NumPy package

Lecture 3: NumPy package of Python

- **M.Sc. Riyadh Seed Agid**
- **Salahaddin University – Erbil**
- **riyadh.agid@su.edu.krd**

Python programming

- Python is a special programming language that includes the functionality required by programmers to write all types of code.

- Python contain a library of functions capable for different tasks.

- A library is a collection of "books" that perform specific tasks and extend your programs functionality. In python a library is a collection of modules

- A module is a file, like a book in the library, that contains functions you can import into your program.

- IDLE : Integrated development and learning environments

python package

NumPy

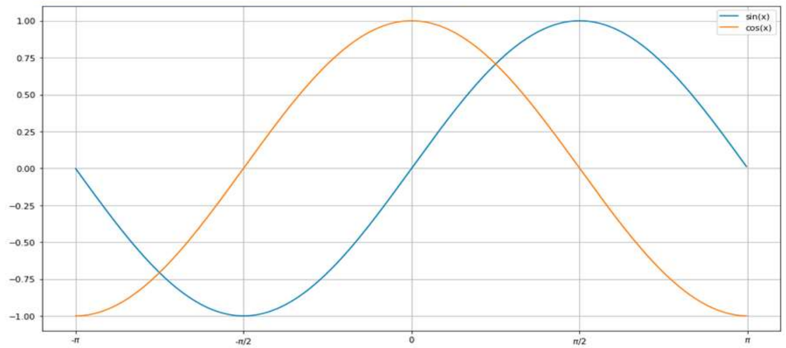
Numpy is a general-purpose array-processing package.

SciPy

SciPy is an open-source Python library which is used to solve scientific and mathematical problems.

matplotlib

is a plotting library



python package

1) **Numpy** (Numeric python) package provides basic routines for manipulating large arrays and matrices of numeric data. It contains array functionality, linear algebra, Fourier transform, and random number capabilities.

Getting NumPy

- Open "**Command Prompt**"
- Then type the following command to install "**Numpy**" package:

```
C:\Users\Your Name> pip install numpy
```

```
import numpy
```

Now NumPy is imported and ready to use.

Example 1a:

from numpy import*

```
a = pi
print(a)
Result
3.14

B = e
print(B)
Result
2.71
```

Example 2a:

import numpy as np

```
a =np. pi
print(a)
Result
3.14

B =np.e
print(B)
Result
2.71
```

Example 2:

```
from numpy import*

print arange(0.0, 1.0, 0.1)

[ 0.  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9]
```

2) **Scipy** (Scientific python) Open-source Python software for mathematics, science, and engineering. This contains a large number of packages which can perform some fairly complex analysis. The additional benefit of basing SciPy on Python is that this also makes a powerful programming language available for use in developing sophisticated programs and specialized applications.

SciPy is organized into subpackages covering different scientific computing domains.

These are summarized in the following table:

Subpackage	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear algebra
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions

from Scipy import linalg, optimize

3) **Matplotlib :** Python 2D plotting library. It tries to make easy things easy and hard things possible. This provides a straightforward way to generate and save plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

Syntax summary we will provide a syntax summary that lists all of the functions and computational methods that are introduced in each lecture. This is available in the python math package, and the same functions (with additional features) are available in NumPy. We will be using the NumPy versions

Function	Syntax	Alternative Syntax
Addition	x + y	add(x,y)
Subtraction	x - y	subtract(x,y)
Multiplication	x * y	multiply(x,y)

Function	Syntax	Alternative Syntax
Division	x / y	divide(x,y)
Remainder	x % y	remainder(x,y)
Test of equality	x == y	
Tests of inequality	x > y; x < y	x >= y; x <= y
Absolut value	abs()	
Power	x**y	pow(x,y)
Scientific notation	2.5*10**7	2.5e7
Square root	sqrt(x)	
Log_e	log(x)	
Log_{10}	log10(x)	
Exponential	exp(x)	e**x
Sine	sin(x)	
Cosine	cos(x)	
Tangent	tan(x)	

Function	Syntax	Alternative Syntax
Inverse Sine	**arcsin(x)**	
Inverse Cosine	**arccos(x)**	
Inverse Tangent	**arctan(x)**	
Hyperbolic Sine	**sinh(x)**	
Hyperbolic Cosine	**cosh(x)**	
Hyperbolic Tangent	**tanh(x)**	
Inverse Hyperbolic Sine	**arcsinh(x)**	
Inverse Hyperbolic Cosine	**arccosh(x)**	
Inverse Hyperbolic Tangent	**arctanh(x)**	
Convert angle from degrees to radians	**deg2rad(x)**	**radians(x)**
Convert angle from radians to degrees	**rad2deg(x)**	**degrees(x)**

operations

#Addition:
4+5
add(4,5)   #don't forget to import numpy!

#Subtraction:
6-2
subtract(6,2)

#Multiplication:
7*9
multiply(7,9)

#Division:

7/3   #or

divide(7,3)   did the division gives what you expected? no why?

Use remainder function   remainder(7,3)!

As we have seen, Python will by default treat whole numbers as integers. In order to have the computer work with decimals (floating point numbers), you have to tell Python to store it as a float (a floating-point number) rather than an int (integer or whole number).

Example

divide(7.0,3.0) = 2.33

#***The easiest way to simply put a decimal point after every number you want treated as a float such as 3., 7., 10.***

The equal sign = is used to assign a value to a variable. Afterward, no result is displayed before the next interactive prompt

Example

```
>>>width = 20
>>>height = 5*9
>>>Width *height
900
```

If a variable is not "defined" (assigned a value), trying to use it will give you an error.

