Computer Architecture I

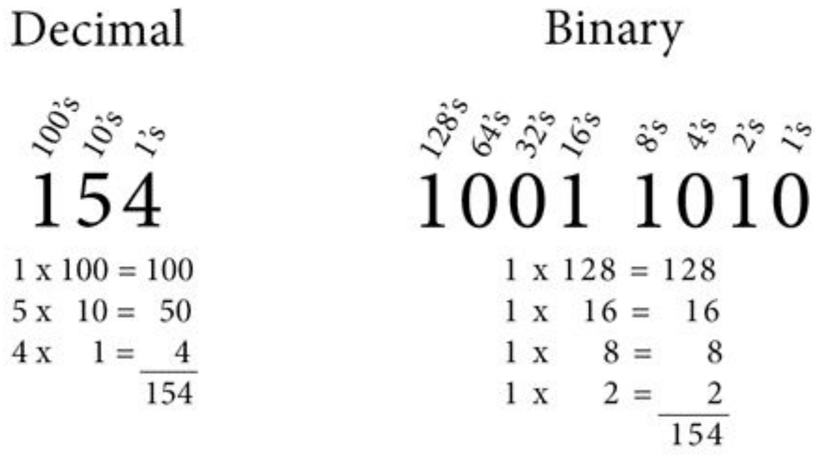
AGENDA

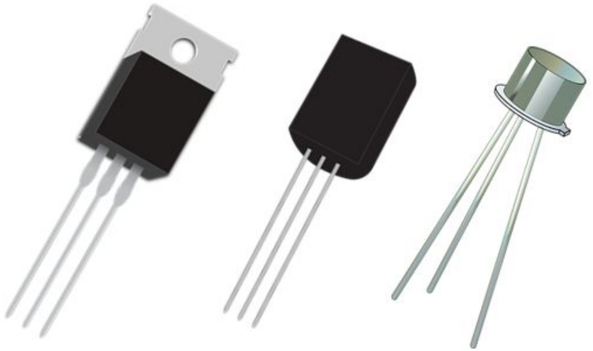
- Number Bases and Conversions
- Virtual Machines
- Building a Simple Data-driven Machine

Number Bases and Conversions

BASE TWO / BINARY

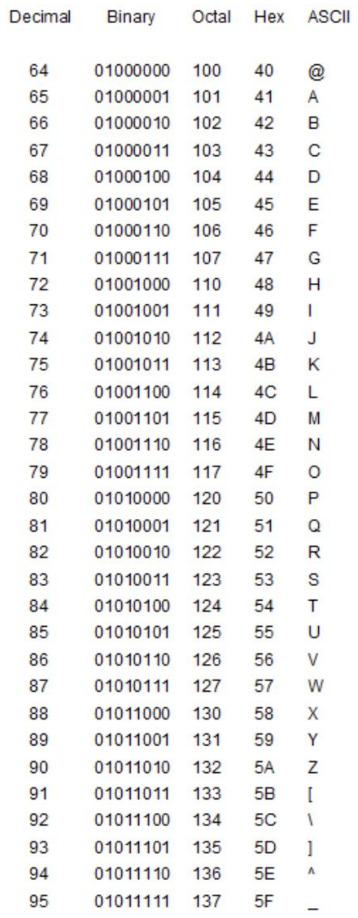
- A computer is built out of transistors
- A transistor can only represent two states: on (1/true) or off (0/false), hence why computers use binary
- This means everything must be able to be represented as binary for computers!





TEXT IN BINARY

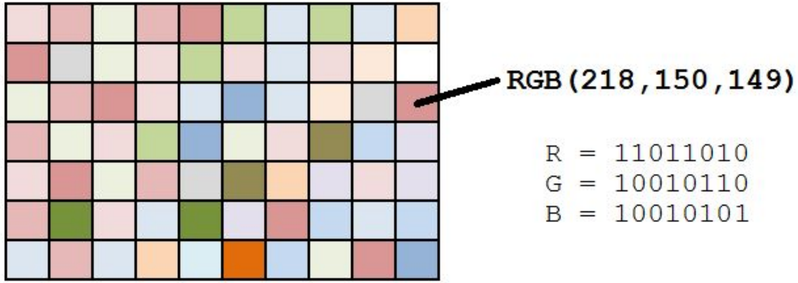
- To represent text, each computer maps a number to a certain character
- There are two main character encodings: Unicode, ASCII
- ASCII can represent $2^7 = 128$ characters
- Unicode can represent 2^{21} = ~2.1M characters



