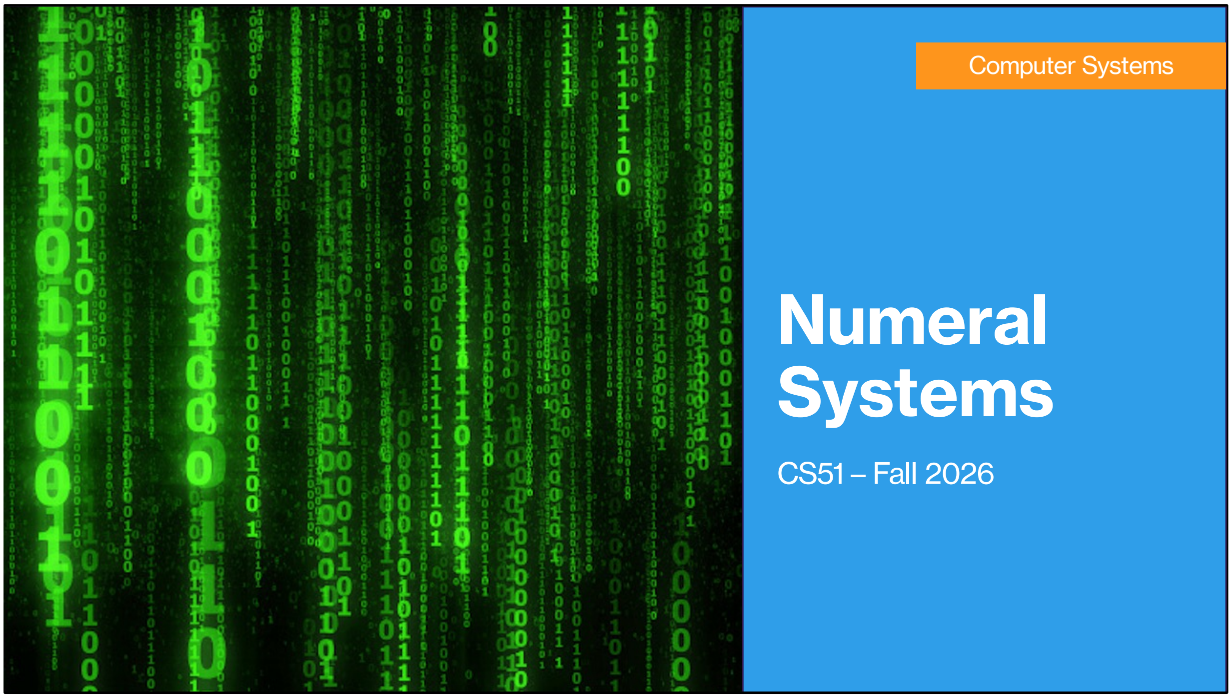

Last time we talked about how physical components like transistors allow us to represent data in computers in one of two binary states, 0 or 1, False or True etc. Working with binary data is just one example of a numeral system which we will tackle today. But first, I want us to see how to work with Boolean variables in Python.

Boolean Algebra in Python

2

Boolean variables in Python

- In Python, Boolean variables are represented through the `bool` data type. A bool variable can either be `True` or `False`. These values can also be seen as equal to 1 and 0, respectively.
- For example, `raining = False` declares a variable called `raining` and assigns to it the value `False.`
- The comparison operators ==, !=, >, <, >=, <= return bools. For example,
  - `10 < 0` evaluates to `False`
  - `11 >= 11` evaluates to `True`
  - `11 > 11.0` evaluates to `False`
  - `10 == 10.0` evaluates to `True`
  - `10 != 10.0` evaluates to `False`

3

So far, we have been talking about 0s and 1s either as Boolean variables and have learned we can combine them with logical operators (e.g., AND, OR, XOR, NOT) in Boolean Algebra. We also saw that there is an equivalent representation in circuits, which we will explore in more depth in subsequent class meetings.. In Python, you may have already worked with Boolean variables through the bool data type. A bool variable can be True or False, equivalently to 1 and 0. If you have worked with comparison operators, you have also evaluated Boolean expressions.

