Phys 444
Computational PhysicsII

Introduction

Syllabus

COURSE OUTLINE

PHYS 444
Computational Physics II
Instructor: Prof. Dr. A. Murat Güler
Room: 333, e-mail: amg@metu.edu.tr
Class Schedule:

Thursday	12:40-15:30

Zoom Class: Meeting ID: 430 332 7559, Passcode: 327

Reference
-Computational Physics by Mark Newman
(http://www-personal.umich.edu/~mejn/cp/index.html)
-Numerical Methods for Engineers
Steven C. Chapra, Raymond P. Canale

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
- Root finding
  - Bisection method
    - The False-Position Method
  - Newton method
  - Secant method
  - Müller method
- Solution of Linear Systems
  - Gaussian elimination method
  - Gauss-Jordan method
- Numerical Differentiation
  - Finite Difference Method
    - Forward-Backward and Central difference
  - Richardson Extrapolation

Syllabus

- Numerical Integration
  - Midpoint method
  - Trapezoid method
  - Romberg method
  - Simpson method
  - Gaussian Quadrature
- Ordinary Differential Equations
  - ODE's types
  - Single Step Methods
    - Euler's method
    - Runge-Kutta method
  - Midpoint and Heun's Methods
  - Solution of Boundary-Value Problems
    - Shooting Method
    - Finite Difference Method
- Partial Differential Equations
  - PDE's types
  - Finite Difference Method
  - Crank-Nicolson Method
- Monte Carlo Techniques
- Fourier techniques

Grading

- Grading
  - Attandance : 5%
  - Homework : 25%
  - Midterm: 35 %
  - Final: 35%

Numerical Methods

Numerical Methods:

Algorithms that are used to obtain numerical solutions of a mathematical problem.

Why do we need them?

1. No analytical solution exists,
2. An analytical solution is difficult to obtain or not practical.

What do we need?

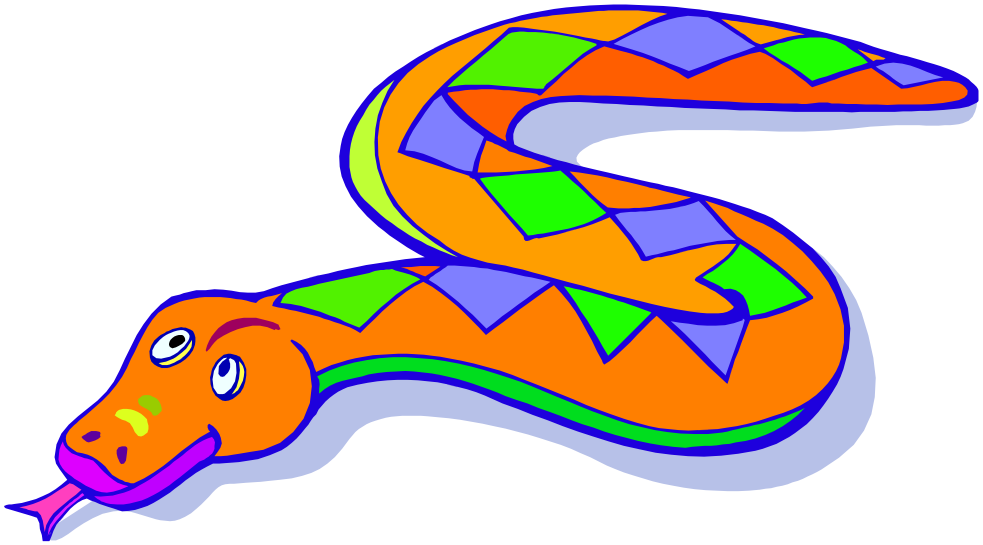
Basic Needs in the Numerical Methods:

- Practical:

  Can be computed in a reasonable amount of time.

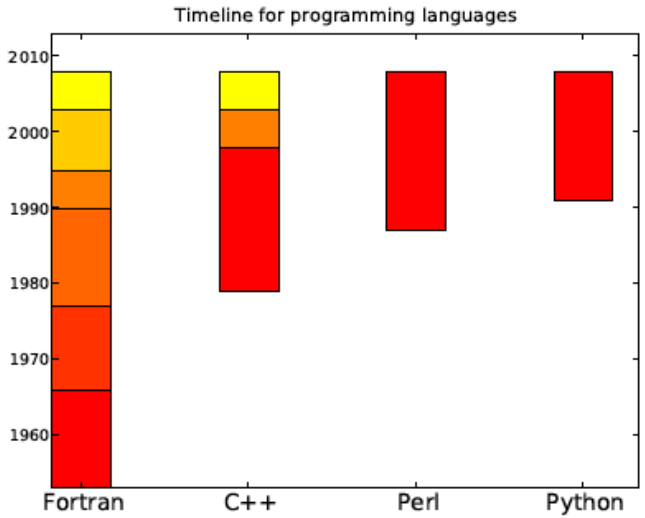
- Accurate:
  - Good approximate to the true value,
  - Information about the approximation error (Bounds, error order,…).

