

# **DSC 102**

# **Systems for Scalable Analytics**

Fall 2023

Haojian Jin

Now for the course logistics ...

# Prerequisites

- ❖ **DSC 100** (or equivalent) is necessary
- ❖ Transitively **DSC 80**; a mainstream ML algorithmics course is necessary
- ❖ Proficiency in Python programming
- ❖ For all other cases, email me with proper justification; a waiver can be considered

<https://haojian.github.io/DSC102FA23/>

# Components and Grading

- ❖ **3 Programming Assignments: 40%** (8% + 16% + 16%)
  - ❖ No late days! Plan your work well ahead.
  - ❖ **Plan your credit as well!**
- ❖ **Midterm Exam: 15%**
  - ❖ TBD; in-class only (50min)
- ❖ **Cumulative Final Exam: 35%**
  - ❖ Dec. 15; Canvas Quiz only (3hrs long but 4hrs limit)
- ❖ **10 (of 12) Peer Instruction Activities: 10%**
- ❖ **Extra Credit Peer Evaluation Activities: 4%** (likely)
- ❖ LMK ahead of time if you need makeup exam slot

<https://haojian.github.io/DSC102FA23/>

# Grading Scheme

Hybrid of relative and absolute; grade is better of the two

Grade	Relative Bin (Use strictest)	Absolute Cutoff (>=)
A+	Highest 5%	95
A	Next 10% (5-15)	90
A-	Next 15% (15-30)	85
B+	Next 15% (30-45)	80
B	Next 15% (45-60)	75
B-	Next 15% (60-75)	70
C+	Next 5% (75-80)	65
C	Next 5% (80-85)	60
C-	Next 5% (85-90)	55
D	Next 5% (90-95)	50

**Example:** Score 82 but 33%ile; Rel.: B-; Abs.: B+; so, B+

# Programming Assignments

- ❖ **PA0: Setting up AWS and Dask**

- ❖ Oct 6 to Oct 22

- ❖ **PA1: Data Exploration with Dask**

- ❖ Oct 23 to Nov 12

- ❖ **PA2: Feature Eng. and Model Selection with Spark**

- ❖ Nov 13 to Dec 04

- ❖ **Expectations on the PAs:**

- ❖ Teams of 1-3; see webpage on academic integrity

- ❖ I will cover the concepts and tools' tradeoffs in the lectures

- ❖ TAs will explain and demo the tools; handle all Q&A

- ❖ You are expected to put in the effort to learn the details of the tools' APIs using their documentation on your own!

<https://haojian.github.io/DSC102FA23/>

# Course Administtrivia

- ❖ **Lectures: MWF 3pm-3:50pm PT at CENTR 214**
  - ❖ Attendance optional but encouraged; podcast available
  - ❖ No need for clickers.
- ❖ **Discussions: Friday 9-9:50am PT on Zoom**
  - ❖ Only for talks on PAs by TAs, for pre-exam review by me
- ❖ **Instructor:** Haojian Jin; haojian@ucsd.edu
  - ❖ OHs: **Wednesday 4-5 pm PT at 341 HDSI**
- ❖ **Slack** for all communications
- ❖ **Canvas** for PA submission, Peer Evaluation Activities, Final Exam

<https://haojian.github.io/DSC102FA23/>

# Office hours

- ❖ Rohit Ramaprasad's OHs: Tuesday 1:00 PM - 2:00 PM
- ❖ Sai Sree Harsha's OHs: Thursday 1:00 PM - 2:00 PM
- ❖ Golokesh Patra's OHs: Monday 11:00 AM - 12:00 PM
- ❖ Post questions to the ta-public channel.
- ❖ Avoid asking repetitive questions.

<https://haojian.github.io/DSC102FA23/>



# General Dos and Do NOTs

## **Do:**

- ❖ Follow all announcements on Piazza
- ❖ Try to join the lectures/discussions live
- ❖ Raise your hand before speaking
- ❖ View/review podcast videos asynchronously by yourself
- ❖ To contact me/TAs, use private Slack; if you really need to email, use “DSC 102:” as subject prefix

## **Do NOT:**

- ❖ Harass, intimidate, or intentionally talk over others
- ❖ Violate academic integrity on the PAs, exams, or other components; I am *very strict* on this matter!

# Reasonable person.

- (1) Everyone will be reasonable.
- (2) Everyone expects everyone else to be reasonable.
- (3) No one is special.
- (4) Do not be offended if someone suggests you are not being reasonable.

Now for the course structure ...

## DSC 102 will get you thinking about the fundamentals of systems for scalable analytics

1. **“Systems”**: What resources does a computer have? How to store and efficiently compute over large data? What is cloud?
2. **“Scalability”**: How to scale and parallelize data-intensive computations?
3. **For “Analytics”**:
  1. **Source**: Data acquisition & preparation for ML
  2. **Build**: Model selection & deep learning systems
  3. **Deploying** ML models
4. Hands-on experience with scalable analytics tools

# Data Systems Concerns in ML

## Key concerns in ML:

**Q:** How do “ML Systems” relate to ML?

## Runtime efficiency (sometimes)

## Additional key *practical* concerns in ML Systems:

ML Systems : ML :: Computer Systems : TCS

# Usability

# Manageability

# Developability

*Long-standing  
concerns in the  
**DB systems**  
world!*

**Q. Q. What do you think will be the top 5 emerging technologies for the next 5 years?**

# Conceptual System Stack Analogy

## Relational DB Systems

## ML Systems

### Theory

First-Order Logic  
Complexity Theory

Learning Theory  
Optimization Theory

### Program Formalism

Relational Algebra

Tensor Algebra  
Gradient Descent

### Program Specification

SQL

TensorFlow?  
Scikit-learn?

### Program Modification

Query Optimization

???

