

## Representing Data/Information

- What is the decimal based number system.
- How do other, non-decimal, systems work (binary, octal and hex).
- How to convert from a non-decimal number systems to decimal.
- How are negative and rational numbers represented on the computer.

James Tam

## Information Vs. Data

### •Data:

- The raw information without any context
- Examples (you don't have to interpret this data unless you are taught how to do this sometime during the semester).
  - 12, 07, 1941 (or Dec. 7, 1941)
  - 04, 05, 2063 (or April 5, 2063)
  - 01111000 00110010 01000101 (computer stores all data as binary)

### •Information:

- Data that has been processed or manipulated in order to provide a meaningful context
- Context for the above examples of data:
  - World War II: Japan launches a surprise attack on the American naval base at Pearl Harbor resulting on the latter's entry into the war.
  - Star Trek: The first contact of an extraterrestrial race (Vulcans) with humans and this leads to the eventual founding of the United Federation of planets.
  - Third example: If the bits are interpreted as text: x2E
  - Third example: if the bits are interpreted as (RGB) colors:



James Tam

## Data Vs. Information: Using A Computer

- Stores information as data (binary patterns) e.g.  $010 \times 011$
- If needed process the data e.g.  $010 \times 011 = 110$



User communicates information  
e.g. 1 what is  $2*3$   
e.g. 2, set color to red



Translate and display the information as output to the user  
e.g. 1 what is  $2*3=6$   
e.g. 2, change color of shape to red



James Tam

## Inappropriate Conversion: Data To Text

- Opening an image in a text-only program (Notepad)
- This is a pic of your course instructor - see the resemblance? ;)
  - It's 'garbage' because the bits are not interpreted as an image (e.g. 24 bits=a pixel) but interpreted as ASCII (e.g. 8 bits=a single character).

[illegible]

James Tam

## Alternative Presentation Of These Concepts

- To ensure that you don't "miss anything" I've included in the next three screens how Richard Zhao and Jonathan Hudson envisioned how this information was to be communicated in this course.

James Tam

## What is Data?



**Data:** raw facts, representation of information, no context



**Encoding:** The translation of information into data

(Decoding the other direction)



**Data represents information**

James Tam

## Information Processing

A change of information in any manner detectable by an observer



Using a computer?

Encode information  
into data

Process the data

Translate data back  
into information



**Moral: computers process *data*, not information – it is our responsibility to interpret the data correctly.**

James Tam

## Storing Data

All data in a computer is either a 0 or 1

Called a bit

Electrically, this is a switch  
that is either open or closed



**Encoding schemes translate integers, real numbers, letters, pictures, ... into bits**

James Tam

## Bit (*Binary Digit*)

- You have probably heard of this term in some context e.g. a hard drive is 1 Terabyte in capacity and there's 8 bits in 1 byte i.e. that hard drive is (1 byte = 8 bits so multiply capacity in bytes to determine capacity in bits – not commonly done) 8 Terabit capacity drive.
- All digital information is stored in the form of a bit (bits are **not just zeros and ones** as you may have been lead/mislead to believe).
- The key characteristic of a bit is that **information is stored in one of two states**.

James Tam

## Bits: Example Implementations

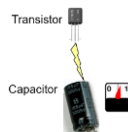
- Computer bus: wires that transit information via electrical signals on wires.



- Optical drives: CD/DVD



- RAM (requires the computer to be on to maintain power)



- Boolean variables (python examples)

```
isDone = True  
isDone = False
```

James Tam

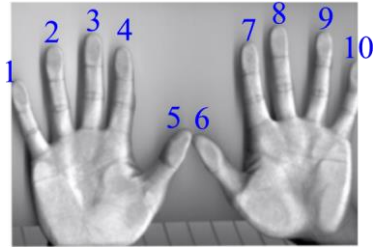
## What Is Decimal?

- Base 10

- 10 unique symbols are used to represent values

0
1
2
3
4
5
6
7
8
9
10
:

The number of digits is based on...the number of digits



The largest decimal value that can be represented by a single decimal digit is 9  
=  $\text{base}(10) - 1$

James Tam

## Representing Integer Information

- What you have been taught is the decimal-based system.

0	<p>Column 1 counts through all 10 possible values</p> <p>For the next value, column 1 resets back to zero and column 2 increases by one</p>	20
1		21
2		22
3		23
4		24
5		25
6		26
7		27
8		28
9		29
10	<p>Column 1 counts through all 10 possible values, column 2 remains unchanged</p>	30
11		31
12		Etc.
13		
14		
15		
16		
17		
18		
19		

For the next value, column 1 resets back to zero and column 2 increases by one

James Tam

## Decimal

- “Base ten”
- Employs ten unique symbols (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Each digit can only take on the value from 0 – 9
  - Once a column has traversed all ten values then that column resets back to zero (as does it right hand neighbours) and the column to it's immediate left increases by one.

James Tam

## Recall: Computers Don't Do Decimal!

- Most parts of the computer work in a discrete state:
  - On/off
  - True/false
  - Pitted/smooth
  - Connected/not connected
- These two states can be modeled with the binary number system.

James Tam

## Binary

- Base two
- Employs two unique symbols (e.g. 0 and 1)
- Largest value that can be represented by 1 binary digit =  $1 = \text{base}(2) - 1$ 
  - Confused? Apply this formula to decimal: largest value represented in decimal =  $\text{base}(10) - 1$
- Stepping through binary values.
  - Once a column has traversed both values then that column resets back to zero (as does it right hand neighbours) and the column to its immediate left increases by one.
  - This is similar to decimal but because each digit can only take on the value 0 or the value 1 (instead of 0 – 9 for decimal) increments will affect higher order (left most) columns more quickly.