Boolean Algebra in Python

- We can combine bools and logical operators in Boolean expressions, for example, to evaluate them in **conditionals** (`if` statements) and **loops** (`for` and `while`).
- Python supports three logical operators:
  - `not` for negation (i.e., $\neg$)
  - `and` for conjunction (i.e., $\wedge$)
  - `or` for disjunction (i.e., $\vee$)

4

We can combine bool variables and logical operators in Boolean expressions, for example to evaluate them in conditionals (if statements) and loops, such as the for and while loops. Python also supports not, and, and or.

Short-circuit evaluation in Python

- Python supports **short-circuiting**, a technique that evaluates a logical expression from left to right and *stops evaluation* once the truth can be determined.
- For example, if x=3 and we have the expression:
  - `x>3 and x<24`
  - Once Python evaluates `x>3` being `False` and sees that and follows, then it can immediately evaluate the whole expression to `False`, without needing to evaluate whether x<24.

5

Interestingly, Python supports short-circuiting, a technique that evaluates a logical expression from left to right and stops once the truth can be determined without wasting unnecessarily time by evaluating the rest of the expression. What do you think are the benefits of such approach? that's right, we can optimize our code and not waste unnecessarily time. Here is an example of how short-circuiting works when and is included in the expression. Say we have a variable x set to 3. If we have the expression x>3 and x<24, Python will stop evaluating right once it sees that the False expression x>3 is followed by and because anything that has at least one False, evaluates to False.

Think

- Assume `x=3` again.
- *Can you come up with an expression that includes an* `or` *operator and which would lead to short-circuiting?*

6

Your turn now to create an expression that this time includes an or statement and which would short-circuit.

Think

- Assume `x=3` again.
- *Can you come up with an expression that includes an `or` operator and which would lead to short-circuiting?*
  - `x==3 or x>24`

7

For example x==3 would result to True. Once Python sees that True is followed by an or, it doesn't need to evaluate x>24. It can immediately declare that True or ANYTHING will evaluate to True.

Short-circuit evaluation and optimization

- Because of short-circuit evaluation, we can avoid deeply nested if statements to handle exceptional cases. For example,
  - `if x == 0 or (x-1)/x > 0.5: …`
  - When x is equal to 0, evaluating the second operand of the or statement would cause a divide-by-zero error. But the second disjunct isn't evaluated when x is equal to 0 because the first operand was True!

8

Because of short-circuit evaluation, we can avoid deeply nested if statements to handle exceptional cases. For example, when x is equal to 0, evaluating the second operand of the or statement would cause a divide-by-zero error. But the second disjunct isn't evaluated when x is equal to 0 because the first operand was True!

Short-circuit evaluation and optimization

- We can take advantage of short-circuit evaluation to make our code a bit faster. For example, when writing an if statement that contains the logical operator *and*, you should first write what is simpler to calculate or often false before something that is complex to calculate or often true:
  - `if (simpleOrOftenFalse() and complexOrOftenTrue()):…`

9

Short-circuiting allows us to write more condensed if statements or even make our code faster.

PRACTICE TIME – Boolean expressions

- Assume the variable x currently stores the value 47. What will the following expressions evaluate to?
  - `True and not False`
  - `not True or not False`
  - `True and not 0`
  - `False or not 1`
  - `2 < x and x < 50`
  - `x > 2 and < 20`
  - `(2 > x) or (x == 47)`
  - `not (x == x)`
  - `not (x != 3)`
  - `x < 0 and (y/x) == 3`
  - `x > 0 or (y/x) == 3`

Let’s see whether you can evaluate the following Boolean expressions

ANSWER – Boolean expressions

- Assume the variable x currently stores the value 47. What will the following expressions evaluate to?
  - `True and not False - True`
  - `not True or not False - True`
  - `True and not 0 – True`
  - `False or not 1 - False`
  - `2 < x and x < 50 - True`
  - `x > 2 and < 20 – Invalid syntax`
  - `(2 > x) or (x == 47) - True`
  - `not (x == x) - False`
  - `not (x != 3) – False`
  - `x < 0 and (y/x) == 3 – False`
  - `x > 0 or (y/x) == 3 - True`

11

0 corresponds to False and 1 to True! For the last two, because of shortcircuiting, Python will never find out that y was not declared. It will see x<0 evaluate it to False and since then that is connected with AND it knows that the entire statement has to be False. Similarly, x>0 is True and since it's connected with an OR, it doesn't matter, the whole thing will be evaluated to True.