Python



Why Python?

- Recently it is gaining popularity in the world of scripting.
- Relatively young: first released by Guido van Rossum in 1991 (now at Google).
- 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/

Mar 2021	Mar 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	15.33%	-1.00%
2	1	↓	Java	10.45%	-7.33%
3	3		Python	10.31%	+0.20%
4	4		C++	6.52%	-0.27%
5	5		C#	4.97%	-0.35%
6	6		Visual Basic	4.85%	-0.40%
7	7		JavaScript	2.11%	+0.06%
8	8		PHP	2.07%	+0.05%
9	12	↑	Assembly language	1.97%	+0.72%
10	9	↓	SQL	1.87%	+0.03%
11	10	↓	Go	1.31%	+0.03%
12	18	⇈	Classic Visual Basic	1.26%	+0.49%
13	11	↓	R	1.25%	-0.01%
14	20	⇈	Delphi/Object Pascal	1.20%	+0.48%
15	36	⇈	Groovy	1.19%	+0.94%
16	**14**	↓	**Ruby**	**1.18%**	**+0.13%**
17	17		Perl	1.15%	+0.24%

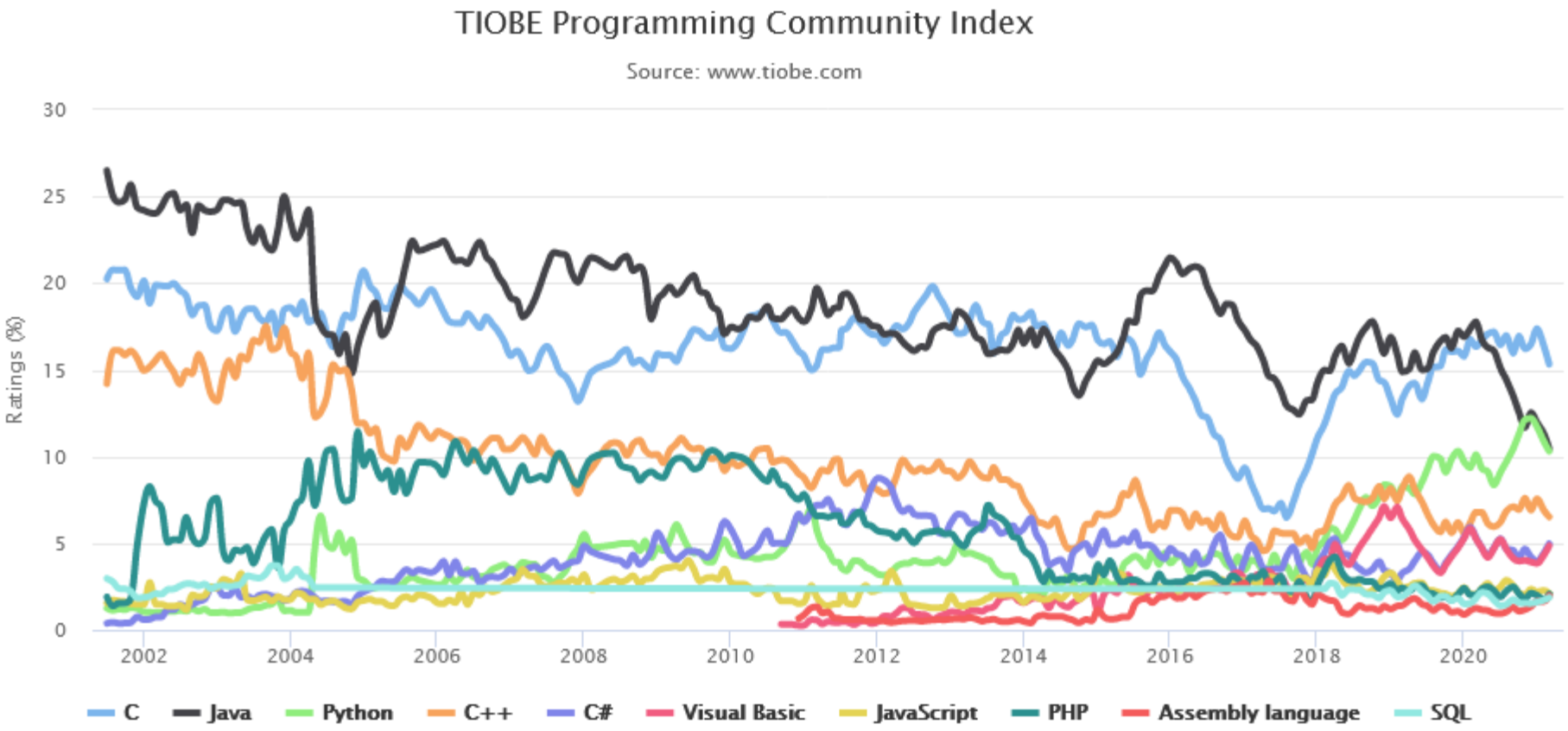
Programming Language Hall of Fame

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

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	28	-	-
Python	3	5	6	7	23	16	-	-
C++	4	3	3	3	2	2	2	8
C#	5	4	5	6	9	-	-	-
JavaScript	6	7	9	9	6	30	-	-
PHP	7	6	4	4	20	-	-	-
R	8	14	35	-	-	-	-	-
SQL	9	-	-	-	-	-	-	-
Go	10	56	15	-	-	-	-	-
Perl	14	8	7	5	4	3	-	-
Lisp	32	23	12	13	16	7	3	2