Decimal	Binary	Octal	Hex	ASCII
64	01000000	100	40	@
65	01000001	101	41	A
66	01000010	102	42	B
67	01000011	103	43	C
68	01000100	104	44	D
69	01000101	105	45	E
70	01000110	106	46	F
71	01000111	107	47	G
72	01001000	110	48	H
73	01001001	111	49	I
74	01001010	112	4A	J
75	01001011	113	4B	K
76	01001100	114	4C	L
77	01001101	115	4D	M
78	01001110	116	4E	N
79	01001111	117	4F	O
80	01010000	120	50	P
81	01010001	121	51	Q
82	01010010	122	52	R
83	01010011	123	53	S
84	01010100	124	54	T
85	01010101	125	55	U
86	01010110	126	56	V
87	01010111	127	57	W
88	01011000	130	58	X
89	01011001	131	59	Y
90	01011010	132	5A	Z
91	01011011	133	5B	[
92	01011100	134	5C	\
93	01011101	135	5D	]
94	01011110	136	5E	^
95	01011111	137	5F	_

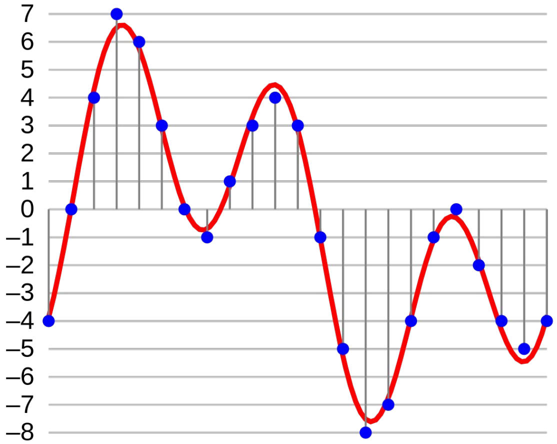
IMAGES AND VIDEO IN BINARY

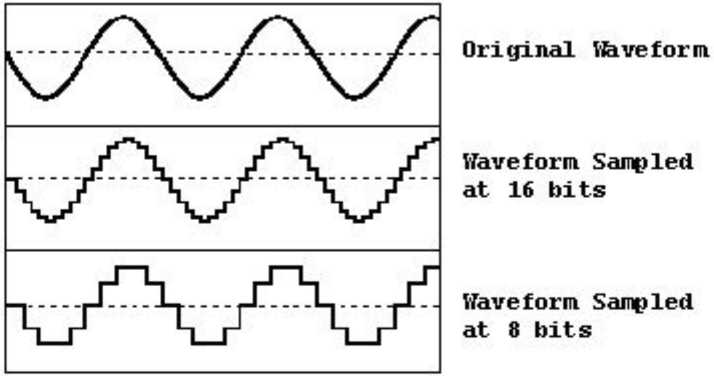
- Images are comprised of *pixels*
- Each pixel is comprised of a Red, Green, Blue (RGB) value which can be represented as a number
- Thus, the RGB value can be represented in binary



AUDIO IN BINARY

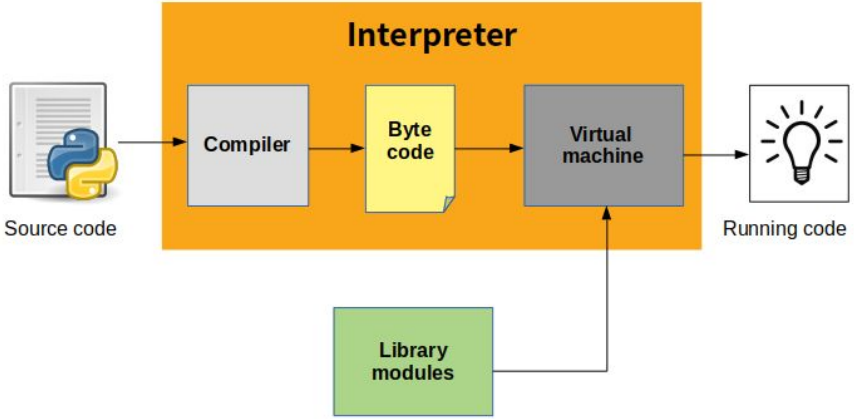
- Audio can be represented as a wave
- A wave can then be represented as a series of numbers which can be represented in binary





