Lecture 3: Bits, Bytes, Binary

Bits, bytes, binary numbers, and the representation of information

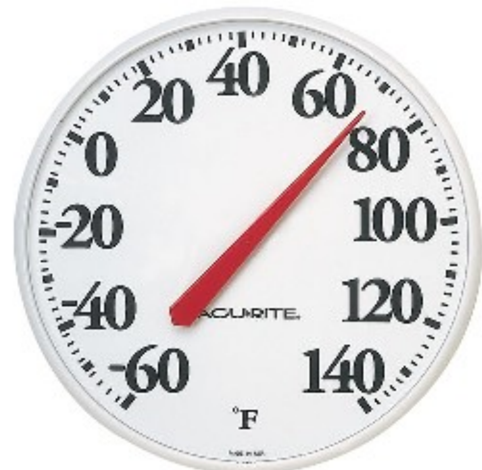
- **computers represent, process, store, copy, and transmit everything as numbers**
  - hence "digital computer"

- **the numbers can represent anything**
  - not just numbers that you might do arithmetic on

- **the meaning depends on context**
  - as well as what the numbers ultimately represent
  - e.g., numbers coming to your computer or phone from your wi-fi connection could be email, movies, music, documents, apps, Zoom meeting, ...

Analog versus Digital

- **analog: "analogous" or "the analog of"**
  - smoothly or continuously varying values
  - volume control, dimmer, faucet, steering wheel
  - value varies smoothly with something else
    - no discrete steps or changes in values
      - small change in one implies small change in another
      - infinite number of possible values
  - the world we perceive is largely analog
- **digital: discrete values**
  - only a finite number of different values
  - a change in something results in sudden change from one discrete value to another
    - digital speedometer, digital watch, push-button radio tuner, …
  - values are represented as <u>numbers</u>



Transducers

- **devices that convert from one representation to another**
  - microphone
  - loudspeaker / earphones
  - camera / scanner
  - printer / screen
  - keyboard
  - mouse
  - touch screen
  - etc.

- **something is usually lost by conversion (in each direction)**
  - the ultimate copy is not as good as the original

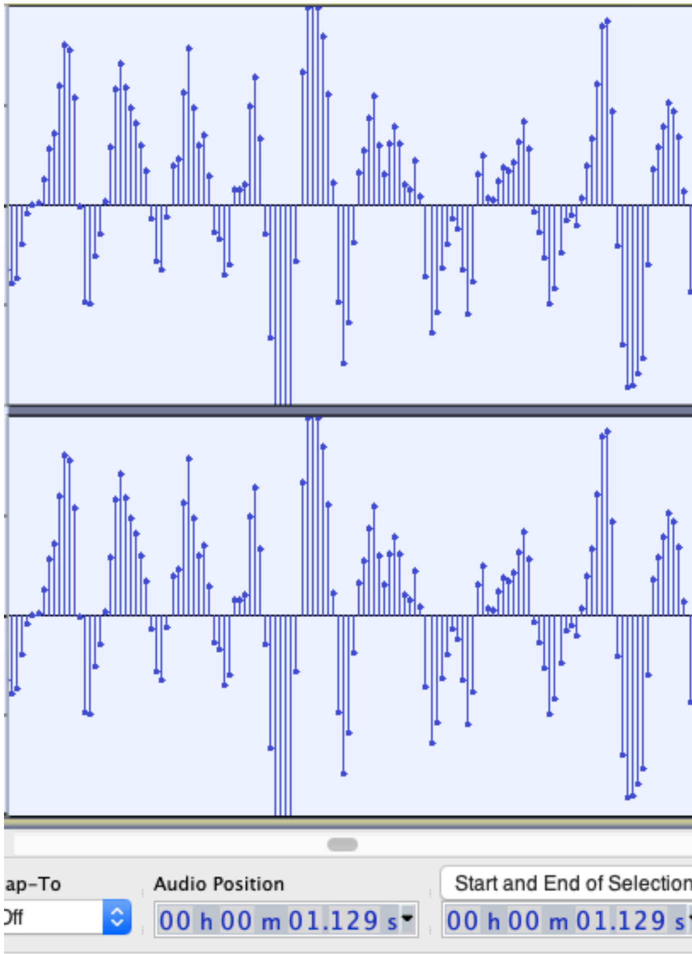
Digital pictures

- divide the picture up into a grid of little rectangles (“pixels”)
- assign a different numeric value to each different color value
- the finer the grid and the finer the color distinctions,
  the more accurate the representation will be