Ada	34	22	20	15	15	5	9	3

Which language is popular?

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



Programming Language Hall of Fame

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

The award is given to the programming language that has the highest rise in ratings in a year.

Year	Winner
2020	Python
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

Why Python?

- Advantages:
  - Free and open-source software, widely spread, with a vibrant community
  - Easy to learn
  - Very rich scientific computing libraries
  - Well thought out language, allowing to write very readable and well structured code.
  - High-level language
  - Dynamically typed language(No need to mention data type based on value assigned, it takes data type)
  - Object-oriented language
- Drawbacks:
  - Not all the algorithms that can be found in more specialized software or toolboxes.

Why Python?

Used by:

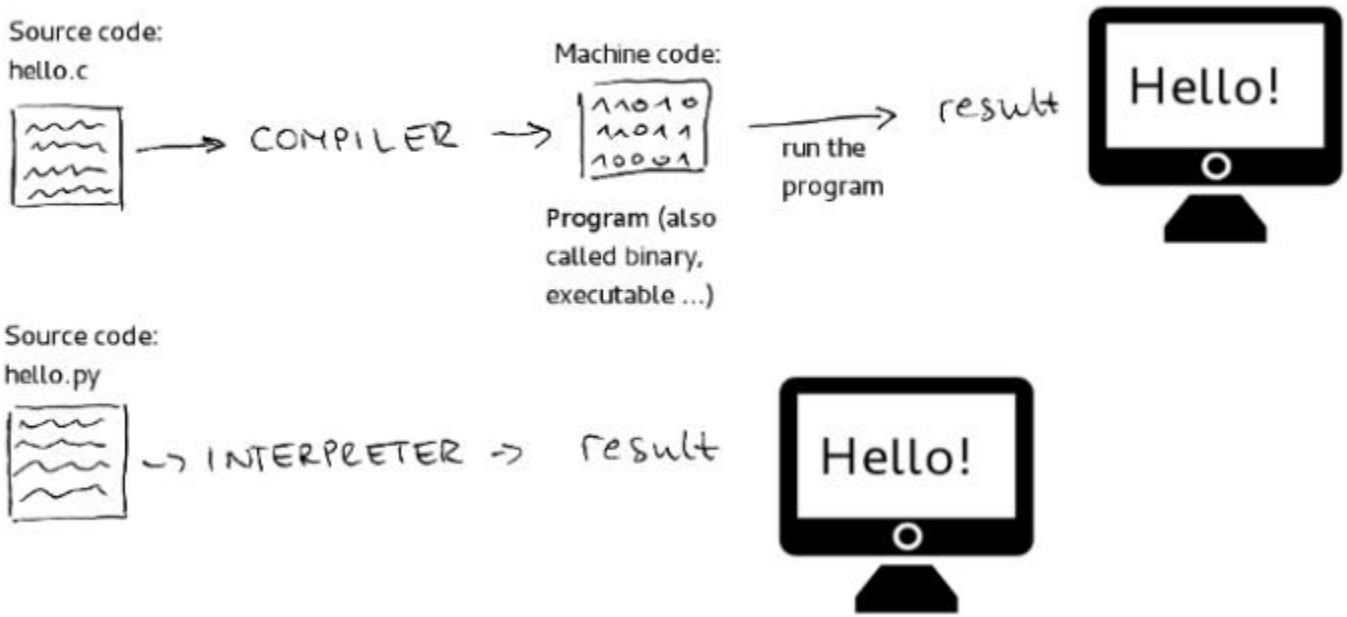
- Google, Yahoo!, Youtube
- Many Linux distributions
- Games and apps (e.g. Eve Online)

“Python is an experiment in how much freedom program-mers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered.”

