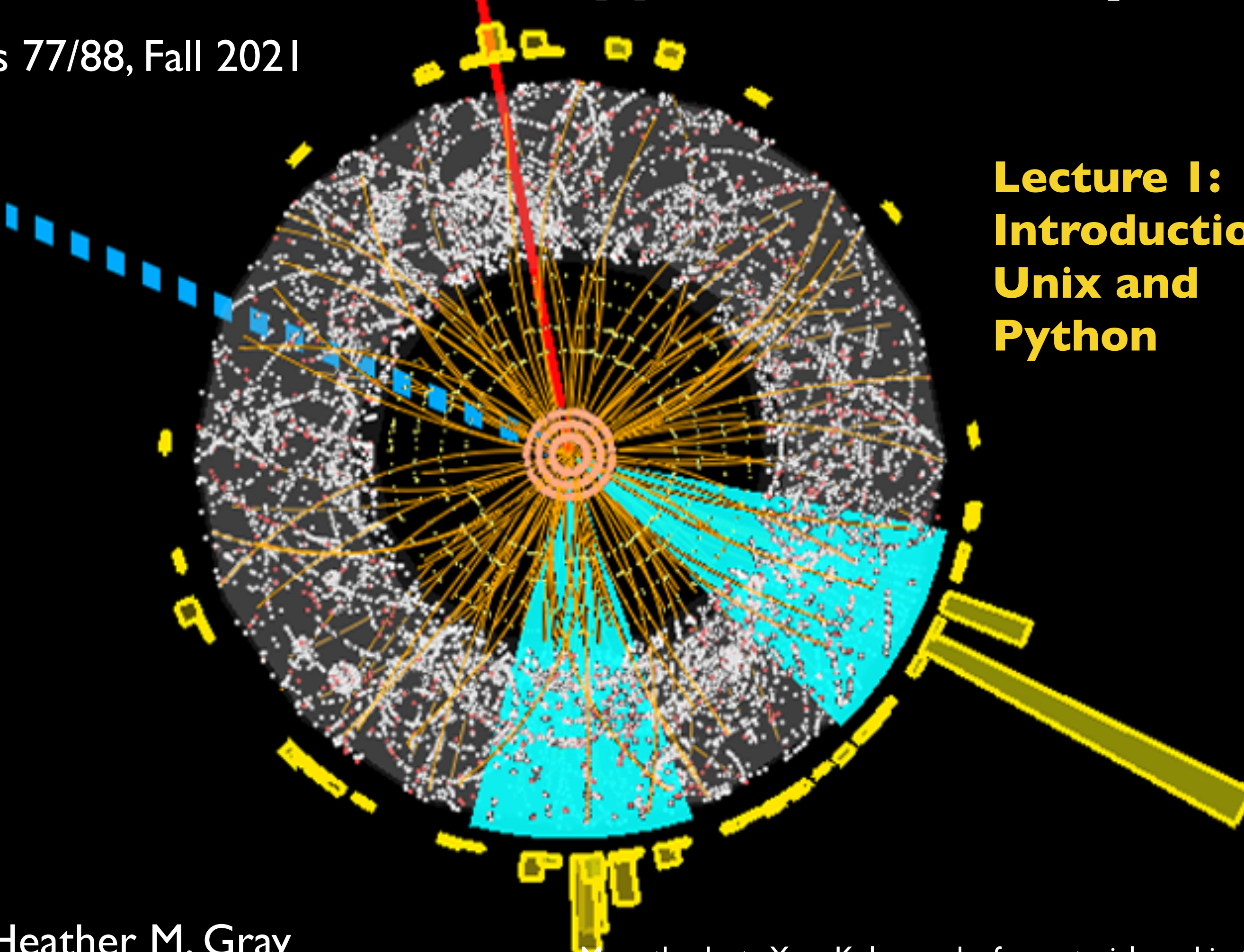
Introduction to Computational Techniques in Physics/Data Science Applications in Physics

Physics 77/88, Fall 2021



Lecture 1: Introduction to Unix and Python

Prof. Heather M. Gray

Many thanks to Yury Kolomensky for material used in this course

The What and Why of Computing

Why Computing?

