

Phys 443

Computational Physics I

Introduction

Syllabus

COURSE OUTLINE

PHYS443

Computational Physics I

Instructor: Prof. Dr. A. Murat Güler

Room: 333, e-mail: mguler@newton.physics.metu.edu.tr

Class Schedule:

Friday 12:40-16:30

Reference Book

- Computational Physics by Mark Newman
(<http://www-personal.umich.edu/~mejn/cp/index.html>)
- Statistical Data Analysis Glen Cowan
- Techniques for Nuclear and Particle Physics Experiments, Leo
- Introduction to Error Analysis J. Taylor

Weekly Schedule

Weekly schedule					
Name:					
Time / period	Monday	Tuesday	Wednesday	Thursday	Friday
8:40-9:30					
9:40-10:30					
10:40-11:30					
11:40-12:30					
12:40-13:30					Phys443
13:40-14:30					Phys443
14:40-15:30					Phys443
15:40-16:30					

Syllabus

PHYS443

Computational Physics I

Content

Errors; distributions; interpolation techniques; linear system of equations; numerical quadrature; estimation of mean and errors; linear least square minimization and data fitting; maximum likelihood; goodness of fit.

Syllabus

PHYS444

Computational Physics II

Content

Numerical solution techniques of nonlinear equations and ordinary differential equations; optimization and non-linear least squares; simulation and random numbers; time series analysis and Fourier techniques; method of finite differences; partial differential equations.

Syllabus

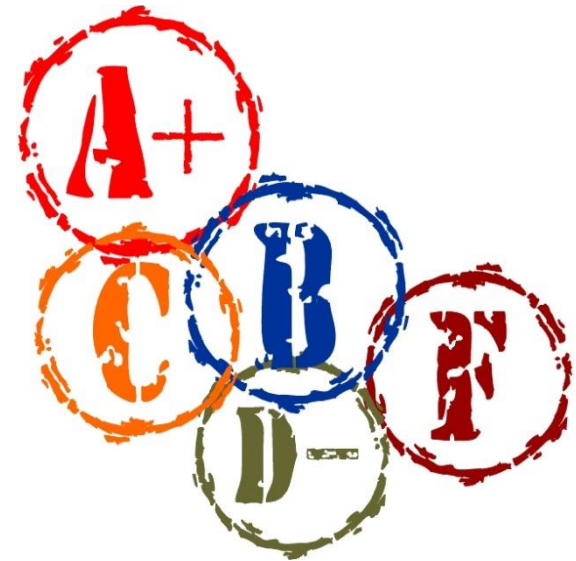
■ Topics

- Introduction: basic concepts
- Python programming
- Probability & Statistics
- Bayes's Theorem
- Basic distributions: Binomial, Poisson Distributions
- Gauss Distributions
- Confidence intervals: definitions, estimation.
- Hypothesis Testing & Regression Analysis
- Advanced parameter estimation: maximum likelihood
- System of linear equations
- Monte Carlo Method
- Root programming
- TensorFlow Deep Neural Learning

Grading

➤ Grading

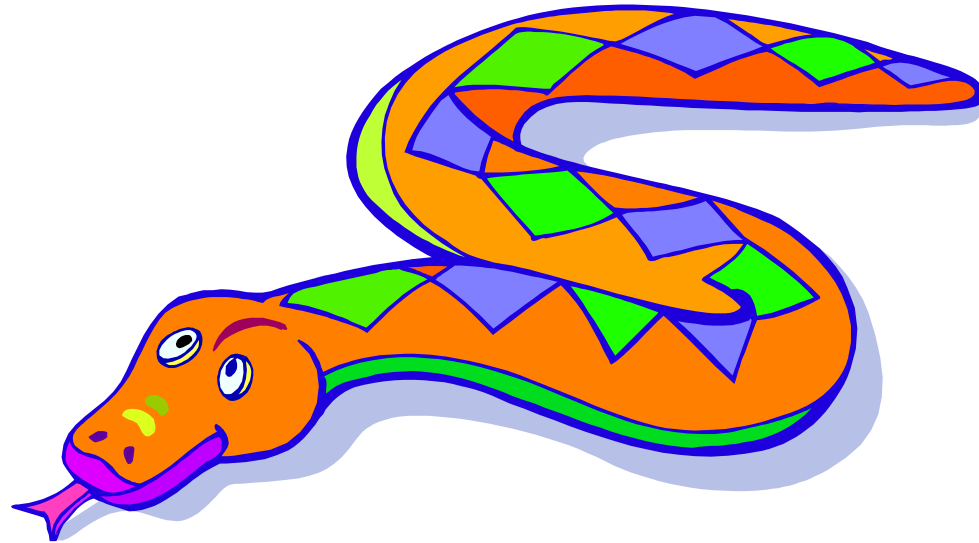
- Attendance : 10%
- Homework : 20%
- Midterm: 35 %
- Final: 35%