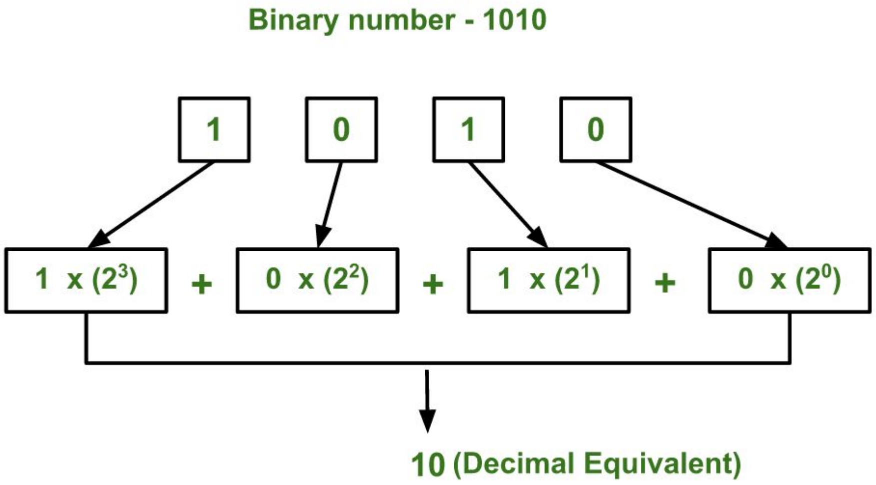
CODE IN BINARY

- There are multiple ways of converting code to binary. Python uses an *interpreter* to convert .py files to lower-level code
- *Compiler* - Translates .py code to ***byte code*** (a lower-level language)
- *Virtual machine* - Translates *byte code* to machine code that can be executed by the CPU
- Find this interesting? Check out this article



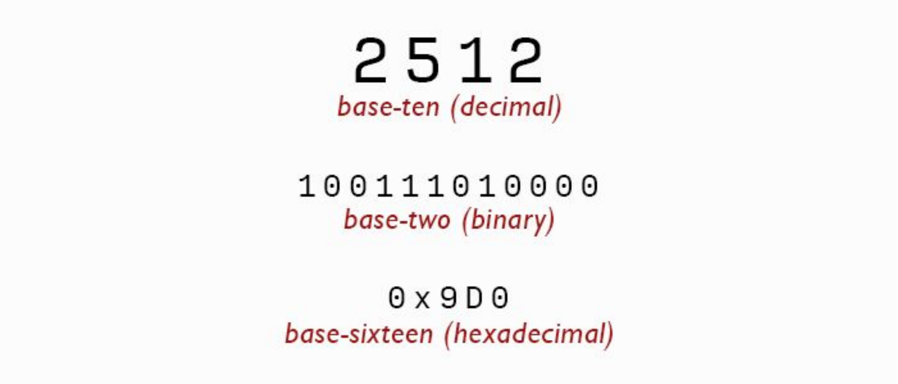
CONVERTING TO/FROM BINARY

- Convert the following from binary to decimal:
  - 10010
  - 11000
- Convert the following from decimal to binary:
  - 53
  - 15
- Max number *n* binary digits can represent is $2^n - 1$



