Physics 29 Fall 23
Week One

Claudio Campagnari

Class Website (instead of Canvas):
http://hep.ucsb.edu/people/claudio/ph29-f23/
Results of the survey

Do you have any previous experience with coding? How much?
38 responses

- 47.4%: Yes, I have done a lot of coding in the past
- 21.1%: Yes, I have done some coding, but I do not feel very confident in my skills
- 13.2%: Yes, I took a class/tryed to teach myself, but I did not get very far
- 18.4%: No, I am a total beginner
Winter 23 grade distribution

Physics 29 Winter 2023

N=64
Mean=79.4
Std=26.4

def letterGrade(s):
    if s >= 97: return "A+
    if s >= 85: return "A"
    if s >= 80: return "A-"
    if s >= 75: return "B+
    if s >= 65: return "B"
    if s >= 60: return "B-"
    if s >= 55: return "C+
    if s >= 45: return "C"
    if s >= 40: return "C-"
    if s >= 35: return "D"
    return "F"
Why programming for physics?

1. Data analysis
   - Manipulate/process data
   - Analyze data (statistics, fitting)
   - Plot/Visualize data

2. Numerical solutions
   - Equations
   - Integrals, derivatives, (partial) differential equations

3. Simulations

4. Data acquisition

5. Equipment control

In this class you will get some basic coding skills and apply them to some examples from categories 1, 2, and 3.
A program is written in pseudo-English with a text editor

```c
double_t xbinsize = 10.;
double_t ybinsize = 10.;
double_t xmin = 250. - xbinsize/2.;
double_t xmax = 450. + xbinsize/2.;
double_t ymin = 100. - ybinsize/2.;
double_t ymax = 300. + ybinsize/2.;
int_t nx = (xmax-xmin)/xbinsize;
int_t ny = (ymax-ymin)/ybinsize;

// Upper limit as a function of gluino and lsp mass
TH2F* ul = new TH2F("ul","ul",nx,xmin,xmin+nx*xbinsize,ny, 
```
```python
# Sensible bins
if histLow == None:
    histLow = min(np.amin(sTrain), np.amin(bTrain), 
                  np.amin(sTest), np.amin(bTest))
if histHigh == None:
    histHigh = max(np.amax(sTrain), np.amax(bTrain), 
                  np.amax(sTest), np.amax(bTest))
bins = np.linspace(histLow, histHigh, nBins+1)
binCenters = 0.5 * (bins[1:] + bins[:-1])

# Now we make a plot that looks very much like what
# comes out of TMVA.
# There is no straightforward way to plot a histogram
# as "points" so for the training samples we need to
# do some gymnastics, sigh
```
```fortran
C--
Log-likelihood fit, therefore error is 0.5
arglis(1) = 0.50d0
call mnexcm(racen, 'SET ERR', arglis, 
       C--
call FCN with flag = 1 at the beginning
arglis(1) = 1.00d0
call mnexcm(racen, 'CALL FCN', arglis, 1, ierr, 0 
       print *, ' --- Done Initializing Minuit ---'
```
```perl
C--
** Check if there is already a root file with that name
set temp1 "res_S_*
set temp2 "root"
set rootFile $dataDirtmp$rodNumber$temp2
puts $rootFile
if {[file exists $rootFile]} {
    set m1 "It will make a file $rootFile. \n" 
    set m2 "Such a file already exists. \n"
    set m3 "Do you want to proceed anyway? \n"
```
Interpreted vs compiled languages

• **Interpreted languages** (perl, tcl, python): run the program (text file) by “invoking it” from the command line (or through a GUI)
  • The text file is “read” by an interpreter, who (effectively) turns it into machine code on the fly

• **Compiled languages** (C, C++, fortran): turn the text file into machine code with “compile” and “linker”, get an “executable” (machine code), run the executable
Python

• Python is (effectively) an interpreted language
• A python program is typically much slower than a compiled C or C++ program
• So why do we use it?
  • In many (not all!) cases speed does not matter much
  • There are “packages” that can be brought into python that are (effectively) compiled C or C++ code that run fast “under the hood”
  • It has a modern nice interface
    • Beauty is in the eye of the beholder
  • It is very popular
    • Popular today. Tomorrow, who knows (Julia anyone?)
    • Can find many examples and help on the internet
  • It has many prepared packages that we can use
Jupyter notebooks

• The jupyter-notebook program starts a server on your computer
• You connect to the server with your browser to open a “notebook”
• The notebook has cells where you can enter (python) code or even comments
• You can run python over all the cells at once
• Or one cell at a time
• You can modify cells, and run it again
• etc.
• A very convenient and efficient way to develop and debug your code
$x = 4$

A location in memory that represents the number 4 with some bit

bound to

a (variable) name
A location in memory that represents the number 4 with some bits joined to a (variable) name.

Another variable:

\[ y = x \]

\[ y \text{ is } x \quad \text{True} \]
A location in memory that represents the number 4 with some bit bound to a (variable) name.

Another variable bound to 20 bound to

y = 20

y is x True
A location in memory that represents the number 4 with some bit bound to a (variable) name.

y = x
y is x True
y = 20

Another variable

x = 4
x
bound to
4

y
bound to
20

x = 4
y = x

Other languages, eg, C

x
bound to
4

y
bound to
4

x = 4
y = x

y
bound to
20

 Doesn’t matter much for simple variables like numbers. Can be important for more complex variables. Can cause bugs. We’ll get to it later.