Bases

Learning Outcomes

- Convert between different bases
- Be familiarised with binary and hexadecimal

A Brief Background on Bases

> "There are these two young fish swimming along and they happen to meet an older fish swimming the other way, who nods at them and says"Morning, boys. How's the water?" And the two young fish swim on for a bit, and then eventually one of them looks over at the other and goes "What the hell is water?"
> **- David Foster Wallace, This Is Water: Some Thoughts, Delivered on a Significant Occasion, about Living a Compassionate Life**

It's not often we need to think about bases, but in computer science it is fundamental. Bases, as the name suggests, are at the core of math we use. For example anything *Decimal* such as money, or anything measured under a metric system will reliably be in **Base 10**. Seconds and Minutes are, oddly enough, in **Base 60**.

There is nothing special about why these bases are chosen as our staples (Would we still like base 10 if we had 6 fingers on each had rather than 5?). Would we have base 60 if ancient Sumerians didn't have the tradition of counting to 12 on one hand using their knuckles, and found base 60 was the most compatible choice for working with fractions?

Just like the seconds or minutes on a clock, **the base is the number you never reach**. You never meet a friend at 60 minutes past 3.

If you were to count to 5 in base 4 you would go:

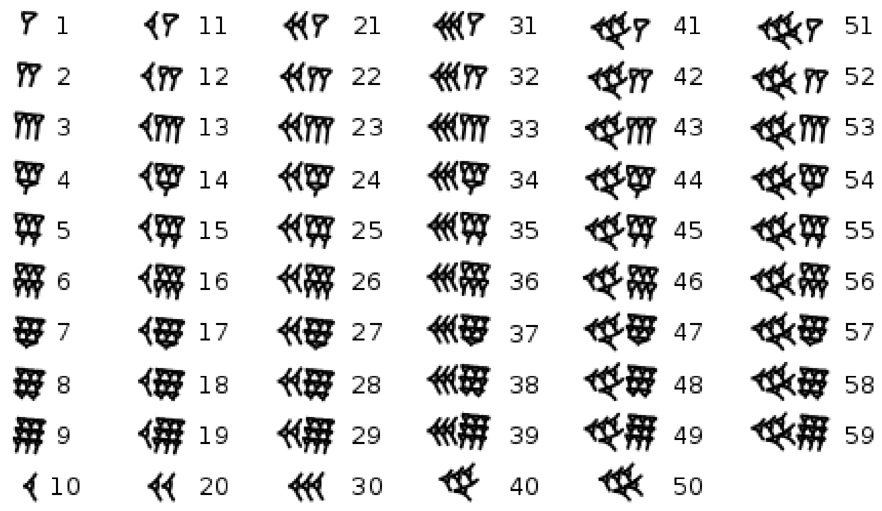

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

Figure 1: Sumerian-inspired Babylonian Numeric System

Base 4	Base 10
0	0
1	1
2	2
3	3
10	4

That is to say, for 10_4 there is a single 4 (the base), and no ones. Base 10 doesn't have a numeral after 9, it just ticks over to 1 ten and 0 ones.

What in the world does he mean by that?

Don't worry aspiring mathematician, we'll get to that

Why are bases useful? Since the dawn of time humankind has developed increasingly clever ways to be lazy; a base allows us to be lazy by putting aside some of the numbers so that we don't need to think about them. A dozen-and-one-eggs saves you from having to think about 13 different eggs, you are just thinking about "a dozen eggs" and "one egg". Similarly now we think about 13 eggs; a group of ten eggs, and a group of three eggs.

Examples of Arbitrary Bases

Bases smaller than 10:

Alright enough self-indulgent intro for now. Here is a visual representation of how a base works.

Base	10 ^{2}			
BasePower	103	102	101	100
=	1000	100	10	1
Numeral	2	3	1	9

So what on Earth does this mean? Well if we had the number 2319_{10} (I will use $_{10}$ interchangeably with *Base 10*, which is also what we refer to as *decimal*), visualise the steps you are taking)

Base	8		
BasePower	82	81	80
=	64	8	1
Numeral	1	7	3

You may now also notice that when evaluating basepower we are displaying the result in base 10.

$$1 \times 64 + 8 \times 7 + 3 \times 1 = 123$$

Show/hide answer

Shortcut to appendix for PDF users

Extending this logic to converting between one arbitrary base to another, say from base 4 to base 7, is left as an exercise to the reader.

Binary

> “So the first distinction is between none and one (or 0 and 1). This is the yin (negative) and yang (positive) polarity of the Book of Changes, which Leibniz read in a Latin translation, and which gave him the idea that all numbers could be represented by the figures 0 and 1, so that for the series 1, 2, 3, 4, 5 we have 001, 010, 100, 101, 110, etc., which is now the arithmetic used by digital computers.” **- Alan Watts**

Maybe people see everything as on-or-off due to the way our brains are wired, with neurons in on/off states to represent the experienced world. Maybe the world can boil down to a complicated expression of relative on/offs. Tall, short, fast, slow, divine, grotesque. In any case computing with 1’s and 0’s trace all the way back to their precursor of *yin* and *yang*, and can be used to describe anything from numbers, to sound, and images. If we can experience it, and model it, we can usually express that model in binary.

It is also it’s on/off property which makes it perfectly suited for computers, which at a circuitry level, consist of transistors in either an on or off state.

Later, we look at binary logic, but for now let’s call binary just base 2.

So with the knowledge of how we handled arbitrary base systems earlier, try converting 0001 0110 from base 2 into base 10.