- Necessary tool
  - Physics = experimental science
  - Observations → laws
    - Make a set of measurements or observations
      - Summarize the results
    - Most conclusions are drawn with some degree of (un)certainty
      - e.g. $F = G_N \frac{m_1 m_2}{r^2}$
      - In reality, we know G_N to some precision
        - e.g. $G_N = 6.6742 \pm 10 \times 10^{-11}\ N m^2/kg^2$
  - Many measurements are a priori uncertain (e.g. quantum physics)
    - Have to be interpreted in probabilistic terms

- Many measurements require automated collection of data and complicated analysis algoritms
  - → Computerization
- (virtually) all modern experiments are computerized
- And theorists use computers too

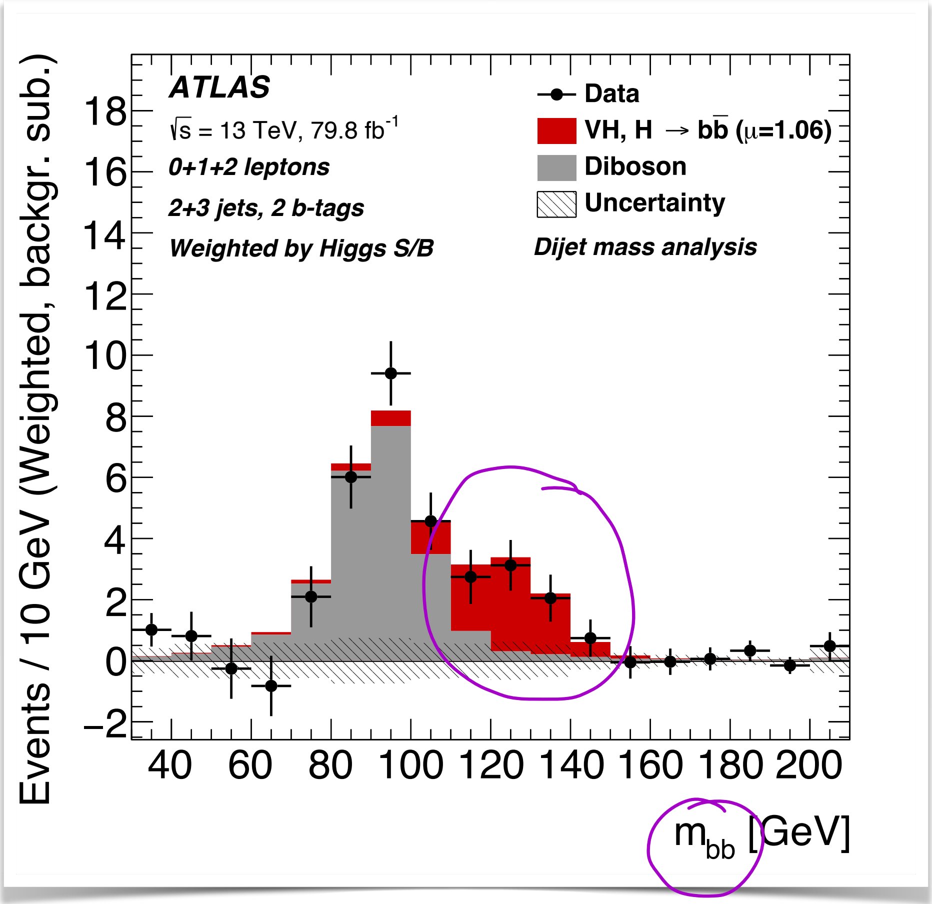
My First Exposure to Computing

IBM personal computer

Floppy disks

BASIC programming

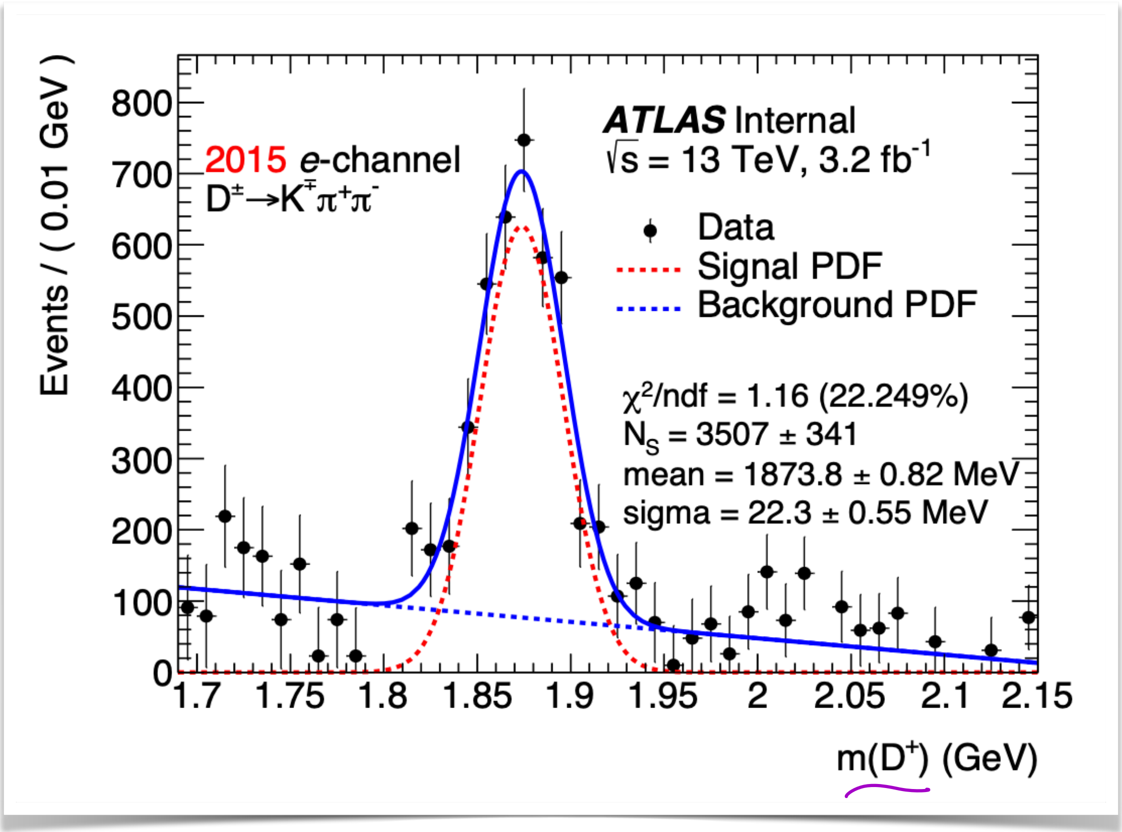
Examples (My Research)



Observation of the Higgs boson.

* Decay to bottom quarks

Examples (My Research)

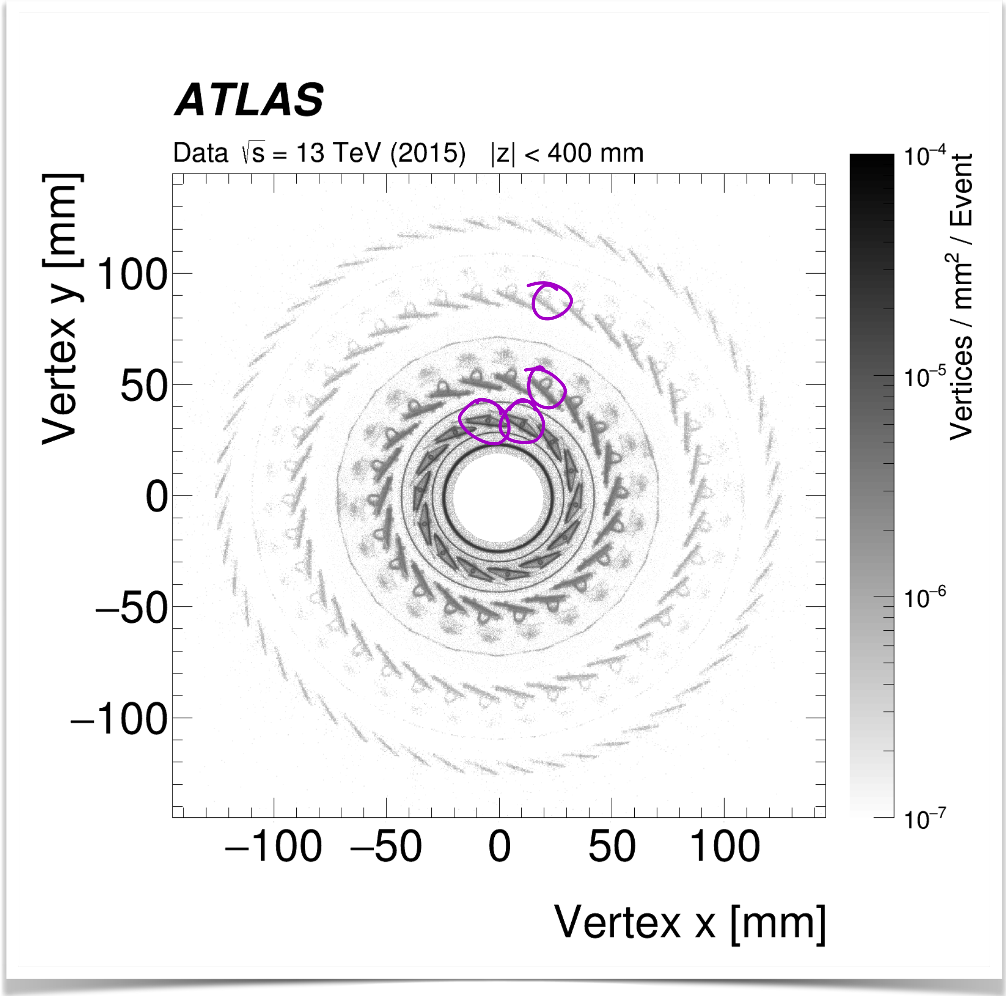


Fit to the data to find particles of a certain type.