James Tam

## Reminder: Incrementing By 1 In Decimal

Incrementing by one: decimal
0
1
2
...
8
9
10
11
12
13
...
98
99
100

Symbols are used up every 10 rows in the lowest order (right most/first) column.

James Tam



## Incrementing By 1: Binary

Increment by one: binary	
0	
1	
10	
11	
100	
101	
110	
111	

“Used up” all symbols in previous column, increase next column to left by 1

Right most column increment through all symbols (in this case 0,1). Symbols are used up every 2 rows.

James Tam

## Counting In Binary

Decimal value	Binary value	Decimal value	Binary value
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

James Tam

## Decimal: Represents Values Using Powers Of Ten

• Example: <sup>2 1 0</sup>482

- Breaking it down:

- 4 of the hundreds units      400
- 8 of the tens units      80
- 2 of the ones units      2
- Total      482

- These values are actually computed with powers of 10

- Hundreds units:  $4 \times 10^2 = 4 \times 100 = 400$
- Tens units:  $8 \times 10^1 = 8 \times 10 = 80$
- Ones units:  $2 \times 10^0 = 2 \times 1 = \underline{2}$
- Sum      482

- You can label the exponents by adding a **super script** above the digits.

James Tam

## Binary: Represents Values Using Powers Of Two

• Example:  $1011_2$  (binary) =  $11_{10}$  (decimal)

- Labeled with a super script (allows us to see the 'powers/exponents)

<sup>3 2 1 0</sup>  
1 0 1 1

- Breaking it down:

- $1 \times 2^3 = 1 \times 8 = 8$
- $0 \times 2^2 = 0 \times 4 = 0$
- $1 \times 2^1 = 1 \times 2 = 2$
- $1 \times 2^0 = 1 \times 1 = \underline{1}$
- Sum      11

James Tam

## You've Learned To Convert: Any Base To Decimal

For 231

- Evaluate the expression: the base raised to some **exponent**, multiply the resulting expression by the corresponding digit and sum the resulting products.

### • General formula:

$$\begin{array}{ccccccc} \textcolor{red}{3} & \textcolor{red}{2} & \textcolor{red}{1} & \textcolor{red}{0} & \textcolor{red}{-1} & \textcolor{red}{-2} & \textcolor{red}{-3} \\ d_7 & d_6 & d_5 & d_4 & d_3 & d_2 & d_1 \\ (d_7 \times b^3) + (d_6 \times b^2) + (d_5 \times b^1) + (d_4 \times b^0) + (d_3 \times b^{-1}) + (d_2 \times b^{-2}) + (d_1 \times b^{-3}) \end{array} \quad \leftarrow \text{Super script (power)}$$

### Example

$$\begin{array}{ccc} \textcolor{red}{1} & \textcolor{red}{0} & \textcolor{red}{-1} \\ 1 & 1 & .1_2 \end{array} \quad \leftarrow \text{Super script (power)}$$

$$\text{Value in decimal} = (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) = (1 \times 2) + (1 \times 1) + (1/2) = 2.5$$

$$1/2 : (1 \times 2^{-1}) = 1 \times 1/2^1 = 0.5$$

Reminder: evaluate a negative exponent by determining the reciprocal of a positive exponent

James Tam

## Students Do: Exercises

For 231

- Convert the following values from binary to decimal

$$100\,000_2 \rightarrow \text{?????}_{10}$$

$$100\,001_2 \rightarrow \text{?????}_{10}$$

$$111\,111_2 \rightarrow \text{?????}_{10}$$

$$0\,000\,001_2 \rightarrow \text{?????}_{10}$$

- To check your work:

- There are various online converter websites
- Alternatively: write a python program to do the calculation (hint: you can use the integer division and modulo operators to extract the digits from the number).

- You'll learn the (more involved) formula for converting from decimal in the next section.

- Quick look ahead: you keep performing integer divisions by the target base (in this case 2) until the remainder is less than the target base.

James Tam

## Why Bother With Binary?

1. It's the method of representing/storing information
  - ASCII (American Standard Code for Information Interchange)
  - UNICODE: it's augmented the original ASCII representations to include additions such as non-English languages.
    - UNICODE is stored on a electronic device using the UTF-8 system.
  - All information is stored on an electronic device is using some of binary e.g. your images, videos are encoded using some form of binary.
2. It's the language of the computer
  - A computer program must be translated to native machine language (specific to hardware/operating system 'platform') in order to run.
    - Interpreted languages such as python, JavaScript are translated each time the program executes.
    - Compiled languages (programs installed on your computer/phone) are translated once and it's the translated version that's installed (and then executed) on your device.

James Tam

## Representing Information: ASCII

- Uses 7 bits to represent characters
- Max number of possibilities =  $2^7 = 128$  characters that can be represented
  - 8 bits were actually used for various reasons: new characters could be represented, storing as 8 bits rather than 7 is easier etc.
- e.g., 'A' is 65 in decimal or 01000001 in binary. In memory it looks like this:

0	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

James Tam

## ASCII Codes

- <https://www.ascii-code.net/>
- Many codes produce a visible character e.g. letters, numbers, symbols.
- Other “control codes” are used for text formatting e.g. tab, new line etc.

CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX
[NUL]	0	00		32	20	@	64	40	'	96	60
[SOH]	1	01	!	33	21	A	65	41	a	97	61
[STX]	2	02	"	34	22	B	66	42	b	98	62
[ETX]	3	03	#	35	23	C	67	43	c	99	63
[EOT]	4	04	\$	36	24	D	68	44	d	100	64
[ENQ]	5	05	%	37	25	E	69	45	e	101	65
[ACK]	6	06	&	38	26	F	70	46	f	102	66
[BEL]	7	07	'	39	27	G	71	47	g	103	67
[BS]	8	08	(	40	28	H	72	48	h	104	68
[HT]	9	09	)	41	29	I	73	49	i	105	69
[LF]	10	0A	*	42	2A	J	74	4A	j	106	6A
[VT]	11	0B	+	43	2B	K	75	4B	k	107	6B
[FF]	12	0C	,	44	2C	L	76	4C	l	108	6C
[CR]	13	0D	-	45	2D	M	77	4D	m	109	6D
[SO]	14	0E	.	46	2E	N	78	4E	n	110	6E
[SI]	15	0F	/	47	2F	O	79	4F	o	111	6F
[DLE]	16	10	0	48	30	P	80	50	p	112	70
[DC1]	17	11	1	49	31	Q	81	51	q	113	71
[DC2]	18	12	2	50	32	R	82	52	r	114	72
[DC3]	19	13	3	51	33	S	83	53	s	115	73
[DC4]	20	14	4	52	34	T	84	54	t	116	74
[NAK]	21	15	5	53	35	U	85	55	u	117	75
[SYN]	22	16	6	54	36	V	86	56	v	118	76
[ETB]	23	17	7	55	37	W	87	57	w	119	77
[CAN]	24	18	8	56	38	X	88	58	x	120	78
[EM]	25	19	9	57	39	Y	89	59	y	121	79
[SUB]	26	1A	:	58	3A	Z	90	5A	z	122	7A
[ESC]	27	1B	;	59	3B	[	91	5B	{	123	7B
[FS]	28	1C	<	60	3C	\	92	5C		124	7C
[GS]	29	1D	=	61	3D	]	93	5D	}	125	7D
[RS]	30	1E	>	62	3E	^	94	5E	~	126	7E
[US]	31	1F	?	63	3F	_	95	5F	[DEL]	127	7F

## The ‘Gist’ Of Some Of The ASCII Codes

ASCII	Decimal	Binary
Invisible (control characters)	0 – 31	00000000 – 00011111
Punctuation, mathematical operations	32 – 47	00100000 – 00101111
Characters 0 - 9	48 - 57	00110000 – 00111001
Comparators and other miscellaneous characters : ; ? @	58 – 64	00111010 – 01000000
Alphabetic (upper case A - Z)	65 - 90	01000001 – 01011010
More miscellaneous characters [ \ ] ^ _ `	91 – 96	01011011 – 01100000
Alphabetic (lower case a - z)	97 – 122	01100001 – 01111010
More miscellaneous characters {   } ~ DEL	123 – 127	01111011 - 01111111

James Tam

## Drawbacks Of ASCII

- A good start but limited in what can be represented.
- “Latin character set” (alphabet).
- Accenting is not possible:  
- È ë Ć

CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX
[NUL]	0	00		32	20	@	64	40	`	96	60
[SOH]	1	01	!	33	21	A	65	41	a	97	61
[STX]	2	02	"	34	22	B	66	42	b	98	62
[ETX]	3	03	#	35	23	C	67	43	c	99	63
[EOT]	4	04	\$	36	24	D	68	44	d	100	64
[ENQ]	5	05	%	37	25	E	69	45	e	101	65
[ACK]	6	06	&	38	26	F	70	46	f	102	66
[BEL]	7	07	'	39	27	G	71	47	g	103	67
[BS]	8	08	(	40	28	H	72	48	h	104	68
[HT]	9	09	)	41	29	I	73	49	i	105	69
[LF]	10	0A	*	42	2A	J	74	4A	j	106	6A
[VT]	11	0B	+	43	2B	K	75	4B	k	107	6B
[FF]	12	0C	,	44	2C	L	76	4C	l	108	6C
[CR]	13	0D	-	45	2D	M	77	4D	m	109	6D
[SO]	14	0E	.	46	2E	N	78	4E	n	110	6E
[SI]	15	0F	/	47	2F	O	79	4F	o	111	6F
[DLE]	16	10	0	48	30	P	80	50	p	112	70
[DC1]	17	11	1	49	31	Q	81	51	q	113	71
[DC2]	18	12	2	50	32	R	82	52	r	114	72
[DC3]	19	13	3	51	33	S	83	53	s	115	73
[DC4]	20	14	4	52	34	T	84	54	t	116	74
[NAK]	21	15	5	53	35	U	85	55	u	117	75
[SYN]	22	16	6	54	36	V	86	56	v	118	76
[ETB]	23	17	7	55	37	W	87	57	w	119	77
[CAN]	24	18	8	56	38	X	88	58	x	120	78
[EM]	25	19	9	57	39	Y	89	59	y	121	79
[SUB]	26	1A	:	58	3A	Z	90	5A	z	122	7A
[ESC]	27	1B	;	59	3B	[	91	5B	{	123	7B
[FS]	28	1C	<	60	3C	\	92	5C		124	7C
[GS]	29	1D	=	61	3D	]	93	5D	}	125	7D
[RS]	30	1E	>	62	3E	^	94	5E	~	126	7E
[US]	31	1F	?	63	3F	_	95	5F	[DEL]	127	7F

James Tam

## ASCII Encoding

- What you should already know.
- Basic character information for the first 128 combinations (0-127) is specified by ASCII codes.
  - It includes upper/lower case alphabetic characters, all ten digits, punctuation and more...characters used in basic communications in English (and other languages which use alphabetic characters).
- 8 bits are used even though only 7 bits are needed ( $2^7 = 128$ )
  - The left most bit (most significant bit) is either used for error checking or simply set to zero.