### Execution Primitives

Parallel Relational  
Operator Dataflows

Depends on ML Algorithm

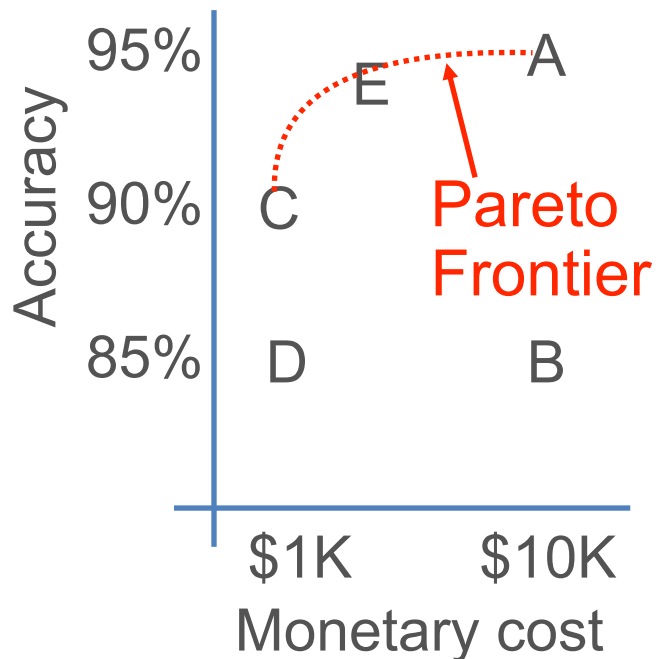
### Hardware

CPU, GPU, FPGA, NVM, RDMA, etc.

# Real-World ML: Pareto Surfaces

*Q: Suppose you are given ad click-through prediction models A, B, C, and D with accuracies of 95%, 85%, 90%, and 85%, respectively. Which one will you pick?*

*Q: What about now?*



- ❖ Real-world ML users must grapple with multi-dimensional *Pareto surfaces*: accuracy, monetary cost, training time, scalability, inference latency, tool availability, interpretability, fairness, etc.
- ❖ *Multi-objective optimization* criteria set by application needs / business policies.

# Learning Outcomes of this course

- ❖ **Explain** the basic principles of the memory hierarchy, parallelism paradigms, scalable data systems, and cloud computing.
- ❖ **Identify** the abstract data access patterns of, and opportunities for parallelism and efficiency gains in, data processing and ML algorithms at scale.
- ❖ **Outline** how to use cluster and cloud services, dataflow (“Big Data”) programming with MapReduce and Spark, and ML tools at scale.
- ❖ **Apply** the above programming skills to create end-to-end pipelines for data preparation, feature engineering, and model selection on large-scale datasets.
- ❖ **Reason** critically about practical tradeoffs between accuracy, runtimes, scalability, usability, and total cost.



# What this course is NOT about

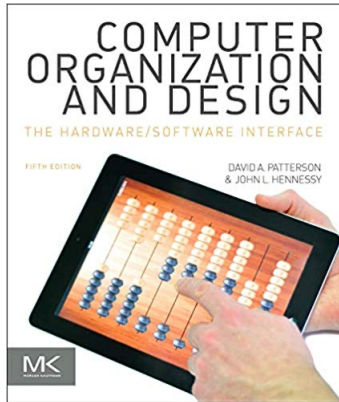
- ❖ NOT a course on databases, relational model, or SQL
  - ❖ Take DSC 100 instead (pre-requisite)
- ❖ NOT a course on internal details of RDBMSs
  - ❖ Take CSE 132C instead
- ❖ NOT a training module for how to use Spark
- ❖ NOT a course on ML or data mining *algorithmics*; instead, we focus on ML *systems*

# Tentative Course Schedule

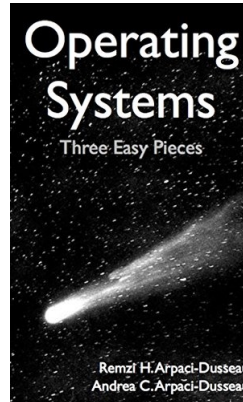
Week	Topic	
Systems Principles	Basics of Machine Resources: Computer Organization	
	Basics of Machine Resources: Operating Systems	
	4	Basics of Cloud Computing
4-5	Parallel and Scalable Data Processing: Parallelism Basics	
Scalability Principles	Midterm Exam on TBD	
	Parallel and Scalable Data Processing: Scalable Data Access	
	6-7	Parallel and Scalable Data Processing: Data Parallelism
7-8		
9	Scalable Analytics Systems	Dataflow Systems
10		ML Model Building Systems
11		Final Exam on Dec 15

There will be 2 industry guest lectures (maybe 3)

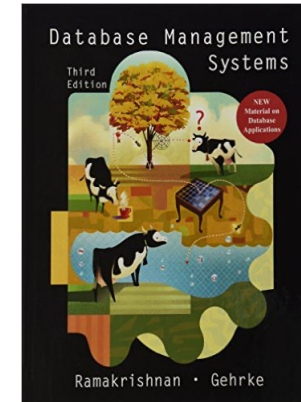
# Suggested Textbooks



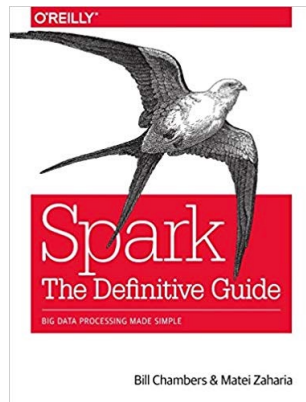
Aka “CompOrg Book”



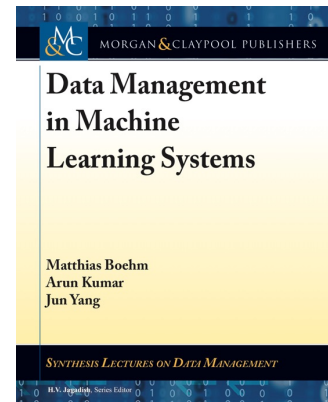
Aka “Comet Book”



Aka “Cow Book”



Aka “Spark Book”

