Computer Organization: Data Representation

DS 5110: Big Data Systems (Spring 2023)

Lecture 2a

Yue Cheng

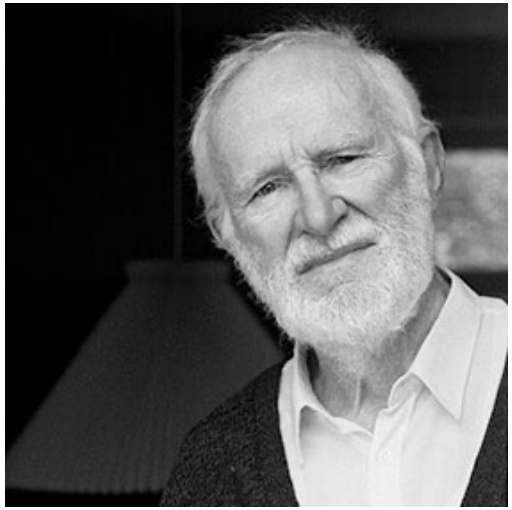
Some material taken/derived from:

- UC San Diego DSC 102 by Arun Kumar.

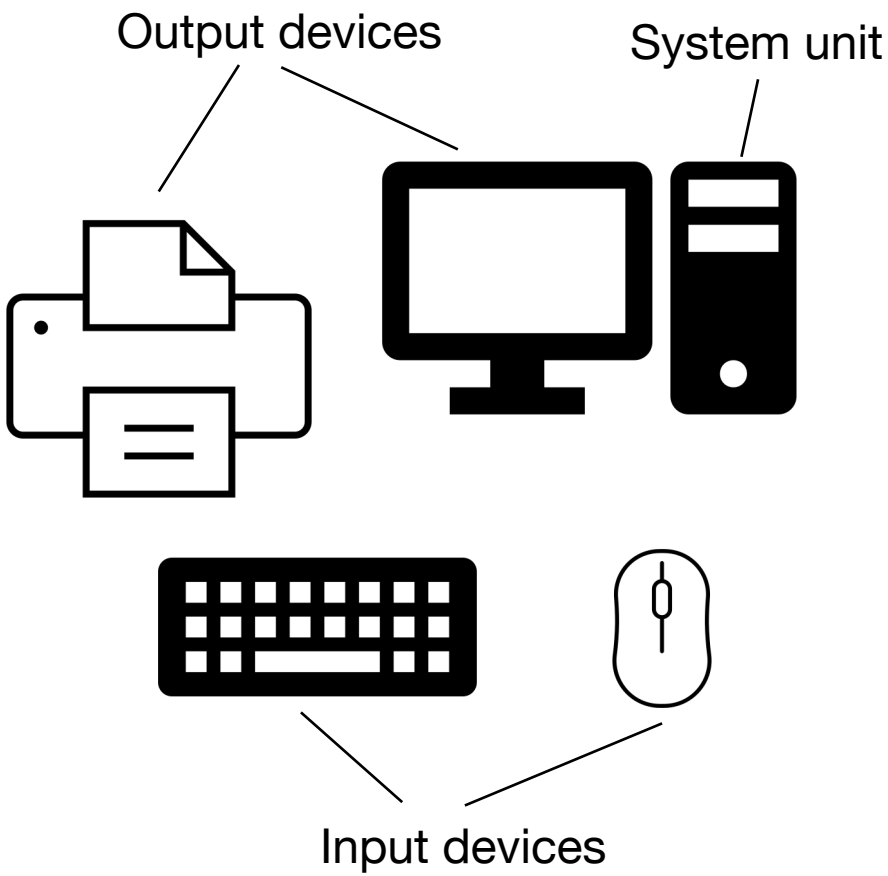
What is a computer?