Phys 307

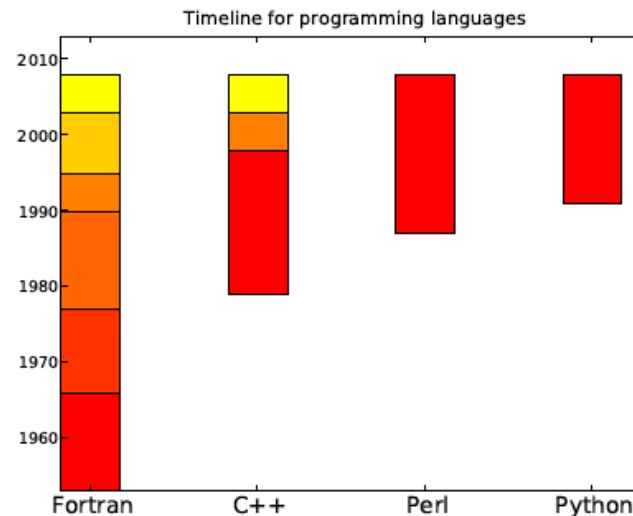
Applied Modern physics

Python I



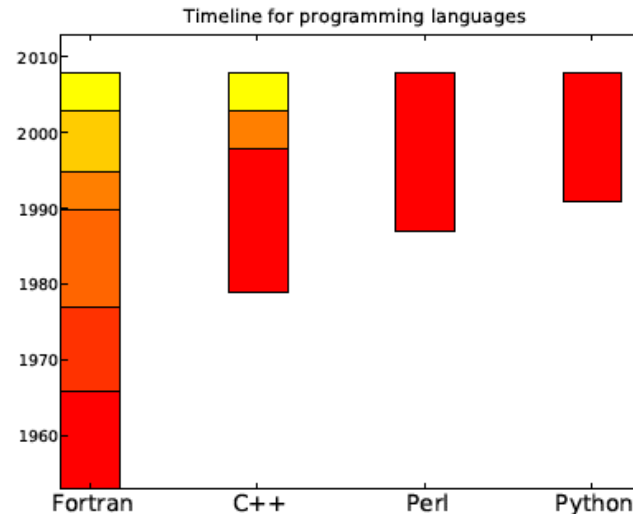
Why Python?

- Completely open source
- Relatively easy to learn
- Many packages for science and data analysis



Why Python?

- Recently it is gaining popularity in the world of scripting.
- Relatively young: first released by Guido van Rossum in 1991.
- **Interpreted languages are often more flexible and changes can be easily made to the program at runtime.**



Which language is popular?