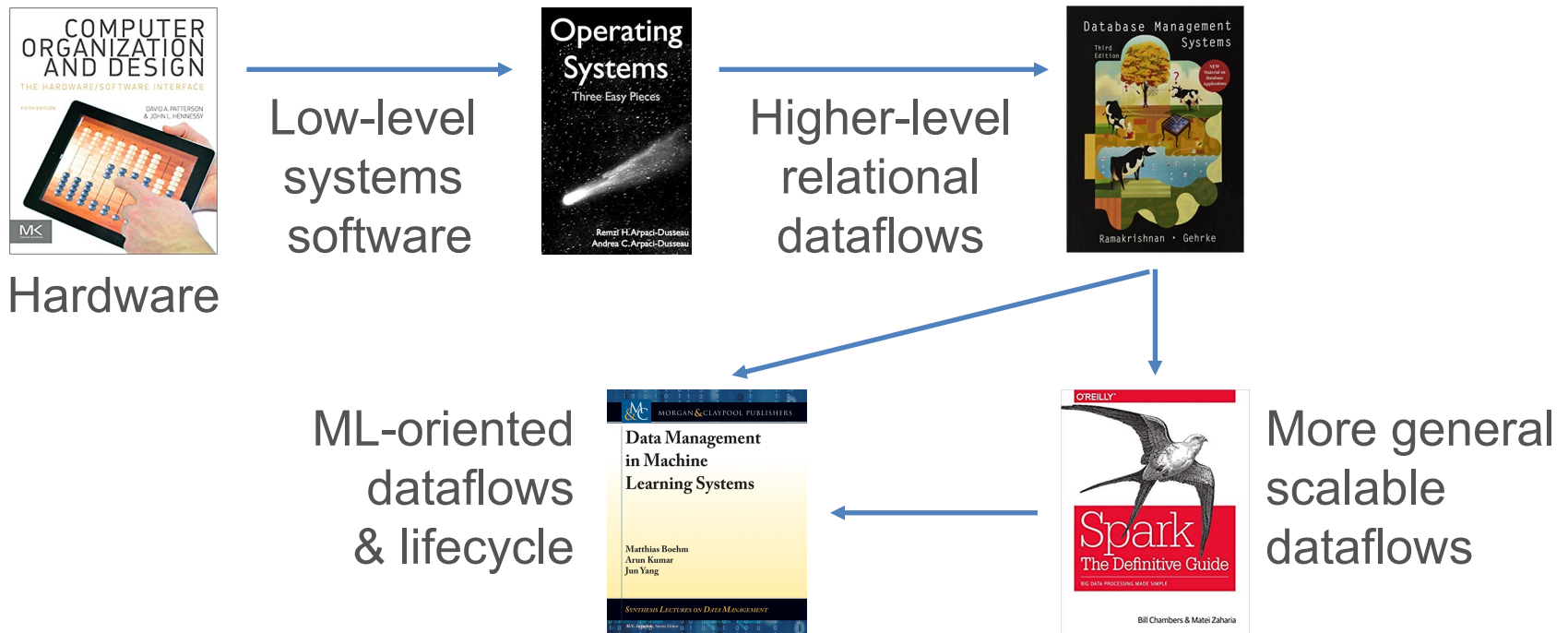


Aka “MLSys Book”

(Free PDFs available online; also check out our library)

# Why so many textbooks?!

1. Computer systems are about carefully layering *levels of abstraction*.



2. Analytics/ML Systems is a recent/emerging area of research.

3. Also, DSC 102 is the first UG course of its kind in the world!

# Tentative Course Schedule

Week	Topic
1-2	Basics of Machine Resources: Computer Organization
Systems Principles	Basics of Machine Resources: Operating Systems
	Basics of Cloud Computing
4-5	Parallel and Scalable Data Processing: Parallelism Basics
6	<b>Midterm Exam on TBD</b>
6-7	Parallel and Scalable Data Processing: Scalable Data Access
7-8	Parallel and Scalable Data Processing: Data Parallelism
9	Dataflow Systems
10	ML Model Building Systems
11	<b>Final Exam on Dec 15</b>

There will be 2 industry guest lectures (maybe 3)

# **DSC 102**

## **Systems for Scalable Analytics**

Topic 1: Basics of Machine Resources  
Part 1: Computer Organization

Ch. 1, 2.1-2.3, 2.12, 4.1, and 5.1-5.5 of CompOrg Book

***Q: What is a computer?***

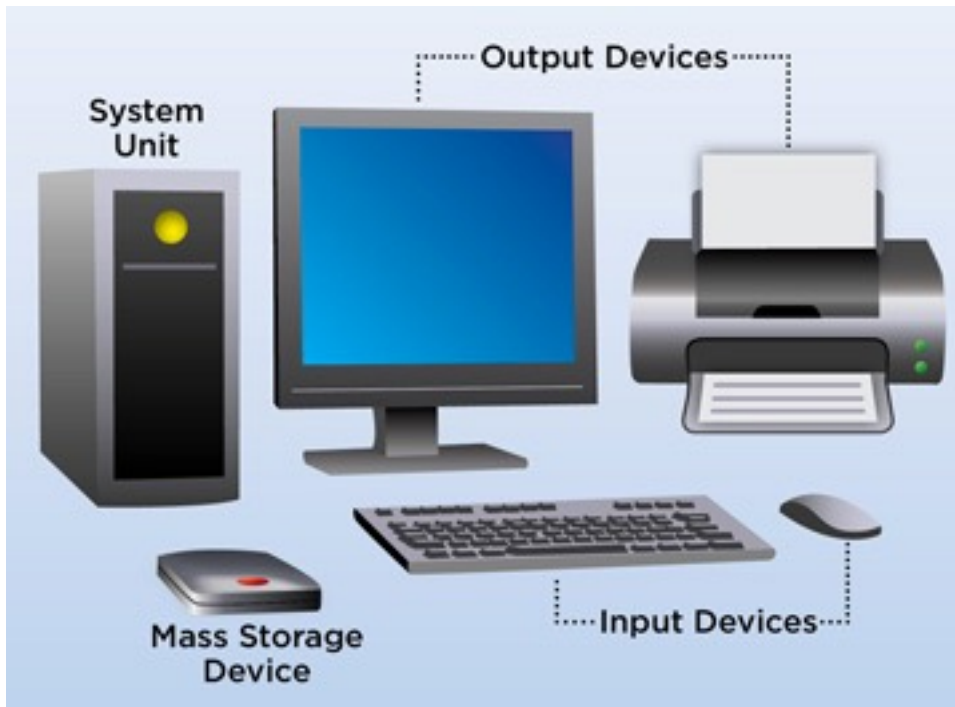
A programmable electronic device that can store, retrieve,  
and process digital data.