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

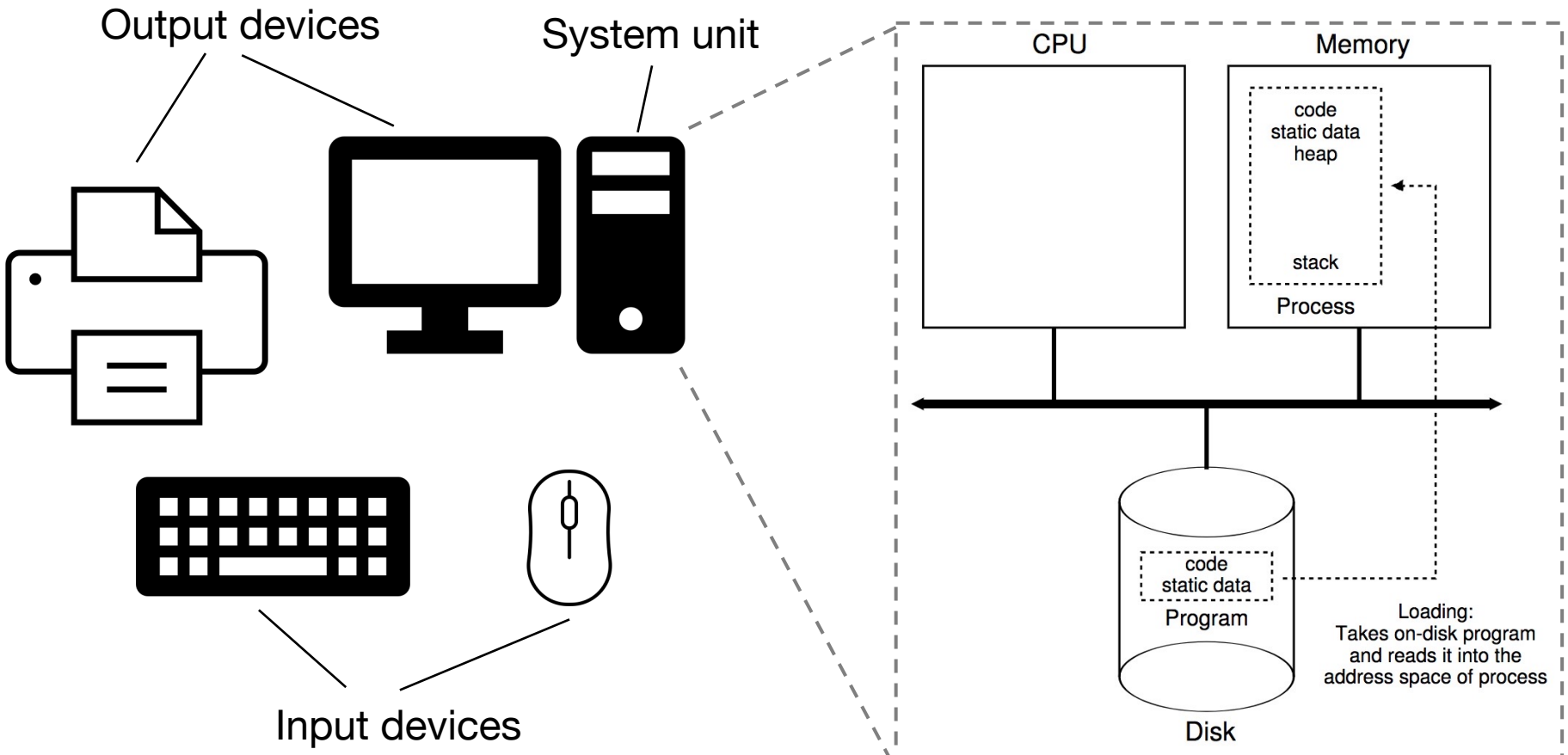
Computer science aka “Datalogy”



What is in a computer?