M. Muskinja (PostDoc)

Examples (My Research)

Map out the detector



Describing the Data

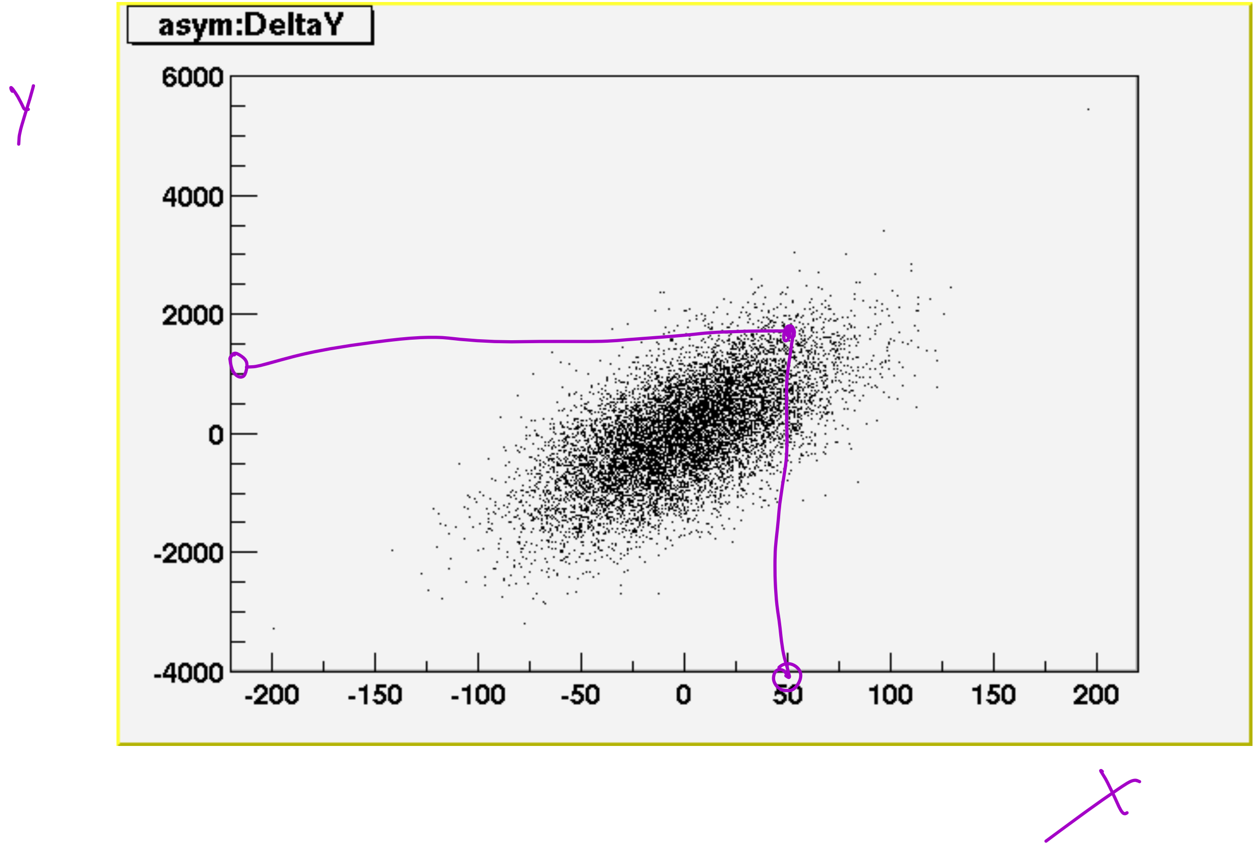
- Data: results of measurements
  - In physics we mostly deal with quantitative data, i.e. numbers
  - Other fields may deal with qualitative data
    - An American Robin has gray upper parts and head, and orange under parts, usually brighter in the male
  - Numbers are easier to handle mathematically
    - We will mostly deal with quantitative measurements
- Types of quantitative data
  - Discrete data, e.g. integers (count)
  - Continuous data, e.g. energies, momenta
    - Some precision, e.g. from measuring apparatus
  - Lexical data, e.g. words
  - Sets of data : arrays, tuples, associative sets → databases