# Outline

- ➔ ❖ Basics of Computer Organization
  - ❖ Digital Representation of Data
  - ❖ Processors and Memory Hierarchy
- ❖ Basics of Operating Systems
  - ❖ Process Management: Virtualization; Concurrency
  - ❖ Filesystem and Data Files
  - ❖ Main Memory Management
- ❖ Persistent Data Storage



# Parts of a Computer



## **Hardware:**

The electronic machinery (wires, circuits, transistors, capacitors, devices, etc.)

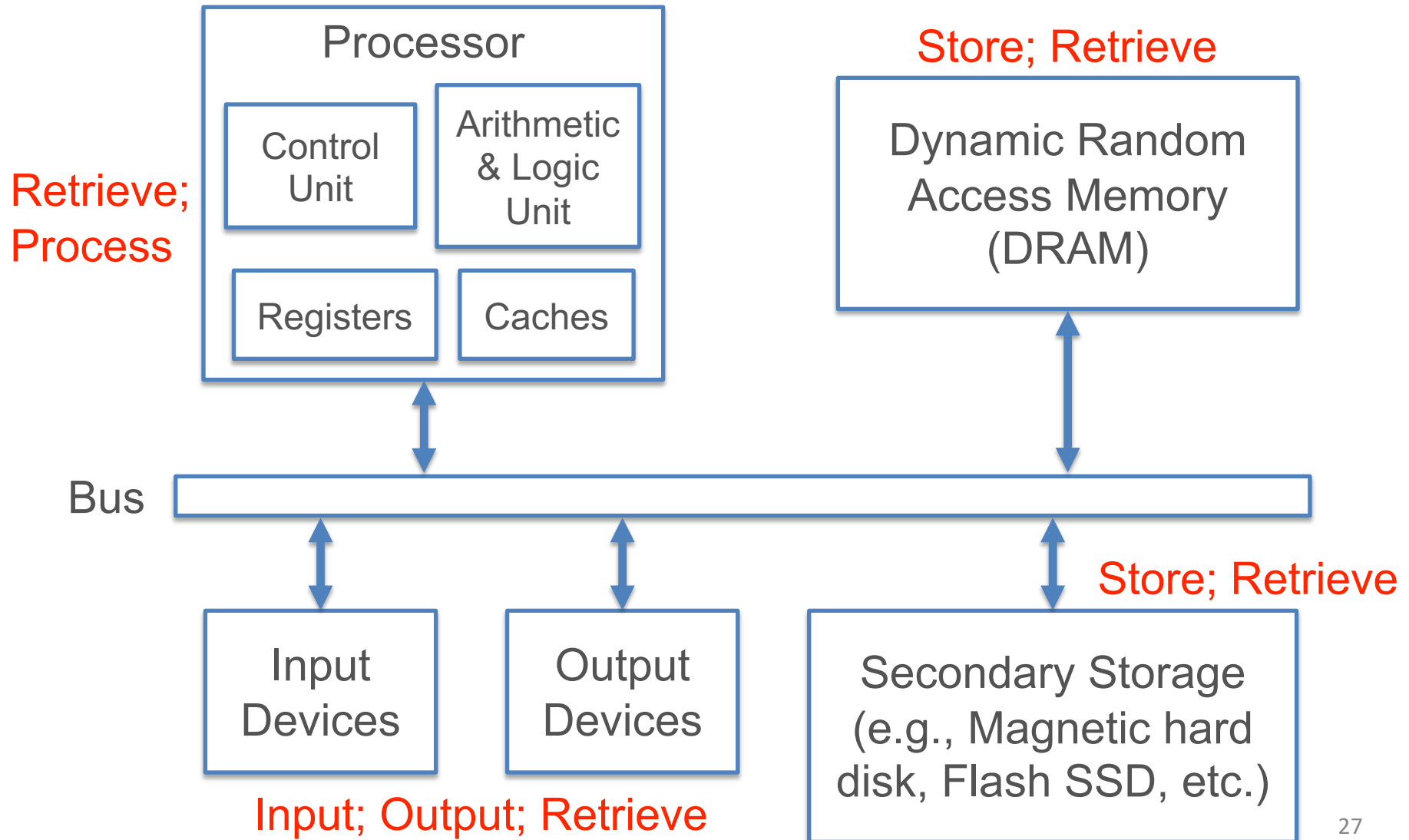
## **Software:**

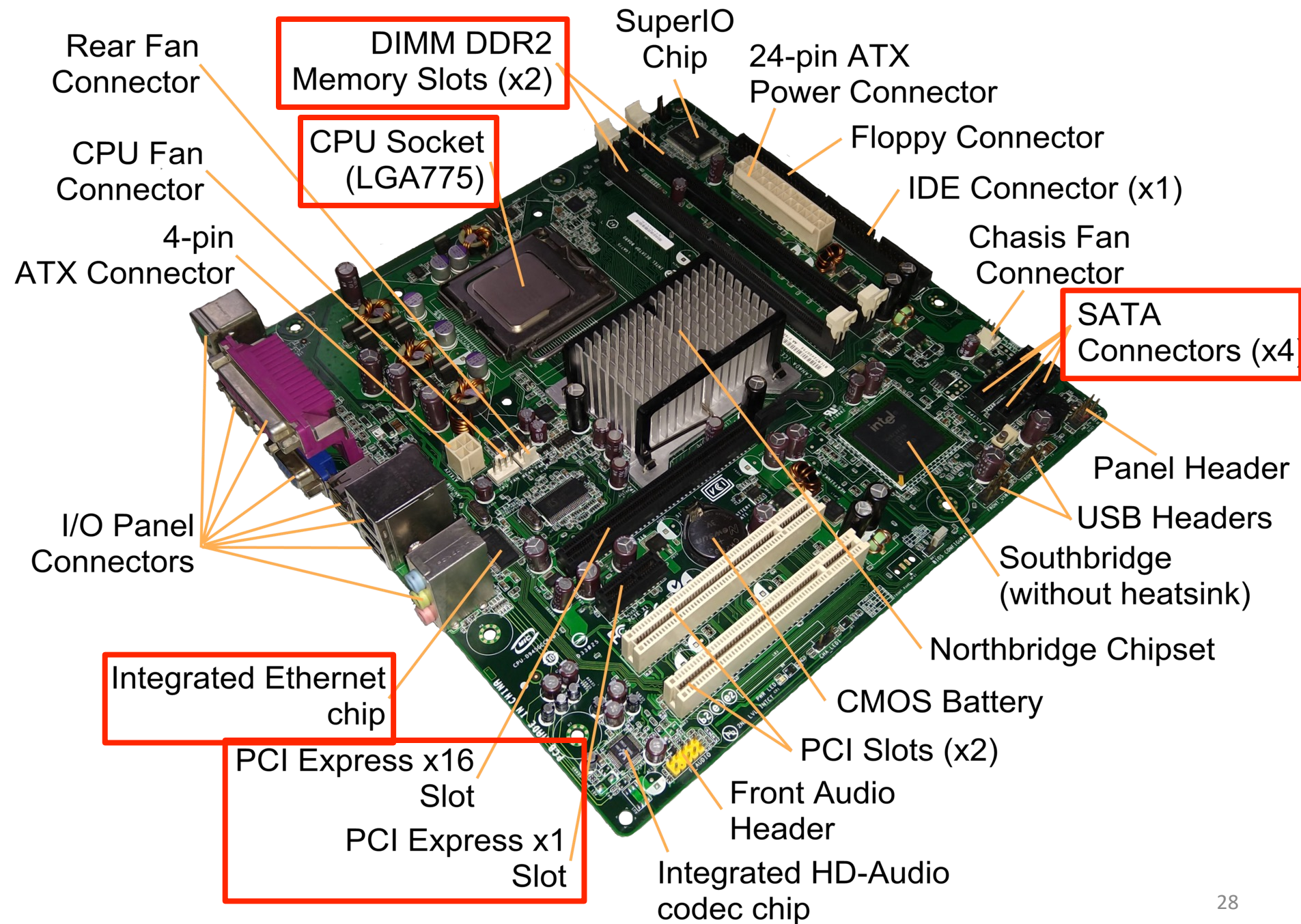
Programs (instructions) and data

# Key Parts of Computer Hardware

- ❖ **Processor** (CPU, GPU, etc.)
  - ❖ Hardware to orchestrate and execute *instructions* to manipulate *data* as specified by a *program*
- ❖ **Main Memory** (aka Dynamic Random Access Memory)
  - ❖ Hardware to store *data* and *programs* that allows very fast location/retrieval; byte-level *addressing* scheme
- ❖ **Disk** (aka secondary/persistent storage)
  - ❖ Similar to memory but *persistent*, *slower*, and higher capacity / cost ratio; various addressing schemes
- ❖ **Network** interface controller (NIC)
  - ❖ Hardware to send data to / retrieve data over network of interconnected computers/devices

# Abstract Computer Parts and Data





# Key Aspects of Software

## ❖ Instruction

- ❖ A command understood by hardware; finite vocabulary for a processor: Instruction Set Architecture (ISA); bridge between hardware and software

