Programming for Scientific Applications

based on slides from Gordon Erlebacher and Xiaquiang Wang

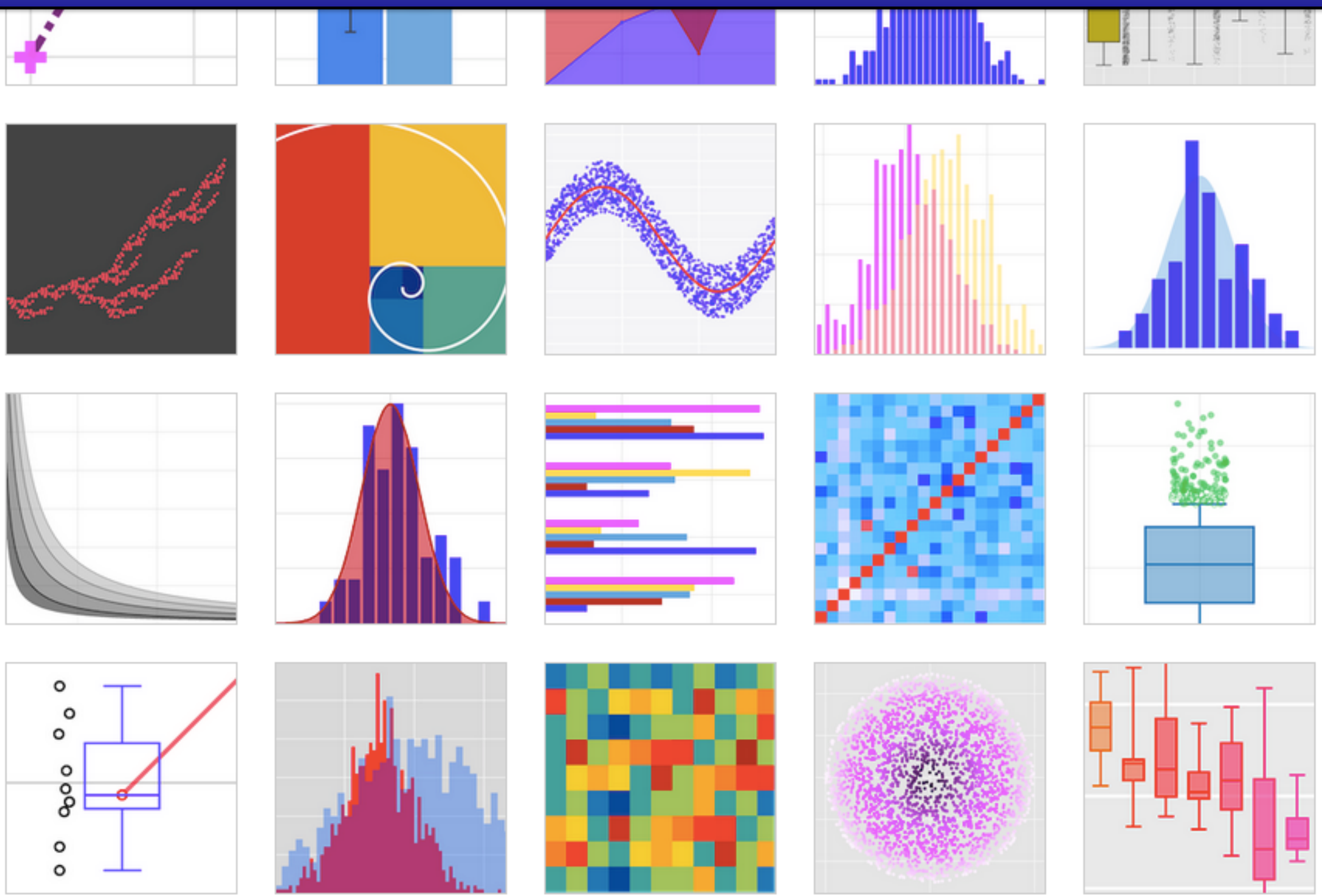

image from plotly

Syllabus

- Class participation (questions, suggestions, comments): might increase your grade if otherwise too low
- Homeworks (weekly with lapses): 30%
- Quizzes: 10%
- Labs: 30%
- Midterm: 20%
- Final: 20%

Class Attendance

- Attendance Mandatory
- Keep your cell phone in your pocket or purse

Class Objectives

- to understand the benefits of interpreted and compiled languages and know when to use each one to best advantage
- to understand Python sufficiently to program applications with confidence
- to understand C++ sufficiently to program applications
- learn to interface C++ and Python to each other, to take advantage of the best features of both languages
- through lab work, develop the skills to apply Python and C++ to a range of practical scientific applications, ranging from graphical user interfaces, web-based display of results, processing of scientific data, and visualization

Honor Code

- The Academic Honor System of The Florida State University is based on the premise that each student has the responsibility 1) to uphold the highest standards of academic integrity in the student's own work, 2) to refuse to tolerate violations of academic integrity in the University community, and 3) to foster a high sense of integrity and social responsibility on the part of the University community. Please note that violations of this Academic Honor System will not be tolerated in this class. Specifically, incidents of plagiarism of any type or referring to any unauthorized material during examinations will be rigorously pursued by this instructor. Before submitting any work for this class, please read the ``Academic Honor System" in its entirety (as found in the *FSU General Bulletin* and in the *FSU Student Handbook* and ask the instructor to clarify any of its expectations that you do no understand.