James Tam

## UTF-8 Encoding

- Previously you were given a basic introduction.
- This form of encoding is used to represent additional text information which includes but is not limited to:
  - Emoji's/emoticons
  - Languages which use their own symbols rather than alphabetic characters (Latin character set).
- To allow backward compatibility: The first 128 UTF-8 codes are identical to the ASCII codes.
  - To save space only 1 byte is used to encode 0 - 127
- Other characters will require 2 to 4 bytes to encode.
  - 2 bytes are used to encode 128 – 65,535,  $2^{16} = 65,536$  combinations.
  - 3 bytes are used to encode 0-16,777,215,  $2^{24} = 16,777,216$  combinations.
  - 4 bytes are used to encode combinations above 16,777,216.
    - FYI for those who will ask:  $2^{32} =$  over 4 billion combinations (4.294.967.296)

James Tam

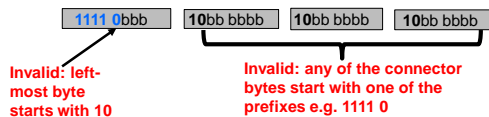
## Space Needed To Represent A Character.

- A character can be represented with one of the following possible lengths: 1 byte (8 bits), 2 bytes (16 bits), 3 bytes (24 bits), 4 bytes (32 bits).
- How do we 'know' the number of bytes used?
- The '**prefix**' (**left most bits**) is what distinguishes the 4 cases.
  - 1 byte: **0** 7 more bits **0bbb bbbb** b: bit can either be 0 or 1
  - 2 bytes: **110** 13 more bits **110b bbbb** **10bb bbbb**
  - 3 bytes: **1110** 20 more bits **1110 bbbb** **10bb bbbb** **10bb bbbb**
  - 4 bytes: **11110** 27 more bits **1111 0bbb** **10bb bbbb** **10bb bbbb** **10bb bbbb**
- The other bytes (if any) cannot contain these bit patterns at the start.
  - They must begin with **10** while the other 6 bits in the byte can either be 0 or 1.
  - "Continuation bytes" (not left most) start with the pattern **10**.

James Tam

## Benefits Gained Using UTF-8 Encoding

- Space efficiency.
  - More frequently occurring language sets only using common characters e.g. A-Z, a-z only require 1 byte to store.
- “Self synchronization” [JT: error handling]
  - If bit patterns appear in unexpected locations you know the information for the character is **invalid** (corrupted during storage or transmission).



- It's the standard encoding used by the WWW: default for html (web pages end in '.html' or 'htm' (older suffix) and most web protocols (protocol a method of transmission).
- Supported with many programming languages such as but not limited to python (avoids bugs due to mismatched encoding).

James Tam

## Python And UTF-8

- With python 3.x UTF-8 is the default method of encoding.
  - Just type the desired character and it will be properly displayed as the character is encoded/represented in the extended (beyond ASCII) approach i.e. more than just the basic alphabetic characters can be represented.

- **Name of the full example:** UTF8\_example\_multi\_lingual\_hello

```
print("Hello")
print("Bonjour")      #French
print("Hello po")     #Tagalog
print("Hallo")        #Dutch, German
print("Hola")         #Spanish

print("Allô")         #Quebecois French
print("你好")          #Traditional Chinese
print("مرحبا")         #Arabic
print("שלום")         #Hebrew
```

James Tam



## Unicode Vs. UTF-8

- Similar to ASCII each character is mapped to a unique numeric value using Unicode mappings.
  - Unlike ASCII Unicode includes more than just 128 mappings although the original mappings are used for these first combinations to ensure backward compatibility e.g. 'A' maps to 65 using ASCII or Unicode.
- UTF-8: the numeric Unicode is then stored using UTF-8 standards for encoding: 1 – 4 bytes.
  - E.g. 1, B Unicode=66 is stored using 1 byte.
  - E.g. 1, ċ Unicode=263 is stored using 2 bytes according to UTF-8 (Co-pilot)

