} : {name}".format(iteration=value+1, name=names[value]))
    value+=1
```

```
Centrally Funded Technical Institute:
1 : NIT Delhi
2 : JMI Delhi
3 : IIT Delhi
4 : IIITM Gwalior
```

1.6. Defining Function - Just write def function_name(parameters)

Just as in C, Python also provides various types of functions. It gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

In the following some simple examples of functions are given.

In [17]:

```
def add(x,y):
    print ('Addition is :', x+y)

add(12,-10)
```

```
Addition is : 2
```

1.7 Dictionary - Dictionary is an associative array of python

As given in (https://docs.python.org/3/tutorial/datastructures.html (https://docs.python.org/3/tutorial/datastructures.html)) :

"Dictionary is a useful data type built into Python, similar to “associative memories” or “associative arrays” in some languages. A dictionary is indexed by keys, Unlike a sequence, which is indexed by a range of numbers. A Key can be any immutable type, strings and numbers can always be keys.

So a dictionary can be considered as an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). Each key is separated from its value by a colon (:), the items are separated by commas, and the entire unordered ke:value pair in enclosed within curly braces.

A dictionary can be initialized to be an empty dictionary by using a pair of braces : {}. Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary; this is also the way dictionaries are written on output.

Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key. You can't use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like append() and extend().

- The main operations on a dictionary are storing a value with some key and extracting the value given the key.
- To delete a key:value pair you can use 'del'. If you store a value using a key that is already in use, then the old value associated with that key is overwritten.
- Use list(d.keys()) to obtain a list of all the keys used in the dictionary, in arbitrary order.
- Use sorted(d.keys()) if you wanted it to be in a sorted order.
- To check whether a single key is in the dictionary, use the 'in' keyword.

Here are some examples of code pieces using dictionaries:

In [18]:

```
prices = {'apple': 100, 'banana': 50}
my_purchase = {'apple': input('apple :'),'banana': input('banana :')}
grocery_bill = sum(prices[fruit] * int(my_purchase[fruit])
                   for fruit in my_purchase)
print ('Total Bill : %.2f' % grocery_bill +' rupees')
```

```
apple :200
banana :50
Total Bill : 22500.00 rupees
```

1.8 List -

Python offers a range of compound datatypes often referred to as sequences.

List is one of the most frequently used datatypes used in Python.

- A list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.
- It can have any number of items and they may be of different types (integer, float, string etc.).

Accessing the elements of the list

You can use the index operator [ ] to access an item in a list. Index starts from 0. So, a list having 5 elements will have indices ranging from 0 to 4.

In [19]:

```
dynamic_languages = ['Python', 'Ruby', 'Groovy']
dynamic_languages.append('Lisp')
dynamic_languages
```

Out[19]:

```
['Python', 'Ruby', 'Groovy', 'Lisp']
```

1.9 Exception Handling - try and except blocks are used for exception handling

In [21]:

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

```
Please enter a number: h
Oops!  That was no valid number.  Try again...
Please enter a number: 6
```

Python Packages

In [22]:

```
import math
```

In [23]:

```
a = math.sqrt(100)
print(a)
```

```
10.0
```

In [24]:

```
a = math.pow(100, 0.5)
print(a)
```

```
10.0
```

In [25]:

```
x = 100
y = 1
for i in range(1, x):
  y *= i
print('Factorial of', x, 'is', y)
```

```
Factorial of 100 is 93326215443944152681699238856266700490715968264381
6214685929638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000
```

In [26]:

```
y = math.factorial(x)
```

In [27]:

```
import math as m
```

In [28]:

```
y = m.factorial(x)
```

In [29]:

```
from math import factorial
```

In [30]:

```
y = factorial(x)
```

In [31]:

```
import time

vals = list(range(1, 100))
tic = time.time()
for x in vals:
  y = 1
  for i in range(1, x):
    y *= i
toc = time.time()
print('Elapsed time in secs without own function', toc - tic)

tic = time.time()
for x in vals:
  y = math.factorial(x)
toc = time.time()
print('Elapsed time in secs with own function', toc - tic)
```

```
Elapsed time in secs without own function 0.0008814334869384766
Elapsed time in secs with own function 0.00012612342834472656
```

In [32]:

```
!echo 'def hello():' > my_first_module.py
!echo '    print("hello, i am living in a different file!!!")' >> my_first_module.py
```

In [33]:

```
!cat my_first_module.py
```

```
def hello():
    print("hello, i am living in a different file!!!")
```

In [34]:

```
import my_first_module
```

In [35]:

```
my_first_module.hello()
```

```
hello, i am living in a different file!!!
```

In [36]:

```
from my_first_module import hello
```

In [37]:

```
hello()
```

```
hello, i am living in a different file!!!
```

File handling

In [38]:

```
!wget https://www.dropbox.com/s/go8m1pbuiphzgi8/mobile_cleaned.csv
```

```
--2021-08-26 09:35:59--  https://www.dropbox.com/s/go8m1pbuiphzgi8/mob
ile_cleaned.csv (https://www.dropbox.com/s/go8m1pbuiphzgi8/mobile_clea
ned.csv)
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18, 2620:100:
601d:18::a27d:512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443... c
onnected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/go8m1pbuiphzgi8/mobile_cleaned.csv [following]
--2021-08-26 09:35:59--  https://www.dropbox.com/s/raw/go8m1pbuiphzgi
8/mobile_cleaned.csv (https://www.dropbox.com/s/raw/go8m1pbuiphzgi8/mo
bile_cleaned.csv)
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc99b25a0c10d1ee0e2fddc8310f.dl.dropboxusercontent.c
om/cd/0/inline/BU-kFbrbuA-mG3nhOgqo4sbl4-h304UPQmp95EvgjkgCKNZA1Mxpfx7
i4oVybjPUS1je4gQCwjjsR41mXUmn5bSvWM0nL30_1q3NNTRPo32PywHKQTAl1JCfYO2Lb
irLQ_Itoskip1aSoNPSig4KWwp9/file# (https://uc99b25a0c10d1ee0e2fddc8310
f.dl.dropboxusercontent.com/cd/0/inline/BU-kFbrbuA-mG3nhOgqo4sbl4-h304
UPQmp95EvgjkgCKNZA1Mxpfx7i4oVybjPUS1je4gQCwjjsR41mXUmn5bSvWM0nL30_1q3N
NTRPo32PywHKQTAl1JCfYO2LbirLQ_Itoskip1aSoNPSig4KWwp9/file#) [followin
g]
--2021-08-26 09:36:00--  https://uc99b25a0c10d1ee0e2fddc8310f.dl.dropb
oxusercontent.com/cd/0/inline/BU-kFbrbuA-mG3nhOgqo4sbl4-h304UPQmp95Evg
jkgCKNZA1Mxpfx7i4oVybjPUS1je4gQCwjjsR41mXUmn5bSvWM0nL30_1q3NNTRPo32Pyw
HKQTAl1JCfYO2LbirLQ_Itoskip1aSoNPSig4KWwp9/file (https://uc99b25a0c10d
1ee0e2fddc8310f.dl.dropboxusercontent.com/cd/0/inline/BU-kFbrbuA-mG3nh
Ogqo4sbl4-h304UPQmp95EvgjkgCKNZA1Mxpfx7i4oVybjPUS1je4gQCwjjsR41mXUmn5b
SvWM0nL30_1q3NNTRPo32PywHKQTAl1JCfYO2LbirLQ_Itoskip1aSoNPSig4KWwp9/fil
e)
Resolving uc99b25a0c10d1ee0e2fddc8310f.dl.dropboxusercontent.com (uc99
b25a0c10d1ee0e2fddc8310f.dl.dropboxusercontent.com)... 162.125.2.15, 2
620:100:601d:15::a27d:50f
Connecting to uc99b25a0c10d1ee0e2fddc8310f.dl.dropboxusercontent.com
 (uc99b25a0c10d1ee0e2fddc8310f.dl.dropboxusercontent.com)|162.125.2.15
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14044 (14K) [text/plain]
Saving to: ‘mobile_cleaned.csv’

mobile_cleaned.csv  100%[===================>]  13.71K  --.-KB/s    in
0s

2021-08-26 09:36:00 (325 MB/s) - ‘mobile_cleaned.csv’ saved [14044/140
44]
```

In [39]:

```
!ls
```

```
mobile_cleaned.csv  my_first_module.py  __pycache__  sample_data
```

In [40]:

```
file = open('mobile_cleaned.csv', 'r')
```

In [41]:

```
s = file.readline()
```

In [42]:

```
print(s)
```

```
sim_type,aperture,gpu_rank,weight,stand_by_time,processor_frequency,th
ickness,flash_type,front_camera_resolution,auto_focus,screen_size,fram
es_per_second,FM,no_of_reviews_in_gsmarena_in_week,os,phone_height,scr
een_protection,sim_size,price,talk_time,video_resolution,display_resol
ution,removable_battery,display_type,primary_camera_resolution,battery
_type,ram_memory,internal_memory,brand_rank,no_of_cores,micro_sd_slot,
screen_pixel_density,water_proof_rate,phone_width,expandable_memory,ve
rsion,usb_type,battery_capacity,processor_rank,is_liked
```

In [77]:

```
print(len(s.split(',')))
```

```
40
```

In [76]:

```
print(s.split(','))
```

```
['sim_type', 'aperture', 'gpu_rank', 'weight', 'stand_by_time', 'proce
ssor_frequency', 'thickness', 'flash_type', 'front_camera_resolution',
'auto_focus', 'screen_size', 'frames_per_second', 'FM', 'no_of_reviews
_in_gsmarena_in_week', 'os', 'phone_height', 'screen_protection', 'sim
_size', 'price', 'talk_time', 'video_resolution', 'display_resolutio
n', 'removable_battery', 'display_type', 'primary_camera_resolution',
'battery_type', 'ram_memory', 'internal_memory', 'brand_rank', 'no_of_
cores', 'micro_sd_slot', 'screen_pixel_density', 'water_proof_rate',
'phone_width', 'expandable_memory', 'version', 'usb_type', 'battery_ca
pacity', 'processor_rank', 'is_liked\n']
```

In [44]:

```
file.close()
```

In [45]:

```
with open('mobile_cleaned.csv', 'r') as file:
  print(file.readline())
```

```
sim_type,aperture,gpu_rank,weight,stand_by_time,processor_frequency,th
ickness,flash_type,front_camera_resolution,auto_focus,screen_size,fram
es_per_second,FM,no_of_reviews_in_gsmarena_in_week,os,phone_height,scr
een_protection,sim_size,price,talk_time,video_resolution,display_resol
ution,removable_battery,display_type,primary_camera_resolution,battery
_type,ram_memory,internal_memory,brand_rank,no_of_cores,micro_sd_slot,
screen_pixel_density,water_proof_rate,phone_width,expandable_memory,ve
rsion,usb_type,battery_capacity,processor_rank,is_liked
```

In [46]:

```
with open('mobile_cleaned.csv', 'r') as file:
  print(file.read())
```

```
sim_type,aperture,gpu_rank,weight,stand_by_time,processor_frequency,t
hickness,flash_type,front_camera_resolution,auto_focus,screen_size,fr
ames_per_second,FM,no_of_reviews_in_gsmarena_in_week,os,phone_height,
screen_protection,sim_size,price,talk_time,video_resolution,display_r
esolution,removable_battery,display_type,primary_camera_resolution,ba
ttery_type,ram_memory,internal_memory,brand_rank,no_of_cores,micro_sd
_slot,screen_pixel_density,water_proof_rate,phone_width,expandable_me
mory,version,usb_type,battery_capacity,processor_rank,is_liked
0,12,55,155.0,250,1.3,10.5,5,2.0,0,4.5,15,3,0,0,132.9,26,3,3870,9,48
0,12,3,11,5.0,1,1,7,29,6,4,2,3,67.8,64.0,5,3,2000,125,1
0,1,55,132.0,300,1.3,10.6,5,0.3,1,4.0,30,2,6,0,124.5,26,5,4059,9,720,
15,3,11,5.0,1,1,7,11,6,4,4,6,64.0,32.0,5,3,2000,165,1
0,9,55,142.0,329,1.5,8.5,5,2.0,3,5.0,30,2,20,0,145.5,4,3,4777,10,108
0,4,3,1,5.04,2,1,7,27,6,4,9,6,72.0,32.0,6,3,2500,164,0
0,8,55,152.0,385,1.3,8.0,5,2.0,3,5.0,15,3,0,0,147.5,26,3,5799,19,720,
17,3,2,5.0,1,1,7,4,6,4,1,3,75.1,32.0,6,3,3000,165,1
1,1,55,234.0,385,1.3,7.9,5,1.92,3,5.0,15,3,0,0,179.0,18,3,5990,11,72
0,17,3,1,5.0,1,1,7,4,6,4,1,6,91.0,32.0,6,3,3000,165,0
0,14,55,179.0,280,1.3,7.9,5,5.0,3,5.5,30,3,6,0,150.0,5,3,5999,22,720,
0,0,3,8,0,1,1,7,4,6,4,7,3,71.0,32.0,6,3,2000,165,0
```

In [47]:

```
with open('mobile_cleaned.csv', 'r') as file:
  for line in file:
    for word in line.split(','):
      print(word)
    print('--------')
```

```
sim_type
aperture
gpu_rank
weight
stand_by_time
processor_frequency
thickness
flash_type
front_camera_resolution
auto_focus
screen_size
frames_per_second
FM
no_of_reviews_in_gsmarena_in_week
os
phone_height
screen_protection
sim_size
price
```

In [48]:

```
with open('my_first_file_output.txt', 'w') as file:
  file.write('hello world from python code')
```

In [49]:

```
!cat my_first_file_output.txt
```

```
hello world from python code
```

1.10- NumPy - **"NumPy is a general-purpose fundamental package for scientific computing with Python. It contains various features including these important ones:

> A powerful N-dimensional array object

> Sophisticated (broadcasting) functions

> Tools for integrating C/C++ and Fortran code

> Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases."

- Source: Official website of NumPy (www.numpy.org (http://www.numpy.org))
- *Let us consider some examples to understand this powerful package.**

In [50]:

```python
# First you need to import the NumPy Library
import numpy as np

b1 = np.array([1,2,3,5,5,8])    #Declaring a NumPy Array
b2 = np.array([4,5,6,7,7,9])

print(b1+b2)

print(b2 * 3)
print("No. of dimensions: ", b1.ndim)  # Rows in array, considered as a matrix.
# Printing shape of array
print("Shape of array: ", b1.shape)  # Dimension

#reshaping an array
r_b1=b1.reshape(2,3)

print("Reshaped array: ", r_b1)
print("Shape of array: ", r_b1.shape)

# Printing size (total number of elements) of array
print("Size of array: ", b1.size) # elements in a row or column elements.

# Printing the datatype of elements in array
print("Array stores elements of type: ", b1.dtype)
```

```
[ 5  7  9 12 12 17]
[12 15 18 21 21 27]
No. of dimensions:  1
Shape of array:  (6,)
Reshaped array:  [[1 2 3]
 [5 5 8]]
Shape of array:  (2, 3)
Size of array:  6
Array stores elements of type:  int64
```

1.11 Operations on Arrays - You can create arrays in NumPy in various ways.

Array creation: You can create arrays in NumPy in various ways.

For example, you can create an array from a regular Python list or tuple using the array function.

The type of the resulting array is deduced from the type of the elements in the sequences.

Often, we need to declare arrays whose sizes are known but elements are initially unknown. Hence, NumPy offers many functions to create arrays with initial placeholder content. This minimizes the necessity of growing arrays, which is generally an expensive operation.

- Examples: *np.zeros, np.ones, np.full, np.empty*, etc.

To create sequences of numbers, NumPy provides a function analogous to range that returns arrays instead of lists.

arange: returns evenly spaced values within a given interval. In this, step size is specified.

linspace: returns evenly spaced values within a given interval. Number of elements are returned.

Reshaping array: The reshape method is used to reshape an array. If you have an array (a1, a2, a3, …, aN) and you want to reshape and convert it into another array of shape (b1, b2, b3, …, bM), you can do it easily using the reshape method. But the only precondition is that a1 x a2 x a3 … x aN = b1 x b2 x b3 … x bM . (i.e. , the total number of elements in the array should be the same, or the original size of array should remain unchanged.)

Flatten array: The flatten method is used to convert an array into one dimension. It accepts order argument. Default value is ‘C’ (for row-major order), and you can use ‘F’ to use the flatten method for column major order.

- Let us see some examples.

In [51]:

```
# array creation
import numpy as np

# Creating array from a list with type float
A = np.array([[1, 2, 4], [5, 8, 7]], dtype = 'float')
print("A : ",A)
# Create a 3X4 array with all zeros. Please note, we have used double paranthesis.
B = np.zeros((3, 4))
print("B : ",B)
# Create an arrary of complex numbers
C = np.full((3, 3), 6, dtype = 'complex')
print("C : ",C)
# Create an array with random values
np.random.seed(1) # A seed is set to ensure that the results are consistent if you u