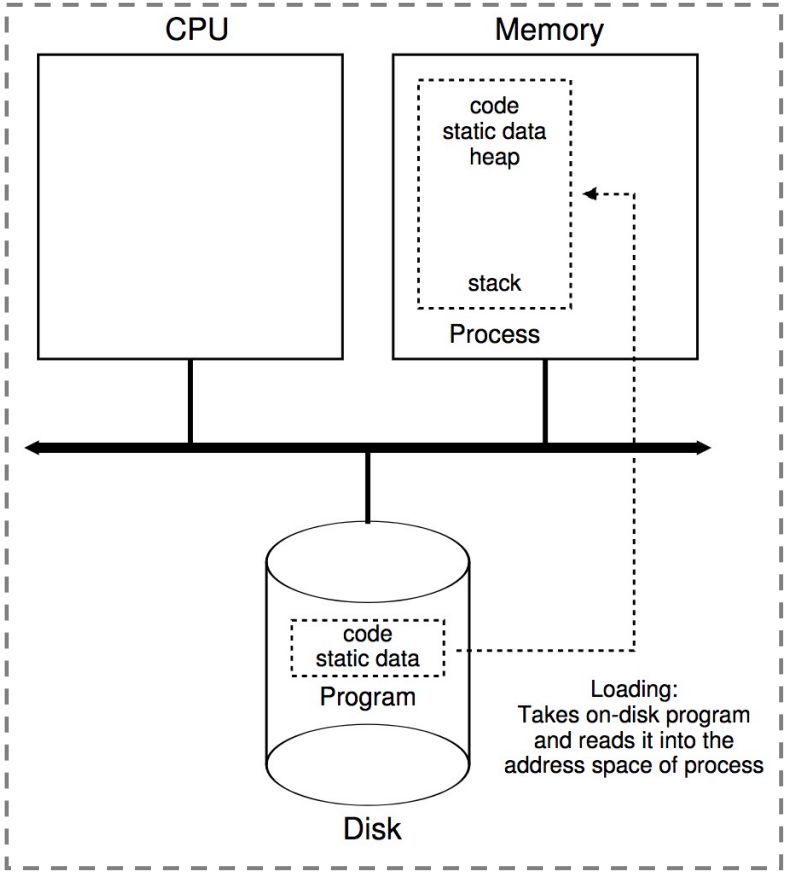


What is in a computer?



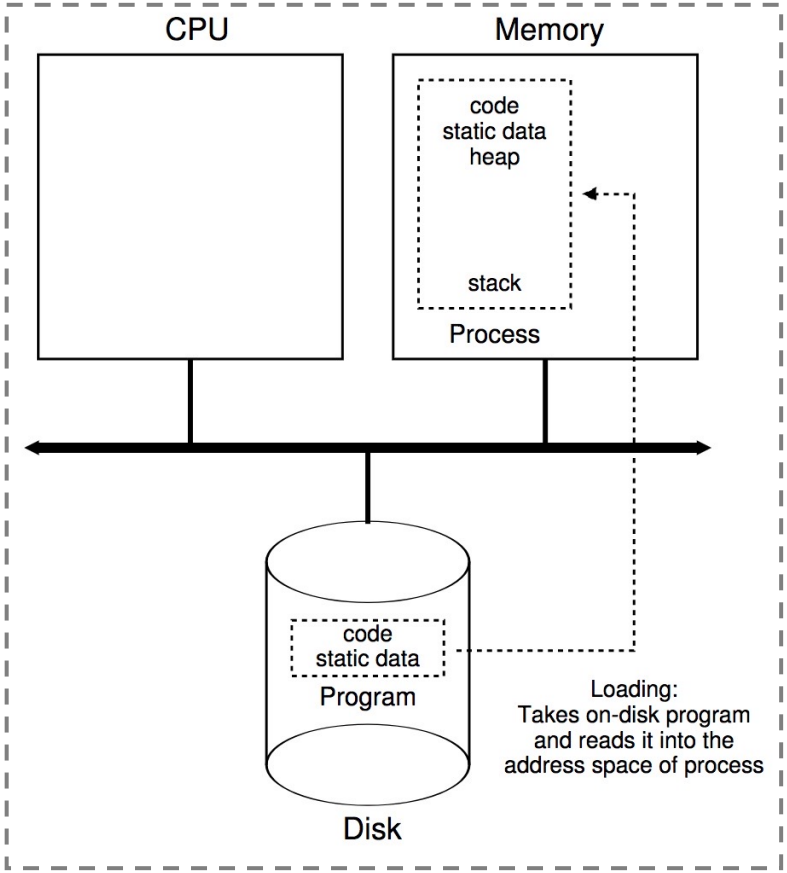
Key parts of computer hardware

- **CPU**
  - Hardware to execute instructions to manipulate data as specified by a program