Numeral Systems

12

Decimal numbers

- We are used to numbers that use ten digits: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.
- This arithmetic is called **base $\mathbf{10}$** (denoted as a subscript) and such numbers are called **decimal**.
- To avoid confusion, we use subscripts to indicate a base of a number.
  - For example, instead 9247 we will write the number 9247_{10}.

13

Although computer scientists love binary data, we are not used to them in everyday life. Instead, we are accustomed to talking about numbers that use ten digits, 0,…9, which we combine to form longer numbers. This type of arithmetic is called base 10 or decimal (deci means one tenth). To avoid confusion, we will use subscripts to indicate the base of a number, e.g., 9247_{10}

Think

- *What number is 9247_{10}?*

14

But if you think about it, what number is 9247?

Think

- *What number is 9247_{10}?*
  - $9247_{10} = 9000 + 200 + 40 + 7 =$
  - $= 9\times10^3 + 2\times10^2 + 4\times10^1 + 7\times10^0$

15

In school, you learned that you can break it down as $9000 + 200 + 40 + 7$ which if we think in terms of base 10 can be written as $9\times10^3 + 2\times10^2 + 4\times10^1 + 7\times10^0$

Decimal numbers

- The n digit number $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer

$$d_{n-1}10^{n-1} + \ldots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$$

- For example, the number 9247_{10} can be written as the sum of each of its digits multiplied by the weight of the corresponding column:

$10^3 = 1000$	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$
9	2	4	7

  - $9247_{10} = 9\times10^3 + 2\times10^2 + 4\times10^1 + 7\times10^0$

16

More formally, if we take a decimal number with n digits we can break it down as the sum of individual digits that are multiplied with increasingly larger powers of 10. Starting with the right-most digit, that would be multiplied with 1 or 10^0. The second to last would be multiplied with 10 or 10^1. Third from the right, would be multiplied with 100 or 10^2 and so on. Notice that each column is 10 times larger than the one on the right.

Think

- The n digit number $d_{n-1} \dots d_2\, d_1 d_0$ represents the integer

$$d_{n-1}10^{n-1} + \dots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$$

- *If we had 3 digits, how many decimal numbers could we represent?*

17

If we know that with n digits, $d_{n-1} \dots d_2\, d_1 d_0$ represents the integer $d_{n-1}10^{n-1} + \dots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$ and we have 3 digits, how many decimal numbers can we represent?

Think

- The n digit number $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer

$$d_{n-1}10^{n-1} + \ldots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$$

- *If we had 3 digits, how many decimal numbers could we represent?*
  - 1000 numbers: 0, 1, 2, 3,...,999

18

We could represent 1000 numbers: 0, 1, 2, 3, all the way to 999

Think

- The n digit number $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer

$$d_{n-1}10^{n-1} + \ldots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$$

- *If we had 3 digits, how many decimal numbers could we represent?*
  - 1000 numbers: 0, 1, 2, 3,…,999
- *If we had n digits, how many decimal numbers could we represent?*

What if we had n digits?

Think

- The n digit number $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer

$$d_{n-1}10^{n-1} + \ldots + d_2\, 10^2 + d_1 10^1 + d_0 10^0$$

- *If we had 3 digits, how many decimal numbers could we represent?*
  - 1000 numbers: 0, 1, 2, 3,...,999
- *If we had n digits, how many decimal numbers could we represent?*
  - 10^n numbers: $0, 1, 2, 3, \ldots, 10^n - 1$

20

It would be 10^n, in the range of 0, 1, 2, …, 10^n -1, if we assume only positive numbers

Think

- *What digits can we use if we are working on base 5?*

21

Let's try to extrapolate a bit from the decimal system. In the decimal system, we worked with 10 digits: 0, 1, 2, …, 9. If we were to apply the same thinking, which are the digits we would work with in base 5?