## ❖ Program (aka code)

- ❖ A collection of instructions for hardware to execute

## ❖ Programming Language (PL)

- ❖ A human-readable *formal* language to write programs; at a much higher level of *abstraction* than ISA

## ❖ Application Programming Interface (API)

- ❖ A set of functions (“interface”) exposed by a program/set of programs for use by humans/other programs

## ❖ Data

- ❖ Digital representation of *information* that is stored, processed, displayed, retrieved, or sent by a program

# Main Kinds of Software

## ❖ Firmware

- ❖ Read-only programs “baked into” a device to offer basic hardware control functionalities

## ❖ Operating System (OS)

- ❖ Collection of interrelated programs that work as an intermediary platform/service to enable application software to use hardware more effectively/easily
- ❖ Examples: Linux, Windows, MacOS, etc.

## ❖ Application Software

- ❖ A program or a collection of interrelated programs to manipulate data, typically designed for human use
- ❖ Examples: Excel, Chrome, PostgreSQL, etc.

# Outline

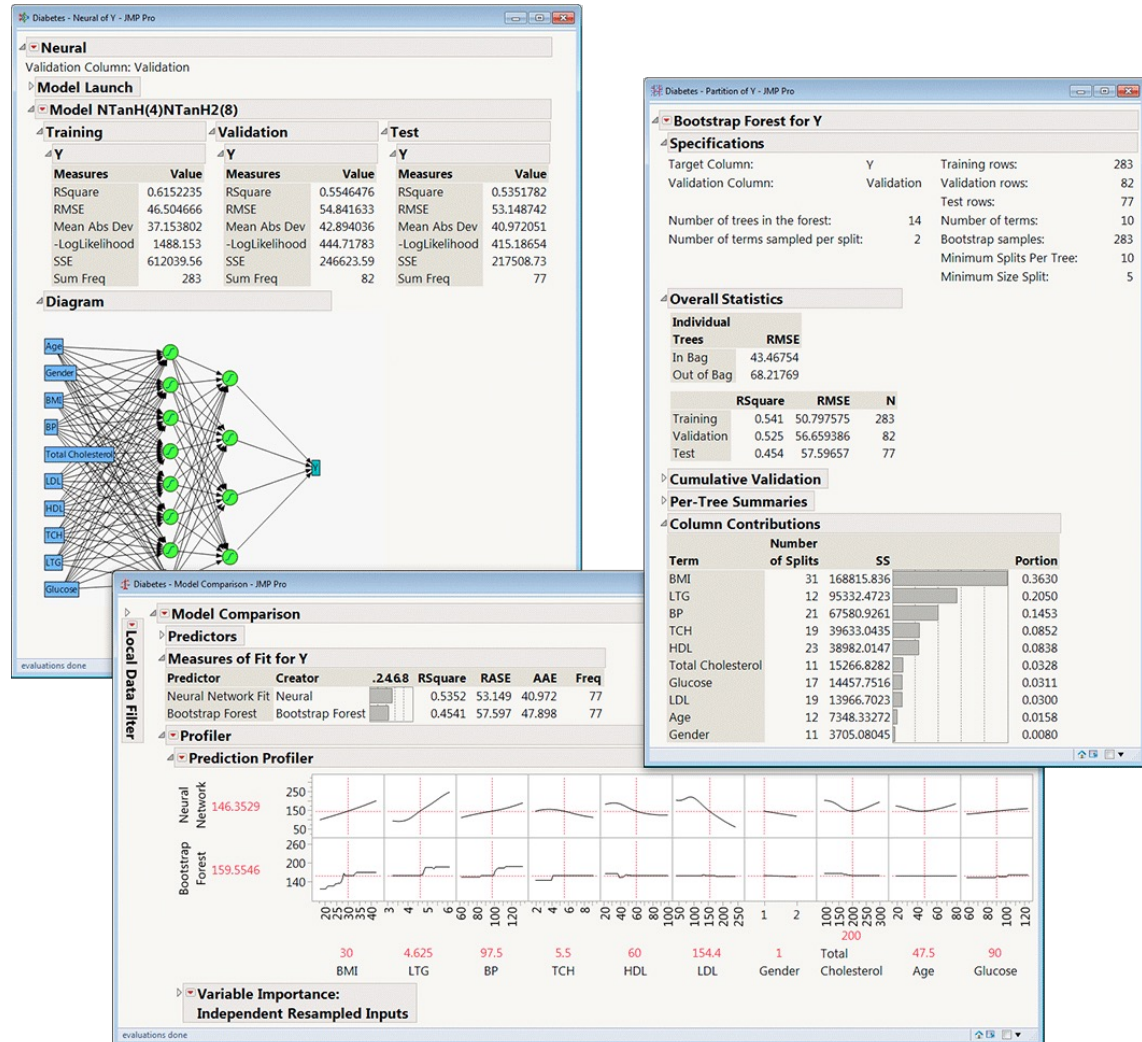
- ❖ Basics of Computer Organization
  - ❖ Digital Representation of Data
  - ❖ Processors and Memory Hierarchy
- ❖ Basics of Operating Systems
  - ❖ Process Management: Virtualization; Concurrency
  - ❖ Filesystem and Data Files
  - ❖ Main Memory Management
- ❖ Persistent Data Storage