Visualizing the Data

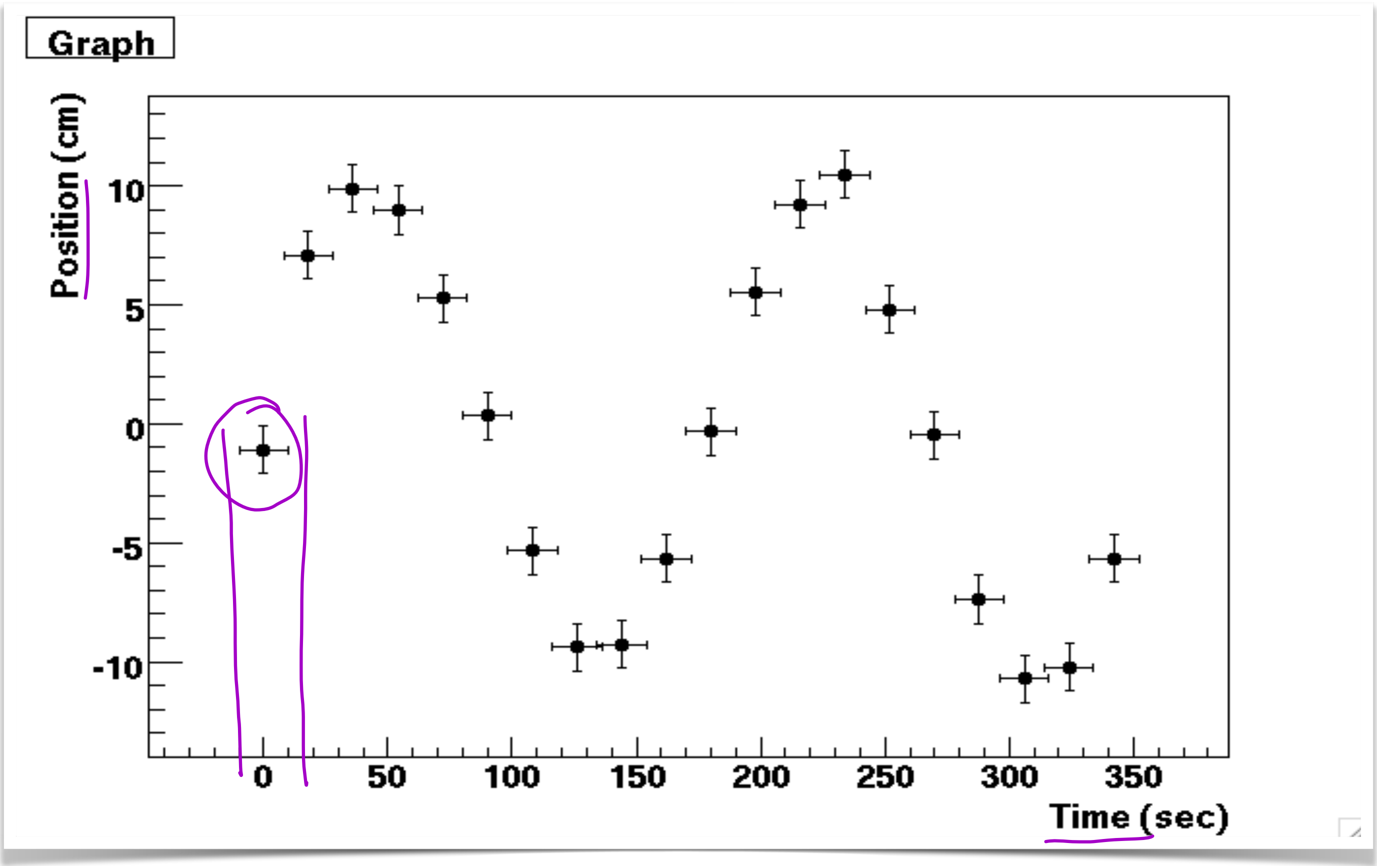
- Tables
  - Large datasets can be difficult to parse
- Graphs
  - Trends, variable dependence
- Charts
  - Frequency distribution