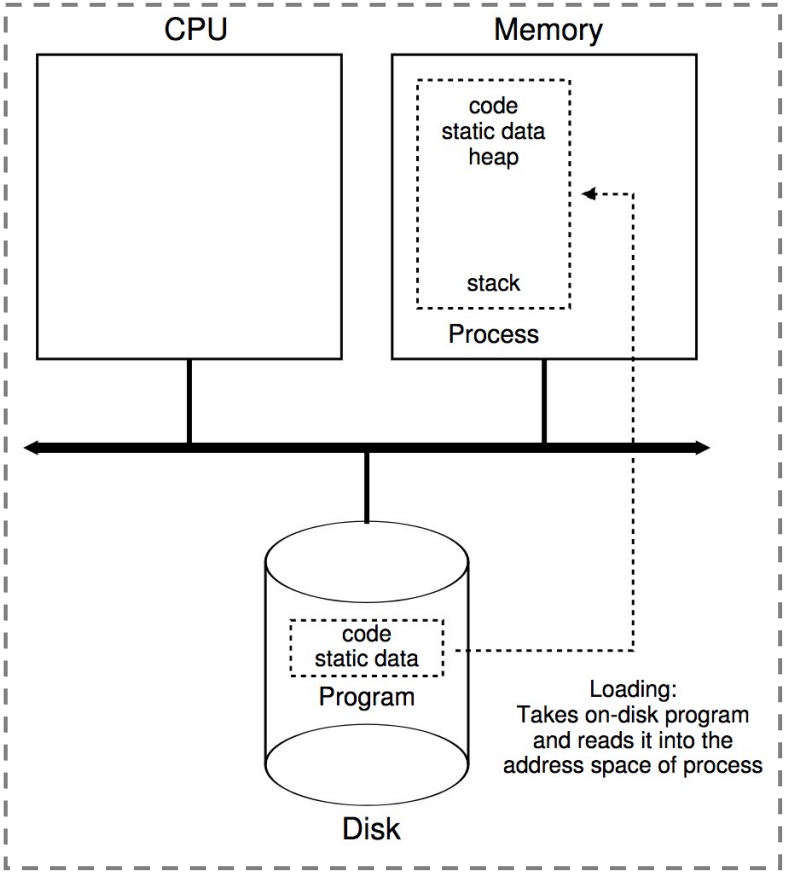
Key parts of computer hardware

- **CPU**
  - Hardware to execute instructions to manipulate data as specified by a program
- **Memory (DRAM)**
  - Hardware to store data and programs that allow fast storage/retrieval (byte addressable)