***Q: But why bother learning such low-level computer sciencey stuff in Data Science?***



# Luxury of “Statisticians”/“Analysts” of Yore

- ❖ **Methods:** Sufficed to learn just math/stats, maybe some SQL
- ❖ **Types:** Mostly tabular (relational), maybe some time series
- ❖ **Scale:** Mostly small (KBs to few GBs)
- ❖ **Tools:** Simple GUIs for both analysis and deployment; maybe an R-like console



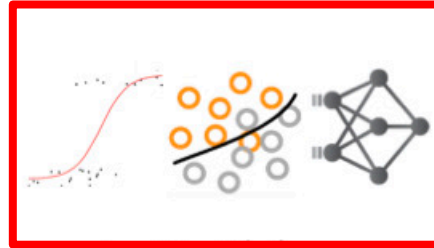
# Reality of Today's "Data Scientists"



Data Scientist/  
ML Engineer



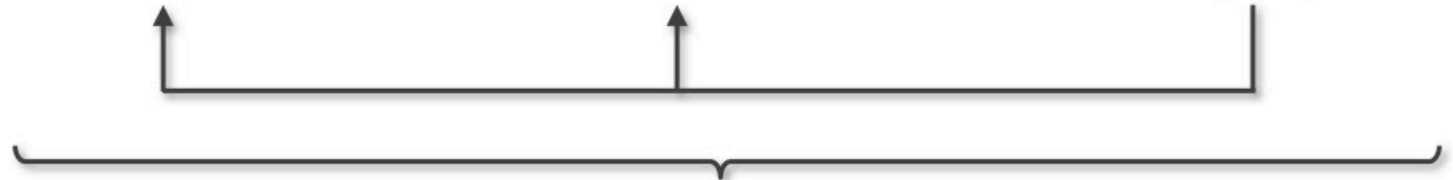
Source



Build



Deploy



Data acquisition  
Data preparation

Feature Engineering  
Training & Inference  
Model Selection

Serving  
Monitoring

# Why bother with these in Data Science?

- ❖ Basics of Computer Organization

You will face myriad  
and new data types

- ❖ Digital Representation of Data

- ❖ Processors and Memory Hierarchy

Compute hardware  
is evolving fast

- ❖ Basics of Operating Systems

- ❖ Process Management: Virtualization; Concurrency

- ❖ Filesystem and Data Files

- ❖ Main Memory Management

You will need to use new  
methods on evolving data file  
formats on clusters / cloud

- ❖ Persistent Data Storage

Storage hardware  
is evolving fast

# How much does a Statistician make?

Updated Jan 4, 2022

Industry

All industries

▼

Employer Size

All company sizes

▼

Experience

All years of Experience

▼

 Very High Confidence

\$88,989 /yr

Average Base Pay

2,398 salaries





## Data Scientist Salaries United States ▼

Overview

**Salaries**

Interviews

Insights

Career Path

### How much does a Data Scientist make?

Updated Jan 4, 2022

Industry



All industries



Employer Size



All company sizes



Experience



All years of Experience



To filter salaries for Data Scientist, [Sign In](#) or [Register](#).

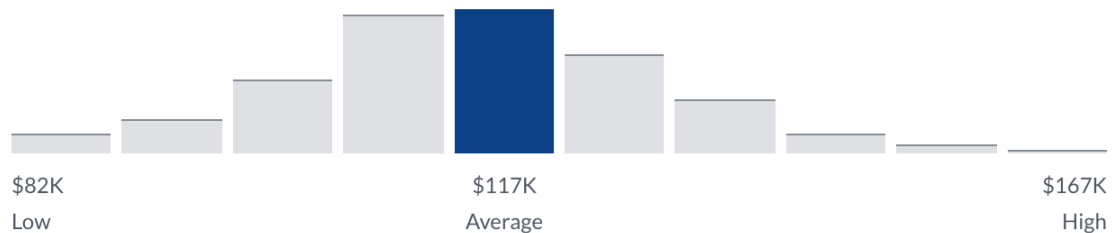


Very High Confidence

**\$117,212** /yr

Average Base Pay

18,354 salaries



— 88,989

= 28,223!