	SOLAR CHAR		COSMIC CHAR		SOLAR CHAR		COSMIC CHAR	
DSOL	1	0.00	32	20	64	40	5	80
DSOL2	1	0.01	33	21	65	41	6	81
DSOL3	1	0.02	34	22	66	42	7	82
DSOL4	1	0.03	35	23	67	43	8	83
DSOL5	1	0.04	36	24	68	44	9	84
DSOL6	1	0.05	37	25	69	45	10	85
DSOL7	1	0.06	38	26	70	46	11	86
DSOL8	1	0.07	39	27	71	47	12	87
DSOL9	1	0.08	40	28	72	48	13	88
DSOL10	1	0.09	41	29	73	49	14	89
DSOL11	1	0.10	42	30	74	50	15	90
DSOL12	1	0.11	43	31	75	51	16	91
DSOL13	1	0.12	44	32	76	52	17	92
DSOL14	1	0.13	45	33	77	53	18	93
DSOL15	1	0.14	46	34	78	54	19	94
DSOL16	1	0.15	47	35	79	55	20	95
DSOL17	1	0.16	48	36	80	56	21	96
DSOL18	1	0.17	49	37	81	57	22	97
DSOL19	1	0.18	50	38	82	58	23	98
DSOL20	1	0.19	51	39	83	59	24	99
DSOL21	1	0.20	52	40	84	60	25	100
DSOL22	1	0.21	53	41	85	61	26	101
DSOL23	1	0.22	54	42	86	62	27	102
DSOL24	1	0.23	55	43	87	63	28	103
DSOL25	1	0.24	56	44	88	64	29	104
DSOL26	1	0.25	57	45	89	65	30	105
DSOL27	1	0.26	58	46	90	66	31	106
DSOL28	1	0.27	59	47	91	67	32	107
DSOL29	1	0.28	60	48	92	68	33	108
DSOL30	1	0.29	61	49	93	69	34	109
DSOL31	1	0.30	62	50	94	70	35	110
DSOL32	1	0.31	63	51	95	71	36	111
DSOL33	1	0.32	64	52	96	72	37	112
DSOL34	1	0.33	65	53	97	73	38	113
DSOL35	1	0.34	66	54	98	74	39	114
DSOL36	1	0.35	67	55	99	75	40	115
DSOL37	1	0.36	68	56	100	76	41	116
DSOL38	1	0.37	69	57	101	77	42	117
DSOL39	1	0.38	70	58	102	78	43	118
DSOL40	1	0.39	71	59	103	79	44	119
DSOL41	1	0.40	72	60	104	80	45	120
DSOL42	1	0.41	73	61	105	81	46	121
DSOL43	1	0.42	74	62	106	82	47	122
DSOL44	1	0.43	75	63	107	83	48	123
DSOL45	1	0.44	76	64	108	84	49	124
DSOL46	1	0.45	77	65	109	85	50	125
DSOL47	1	0.46	78	66	110	86	51	126
DSOL48	1	0.47	79	67	111	87	52	127
DSOL49	1	0.48	80	68	112	88	53	128
DSOL50	1	0.49	81	69	113	89	54	129
DSOL51	1	0.50	82	70	114	90	55	130
DSOL52	1	0.51	83	71	115	91	56	131
DSOL53	1	0.52	84	72	116	92	57	132
DSOL54	1	0.53	85	73	117	93	58	133
DSOL55	1	0.54	86	74	118	94	59	134
DSOL56	1	0.55	87	75	119	95	60	135
DSOL57	1	0.56	88	76	120	96	61	136
DSOL58	1	0.57	89	77	121	97	62	137
DSOL59	1	0.58	90	78	122	98	63	138
DSOL60	1	0.59	91	79	123	99	64	139
DSOL61	1	0.60	92	80	124	100	65	140
DSOL62	1	0.61	93	81	125	101	66	141
DSOL63	1	0.62	94	82	126	102	67	142
DSOL64	1	0.63	95	83	127	103	68	143
DSOL65	1	0.64	96	84	128	104	69	144
DSOL66	1	0.65	97	85	129	105	70	145
DSOL67	1	0.66	98	86	130	106	71	146
DSOL68	1	0.67	99	87	131	107	72	147
DSOL69	1	0.68	100	88	132	108	73	148
DSOL70	1	0.69	101	89	133	109	74	149
DSOL71	1	0.70	102	90	134	110	75	150
DSOL72	1	0.71	103	91	135	111	76	151
DSOL73	1	0.72	104	92	136	112	77	152
DSOL74	1	0.73	105	93	137	113	78	153
DSOL75	1	0.74	106	94	138	114	79	154
DSOL76	1	0.75	107	95	139	115	80	155
DSOL77	1	0.76	108	96	140	116	81	156
DSOL78	1	0.77	109	97	141	117	82	157
DSOL79	1	0.78	110	98	142	118	83	158
DSOL80	1	0.79	111	99	143	119	84	159
DSOL81	1	0.80	112	100	144	120	85	160
DSOL82	1	0.81	113	101	145	121	86	161
DSOL83	1	0.82	114	102	146	122	87	162
DSOL84	1	0.83	115	103	147	123	88	163
DSOL85	1	0.84	116	104	148	124	89	164
DSOL86	1	0.85	117	105	149	125	90	165
DSOL87	1	0.86	118	106	150	126	91	166
DSOL88	1	0.87	119	107	151	127	92	167
DSOL89	1	0.88	120	108	152	128	93	168
DSOL90	1	0.89	121	109	153	129	94	169
DSOL91	1	0.90	122	110	154	130	95	170
DSOL92	1	0.91	123	111	155	131	96	171
DSOL93	1	0.92	124	112	156	132	97	172
DSOL94	1	0.93	125	113	157	133	98	173
DSOL95	1	0.94	126	114	158	134	99	174
DSOL96	1	0.95	127	115	159	135	100	175
DSOL97	1	0.96	128	116	160	136	101	176
DSOL98	1	0.97	129	117	161	137	102	177
DSOL99	1	0.98	130	118	162	138	103	178
DSOL100	1	0.99	131	119	163	139	104	179
DSOL101	1	1.00	132	120	164	140	105	180
DSOL102	1	1.01	133	121	165	141	106	181
DSOL103	1	1.02	134	122	166	142	107	182
DSOL104	1	1.03	135	123	167	143	108	183
DSOL105	1	1.04	136	124	168	144	109	184
DSOL106	1	1.05	137	125	169	145	110	185
DSOL107	1	1.06	138	126	170	146	111	186
DSOL108	1	1.07	139	127	171	147	112	187
DSOL109	1	1.08	140	128	172	148	113	188
DSOL110	1	1.09	141	129	173	149	114	189
DSOL111	1	1.10	142	130	174	150	115	190
DSOL112	1	1.11	143	131	175	151	116	191
DSOL113	1	1.12	144	132	176	152	117	192
DSOL114	1	1.13	145	133	177	153	118	193
DSOL115	1	1.14	146	134	178	154	119	194
DSOL116	1	1.15	147	135	179	155	120	195
DSOL117	1	1.16	148	136	180	156	121	196
DSOL118	1	1.17	149	137	181	157	122	197
DSOL119	1	1.18	150	138	182	158	123	198
DSOL120	1	1.19	151	139	183	159	124	199
DSOL121	1	1.20	152	140	184	160	125	200
DSOL122	1	1.21	153	141	185	161	126	201
DSOL123	1	1.22	154	142	186	162	127	202
DSOL124	1	1.23	155	143	187	163	128	203
DSOL125	1	1.24	156	144	188	164	129	204
DSOL126	1	1.25	157	145	189	165	130	205
DSOL127	1	1.26	158	146	190	166	131	206
DSOL128	1	1.27	159	147	191	167	132	207
DSOL129	1	1.28	160	148	192	168	133	208
DSOL130	1	1.29	161	149	193	169	134	209
DSOL131	1	1.30	162	150	194	170	135	210
DSOL132	1	1.31	163	151	195	171	136	211
DSOL133	1	1.32	164	152	196	172	137	212
DSOL134	1	1.33	165	153	197	173	138	213
DSOL135	1	1.34	166	154	198	174	139	214
DSOL136	1	1.35	167	155	199	175	140	215
DSOL137	1	1.36	168	156	200	176	141	216
DSOL138	1	1.37	169	157	201	177	142	217
DSOL139	1	1.38	170	158	202	178	143	218
DSOL140	1	1.39	171	159	203	179	144	219
DSOL141	1	1.40	172	160	204	180	145	220
DSOL142	1	1.41	173	161	205	181	146	221
DSOL143	1	1.42	174	162	206	182	147	222
DSOL144	1	1.43	175	163	207	183	148	223
DSOL145	1	1.44	176	164	208	184	149	224
DSOL146	1	1.45	177	165	209	185	150	225
DSOL147	1	1.46	178	166	210	186	151	226
DSOL148	1	1.47	179	167	211	187	152	227
DSOL149	1	1.48	180	168	212	188	153	228
DSOL150	1	1.49	181	169	213	189	154	229
DSOL151	1	1.50	182	170	214	190	155	230
DSOL152	1	1.51	183	171	215	191	156	231
DSOL153	1	1.52	184	172	216	192	157	232
DSOL154	1	1.53	185	173	217	193	158	233
DSOL155	1	1.54	186	174	218	194	159	234
DSOL156	1	1.55	187	175	219	195	160	235
DSOL157	1	1.56	188	176	220	196	161	236
DSOL158	1	1.57	189	177	221	197	162	237
DSOL159	1	1.58	190	178	222	198	163	238
DSOL160	1	1.59	191	179	223	199	164	239
DSOL161	1	1.60	192	180	224	200	165	240
DSOL162	1	1.61	193	181	225	201	166	241
DSOL163	1	1.62	194	182	226	202	167	242
DSOL164	1	1.63	195	183	227	203	168	243
DSOL165	1	1.64	196	184	228	204	169	244
DSOL166	1	1.65	197	185	229	205	170	245
DSOL167	1							