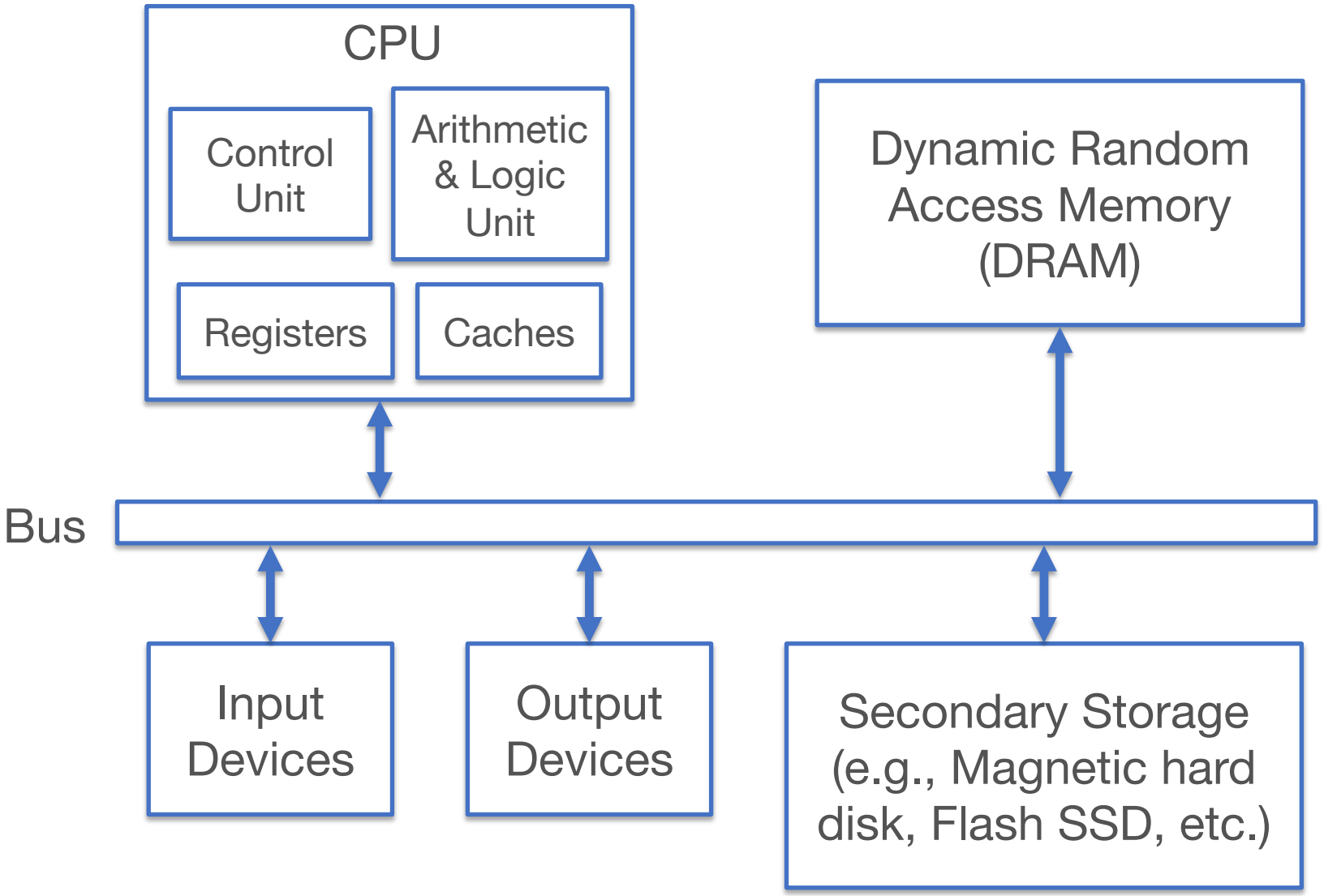


Key parts of computer hardware

- **CPU**
  - Hardware to execute instructions to manipulate data as specified by a program
- **Memory (DRAM)**
  - Hardware to store data and programs that allow fast storage/retrieval (byte addressable)
- **Disk (second storage)**
  - Persistent, slower storage with higher capacity (block addressable)



How different parts interact



Key aspects of software

- **Instruction**
  - A command understood by hardware
  - Finite vocabulary for a CPU: Instruction Set Architecture (ISA)
  - Bridge between hardware and software

Key aspects of software

- **Instruction**
  - A command understood by hardware
  - Finite vocabulary for a CPU: Instruction Set Architecture (ISA)
  - Bridge between hardware and software
- **Program**
  - A collection of instructions for hardware to execute

Key aspects of software

- **Instruction**
  - A command understood by hardware
  - Finite vocabulary for a CPU: Instruction Set Architecture (ISA)
  - Bridge between hardware and software
- **Program**
  - A collection of instructions for hardware to execute
- **Programing language (PL)**
  - A human-readable formal language to write programs
  - At a much higher level than ISA

Key aspects of software

- **Instruction**
  - A command understood by hardware
  - Finite vocabulary for a CPU: Instruction Set Architecture (ISA)
  - Bridge between hardware and software
- **Program**
  - A collection of instructions for hardware to execute
- **Programing language (PL)**
  - A human-readable formal language to write programs
  - At a much higher level than ISA
- **Application programming interface (API)**
  - A set of functions (“interfaces”) exposed by a program for use by human/other programs

Key aspects of software

- **Instruction**
  - A command understood by hardware
  - Finite vocabulary for a CPU: Instruction Set Architecture (ISA)
  - Bridge between hardware and software
- **Program**
  - A collection of instructions for hardware to execute
- **Programing language (PL)**
  - A human-readable formal language to write programs
  - At a much higher level than ISA
- **Application programming interface (API)**
  - A set of functions (“interfaces”) exposed by a program for use by human/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

Main kinds of software

- **Firmware**
  - Read-only programs “baked into” a device to offer basic hardware control functionalities
- **Operating system (OS)**
  - Sophisticated (kernel-space) software programs that collectively work as an intermediary/manager to enable application programs to use hardware efficiently

Main kinds of software

- **Firmware**
  - Read-only programs “baked into” a device to offer basic hardware control functionalities
- **Operating system (OS)**
  - Sophisticated (kernel-space) software programs that collectively work as an intermediary/manager to enable application programs to use hardware efficiently
- **Application software programs**
  - A (user-space) program or a collection of (user-space) programs to perform a certain task for human use
  - Examples: Office, Chrome, Zoom

What is data?

Digital representation of data

- **Bits:** All digital data are sequences of 0s and 1s (binary)
  - Amenable to high-low/on-off electromagnetism

