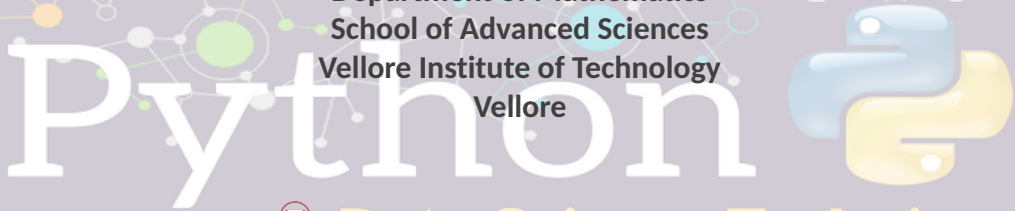
PMDS508L - Python Programming

Dr. B.S.R.V. Prasad
Department of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Vellore



srvprasad.bh@gmail.com (Personal)
srvprasad.bh@vit.ac.in (Official)
+91-8220417476

Introduction to

Python

Jupyter Modifications

IPython Interactive Shell//Multiple Outputs in Jupyter Notebook

To obtain the multiple output of all the commands that we run in a single a cell in Jupyter notebook we can activate the Interactive Shell by running the following two lines of the code in the beginning of the Notebook.

```
1 from IPython.core.interactiveshell import
     InteractiveShell
2 InteractiveShell.ast_node_interactivity = "all"
```

IPython Magic Commands

- The magic function system provides a series of functions which allow you to control the behavior of IPython itself, plus a lot of system-type features.
- There are two kinds of magics, line-oriented and cell-oriented.
- Line magics are prefixed with the % character and work much like OS command-line calls.
  - `%timeit range(100)`
- Cell magics are prefixed with the %%

```
%%timeit
x = numpy.matrix ([[1 ,2] ,[2 ,4]])
eig = numpy.linalg.eigvals(x)
```

IPython Magic Commands

- To get help on Magic commands one can type `%magic`
- Few interesting magic commands
  - `%pwd`
  - `%ls`
  - `%cd <direcotry>`
  - `%history`
  - `%mkdir`
- To see all the available magics one can use `%lsmagic`
- If `automagic` is turned on (which is by default) we can neglect % before the commands

IPython Magic Commands

- Profiling and Timing Codes
  - `%time` – Time the execution of a single statement
  - `%timeit` – Time repeated execution of a single statement for more accuracy
  - `%prun` – Run code with profiler
  - `%lprun` – Run code with the line-by-line profiler
  - `%memit` – Measure the memory use of a single statement
  - `%mprun` – Run code with the line-by-line memory profiler

Timing Code Sinppets
%timeit and %time


```
%timeit sum(range(100))

1.46 µs \pm 162 ns per loop (mean \pm std. dev. of 7 runs, 100000
   loops each)
```

%timeit automatically does a large number of repetitions. For slower commands, %timeit will automatically adjust and perform fewer repetitions

```
%%timeit
total = 0
for i in range(1000):
    for j in range(1000):
        total += i * (-1) ** j

432 ms \pm 114 ms per loop (mean \pm std. dev. of 7 runs, 1 loop
   each)
```