BASE 16 / HEXADECIMAL

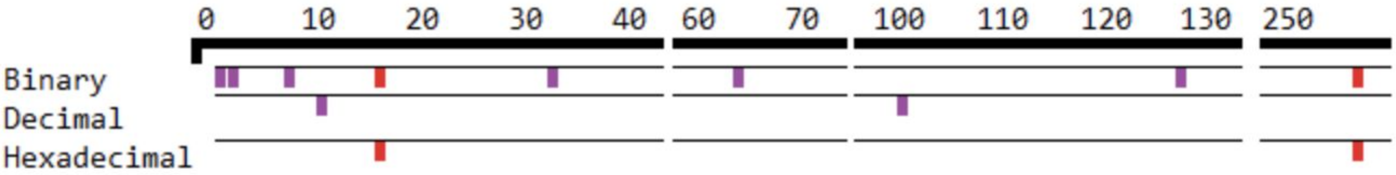
- A more readable and concise representation of binary
- Each digit can be *0-9* or *a-f*
- “But decimal is more readable!”



2512
base-ten (decimal)

100111010000
base-two (binary)

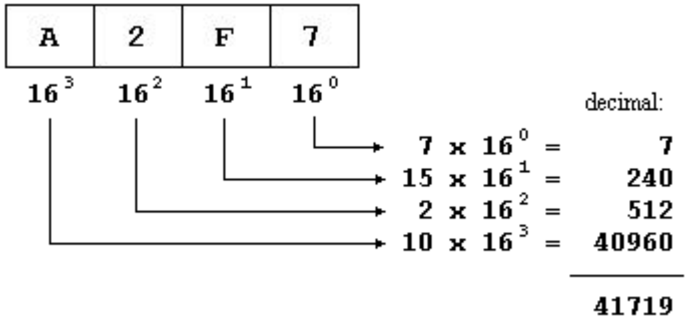
0x9D0
base-sixteen (hexadecimal)



Each purple tick is when a new digit is added when representing numbers

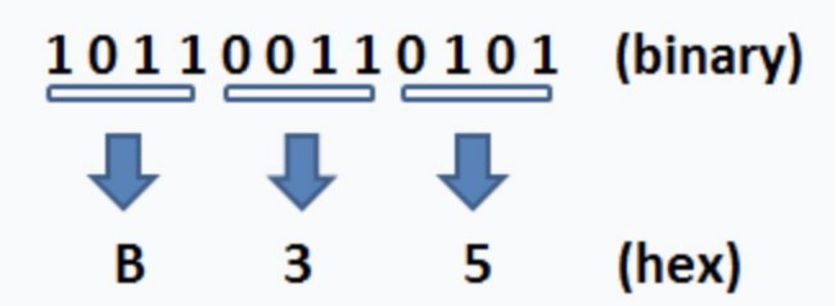
CONVERTING FROM HEXADECIMAL

- Convert the following numbers to decimal:
  - D34D
  - B33F
- Max number of n hex digits can represent is $16^n - 1$



CONVERTING BETWEEN BINARY AND HEXADECIMAL

- To convert binary to hex:
  - Divide the binary digits into groups of four (starting from the right)
  - Convert each group to hexadecimal
- To convert hex to binary:
  - Convert each hex digit to its binary equivalent
- Convert the following:
  - F12A to binary
  - 10111 to hex