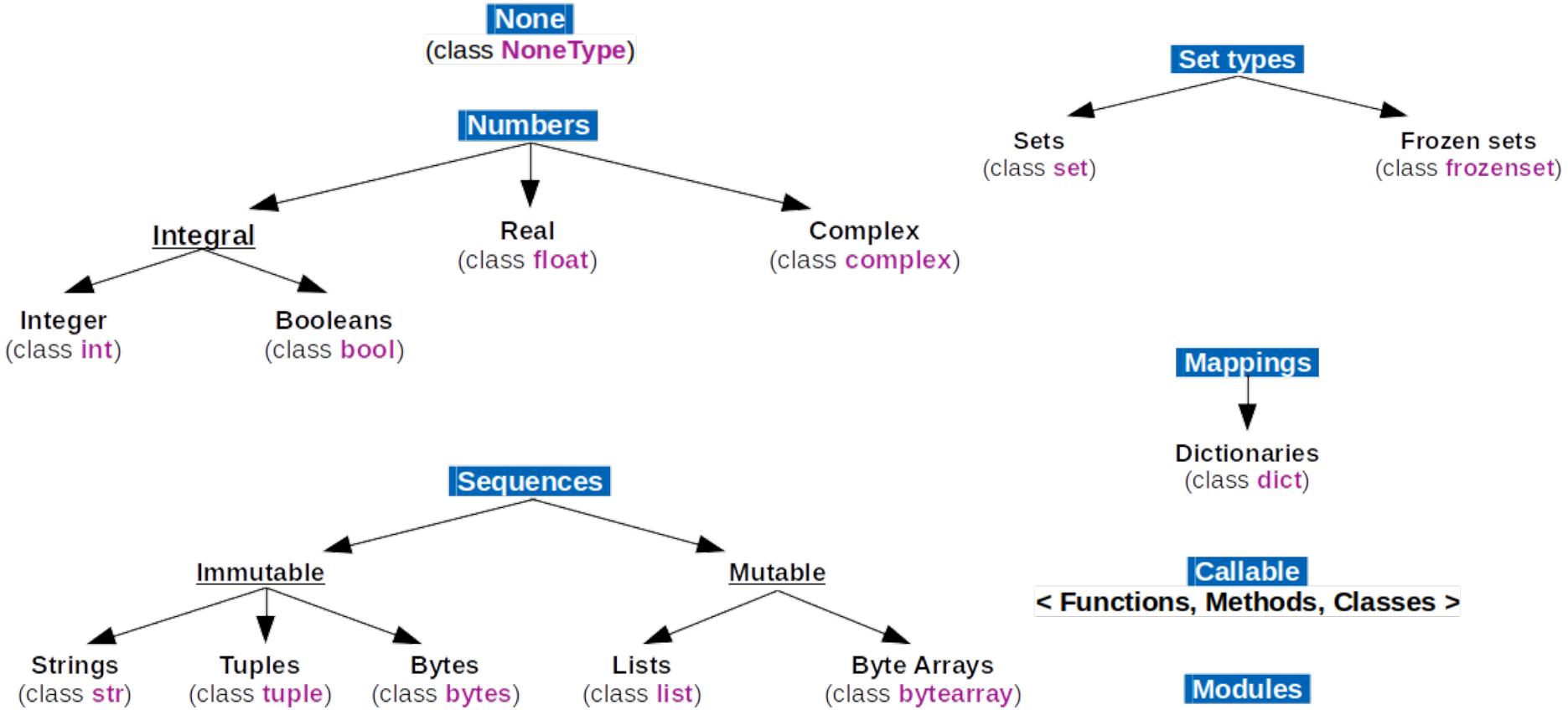
Digital representation of data

- **Bits:** All digital data are sequences of 0s and 1s (binary)
  - Amenable to high-low/on-off electromagnetism
- **Data types:** First layer of abstraction to interpret a bit sequence with human-understandable category of information
  - Example data types: Boolean, byte, integer, floating point number (float), character, string

Digital representation of data

- **Bits:** All digital data are sequences of 0s and 1s (binary)
  - Amenable to high-low/on-off electromagnetism
- **Data types:** First layer of abstraction to interpret a bit sequence with human-understandable category of information
  - Example data types: Boolean, byte, integer, floating point number (float), character, string
- **Data structures:** A second layer of abstraction to organize multiple instances of same or varied data types as a more complex object with specific properties
  - Examples: Array, dictionary (hash table), tree, graph

Data types in Python 3



Data types

- **Byte:** an 8-bit unit

Data types

- **Byte:** an 8-bit unit
- **Boolean:** Yes/No, True/False
  - Information encoded in a Boolean type is just 1 bit, but actual size of Boolean typed variable is always 1 byte (7 bits are wasted!)

Data types

- **Byte:** an 8-bit unit
- **Boolean:** Yes/No, True/False
  - Information encoded in a Boolean type is just 1 bit, but actual size of Boolean typed variable is always 1 byte (7 bits are wasted!)
- **Integer:**
  - Examples: count of something – # friends
  - Typically 4B but many variants (short, unsigned, etc.)
    - Java `int`: -2^{31} to $(2^{31}-1)$
    - C `unsigned int`: 0 to $(2^{32}-1)$
    - Python3 `int`: effectively no max limit

Binary examples

Q: How many unique data items can be represented by 4 bytes?

Binary examples

Q: How many unique data items can be represented by 4 bytes?

- Given k bits, we can represent 2^k unique data items
- 4 bytes = 32 bits → 2^{32} items
- Common approximation: $2^{10} = 1024 \sim 10^3$ (i.e., 1000); recall kibibyte (KiB) vs. kilobyte (KB) and so on

Binary examples

Q: How many unique data items can be represented by 4 bytes?