<https://www.tiobe.com/tiobe-index/>

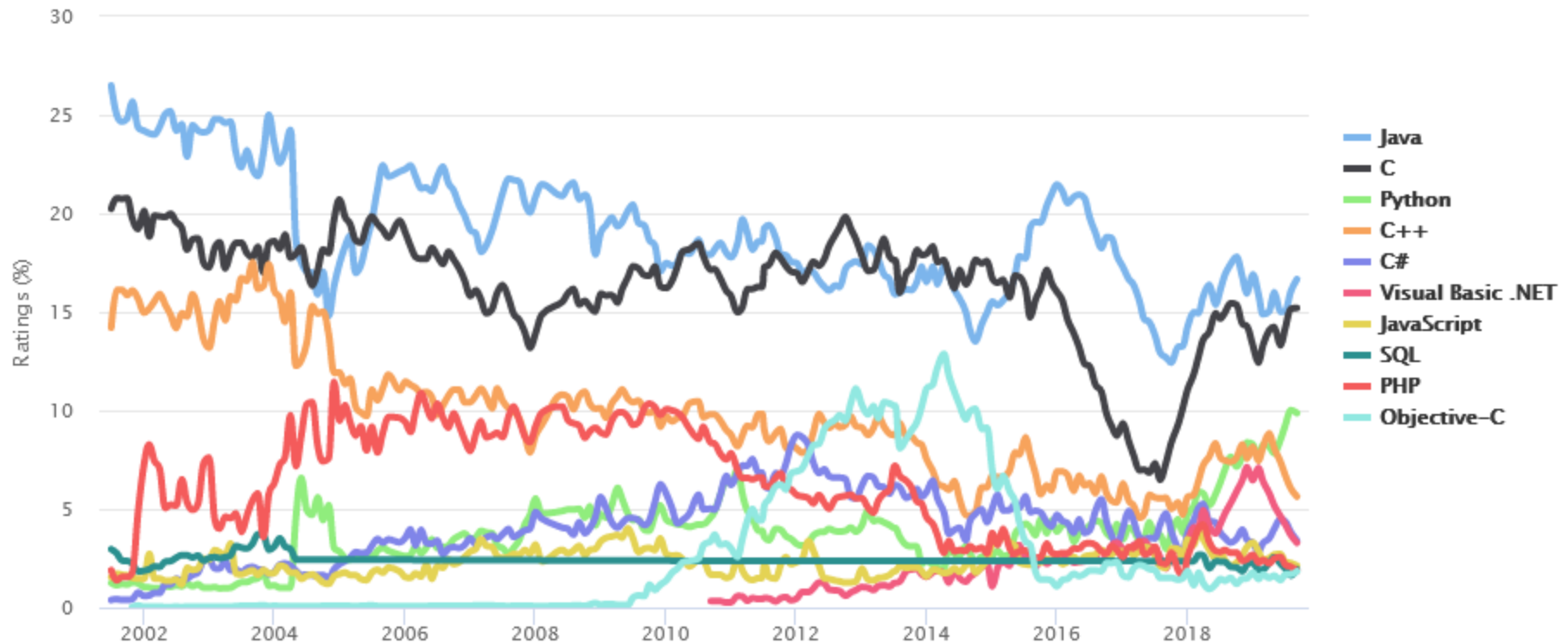
Sep 2019	Sep 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	▲	C#	3.399%	+0.10%
6	5	▼	Visual Basic .NET	3.291%	-2.02%
7	8	▲	JavaScript	2.128%	-0.00%
8	9	▲	SQL	1.944%	-0.12%
9	7	▼	PHP	1.863%	-0.91%
10	10		Objective-C	1.840%	+0.33%
11	34	▲▲	Groovy	1.502%	+1.20%
12	14	▲	Assembly language	1.378%	+0.15%
13	11	▼	Delphi/Object Pascal	1.335%	+0.04%
14	16	▲	Go	1.220%	+0.14%
15	12	▼	Ruby	1.211%	-0.08%
16	15	▼	Swift	1.100%	-0.12%
17	20	▲	Visual Basic	1.084%	+0.40%
18	13	▼▼	MATLAB	1.062%	-0.21%
19	18	▼	R	1.049%	+0.03%

Which language is popular?

<https://www.tiobe.com/tiobe-index/>

TIOBE Programming Community Index

Source: www.tiobe.com



Why Python?

- Python is one of the easiest languages to learn and use.
- At the same time it is very powerful:
 - It is used by many of the most highly productive professional programmers.
 - Robust support for object-oriented programming
 - Support for integration with other languages.