- Guido van Rossum

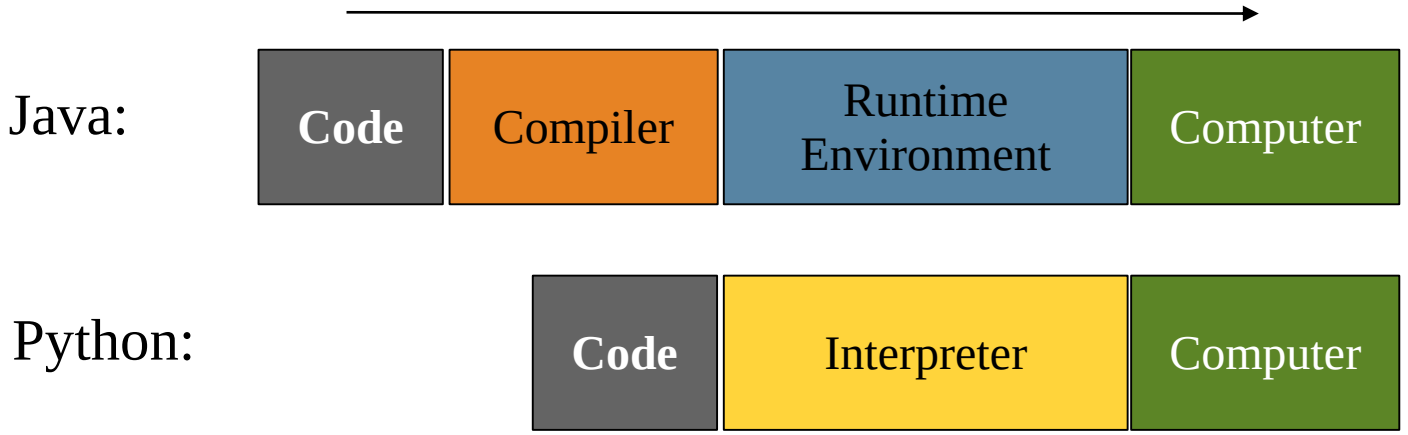
Interpreted Languages

- **interpreted**
  - Not compiled like Java, C, C++,...
  - Code is written and then directly executed by an **interpreter**
  - Type commands into interpreter and see immediate results



Compiled/interpreted

- 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. One statement at a time



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

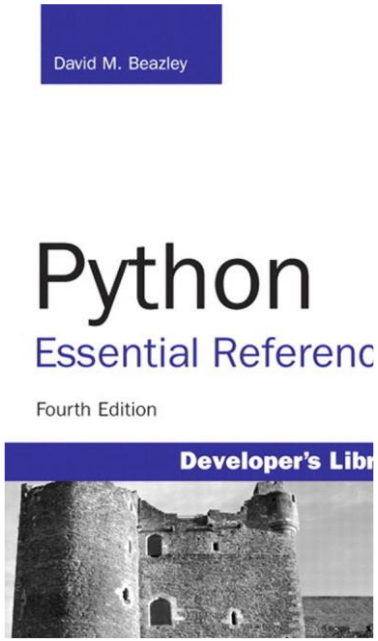
- Python 3.0 does not maintain backward compatibility with the older versions of Python
  - Thus, code developed for Python 2.X may not work with Python 3.X, and vice-versa.
- So much code and so many libraries have been developed for Python 2.X that most persons in the scientific community still use 2.X.
- Python 2.7, includes many features of Python 3.X, but maintains backward compatibility with older versions.

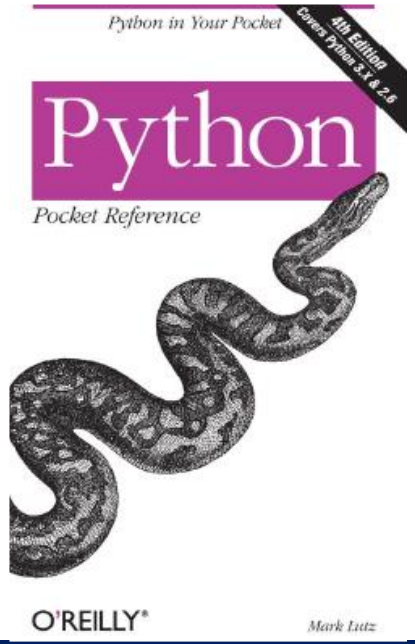
Python Documantation

- http://python.org/

documentation, tutorials, beginners guide, core distribution

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





Versions for our lecture

- Use the following versions & modules
  - Python 2.7 or Python 3.x
  - Numpy, Scipy, Matplotlib

In Class Exercise

- Create a Numpy arrray which starts at 0 and ends at 100 spaced by 1.

$y = v_o t + \frac{1}{2} g t^2$ (take $v_0 = 5$, $g = 10$)

- Plot a graph of y versus t using Pyplot
- Fit the plot with a polynomial

(Be sure to try out xlabel, ylabel, title, and grid commands)