- Given k bits, we can represent 2^k unique data items
- 4 bytes = 32 bits → 2^{32} items
- Common approximation: $2^{10} = 1024 \sim 10^3$ (i.e., 1000); recall kibibyte (KiB) vs. kilobyte (KB) and so on

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

Binary examples

Q: How many unique data items can be represented by 4 bytes?

- Given k bits, we can represent 2^k unique data items
- 4 bytes = 32 bits → 2^{32} items
- Common approximation: $2^{10} = 1024 \sim 10^3$ (i.e., 1000); recall kibibyte (KiB) vs. kilobyte (KB) and so on

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

- For k unique items, invert the exponent: $\log_2(k)$
- #bits should be integer, so we do $\lceil \log_2(k) \rceil$
- $\lceil \log_2(97) \rceil$: 97 → 128 = 2^7, so, 7 bits

Binary examples

Q: How to convert a decimal to a binary?

Binary examples

Q: How to convert a decimal to a binary?

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 0 till position 0

Binary examples

Q: How to convert a decimal to a binary?

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 0 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

Binary examples

Q: How to convert a decimal to a binary?

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 0 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; repeat step 1-2

Binary examples

Q: How to convert a decimal to a binary?

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 0 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; repeat step 1-2
4. Fill remaining positions in between with 0s

Binary examples

Q: How to convert a decimal to a binary?

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 0 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; repeat step 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
8_{10}									
26_{10}									
163_{10}									

Hexadecimal examples

- 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

Hexadecimal examples

- 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	
47_{10}	$10\ 1111_2$	
163_{10}	$1010\ 0011_2$	
16_{10}	$1\ 0000_2$	

Hexadecimal examples

- 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}

Hexadecimal examples

- 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 $A3_H$

Float

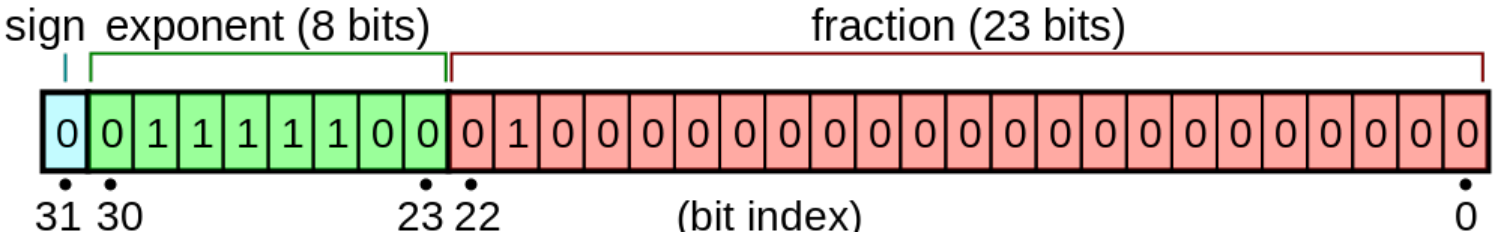
- Float
  - Examples: salary, model weights
  - IEEE-754 single-precision format is 4B long; double-precision format is 8B long
  - Java and C float is single; Python float is double

Float

- Float
  - Standard IEEE format for single (aka binary32)

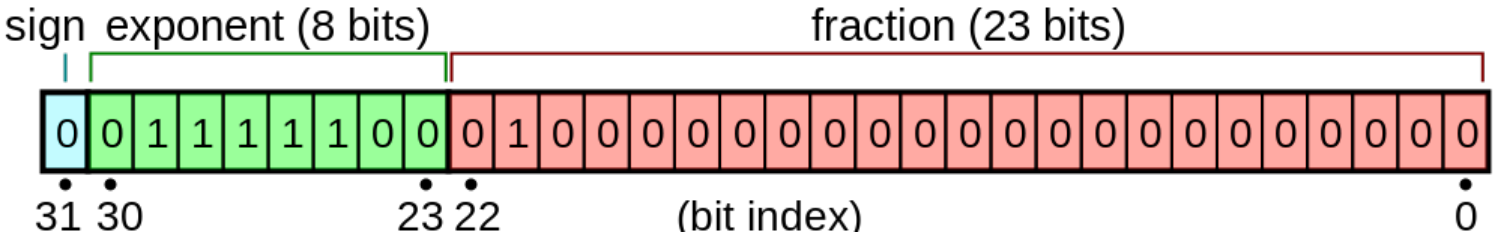
Float

- Float
  - Standard IEEE format for single (aka binary32)