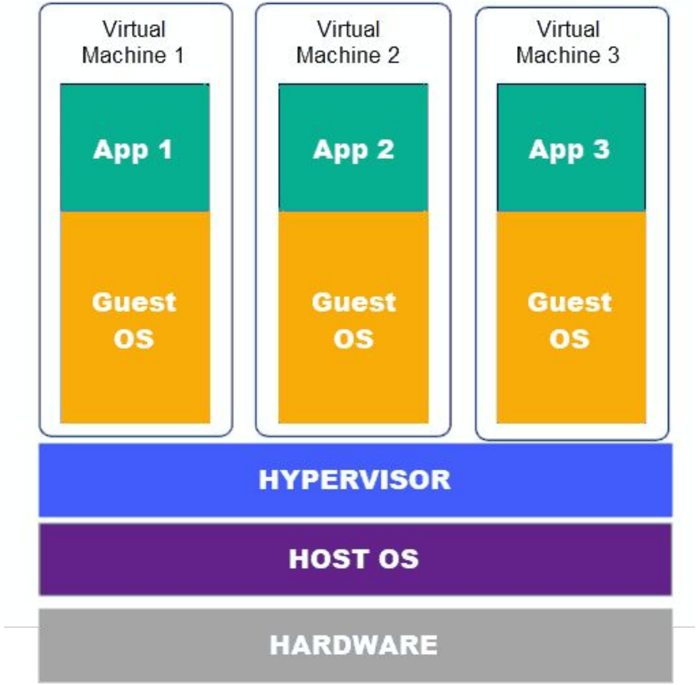


10 Minute Break

Virtual Machines

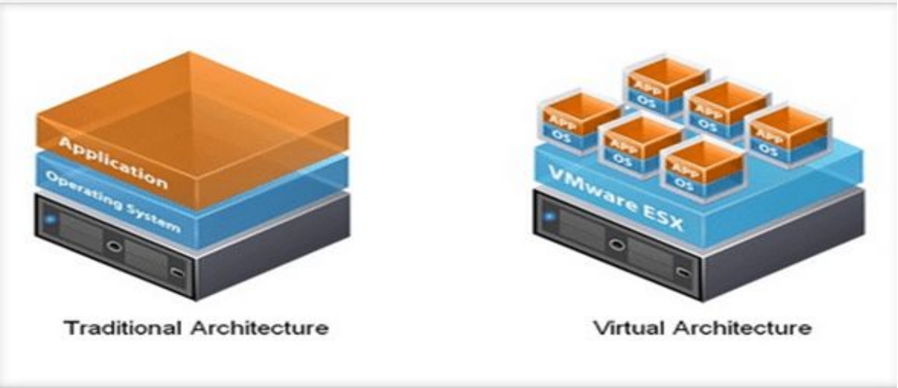
VIRTUAL MACHINES

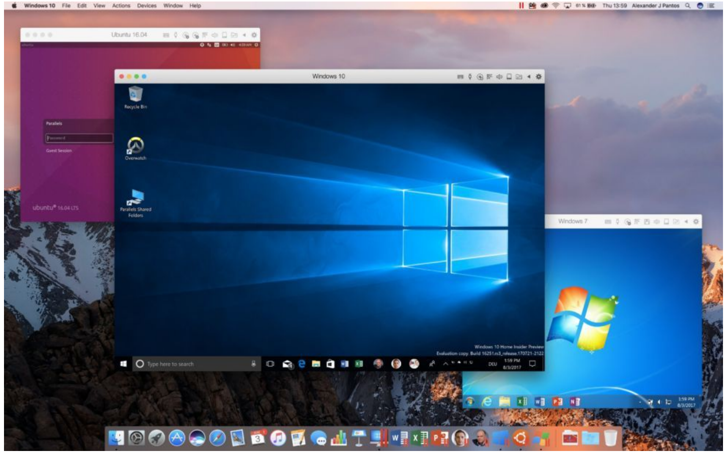
- *“...a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage” - Red Hat*
- They allow you to run a guest OS within another OS. The guest OS behaves like a full, separate computer



SAMPLES OF VIRTUAL MACHINES

- ***server virtualization*** - a single host running multiple virtual machines (e.g. VMWare)
- ***mac virtualization apps*** - Parallels, OracleVM, VMWare Player, etc.
- ***python interpreter virtual machine*** - translate lower-level byte code (from .py files) to machine code that can be executed by the CPU





Building a Simple Data-driven Machine

BUILDING A SIMPLE DATA-DRIVEN MACHINE

- A simple machine that reads instructions and values from memory
- This will be very similar to your project

Building a Simple Data-driven Machine Demo

BUILDING A SIMPLE DATA-DRIVEN MACHINE RECAP

- ***opcode*** - specifies the operation to be performed (e.g. HALT, SAVE, ADD)
- ***registers, random access memory (RAM)*** - temporary storage areas
- ***program counter*** - contains the location of the instruction being executed at the current time
    - This also determines the location of arguments
    - We advance this counter after executing the command

YOUR PROJECT: LS8

- A simple machine that executes instructions from a program
- This project will teach you concepts that are used by the CPU to run programs
- Make sure to read the spec!