D = np.random.randn(2, 2)
print("D : ",D)
E = np.random.random((2, 2))
print("E : ",E)
F = np.random.randint(3, 15, size=(2, 4))
print("F : ",F)
```

```
A :  [[1. 2. 4.]
 [5. 8. 7.]]
B :  [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
C :  [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]
D :  [[ 1.62434536 -0.61175641]
 [-0.52817175 -1.07296862]]
E :  [[0.39676747 0.53881673]
 [0.41919451 0.6852195 ]]
F :  [[14 13  5  7]
 [10 10 12  4]]
```

1.12 Array Reshaping -

In [55]:

```
# Reshaping 3X4 array to 2X2X3 array
A = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 1, 2, 3]])

new_A = A.reshape(2, 2, 3)

# Flatten array
B = np.array([[1, 2, 3], [4, 5, 6]])
flat_B= B.flatten()
# column_flat_B('C')

print ("\nOriginal array:\n", A)
print ("Reshaped array:\n", new_A)
print ("\nOriginal array:\n", B)
print ("Fattened array:\n", flat_B)
#print ("Column Fattened array:\n", column_flat_B)
```

```
Original array:
 [[1 2 3 4]
 [5 6 7 8]
 [9 1 2 3]]
Reshaped array:
 [[[1 2 3]
  [4 5 6]]

 [[7 8 9]
  [1 2 3]]]

Original array:
 [[1 2 3]
 [4 5 6]]
Fattened array:
 [1 2 3 4 5 6]
```

1.13 Broadcasting -

The term broadcasting refers to managing arrays of different shapes during arithmetic operations in NumPy.

Arithmetic operations on arrays are usually performed on corresponding elements, based on the law of matrices \ arrays.

NumPy provides a way by which dimension mismatch is taken care of. When operating on two arrays, NumPy compares their shapes element-wise. It starts with the trailing dimensions, and works its way forward. Two dimensions are compatible when they are equal, or one of them is 1

For example, if you want to add an array 'b' of size (4,1) to a matrix 'A' of size (4,3), then the values in each row of 'b' are broadcasted and added in each column of the matrix 'A'. Let us see how it is done.

1.14 Python Classes-

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.

In [56]:

```
class MobilePhone:
  """This is a sample class to illustrate how Python classes work"""
  def __init__(self, name, is_android = False, screen_size = 4.3):
    self.name = name
    self.is_android = is_android
    self.screen_size = screen_size
    self.rating = -1

  def has_rating(self):
    return self.rating > -1
```

In [57]:

```
new_phone = MobilePhone('iPhone 5s')
```

In [58]:

```
type(new_phone)
```

Out[58]:

```
__main__.MobilePhone
```

In [59]:

```
print(new_phone.name, new_phone.is_android, new_phone.screen_size)
```

```
iPhone 5s False 4.3
```

In [60]:

```
new_phone.screen_size = 4
```

In [61]:

```
new_phone.has_rating()
```

Out[61]:

```
False
```

In [62]:

```
new_phone.rating = 3.9
```

In [63]:

```
MobilePhone.__doc__
```

Out[63]:

```
'This is a sample class to illustrate how Python classes work'
```

In [64]:

```
class iPhone(MobilePhone):
  def __init__(self, name):
    MobilePhone.__init__(self, name, False, 4)

  def __str__(self):
    return self.name + " " + str(self.is_android) + " " + str(self.screen_size)
```

In [65]:

```
new_iphone = iPhone('iPhone 5s')
```

In [66]:

```
new_iphone.is_android
```

Out[66]:

```
False
```

In [67]:

```
print(new_iphone)
```

```
iPhone 5s False 4
```

In [68]:

```
a = np.array([[1,1,1],[1,1,1], [1,1,1]])
b = np.array([2,2,2])
c = a + b
c1= a*b #element wise multiplication. This will also be broadcasted accordingly.

print(c)
print(c1)
```

```
[[3 3 3]
 [3 3 3]
 [3 3 3]]
[[2 2 2]
 [2 2 2]
 [2 2 2]]
```

1.15 Files I/O - The file object provides a set of methods to read and write on files. Some examples are given below.

In [69]:

```
# File open, read and write.

file1 = open("sample.txt", "w+")
file1.write("Welcome to the couse on Machine Learning at NIT Delhi.");
file1.close()