Base	2				
BasePower	24	23	22	21	20

Base	2				
=	16	8	4	2	1
Numeral	1	0	1	1	0

$16 + 4 + 2 = 22$

*It's good practice to divide binary into groups of 4 bits at a time for ease of visibility (e.g. 1100 0011). A 'byte' in computing (yes, the same byte in **megabyte**) refers to 8 such bits of data*

Show/hide answer

Shortcut to appendix for PDF users

Mini-challenge: Now convert 7 into binary. How could you tell from looking at a string of bits whether a number in binary was odd or even?

Arbitrary bases larger than 10

Now it may seem trivial that 10 comes after 9, but what is 10 *really*? 10 isn't a numeral; it's 1 "ten" and zero "ones".

Well now he's just being confusing!
(Hang in there)

Say we wanted to represent 10 in base 12. How would we write it? Can we just write `10`?

12^1	12^0
1	0

Well that just doesn't work.

$$1 \times 12 + 0 \times 1 = 12$$

We've unintentionally written 12_{10} by thinking $10_{10} = 10_1 2$.
So what do we do when we run out of numerals after 9? Just use the alphabet.

`0, 1, 2,...,9, A, B, 10, 11`

Seeing as B is the 11th number, the 12th number becomes 10, which is to say there is 1×12^1 and 0×12^0.

There we go, we just counted to 13 in base 12.

Try counting to 15 in base 13.

Hexadecimal: How I Learned To Stop Worrying and Love Base 16

As mentioned previously (If spoiler tags are working, and you're not viewing this a book) a single byte is 8 bits, usually written out as two blocks of 4 bits. What's the largest number that can be represented by 4 bits?

$$1111 = 8 + 4 + 2 + 1 \quad = \ ?$$

That's right! it's 15!

Hexidecimal (Street-lingo: hex) is commonly used to represent a byte, as rather than looking at a large batch of 1's and 0's you can look at a quarter as many 0-Fs. A few of you have seen hex used for representing color. For color hex is used as the intensity of Red, Green, and Blue; in the form `#RRGGBB`, e.g. : `#2a7a72` (Teal),

Side note: Numbers in hex (hexidecimal) are often shown in the form 0x___, such as 0x01.

1. What is $0x1E_{16}$ in decimal?
   - What is 127_{10} in decimal?
   - What is $6F_{15}$ in binary?
2. What is the largest value you can represent with a byte?
   - What does it look like in binary?
   - What does it look like in Hex?
   - How many different values can a byte represent?
3. What colour do you think (red, green, blue) = (247, 85, 49) would look like?
   - What would its value be as a hex code (Hint: `#__ __ __`)

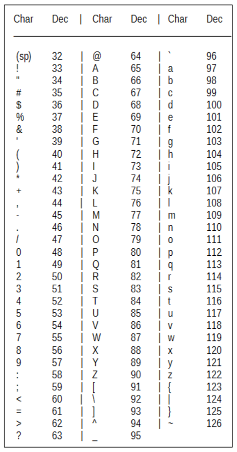
Information Overload: Funnily enough, every single thing stored on your computer is actually a pair of hex. There will be an address, like a street address, of where the information is stored. The number of the address is in hex, as in the information contained at that address.

```
0x0000: 14 00 20 00  0A 00 01 00  69 00 0F 00
0x000C: 4D 00 6F 00  72 00 6D 00  61 00 6C 00
.
.
.
0xFFF4: 0C 04 46 00  48 00 65 00  61 00 64 00
```

On the left are the addresses, on the right is the data stored at that address. The bits are converted back into a number, character, or other data type when they are accessed and used. How this occurs depends on the programming language.

Challenge:

You have intercepted the following message being sent. You realise that it is likely to be text and you are going to attempt to translate it back to ASCII (plain, readable text). First convert each hexadecimal number to decimal (base 10), then find which ASCII letter it maps to.



Char	Dec	Char	Dec	Char	Dec
(sp)	32	@	64	`	96
!	33	A	65	a	97
"	34	B	66	b	98
#	35	C	67	c	99
$	36	D	68	d	100
%	37	E	69	e	101
&	38	F	70	f	102
'	39	G	71	g	103
(	40	H	72	h	104
)	41	I	73	i	105
*	42	J	74	j	106
+	43	K	75	k	107
,	44	L	76	l	108
-	45	M	77	m	109
.	46	N	78	n	110
/	47	O	79	o	111
0	48	P	80	p	112
1	49	Q	81	q	113
2	50	R	82	r	114
3	51	S	83	s	115
4	52	T	84	t	116
5	53	U	85	u	117
6	54	V	86	v	118
7	55	W	87	w	119
8	56	X	88	x	120
9	57	Y	89	y	121
:	58	Z	90	z	122
;	59	[	91	{	123
<	60	\	92	|	124
=	61	]	93	}	125
>	62	^	94	~	126
?	63	_	95		

Figure 2: A list of ASCII characters and the decimal values they are mapped to

```
In hex:
0x0000: 49 27 6d 20  61 6c 72 65  61 64 79 20  54 72 61 63 65.
```

Hint: You may save time by writing a script in Ruby or JavaScript. This can be done by first making a function responsible for taking a single hex number and returning it as a decimal, then improving the function to convert the hex number straight to a character.

References:

Source: https://www.youtube.com/watch?v=0sZ74NSiJkQ Discrete Maths - Bases

Answers

1.1

Base	8		
BasePower	82	81	80
=	64	8	1
Numeral	1	7	3

1.2

Base	2				
BasePower	24	23	22	21	20
=	16	8	4	2	1
Numeral	1	0	1	1	0

Leibniz had already invented a binary number system before he was introduced to the I Ching by Joachim Bouvet. However he was shocked and attributed the original discovery of binary representations to ancient Chinese philosophy

---