Think

- *What digits can we use if we are working on base 5?*
  - 0, 1, 2, 3, 4

22

We would have five digits, 0, 1, 2, 3, 4.

Think

- *What number is* 243_5*?*

23

I will now show you a base-5 number. Thinking similarly to what we did with base 10, what number is 243_5?

Think

- *What number is* 243_5*?*

$5^2 = 25$	$5^1 = 5$	$5^0 = 1$
2	4	3

  - $243_5 = 2\times5^2 + 4\times5^1 + 3\times5^0$

24

You can start by creating a table for powers of 5. Since we have three digits, we would go from 5^2 down to 5^0. And we can break now 243_5 as $2\times5^2 + 4\times5^1 + 3\times5^0$

Think

- *What number is* 243_5?

$5^2 = 25$	$5^1 = 5$	$5^0 = 1$
2	4	3

- $243_5 = 2\times5^2 + 4\times5^1 + 3\times5^0 = 2\times25 + 4\times5 + 3 = 73_{10}$

25

But this is equal to $2\times25 + 4\times5 + 3 = 73_{10}$

Base 5 numbers

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}5^{n-1} + \ldots + d_2\, 5^2 + d_1 5^1 + d_0 5^0$

26

Similarly to how we worked with base 10 numbers, we can represent a base 5 number of n digits.

Think

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}5^{n-1} + \ldots + d_2\, 5^2 + d_1 5^1 + d_0 5^0$
- *If we had 3 digits, how many base 5 numbers could we represent?*

27

Had we had 3 digits, how many base 5 numbers could we represent

Think

- $d_{n-1} \dots d_2\, d_1 d_0$ represents the integer $d_{n-1}5^{n-1} + \dots + d_2\, 5^2 + d_1 5^1 + d_0 5^0$
- *If we had 3 digits, how many base 5 numbers could we represent?*
  - 125 numbers: $0, 1, 2, 3, \dots, 5^3 - 1 = 0, 1, 2, 3, \dots, 124$

28

Assuming positive numbers only, the range would be 0, 12, 3, 5^3-1, that is 125 numbers

Think

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}5^{n-1} + \ldots + d_2\, 5^2 + d_1 5^1 + d_0 5^0$
- *If we had 3 digits, how many base 5 numbers could we represent?*
  - 125 numbers: $0, 1, 2, 3, \ldots, 5^3 - 1 = 0, 1, 2, 3, \ldots, 124$
- *If we had n digits, how many base 5 numbers could we represent?*

29

What if we had n digits?

Think

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}5^{n-1} + \ldots + d_2\, 5^2 + d_1 5^1 + d_0 5^0$
- *If we had 3 digits, how many base 5 numbers could we represent?*
  - 125 numbers: $0, 1, 2, 3, \ldots, 5^3 - 1 = 0, 1, 2, 3, \ldots, 124$
- *If we had n digits, how many base 5 numbers could we represent?*
  - 5^n numbers: $0, 1, 2, 3, \ldots, 5^{\mathrm{n}} - 1$

30

5^n numbers

Numbers in other bases

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1} b^{n-1} + \ldots + d_2\, b^2 + d_1 b^1 + d_0 b^0$

Again, we can extrapolate this to any arbitrary base b

Think

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}b^{n-1} + \ldots + d_2\, b^2 + d_1 b^1 + d_0 b^0$
- *If we had n digits, how many base b numbers could we represent?*

32

So how many base b numbers could we represent with n digits?

Think

- $d_{n-1} \ldots d_2\, d_1 d_0$ represents the integer $d_{n-1}b^{n-1} + \ldots + d_2\, b^2 + d_1 b^1 + d_0 b^0$
- *If we had n digits, how many base b numbers could we represent?*
  - b^n numbers: $0, 1, 2, 3, \ldots, \mathrm{b^n} - 1$

b^n

Think

- *What are the valid digits for base 2 numbers?*

34

Ok let's get more concrete again. If we work with base 2 numbers, what are the valid digits we have at our disposal?

Think

- *What are the valid digits for base 2 numbers?*
  - *0, 1*

35

Did you get 0 and 1?