# Open a file
file_read = open("sample.txt", "r")
str = file_read.read(80);
print("Name of the file: ", file_read.name)
print("Read String from file is : ", str)
# Close opend files
file_read.close()
```

```
Name of the file:  sample.txt
Read String from file is :  Welcome to the couse on Machine Learning a
t NIT Delhi.
```

1.16 Using matplotlib - Matplotlib is a Python 2D plotting library.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits (source: https://matplotlib.org/ (https://matplotlib.org/) , https://matplotlib.org/citing.html (https://matplotlib.org/citing.html))

Here we give a simple example of its use.

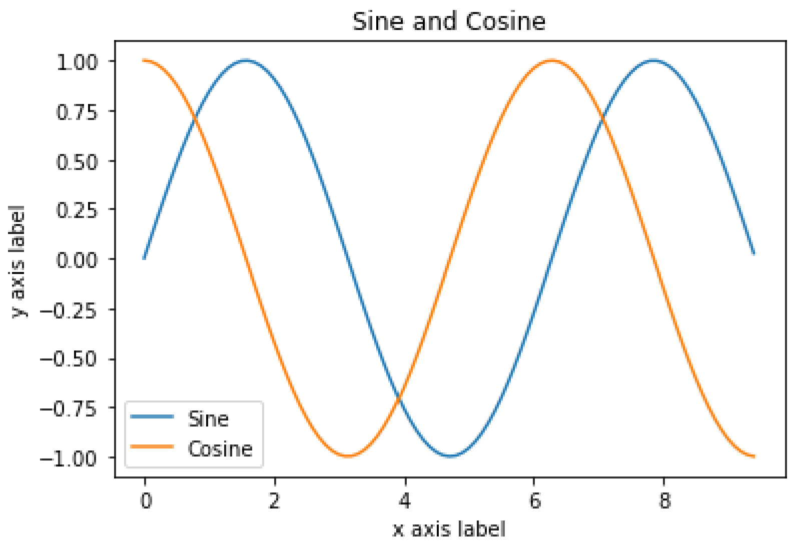
In [70]:

```
import numpy as np
import matplotlib.pyplot as plt

# Computes x and y coordinates for
# points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])

plt.show()
```



1.17 Using Images -

In [78]:

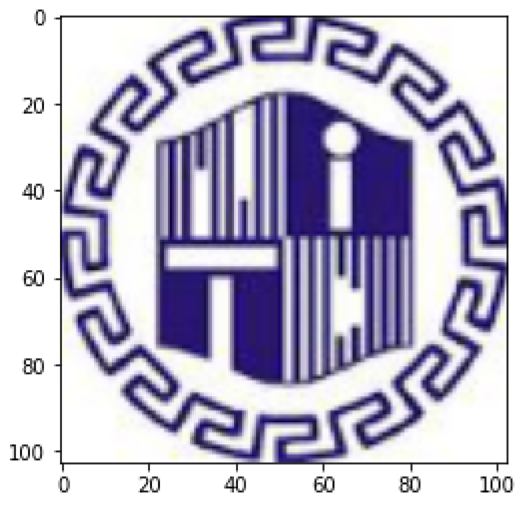
```python
import matplotlib.pyplot as plt
import cv2
import matplotlib.image as mpimg
img=mpimg.imread('NITD.jpeg')

print('Original Dimensions : ',img.shape)

scale_percent = 80 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)
plt.figure(figsize=(4,10))
imgplot = plt.imshow(resized)
plt.show()
```

```
Original Dimensions :  (129, 129, 3)
Resized Dimensions :  (103, 103, 3)
```



In [72]:

```python
# In case any error in the above cell , use either 1 or 2 as per your python enviror

# 1. pip install opencv-python
# 2. pip install cv2
```

Pythonic Way of Coding

In [73]:

```
#this is bad example and not pythonic
data = ["Hello", "I", "am", "in", "Delhi"]
for i in range(len(data)):
    print(data[i])
```

```
Hello
I
am
in
Delhi
```

In [74]:

```
#this is good example and more pythonic
data = ["Hello", "I", "am", "in", "Delhi"]
for word in data:
    print(word)
```

```
Hello
I
am
in
Delhi
```

Type *Markdown* and LaTeX: α^2

Reference: - Machine Learning for Computer Vision2020, E&ICT

In [74]:

```
```