Float

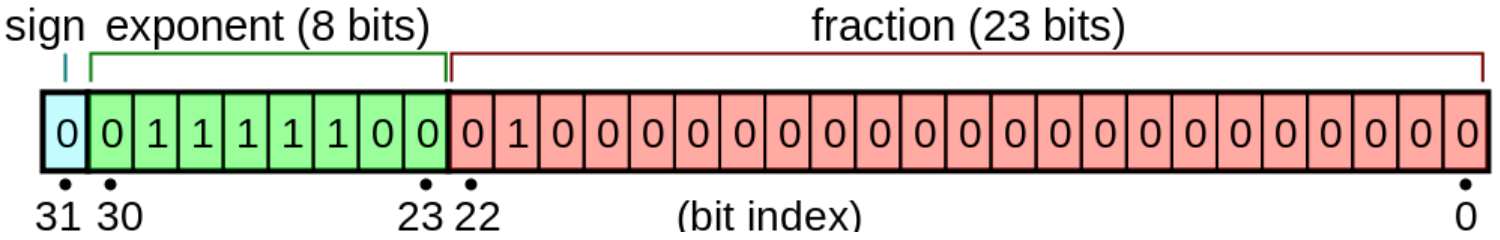
- Float
  - Standard IEEE format for single (aka binary32)



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

Float

- Float
  - Standard IEEE format for single (aka binary32)



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

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

Float

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

```
Yue ds5110-spring23 $ python3
Python 3.9.6 (default, Oct 18 2022, 12:41:40)
[Clang 14.0.0 (clang-1400.0.29.202)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.1 + 0.3
0.4
>>> (0.1 + 0.3) + 0.6
1.0
>>> 0.1 + (0.3 + 0.6)
0.9999999999999999
>>>
```

Float

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

```
Yue ds5110-spring23 $ python3
Python 3.9.6 (default, Oct 18 2022, 12:41:40)
[Clang 14.0.0 (clang-1400.0.29.202)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.1 + 0.3
0.4
>>> (0.1 + 0.3) + 0.6
1.0
>>> 0.1 + (0.3 + 0.6)
0.9999999999999999
>>>
```

- In binary32, special encodings recognized:
  - Exponent 0xFF and fraction 0 is +/- “infinity”
  - Exponent 0xFF and fraction <> 0 is “NaN”
  - Max is 2^{127} x $(2-2^{-23})$, i.e., ~3.4 x 10^{38}

More on float standards

- Double-precision (float64, 8B) and half-precision (float16, 2B)
  - Different #bits for exponent, fraction
- float16 is now common for deep learning parameters

Char and string

- Representing character (`char`) and string
  - Letters, numerals, punctuations, etc.
  - A string is typically just a variable-sized array of `chars`

Char and string

- Representing character (**`char`**) and string
  - Letters, numerals, punctuations, etc.
  - A string is typically just a variable-sized array of **`char`**s
  - C **`char`** is 1B; Java **`char`** is 2B; Python does not have a **`char`** type (use str or bytes)

Char and string

- Representing character (**`char`**) and string
  - Letters, numerals, punctuations, etc.
  - A string is typically just a variable-sized array of **`char`**s
  - C **`char`** is 1B; Java **`char`** is 2B; Python does not have a **`char`** type (use str or bytes)
  - American Standard Code for Information Interchange (ASCII) for encoding characters
    - Initially 7-bit, later extended to 8-bit
    - Examples: ‘D’ is 68, ‘d’ is 100, ‘!’ is 33, ‘?’ is 63

Char and string

- Representing character (**`char`**) and string
  - Letters, numerals, punctuations, etc.
  - A string is typically just a variable-sized array of **`char`**s
  - C **`char`** is 1B; Java **`char`** is 2B; Python does not have a **`char`** type (use str or bytes)
  - American Standard Code for Information Interchange (ASCII) for encoding characters
    - Initially 7-bit, later extended to 8-bit
    - Examples: ‘D’ is 68, ‘d’ is 100, ‘!’ is 33, ‘?’ is 63
  - Unicode UTF-8 subsumes ASCII
    - 4B for ~1.1 million “code points” including many other language scripts, math symbols, emojis, etc.
    - 😄 👍🏻 : https://unicode.org/emoji/charts/full-emoji-list.html

Data structures

- Data structures: A second layer of abstraction to organize multiple instances of same or varied data types as a more complex object with specific properties
  - ML feature vectors: array of floats
  - Neural network weights: set of multi-dimensional arrays (matrices or tensors) of floats
  - Trees: binary trees, N-ary trees
  - Graphs: sets of vertices (integers) and sets of edges (pair of integers) that connect vertices
  - And a lot more…