Think

- *What number is* 1102_2*?*

36

I will now show you a base-2 number. Thinking similarly to what we did with base 10 and 5, what number is 1102_2?

Think

- *What number is* 1102_2*?*

$2^3 = 8$	$2^2 = 4$	$2^1 = 10$	$2^0 = 1$
1	1	0	1

Let's start by populating the table with powers of 2.

Think

- *What number is* 1102_2*?*

$2^3 = 8$	$2^2 = 4$	$2^1 = 10$	$2^0 = 1$
1	1	0	1

- $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

Think

- *What number is* 1102_2*?*

$2^3 = 8$	$2^2 = 4$	$2^1 = 10$	$2^0 = 1$
1	1	0	1

- $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$
- $= 8 + 4 + 0 + 1 = 13_{10}$

39

which translates to 13 in decimal.

Least and most significant bits

- The left-most bit is the **most significant bit** (MSB).
- The right-most is the **least significant bit** (LSB).

You will occasionally hear me talk about least and most significant bits. The leftmost bit is the most significant (think about it, it's the one that contributes the biggest weight). The right most is the least significant.

Think

- $b_n b_{n-1} \ldots b_2\, b_1 b_0$ represents the integer $b_n 2^n + b_{n-1} 2^{n-1} + \ldots + b_2\, 2^2 + b_1 2^1 + b_0 2^0$
- *If we had n digits, how many binary numbers could we represent?*

Going back to the activity we have repeated before, how many binary numbers can we represent with n digits>

Think

- $b_n b_{n-1} \ldots b_2\, b_1 b_0$ represents the integer $b_n 2^n + b_{n-1} 2^{n-1} + \ldots + b_2\, 2^2 + b_1 2^1 + b_0 2^0$
- *If we had n digits, how many binary numbers could we represent?*
  - 2^n numbers: $0, 1, 2, 3, \ldots, 2^{\mathrm{n}} - 1$

42

2^n

Binary numbers and their decimal equivalents

1-bit binary	2-bit binary	3-bit binary	4-bit binary	Decimal equivalent
0	00	000	0000	0
1	01	001	0001	1
	10	010	0010	2
	11	011	0011	3
		100	0100	4
		101	0101	5
		110	0110	6
		111	0111	7
			1000	8
			1001	9
			1010	10
			1011	11
			1100	12
			1101	13
			1110	14
			1111	15

43

Here is what that would look like if we take binary numbers with 1, 2, 3, or 4 bits. You can also see the decimal equivalent. With 1 bit, we can represent 2 values, with 2 bits 4, with 3 bits 8 and 4 bits 16, and so on.

PRACTICE TIME

- Convert the following binary numbers to their decimal representation:
  - 1010_2
  - 1111_2
  - 10011011_2
  - 1100011_2
  - 100010_2

44

Let's practice, by converting the following binary numbers to their decimal representation.

PRACTICE TIME - ANSWER

- Convert the following binary numbers to their decimal representation:
  - $1010_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 8 + 2 = 10_{10}$
  - $1111_2 = 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 = 8 + 4 + 2 + 1 = 15_{10}$
  - $10011011_2 = 1\times2^7 + 0\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 128 + 16 + 8 + 2 + 1 = 155_{10}$
  - $1100011_2 = 1\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 99_{10}$
  - $100010_2 = 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 32 + 2 = 34_{10}$

45

Did you get this?

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.

Divide by 2	Remainder	
84		LSB
		MSB

MSB						LSB

How do we go from decimal numbers to binary? We repeatedly divide the number by 2 and the remainder goes to each column from right to left. Let' see how we can convert the decimal number 84 to binary. We start by writing 84 and we divide it by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.

Divide by 2	**Remainder**	
84	0	LSB
42		
		MSB

						1
						0

MSB LSB

47

84/2 = 42 with a remainder of 0. So 0 goes in the 1s column. We next divide 42 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.

Divide by 2	Remainder	
84	0	LSB
42	0	
21		
		MSB

					2	1
					0	0
MSB						LSB

48

42/2 – 21 with a remainder of 0. So 0 goes to the 2s column. We next divide 21 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s.