Examples

scatter plot

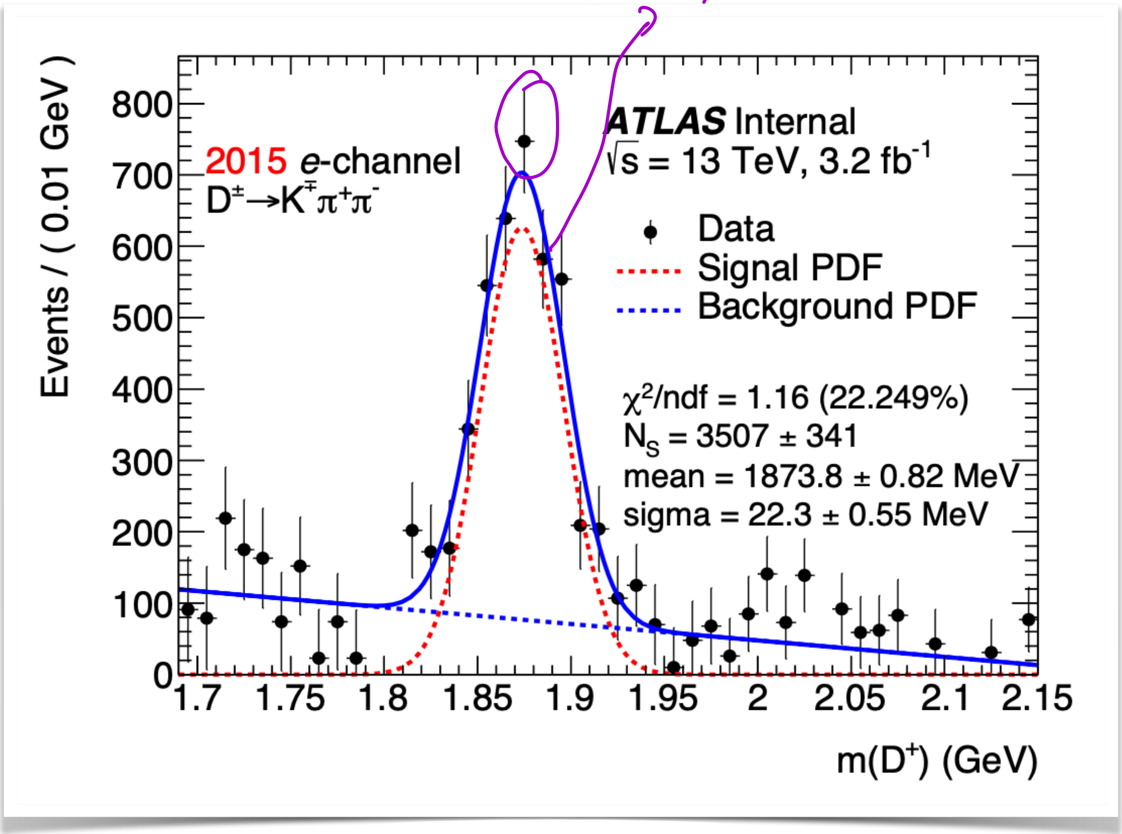


Examples

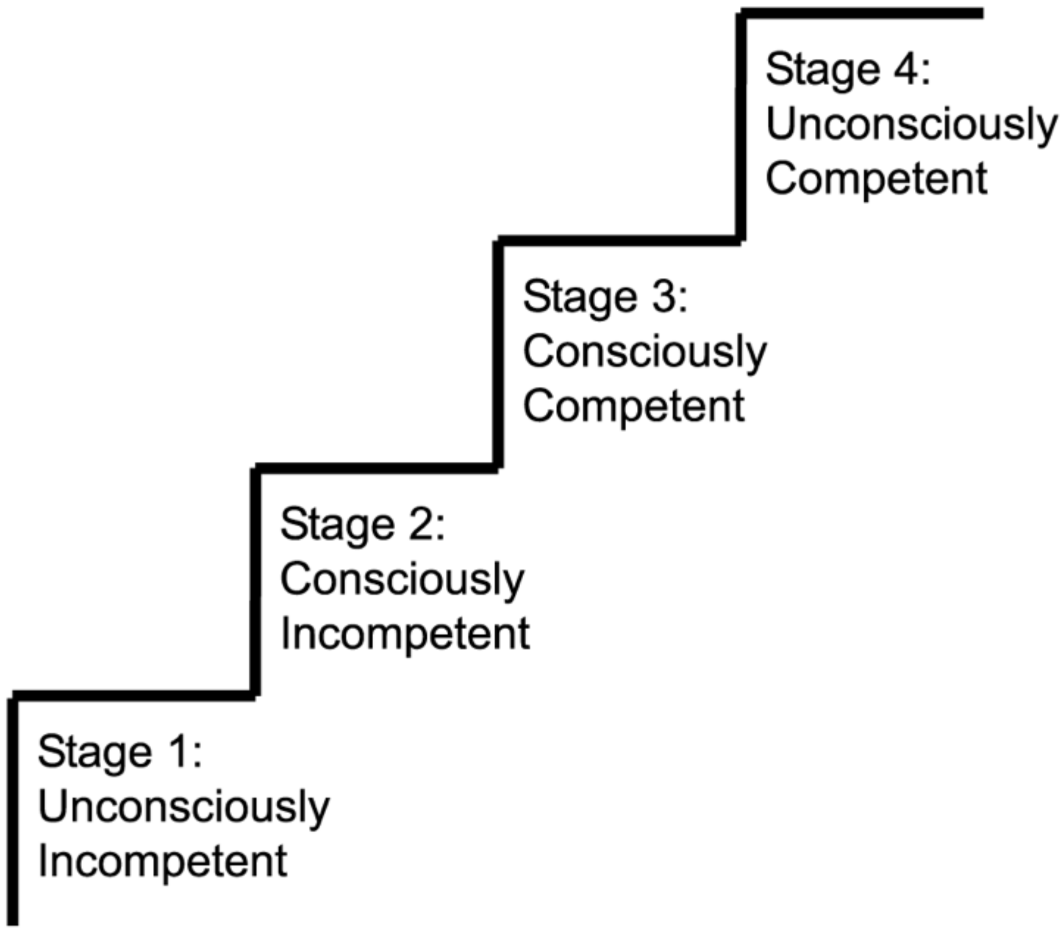

Graph
Position (cm)
10
5
0
-5
-10
0
50
100
150
200
250
300
350
Time (sec)

Examples

fit



How You Should Learn



How to Approach This Course

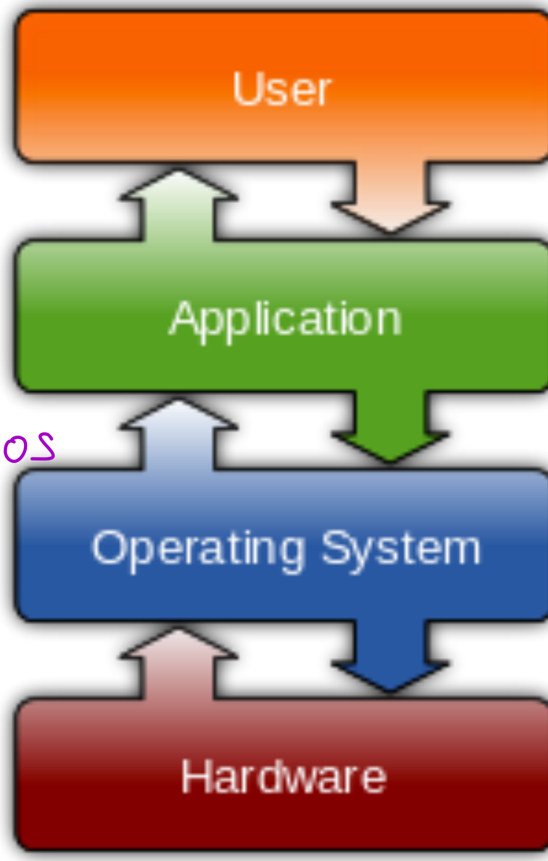
- Active Learning
  - Learn by
- Three main threads to the course
  -
  -
  -
- No single book covers all of it
  - Recommend two (Newman, Hughes & Hase)
  - Many online resources for each step
    - Book → Lecture → Workshop → HW → Book cycle applies

Let's take a look at the jupyter notebook

Operating Systems, Programming Environments, Representations

Brief Spiel: Operating Systems

- Operating System (OS) is an interface between a human and a computer
  - Translate human commands to electronic signals, report results back
  - Optimized for hardware, efficient
    - Low-level code
  - Interface can be graphical (Windows, OSX, iOS, Android), text-based (MS-DOS), or mixed (unix)