Used by:

Google, Yahoo!, Youtube

Many Linux distributions

Games and apps (e.g. Eve Online)



Why Python?

- Compiled languages are generally more efficient and faster than interpreted languages, since the conversion to machine language only occurs one time, and then the machine language code can be executed whenever the program needs to be run.
- In an interpreted language, the interpretation process occurs every time the program is run.

Why Python?

■ Drawbacks:

- Can be a slow language (lot of overhead)
- Lenient syntax, making it error prone
- Memory allocated and is thus limited.

Python building blocks

Unlike Matlab, Scilab or R, **Python does not come with a pre-bundled set of modules for scientific computing. Python, a generic and modern computing language.**

Below are the basic building blocks that can be combined to obtain a scientific computing environment:

- **Ipython**: an advanced Python shell <http://ipython.org/>
- **Numpy** : provides powerful numerical arrays objects, and routines to manipulate them. <http://www.numpy.org/>
- **Scipy** : high-level data processing routines. Optimization, regression, interpolation, ... <http://www.scipy.org/>
- **Matplotlib** : 2-D visualization, plots..
<http://matplotlib.org/>

Python Installation

- Python can be downloaded from the <http://www.python.org/> website.
- Having installed the Python language, install the visual package from <http://www.vpython.org/> (corresponding to python version)
- Next you need to install matplotlib which you can find at matplotlib.org/download.html. Choose the latest version.
- If you use MAC, install also numpy (If you use Windows, numpy will already have been installed when you installed VPython) from sourceforge.net/projects/numpy/files/NumPy

Python Basics

Python integrated in Anaconda

- •Will give you most scientific packages

- Python 3.X does not maintain backward compatibility with the older versions of Python
 - Thus, code developed for Python 2.X may not work with Python 3.X, and vice-versa.
- So much code and so many libraries have been developed for Python 2.X that most persons in the scientific community still use 2.X.
- Python 2.7, includes many features of Python 3.X, but maintains backward compatibility with older versions.

Python Installation

<https://www.python.org/downloads/source/>



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

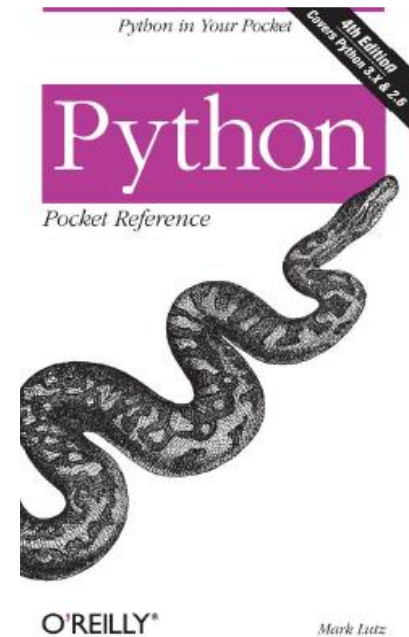
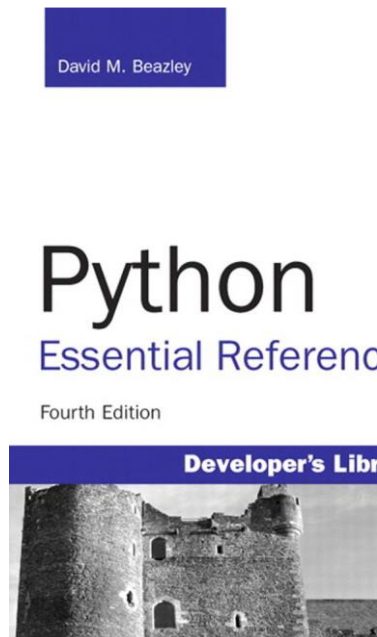
Python >>> Downloads >>> Source code