Digital sound

- need to measure intensity/loudness often enough and accurately enough that we can reconstruct it well enough
- higher frequency = higher pitch
- human ear can hear ~ 20 Hz to 20 KHz
  - taking samples at twice the highest frequency is good enough (Nyquist)
- CD audio usually uses
  - 44,100 samples / second
  - accuracy of 1 in 65,536 (= 2^16) distinct levels
  - two samples at each time for stereo
  - data rate is 44,100 x 2 x 16 bits/sample

    = 1,411,200 bits/sec = 176,400 bytes/sec ~ 10.6 MB/minute
- MP3 audio compresses by clever encoding and removal of sounds that won't really be heard
  - data rate is ~ 1 MB/minute



A review of how decimal numbers work

- **how many digits?**

  we use 10 digits for counting: "decimal" numbers are natural for us

  other schemes show up in some areas

    clocks use 12, 24, 60; calendars use 7, 12

    other cultures use other schemes (quatre-vingts)

- **what if we want to count to more than 10?**

  0 1 2 3 4 5 6 7 8 9

    1 decimal digit represents 1 choice from 10; counts 10 things; 10 distinct values

  00 01 02 … 10 11 12 … 20 21 22 … 98 99

    2 decimal digits represents 1 choice from 100; 100 distinct values

    we usually elide zeros at the front

  000 001 … 099 100 101 … 998 999

    3 decimal digits …

- **decimal numbers are shorthands for sums of powers of 10**

  $1492 = 1 \times 1000 + 4 \times 100 + 9 \times 10 + 2 \times 1$

  $= 1 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

- **counting in "base 10", using powers of 10**

Binary numbers: only use the digits 0 and 1 to represent numbers

- just like decimal except there are only <u>two</u> digits: 0 and 1
- everything is based on powers of 2 (1, 2, 4, 8, 16, 32, …)
  - instead of powers of 10 (1, 10, 100, 1000, …)
- counting in binary or base 2:

  0 1

  1 binary digit represents 1 choice from 2; counts 2 things; 2 distinct values

  00 01 10 11

  2 binary digits represents 1 choice from 4; 4 distinct values

  000 001 010 011 100 101 110 111

  3 binary digits …
- binary numbers are shorthands for sums of powers of 2

  11011 = 1 x 16 + 1 x 8 + 0 x 4 + 1 x 2 + 1 x 1

  $= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
- counting in "base 2", using powers of 2

Binary (base 2) arithmetic

- works like decimal (base 10) arithmetic, but simpler

- addition:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

- subtraction, multiplication, division are analogous

Converting binary to decimal

from right to left:

if bit is 1 add corresponding power of 2

i.e. 2^0, 2^1, 2^2, 2^3

(rightmost power is zero)

$$1101 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3$$
$$= 1 \times 1 + 0 \times 2 + 1 \times 4 + 1 \times 8$$
$$= 13$$

Converting decimal to binary

repeat while the number is > 0:

  divide the number by 2

  write the remainder (0 or 1)

  use the quotient as the number and repeat

the answer is the resulting sequence

  in reverse (right to left) order

divide 13 by 2, write "1", number is 6

divide 6 by 2, write "0", number is 3

divide 3 by 2, write "1", number is 1

divide 1 by 2, write "1", number is 0

answer is 1101

Decimal to binary conversion in Python

```
def dectobinary(num):
  if num == 0:
    return "0"
  binary = ""
  while num > 0:
    remainder = str(num % 2)
    binary = binary + remainder
    num //= 2
  return binary[::-1]

while True:
  num = input("Enter decimal number: ")
  bin = dectobinary(int(num))
  print("Binary representation of " + num + " is " + bin)
```