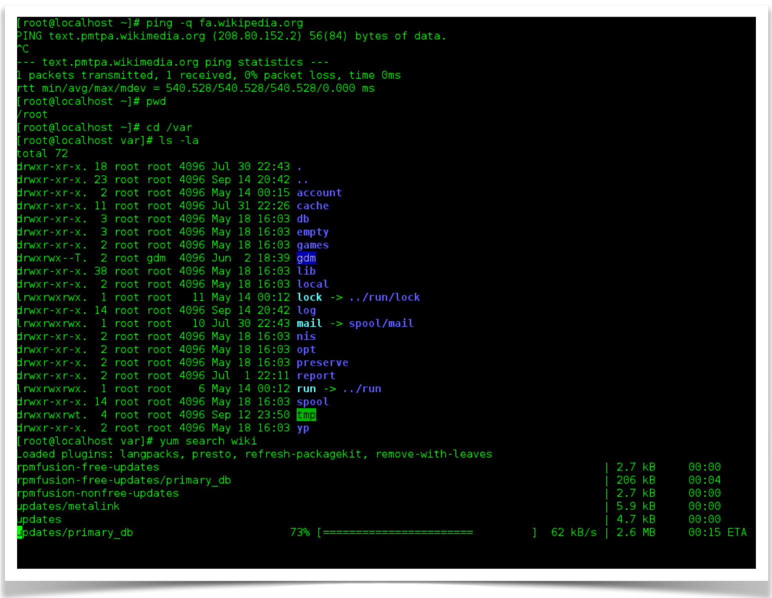
Source: Wikipedia

Unix/Linux

- One of the oldest (surviving) OS ( )
- Initially developed for mainframes , ported to PCs (Linux and spinoffs)
- Variants are now running on a variety of hardware
    - PC (Linux)
    - Mac (OSX)
    - smartphones (Android , iOS)
    - even the occasional microwave
- Robust, efficient
- Designed to be text-based , so basic interface is a command-line shell.

Programming Environments

- Command-line interface
- Graphical User interface GUIs
- Connecting to a server (ssh, terminal)
- Scripts
- Compiled vs interpreted languages



Source: Wikipedia

Brief Spiel: Programming Languages

- Programming languages allow humans to translate sets of instructions (an algorithm) to a form understandable by a computer
- Classifications
  - Procedural
    - BASIC, Fortran, Pascal, C
  - Object-oriented
    - C++, java, python
- Implementations
  - Compiled
    - Fortran, C, C++, java, cobol, pascal
  - Interpreted
    - BASIC, bash/csh, javascript, python

shell scripts

Data Representation

- Data on computers represented in binary format
  - Base 2 representation (as opposed to base 10)
    - $abcd_2 = a \times 2^3 + b \times 2^2 + c \times 2^1 + d 2^0$
      $a \times 10^3 + b \times 10^2 + c \times 10^1 + d\, 10^0$
  - Examples:

Decimal numbers	Binary equivalent	Decimal numbers	Binary equivalent
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

  - Smallest memory cell:
    - 8 bits = 1 byte (B)
  - Measures of memory
    - 1 kB = 1024 B ; 1 MB = 1024 kB, etc

Binary Representation

- Practical consequences
  - All numbers in the digital format are discrete , i.e. they have finite precision
    - This is easy to understand with integers (discrete by construction)
    - Real numbers: think about representation in powers of 2
      - 0.125 = 2^{-3}
    - Could you represent 1/10 in powers of 2 ? What about π ?
  - Basic data types have max and min value, as well as precision, determined by the data type size
    - i.e. how much memory is allocated for each data type
    - Most common: 4 or 8 bytes for integer and real values

Examples: Integer Data

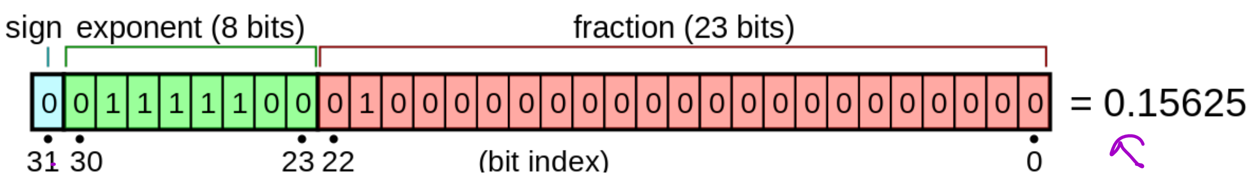
- C/C++

```
root [0] sizeof(int) // number of bytes for interger
(const int)4
root [1] sizeof(long) // number of bytes for long integer
(const int)8
root [2]
```

- This means:
  - Range of signed ints in C is ±2147483647
  - Range of unsigned ints in C is 0..4294967295
  - E.g. time in Unix is represented as — in seconds from Jan 1, 1970 . Hence the impending end of time in
    - (Google "end of unix time")

Examples: Real numbers

- Real (floating points) numbers are represented by 3 fields
  - sign
  - exponent
  - fraction, mantissa



$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}$$

- For example, most languages use 4-byte and 8-byte floating point numbers
  - float and double in C/C++

Precision and Range

Property	Value for float	Value for double
Largest representable number	3.402823466e+38	1.7976931348623157e+308
Smallest number without losing precision	1.175494351e-38	2.2250738585072014e-308
Smallest representable number(*)	1.401298464e-45	5e-324
Mantissa bits	23	52
Exponent bits	8	11
Epsilon(**)	1.1929093e-7	2.220446049250313e-16

http://www.cprogramming.com/tutorial/floating_point/understanding_floating_point_representation.html

What about Python ? Let’s go back to the Notebook