Python Source Releases

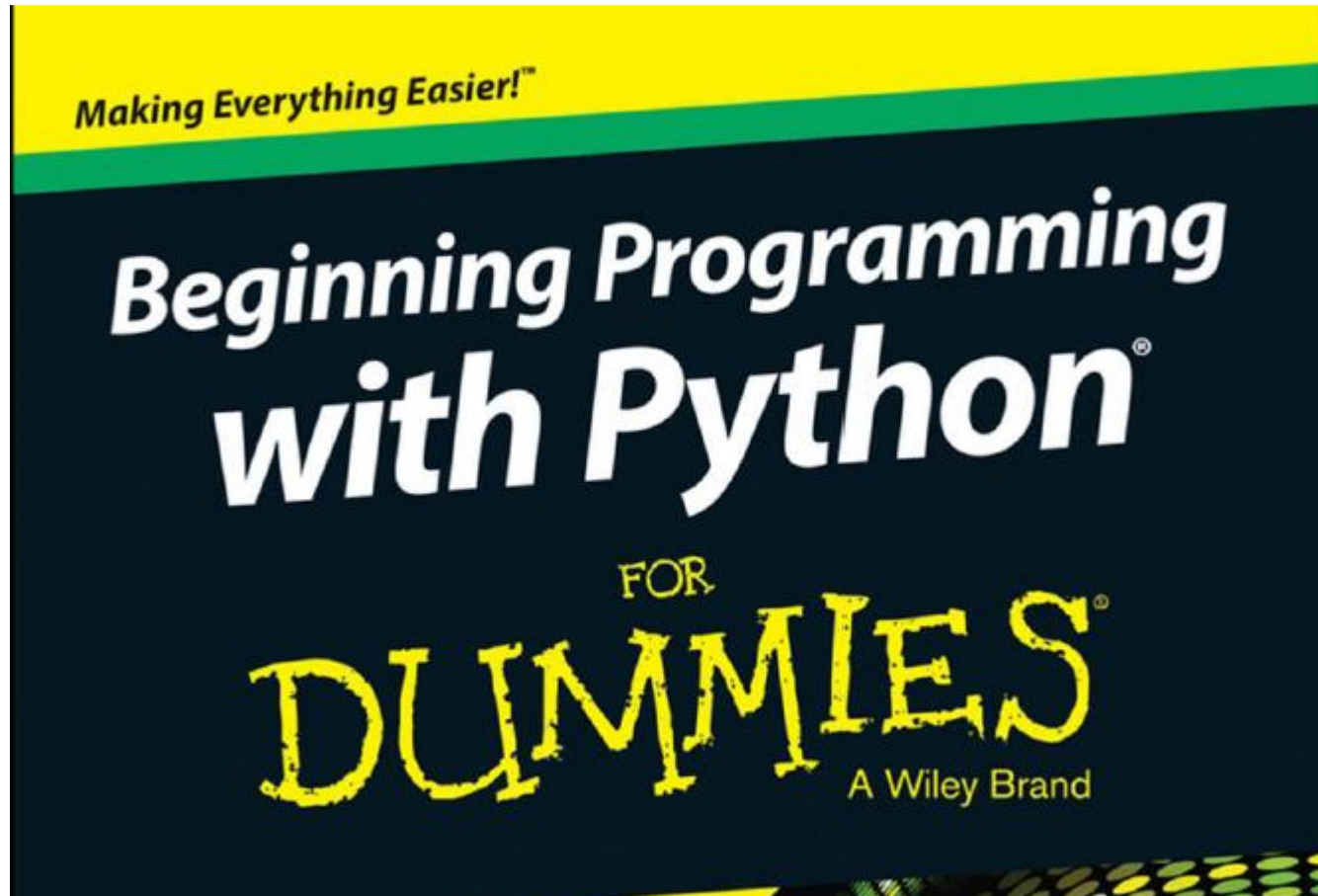
- [Latest Python 3 Release - Python 3.7.4](#)
- [Latest Python 2 Release - Python 2.7.16](#)

Python Documentation

- Books for reference are:
 - Python: Essential Reference(4th ed.) by David M. Beazley.
 - Python Pocket Reference by Mark Lutz.