# Outline

- ❖ Basics of Computer Organization
- ➔ ❖ Digital Representation of Data
  - ❖ Processors and Memory Hierarchy
- ❖ Basics of Operating Systems
  - ❖ Process Management: Virtualization; Concurrency
  - ❖ Filesystem and Data Files
  - ❖ Main Memory Management
- ❖ Persistent Data Storage

***Q: What is data?***



[illegible]

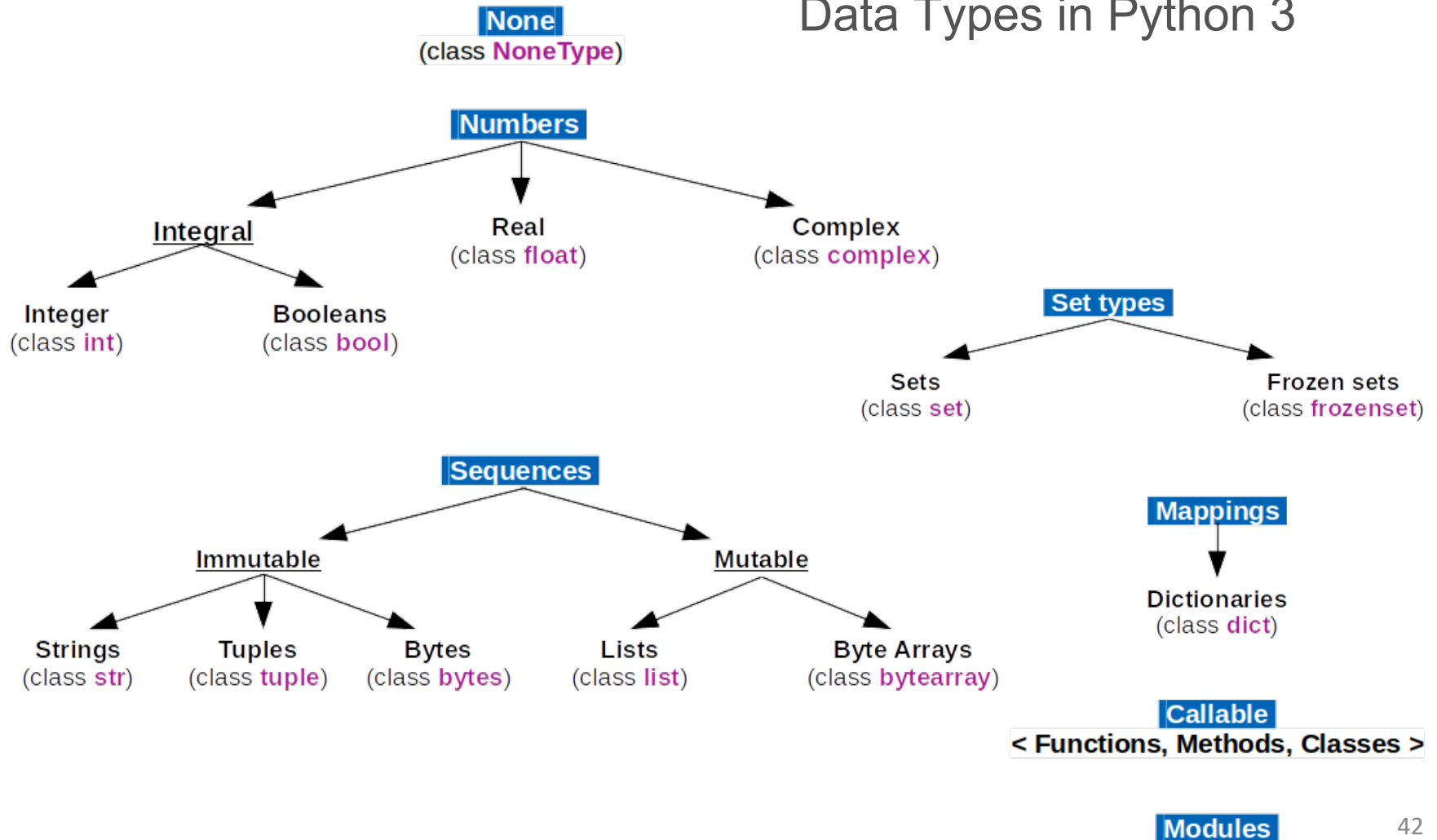


# Digital Representation of Data

- ❖ **Bits:** All digital data are sequences of 0 & 1 (binary digits)
  - ❖ Amenable to high-low/off-on electromagnetism
  - ❖ Layers of *abstraction* to interpret bit sequences
- ❖ **Data type:** First layer of abstraction to interpret a bit sequence with a human-understandable category of information; interpretation fixed by the PL
  - ❖ Example common datatypes: Boolean, Byte, Integer, “floating point” number (Float), Character, and String
- ❖ **Data structure:** A second layer of abstraction to *organize* multiple instances of same or varied data types as a more complex object with specified properties
  - ❖ Examples: Array, Linked list, Tuple, Graph, etc.

# Digital Representation of Data

## Data Types in Python 3



# Digital Representation of Data

- ❖ The *size* and *interpretation* of a data type depends on PL
- ❖ A **Byte** (B; 8 bits) is typically the basic unit of data types
- ❖ **Boolean:**
  - ❖ Examples in data sci.: Y/N or T/F responses
  - ❖ Just 1 bit needed but actual size is almost always 1B, i.e., 7 bits are wasted! (**Q:** *Why?*)
- ❖ **Integer:**
  - ❖ Examples in data science: #friends, age, #likes
  - ❖ Typically 4 bytes; many variants (short, unsigned, etc.)
  - ❖ Java *int* can represent  $-2^{31}$  to  $(2^{31} - 1)$ ; C *unsigned int* can represent 0 to  $(2^{32} - 1)$ ; Python3 *int* is effectively unlimited length (PL magic!)

# Digital Representation of Data

*Q: How many unique data items can be represented by 3 bytes?*

- ❖ Given  $k$  bits, we can represent  $2^k$  unique data items
- ❖ 3 bytes = 24 bits  $\Rightarrow 2^{24}$  items, i.e., 16,777,216 items
- ❖ Common approximation:  $2^{10}$  (i.e., 1024)  $\sim 10^3$  (i.e., 1000); recall kibibyte (KiB) vs kilobyte (KB) and so on

*Q: How many bits are needed to distinguish 97 data items?*

- ❖ For  $k$  unique items, invert the exponent to get  $\log_2(k)$
- ❖ But #bits is an integer! So, we only need  $\lceil \log_2(k) \rceil$
- ❖ So, we only need the next higher power of 2
- ❖  $97 \rightarrow 128 = 2^7$ ; so, 7 bits