Outline

- Class Work (Tuesday, Thursday)
  - fundamentals of Python and C++
  - example-based
  - use of web to supplement class material
- Lab work (Tuesday)
  - illustration of Python and C++ in practical situations
  - 2.5 hours per week
  - write a report about lab (summary, pictures, tables), and also deliver code
  - each lab: 1-2 weeks. Reports due weekly

Class Contents

- Overview (today)
- Modules, Operating system
- lists, dictionaries, sequences
- functions
- text processing
- files, I/O
- numpy for arrays
- classes
- numpy for statistics
- C/C++ and wrapping

What is programming?

- What is a computer program?
  - "A set of coded **instructions** that enables a machine, especially a computer, to perform a desired **sequence** of operations." – American Heritage Dictionary

- Programming instructions are written using a "programming language"
  - Examples: C/C++, Java, Assembly, Fortran, Cobol, BASIC
  - LOTS of programming languages, different uses for different languages

C++

Back to index

```
// Hello World in C++ (pre-ISO)

#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Haskell

Back to index

```
-- Hello World in Haskell

main = putStrLn "Hello World"
```

Lisp

Back to index

```
 ;;; Hello World in Common Lisp
 (defun helloworld ()
   (print "Hello World!")
 )
```

Assembler-Linux

Back to index

```
     ;; Hello World for the nasm Assembler (Linux)

     SECTION .data

     msg  db   "Hello, world!",0xa ;
     len  equ     $ - msg

     SECTION .text
     global main

main:
        mov     eax,4       ; write system call
        mov     ebx,1           ; file (stdou)
        mov     ecx,msg         ; string
        mov     edx,len         ; strlen
     int     0x80       ; call kernel

     mov  eax,1         ; exit system call
        mov     ebx,0
        int     0x80        ; call kernel
```

Terminology

- **Computer program** – a set of instructions that tell a computer exactly what to do
  - The instructions might tell the computer to add up a set of numbers, or compare two numbers and make a decision based on the result, or whatever.
- **Programming language** – a language used by humans to program computers
  - e.g., Fortran, Cobol, Basic, Pascal, C, C++, Java, Perl
- **Compiler** – translates a computer program written in a human-readable computer language (like C++) into a form that a computer can execute
  - You have probably seen .exe files or .app 'files' on your computer.
  - These executable files are the output of compilers.
  - They contain executables -- machine-readable programs translated from human-readable programs.

Programming

- Problem solving
  - Logical/methodical way of solving a problem
- Algorithm/abstraction
  - An algorithm is a series of step-by-step instructions that produces a solution to a problem
- Step wise refinement
  - Incrementally adding functionality to a program

Five steps to writing a program

- Define the problem
- Plan the solution
  - pseudocode
- Code the program
  - Using a programming language
- Test and debug
  - Using a compiler (C++), or interpreter (Python)
- Document

Python vs C++

- Python is interpreted
- C++ is compiled

Python vs C++ and other languages

- Python is interpreted (and Julia)
- C++ is compiled (and D)

Hello World

- In a file named **HelloWorld.py**
  - print ‘Hello World’

Scripting vs traditional

- Traditional programming
  - C, C++, Fortran, Java (kind of), C#
  - mostly computation, networking, other low-level activities
  - traditional code if often wrapped with scripting code for ease of use and integration into scripting framework
- Scripting framework
  - Perl, Python, Ruby, Scheme, Julia, Tcl, ...
  - Integrates text processing, I/O, report writing, and computation
  - Often sections of programs written in scripting languages are translated to traditional code for efficiency

Why Scripting?

- Easier to use
- Faster development time
- Avoid compilation and linking
- Integration of visualization, networking, datea analysis, etc.
- Python makes it simple to *glue together* different applications (plenty of tools, modules)
- Scientific computing is more than number crunching. In addition:
  - data manipulation, data analysis, visualization, format conversion, parametric studies, cataloguing, database access, etc.
  - these tasks are much much easier in Python than C++, Fortran, etc.
- Graphical user interfaces (use of Tk to wrap existing programs)

Language Classification

- Scripting languages: dynamically typed
  - variables types are not declared
  - syntax closer to natural language
- Traditional programming languages: type safe
  - declare variable types
  - syntax closer to the hardware

Efficiency

- Scripting languages
  - first compiled to byte code (independent of OS)
  - byte code is interpreted, line by line
  - better error messages
  - in general, codes run slower (not important for short codes)
  - sometimes speed is optimal (e.g., regular expressions)
  - inefficient sections of code can be rewritten in C/C++/Fortran/Java
- Traditional (compiled) languages
  - source code is translated to machine code (closer to the hardware)
  - machine code is hardware-dependent
  - in general, codes runs MUCH faster

Variable Declaration

- C, C++: type-safe languages
  - protect the user against himself
  - less bugs, safer programming
  - code reuse is harder (types are set in stone)
  - somewhat relaxed with classes and templates (for C++, but not for C)
- Python, Lua, etc.
  - when a variable is needed, assign a value
  - type is determined by the value
  - type conversion is sometimes automatic (Perl), sometimes not (Python)
  - same piece of code can be used in many different contexts