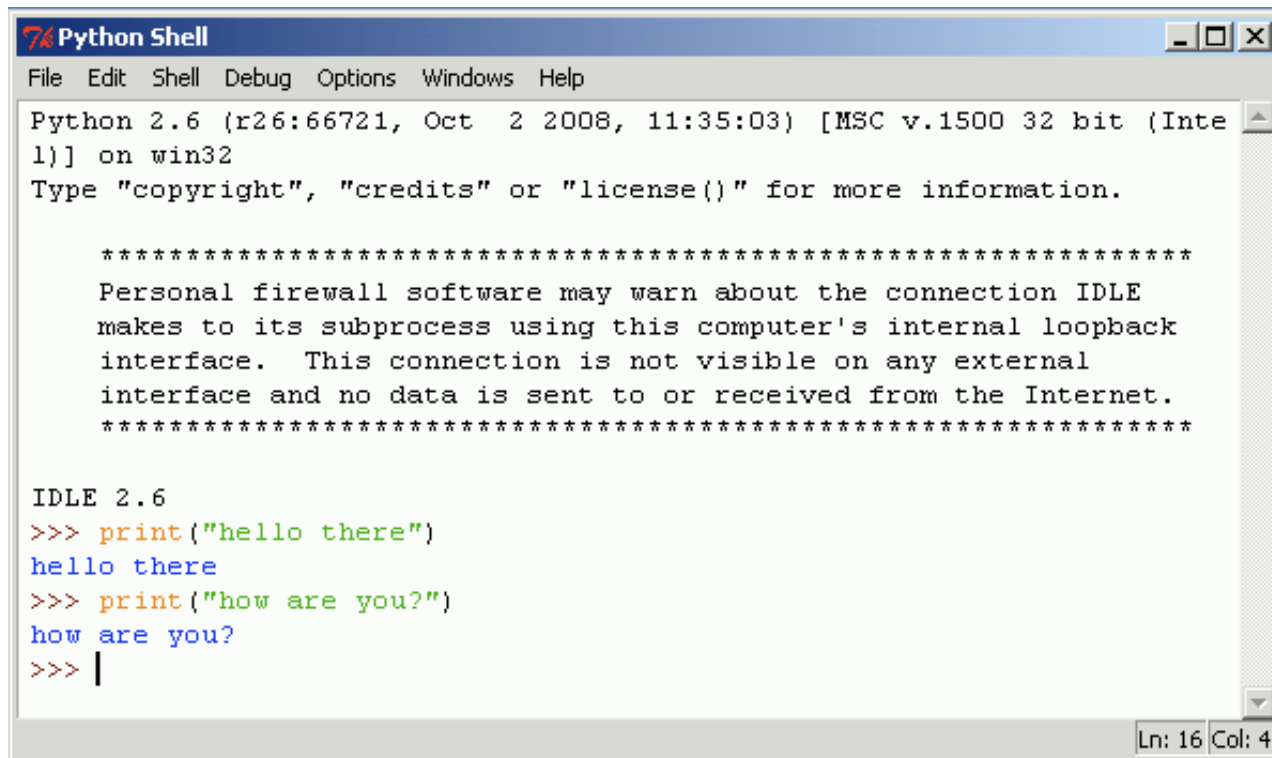


Python Documentation II



Python Basics

- Allows you to type commands one-at-a-time and see results.
- A great way to explore Python's syntax
 - Repeat previous command: Alt+P



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.6 (r26:66721, Oct 2 2008, 11:35:03) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 2.6
>>> print("hello there")
hello there
>>> print("how are you?")
how are you?
>>> |
```

Python Basics

- You can run python programs from files by typing
`>>>python program.py`

at the command line. The file can contain just the python commands.

- Or, one can invoke the program directly by typing the name of the file, “program.py”, if it has as a first line something like