# Digital Representation of Data

*Q: How to convert from decimal to binary representation?*

1. Given decimal  $n$ , if power of 2 (say,  $2^k$ ), put 1 at bit position  $k$ ; if  $k=0$ , stop; else pad with trailing 0s till position 0
2. If  $n$  is not power of 2, identify the power of 2 just below  $n$  (say,  $2^k$ ); #bits is then  $k$ ; put 1 at position  $k$
3. Reset  $n$  as  $n - 2^k$ ; return to Steps 1-2
4. Fill remaining positions in between with 0s

	7	6	5	4	3	2	1	0	Position/Exponent of 2
Decimal	128	64	32	16	8	4	2	1	Power of 2
$5_{10}$						1	0	1	
$47_{10}$			1	0	1	1	1	1	
$163_{10}$	1	0	1	0	0	0	1	1	
$16_{10}$				1	0	0	0	0	

*Q: Binary to decimal?*

# Digital Representation of Data

- ❖ *Hexadecimal* representation is a common stand-in for binary representation; more succinct and readable
  - ❖ Base 16 instead of base 2 cuts display length by ~4x
  - ❖ Digits are 0, 1, ... 9, A ( $10_{10}$ ), B, ... F ( $15_{10}$ )
  - ❖ From binary: combine 4 bits at a time from lowest

Decimal	Binary	Hexadecimal
$5_{10}$	$101_2$	$5_{16}$
$47_{10}$	$10\ 1111_2$	$2F_{16}$
$163_{10}$	$1010\ 0011_2$	$A3_{16}$
$16_{10}$	$1\ 0000_2$	$10_{16}$

Alternative  
notations  
0xA3 or A3H

# Digital Representation of Data

## ❖ Float:

- ❖ Examples in data sci.: salary, scores, model weights
- ❖ IEEE-754 single-precision format is 4B long; double-precision format is 8B long
- ❖ Java and C *float* is single; Python *float* is double!



DEVELOPER

HOME

BLOG

NEWS

FORUMS

DOCS

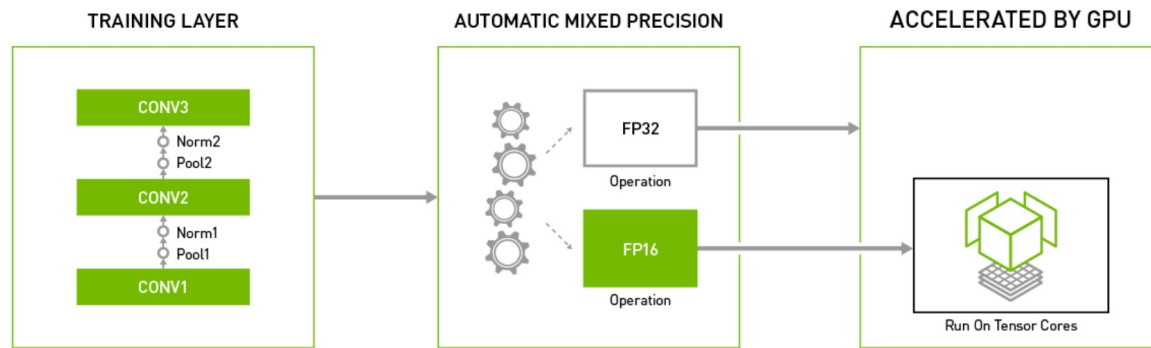
DOWNLOADS

TRAINING



ACCOUNT

available on the [NVIDIA deep learning performance page](#).

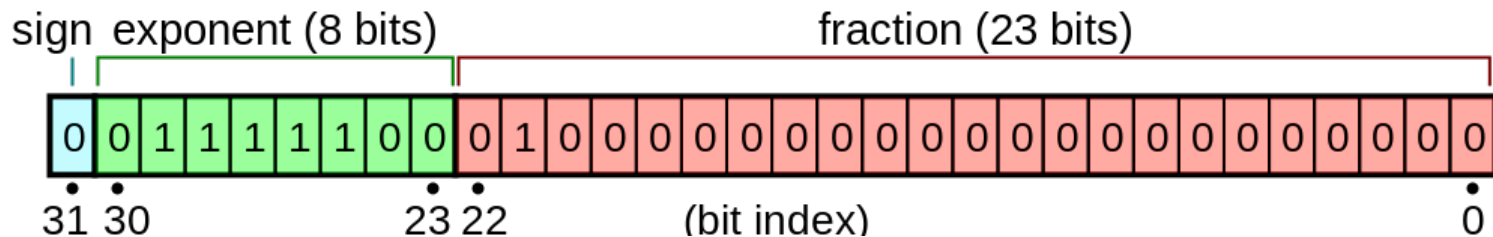


Using Automatic Mixed Precision for Major Deep Learning Frameworks

# Digital Representation of Data

## ❖ Float:

- ❖ Standard IEEE format for single (aka binary32):



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

$$(-1)^0 \times 2^{124-127} \times (1 + 1 \cdot 2^{-2}) = (1/8) \times (1 + (1/4)) = 0.15625$$

**(NB: Converting decimal reals/fractions to float is NOT in syllabus!)**



# Digital Representation of Data

- ❖ Due to representation imprecision issues, floating point arithmetic (addition and multiplication) is not associative!

```
(base) arun@Aruns-MacBook-Pro:~$ python
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.1 + 0.2
0.30000000000000004
>>> (0.1 + 0.2) + 0.7
1.0me
>>> 0.1 + (0.2 + 0.7)
0.9999999999999999
>>>
```

- ❖ In binary32, special encodings recognized:
  - ❖ Exponent 0xFF and fraction 0 is +/- “Infinity”
  - ❖ Exponent 0xFF and fraction  $\neq 0$  is “NaN”
  - ❖ Max is  $\sim 3.4 \times 10^{38}$ ; min +ve is  $\sim 1.4 \times 10^{-45}$

# Digital Representation of Data

- ❖ More float standards: double-precision (float64; 8B) and half-precision (float16; 2B); different #bits for exponent, fraction
- ❖ Float16 is now common for *deep learning* parameters:
  - ❖ Native support in PyTorch, TensorFlow, etc.; APIs also exist for weight quantization/rounding post training
  - ❖ NVIDIA Deep Learning SDK support mixed-precision training; 2-3x speedup with similar accuracy!
- ❖ New processor hardware (FPGAs, ASICs, etc.) enable arbitrary precision, even 1-bit (!), but accuracy is lower