Trace back (most recent call last):
File "<stdin>", line 1, in <module>
Name error: name '…' is not defined

Other operations

1- ***abs() function:*** can return the absolute value of the number

>>> abs(-7.3) return a value of 7.3

2- **pow()** function**:** There are two way to raise a number a to the power n, use ** or the function **pow()**
>>> 7**2 or
>>> pow(7,2) these should be give the same answer 49.
9**(1.0/2) or 9**0.5 should be equivalent to pow(9,1./2)

Exercise:
Try 9** ½ , 9** (1/2) and , 9** 1./2 what is the difference check?

Very small numbers or large numbers are often written in scientific notation, for example 4.5×10^6. In Python you can do this in two ways

4.5*10**6 or using 4.5e6

3- Roots: in python we have a specific function ***sqrt()*** that describes the square root and the answer will be in float.

Example :

```
>>>np.sqrt(9)
3.0
```

4- Exponentials: You can calculate exponentials using the powers method described above, as a value for e is already stored in Python. There is also the built in function

>>>np.***exp().***

Example

```
  e**3 or
>>>np.exp(3)                    #performs the same function.
```

5-Trigonometric: Python can handle trigonometry in much the same way as your calculator.

❖**Note** that by default, it works in radians

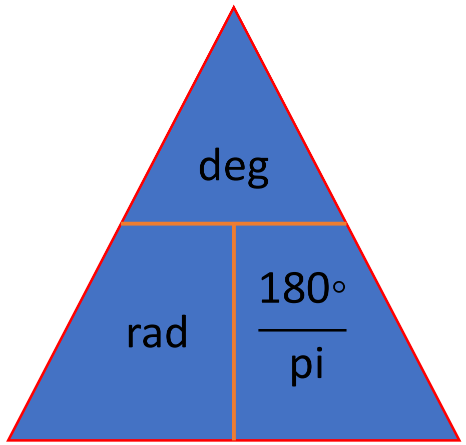
Example **sin(60)** dose not give you the value that you expect.

However, pi is already defined so one can use
Sin(pi/3) instead of sin(60) ….the same is true for other trigonometric functions.

To convert an angle from degrees to radians, use:

```
>>>np.radians(60)

>>>np.deg2rad(60)
```



You have access to all the usual trigonometric functions, inverses and hyperbolic functions.

How python performs order of calculation

An essential point to remember, especially when performing large calculations, is the order in which Python will do each operation. For each line of calculation, Python will operate in the following order

1- **Brackets**. 2- **Indices**. 3- **Division and Multiplication**. 4-**Addition and subtraction**

Division and multiplication are of the same level, so in the absence of brackets they will be performed in the order they are read (i.e. left to right) and the same for addition and subtraction.

Example:

```
>>>X = 10+10*5

>>>X = (10+10)*5                see difference
```

Remember that if all values are entered as integers, calculations involving division may give incorrect answers. The easiest way to avoid this is to get into the habit of entering numbers as floats by adding a decimal point where necessary as we have done in the examples throughout this worksheet.

Defining and Displaying Variables

Variables are containers for storing data values.

You will be familiar with variables from mathematics where a variable is defined as a symbol that represents a quantity in a mathematical expression. In computing, variables serve a similar, although slightly different purpose. Type in, for example:

```
>>> x = 2
```

The variable with name x should appear in the variable explorer as an integer with value 2. It will be remembered that the value of x is assigned to be equal to 2, until this value is overwritten. To display the value of defined variables use the following:

```
>>> print x
```

or simply

```
>>> x
```

Note that variables are case sensitive so that

```
>>> print X
```

will not work. It will return NameError: name 'X' is not defined.

Defining and Displaying Variables

Variables that have been defined can be overwritten, for example, x has previously been defined to have a value of 2. Try the following:

```
>>> print x
2
>>> x = 0
>>> print x
0
```

Incrementing a variable means increasing the value by a step. You can increment the variable x by 1 by typing

```
>>> x=x+1
>>> print x
```

The print x commands are here so that it is clear what is happening.

Dealing with variable

When setting a variable to be a function of another variable, be careful that the one you are changing comes first:

x = y/2 sets the value of x to be half the value of y. y/2 = x will not work -- it gives the error ***SyntaxError: can't assign to operator***

Function	Syntax	Alternative Syntax
Define variable as integer	x = 5	x = int(5)
Define variable as float	x = 5.0	x = float(5)
Display value of variable	print(x)	print x
Increment variable	x = x + 1	

Example

A ball is dropped under the earth gravity with zero initial velocity so its speeds up. IF the variable t represent the time and the constant g is an acceleration. Write a program to find the speed, v, of the ball after one second.

```
>>>t=1.0
>>>g=9.8
>>>v_o =0
>>> v= v_o + g*t
>>> print v
```

If we increment the time by one second , and make the old final velocity be the new initial velocity, and recalculate for the speed after 2 seconds

```
>>>t=t+1
>>>v=v_o
>>> v= v_o + g*t
>>>print v
```

Exercise

a) Calculate $7 \div 2$. Do this both as an integer and a float calculation and note the difference.

b) Calculate the remainder.

c) Define the variable x=55, y= 70, and n=3 then test:

$x+y$

y/x

$x^3 - y^n$

$\sqrt{y}$

y^3. Check this gives the same result as y x y x y

e^{-2n+1} , $\ln(2x)$, $\log_{10}(6y)$

Treating x and y as angles in degrees convert them to radians and calculate the following:

$\sin(x)$

$\cos^2(5y)$

$\cos^2(x)+\sin^2(x)$