```
#!/usr/bin/python
```

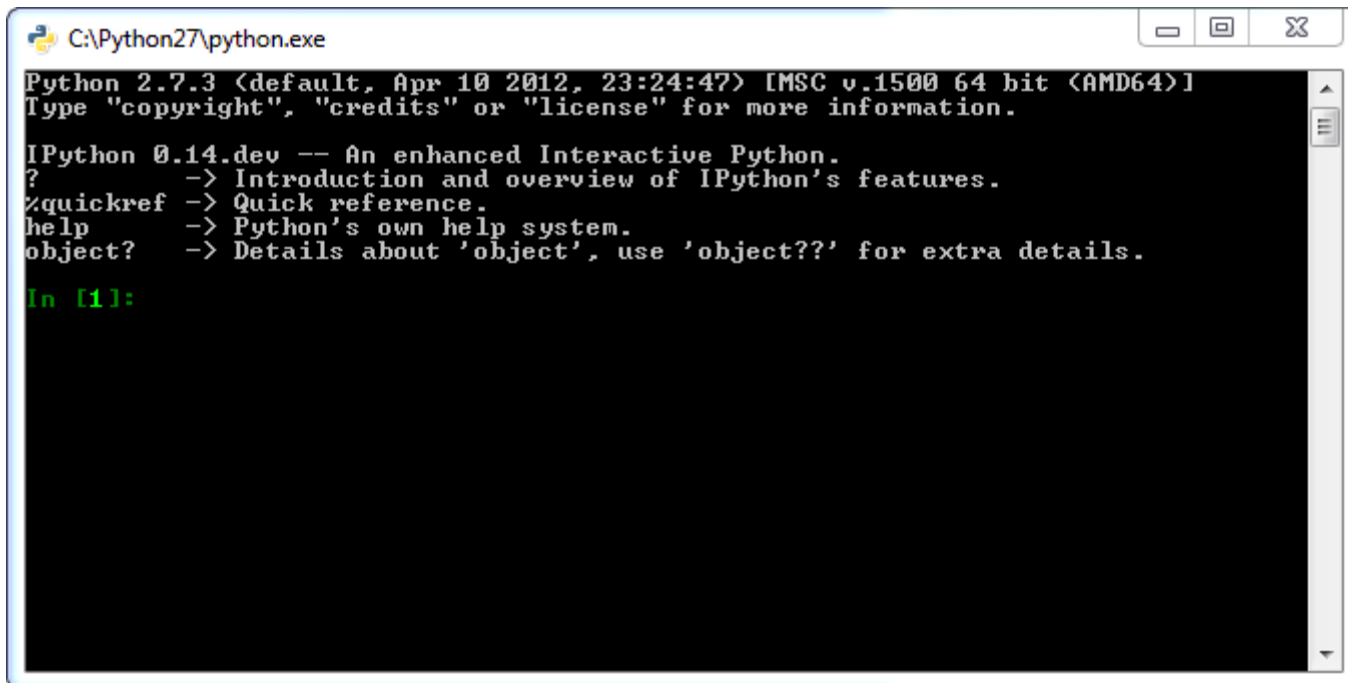
(like a shell script... works as long as the file *has execute permissions set*)

- Alternatively, you can enter a python shell and run python commands interactively, by typing “python”

Python Basics

- Start an interactive Python session, by typing the command “`ipython -i`” followed by a return
- or “`python -i`” <enter> from terminal window (mac, linux)
- or open a window in “Idle” (Windows).

After a few seconds, you will see a welcome message and a prompt:



```
C:\Python27\python.exe
Python 2.7.3 <default, Apr 10 2012, 23:24:47> [MSC v.1500 64 bit <AMD64>]
Type "copyright", "credits" or "license" for more information.

IPython 0.14.dev -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]:
```


Python Packages for science

- *NumPy* arrays, linear algebra, Fourier transforms
- *pandas* data structures, time series analysis
- *Matplotlib* and Seaborn data visualization
- Scikit learn modelling and machine learning
- ***TensorFlow* deep neural networks**

Python in Class Exercise

- Write a python script (“studentid.py”)
 - Read data file which i have sent you
 - Make a histogram of each column
 - Make a Gauss fit to each histogram
 - Sent your code and input file to amg@metu.edu.tr
- Deadline: 17:30 today