Mappings from  
128-255 reserved  
for ASCII

Mappings after 256 are specified using Unicode

## 1 Additional explanations

- <https://askanydifference.com/difference-between-unicode-and-utf-8/> (last viewed 2025)
- <https://blog.hubspot.com/website/what-is-utf-8> (last viewed 2024)

James Tam

## Alternative Presentation Of These Concepts

- Here's an alternative presentation of UTF-8 as an optional resource to help you understand this method of representation.
- The next two screens were created by Richard Zhao and Jonathan Hudson.

James Tam

## Representing More Characters

### •UTF-8

- Another encoding scheme for characters
  - Variable length – 1, 2, 3 or 4 bytes per character
- Compatible with ASCII
- Consider each byte
  - Left most bit is 0? Usual ASCII Character
  - Left most bits are 110? 2 byte character
  - Left most bits are 1110? 3 byte character
  - Left most bits are 11110? 4 byte character
- “tears of joy” emoji
  - 0x F0 9F 98 82  
which is
  - 11110000 10011111 10011000 10000010



James Tam

## UTF-8

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

James Tam

## Encoding Data: References

- ASCII

- <https://www.bbc.co.uk/bitesize/guides/zsnbr82/revision/5>

- UTF-8:

- [https://www.w3schools.com/charsets/ref\\_html\\_utf8.asp](https://www.w3schools.com/charsets/ref_html_utf8.asp) (last viewed fall 2024).

- <https://blog.hubspot.com/website/what-is-utf-8> (last viewed fall 2024).

- "UTF-8 support in the Microsoft GDK". Microsoft Learn. Microsoft Game Development Kit (GDK). Retrieved 2023-03-05.

- "Encoding Standard". encoding.spec.whatwg.org. Retrieved 2020-04-15.

James Tam

## A Problem With Binary

For 231

- 1001 0100 1100 1100?

- 1001 0100 1100 0100?

- 1001 0100 1100 0011?

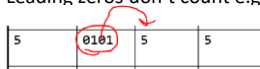
Binary is not intuitive  
for human beings and  
one string of binary  
values can be easily  
mistaken for another

James Tam

## A Shorthand For Binary: Octal

•Machine •language	Octal value
•1010111000000	012700
•1001010000101	011205

- Converting from binary to octal: 3 bits are grouped into one octal digit
  - E.g. 110 = 5
  - You can look the values up in the table provided at the end of these notes.
  - Leading zeros don't count e.g. 0100 in decimal is the same as 00100 or 100.



5	0101	5	5
---	------	---	---

James Tam

## Octal

- Base eight
- Employs eight unique symbols (0 - 7)
- Largest value that can be represented by 1 octal digit = 7 = base(8) - 1

James Tam

## Incrementing By 1: Octal

Increment by one: octal

0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
...
75
76
77
100

"Used up" all symbols in previous column, increase next column to left by 1

Right most column increment through all symbols (in this case 0,1,2,3,4,5,6,7)

James Tam

## Table Of Octal Values

Decimal value	Octal value	Decimal value	Octal value
0	0	8	10
1	1	9	11
2	2	10	12
3	3	11	13
4	4	12	14
5	5	13	15
6	6	14	16
7	7	15	17

James Tam

## Octal: Represents Values Using Powers Of Eight

• Example:  $273_8$  (octal) =  $187_{10}$  (decimal)

- Labeled with a super script lets us to see the powers/exponents.

2 1 0  
2 7 3

- Breaking it down

•  $2 \times 8^2 = 2 \times 64 = 128$

•  $7 \times 8^1 = 7 \times 8 = 56$

•  $3 \times 8^0 = 3 \times 1 = \underline{3}$

• Sum 187

James Tam

## Students Do: Exercises

• Convert the following values from octal to decimal

$6_8 \rightarrow \text{?????}_{10}$

$7_8 \rightarrow \text{?????}_{10}$

$10_8 \rightarrow \text{?????}_{10}$

$26_8 \rightarrow \text{?????}_{10}$

$31_8 \rightarrow \text{?????}_{10}$

$245_8 \rightarrow \text{?????}_{10}$

$712_8 \rightarrow \text{?????}_{10}$

Computer geek joke: Oct 31 is actually Dec 25 (hint: it's an application of the lesson not just some random humor).

James Tam

## Problems With Binary: Got Worse As Computers Got More Powerful

For 231

- 1001 0100 1000 0000 1100 0100 0110 1010?
- Or
- 1001 0100 1000 0000 1100 0100 0110 1011?

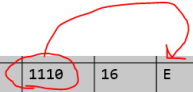
James Tam

## Hexadecimal: An Even More Compact Way Of Representing Binary Instructions

For 231

- | Machine language  | Hexadecimal value |
|-------------------|-------------------|
| • 1010011000001   | 14C1              |
| • 110000011100000 | 60E0              |

- FYI: 4 binary digits are grouped to represent 1 hexadecimal digit
  - e.g. 1110 = E
  - (You can look these values up in the table near the end of these notes).



14	1110	16	E
----	------	----	---

Example from 68000 Family Assembly Language by Clements A

James Tam

## Hexadecimal ('Hex' For Short)

- Base sixteen
- Employs sixteen unique symbols (0 – 9, followed by A - F)
- Largest decimal value that can be represented by 1 hex digit = 15

## Incrementing By 1: Hexadecimal

Increment by one:	Increment by one:
0	99
1	9A
2	9B
3	...
4	9F
5	A0
6	A1
7	A2
8	...
9	F0
A	F1
B	F2
C	F3
D	F4
E	F5
F	F6
10	F7
11	F8
12	F9
13	FA
14	...
15	FE
16	FF
17	100
..	101

"Used up" all symbols in previous column, increase next column to left by 1

Right most column increment through all symbols (in this case 0,1,2,3...D,E,F)



## Table of Hexadecimal Values

Decimal value	Hexadecimal value	Decimal value	Hexadecimal value
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

James Tam

## Hexadecimal: Represents Values Using Powers Of 16

- Example:  $2A8_{16}$  (hex) =  $680_{10}$  (decimal)

- Labeled with a super script (allows us to see the 'powers/exponents')

2 1 0

2 A 8

- Breaking it down:

- $2 \times 16^2 = 2 \times 256 = 512$
- $4 \times 16^1 = 10 \times 16 = 160$
- $8 \times 16^0 = 8 \times 1 = 8$
- Sum 680

James Tam

## Students Do: Exercises

- Convert the following values from hexadecimal to decimal

$6_{16} \rightarrow \text{?????}_{10}$

$A_{16} \rightarrow \text{?????}_{10}$

$C_{16} \rightarrow \text{?????}_{10}$

$F_{16} \rightarrow \text{?????}_{10}$

$26_{16} \rightarrow \text{?????}_{10}$

$245_8 \rightarrow \text{?????}_{10}$

$712_8 \rightarrow \text{?????}_{10}$

James Tam

## Summary (Decimal, Binary, Octal, Hex)

Decimal	Binary	Octal	Hex	Decimal	Binary	Octal	Hex
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

James Tam

## Other Units Of Measurement: Positive Numbers

- By itself a bit is inadequate for representing values.

- Only 2 values (0,1 or zero to  $2^0-1$ ) =  $2^0$

**1 bit/2 combos**

0
1

**2 bit/4 combos**

00
01
10
11

- Larger groupings are needed.

- **Byte** uses 8 bits e.g. 0001 1000

• 256 values =  $2^8$  (range from 0-255 or  $0-2^8-1$ )

- **Half word/short** uses 16 bits:

• 65,536 values =  $2^{16}$  (range from 0-65,535 or  $0-2^{16}-1$ )

- **Word** uses 32 bits:

•  $2^{32}$  ~4 billion values (range from 0- $2^{32}-1$ )

- **Double word/long** uses 64 bits

•  $2^{64}$  possible values (range from 0- $2^{64}-1$ )

James Tam

## Units Of Measurement By Operating System

Value	Windows	MAC
1 byte	8 bits	8 bits
1 Kilobyte	1,000 bytes or $1024 = 2^{10}$	1000 bytes (1 KB)
1 Megabyte	~~1,000,000 bytes or $1,048,576 = 2^{20}$	1,000,000 bytes (1 MB)
1 Gigabyte	~1 billion bytes or $1,073,741,824 = 2^{30}$	1,000,000,000 bytes (1 GB)
1 Terabyte	~1 trillion bytes = $2^{40}$	1,000,000,000,000 bytes (1 TB)

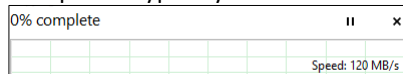
- Keep in mind whether bits or bytes are used when specifications are listed.

- Storage (hard drives), memory (RAM) use bytes

- ISP transmission speeds use bits e.g. 1,000 M**bps** is actually 1,000

M**bits**/second (125 MBs or 125,000,000 bytes, Blu-ray stores ~25 GB or 25,000,000,000 = 200 seconds).

- Computers typically show transmissions as bytes/second.



James Tam

## Units Of Measurement: Some **Computer Applications**

- Bytes (example computer ad):

- Acer Predator P03 Gaming PC
- Intel Core i7-13700F 2.1 / **5.2 GHz**
- **16 GB** of RAM
- **1 TB** HDD/**512 GB**

- Bits:

- (From the above ad)
- RJ45 Ethernet, Wi-Fi 6E, Bluetooth 5.3, 10 USB ports (Front=4: Type A USB **10 Gbps**, Type C USB **20 Gbps**; Back: USB Type A **480 Mbs**x4, USB Type A **5 Gbps** x 2, HDMIx1, Display portx3
- What does this mean:
  - Copying the contents of 1 TB storage device (speeds is the ideal max. – often not realized)
    - o USB C 20 Gbps: 6.67 minutes
    - o USB A 480 Mbps: 277.78 minutes
  - More examples and the formulas used to derive the above times:
    - o [https://cspages.ucalgary.ca/~tam/2025/217F/examples/representations/Transfer\\_speed\\_comparisons.xlsx](https://cspages.ucalgary.ca/~tam/2025/217F/examples/representations/Transfer_speed_comparisons.xlsx)
- Date of last access
  - Last accessed from [www.bestbuy.ca](http://www.bestbuy.ca) January 2024

James Tam

## Representing Negative Numbers

- One bit is used for the sign, the other bits (# bits - 1) is used to represent the magnitude e.g. 4 bits (1 for sign, 3 for magnitude)

- There are different sub-approaches that use a sign bit.

- Sign-magnitude is the one covered in this class now.
- (The others ones and twos complement may be covered later this term or in another class, likely CPSC 355).
- Sign magnitude with 8 bits

- Format:

s   bbb   bbbb

Sign bit:  
0 = positive  
1 = negative

Magnitude:  
Represents quantity (size)

- Examples:

0 000 0011   (positive 3)  
1 000 0100   (negative 4)

James Tam

## Representing Real Numbers

- The approximation of a real number is represented as a float.
- Standard Representation is IEEE 754 Floating Point
  - 0.0002589 becomes  $-2.589 * 10^{-4}$
- 32-bit floating point representation:
  - sign (1 bit), exponent (8 bits), mantissa (23 bits)
- 64-bits:
  - sign (1 bit), exponent (11 bits), mantissa (52 bits)
- Why an approximation?
  - Not all values can be represented e.g.  $0 \leq \text{range} \leq 1$  there's an infinite number of values.
    - Or a single expression can have an infinite number of possible digits e.g.  $1/3$
  - Even with 64 (or more bits) to store a real number some values will missed or stored incorrectly.

James Tam

## Illustrative Example: Why Floats Only Approximate Real Values

- Example of (non IEEE floating point): **5 digits** used to represent the mantissa:
- The exponent affects how much the decimal shifts 'floats'
  - Size of the exponent indicates the number of shifts e.g. squared = 2 shifts
  - Sign of the exponents specifies direction.
    - Negative exponent: shift right when storing as floating point
    - Positive exponent: shift left when store as floating point
  - e.g. One: 123.45 is represented in floating point as  $12345 * 10^{-2}$
  - e.g. Two: 0.12 is represented in floating point as  $12000 * 10^{-5}$
  - e.g. Three: 123456 is represented in floating point as  $12345 * 10^1$
  - Notice: in the last example one digit is lost during storage.
    - This serves to illustrate why floating point (any programming language) only approximates real values.
    - Precision may be lost even with non-repeating rational values.

James Tam

## Programming Lesson: Floating Point Representation

- Note: the **drawbacks** that come from storing real values is **due to the floating point representation** (not unique to python).
- Avoid floats when an integer will do!
  - Example: represent currency as whole cents rather than fractional dollar values.
- Never check for equality when comparing floats.
  - **Name of the full example:** `2_floats_not__same_as_real.py`  
`num = 1.0 - 0.55`  
`print(num == 0.45): #Don't do this!`
- If a float is necessary then consider the use of an epsilon.
  - Example use (if you're interested, it will be covered this semester if the specific need arises):
    - <https://pages.cpsc.ucalgary.ca/~tamj/2021/217P/> (Branching Part II)

James Tam

## Copyright Notification

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.

slide 60

James Tam