Divide by 2	Remainder	
84	0	LSB
42	0	
21	1	
		MSB

				4	2	1
				1	0	0
MSB						LSB

49

21/2 = 10 with a remainder of 1. So 1 goes to the 4s column. We next divide 10 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s.
  - $\frac{10}{2} = 5$ with a remainder of 0, so there is a 0 in the 8's column.

Divide by 2	Remainder	
84	0	LSB
42	0	
21	1	
10	0	
5		
		MSB

			8	4	2	1
			0	1	0	0

MSB LSB

50

10/2 = 5 with a remainder of 0 so 0 goes to the 8s column and we next divide 5 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1’s column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2’s column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s.
  - $\frac{10}{2} = 5$ with a remainder of 0, so there is a 0 in the 8's column.
  - $\frac{5}{2} = 2$ with a remainder of 1, so there is a 1 in 16’s.

Divide by 2	Remainder	
84	0	LSB
42	0	
21	1	
10	0	
5	1	
2		
		MSB

		16	8	4	2	1
		1	0	1	0	0

MSB LSB

51

5/2=2 with a remainder of 1. So 1 goes to the 16s column. We next divide 2 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s.
  - $\frac{10}{2} = 5$ with a remainder of 0, so there is a 0 in the 8's column.
  - $\frac{5}{2} = 2$ with a remainder of 1, so there is a 1 in 16's.
  - $\frac{2}{2} = 1$, with a remainder of 0, so 0 goes in the 32's.

Divide by 2	Remainder	
84	0	LSB
42	0	
21	1	
10	0	
5	1	
2	0	
1		MSB

	32	16	8	4	2	1
	0	1	0	1	0	0

MSB LSB

52

2/2 = 1, with a remainder of 0. So 0 goes in the 32s column. We next divide 1 by 2.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s.
  - $\frac{10}{2} = 5$ with a remainder of 0, so there is a 0 in the 8's column.
  - $\frac{5}{2} = 2$ with a remainder of 1, so there is a 1 in 16's.
  - $\frac{2}{2} = 1$, with a remainder of 0, so 0 goes in the 32's.
  - $\frac{1}{2} = 0$, with a remainder of 1, so 1 goes in the 64's column.

Divide by 2	**Remainder**	
84	0	LSB
42	0	
21	1	
10	0	
5	1	
2	0	
1	1	MSB

64	32	16	8	4	2	1
1	0	1	0	1	0	0

MSB LSB

53

½=0 with a remainder of 1, so 1 goes in the 64s column.

Decimal to binary conversion

- Repeatedly divide the number by 2. The remainder goes to each column, from right to left.
- Example: Convert the decimal number 84_{10} to binary.
  - $\frac{84}{2} = 42$ with a remainder of 0, so 0 goes in the 1's column.
  - $\frac{42}{2} = 21$ with a remainder of 0, so 0 goes in the 2's column.
  - $\frac{21}{2} = 10$ with a remainder of 1, so there is a 1 going to the 4s column.
  - $\frac{10}{2} = 5$ with a remainder of 0, so there is a 0 in the 8's column.
  - $\frac{5}{2} = 2$ with a remainder of 1, so there is a 1 going to the 16's column.
  - $\frac{2}{2} = 1$, with a remainder of 0, so 0 goes in the 32's column.
  - $\frac{1}{2} = 0$, with a remainder of 1, so 1 goes in the 64's column.
- Putting this all together, $84_{10} = 1010100_2$.

Divide by 2	Remainder	
84	0	LSB
42	0	
21	1	
10	0	
5	1	
2	0	
1	1	MSB

64	32	16	8	4	2	1
1	0	1	0	1	0	0

MSB LSB

54

Putting this all together, $84_{10} = 1010100_2$.

PRACTICE TIME

- Convert the following decimal numbers to their binary representation.
  - 125_{10}
  - 339_{10}
  - 75_{10}

55

Practice converting the following decimal numbers to binary.

PRACTICE TIME - ANSWER

- Convert the following decimal numbers to their binary representation.
  - $125_{10} = 1111101_2$ $339_{10} = 101010011_2$ $75_{10} = 1001011_2$

Divide by 2	Remainder	
125	1	LSB
62	0	
31	1	
15	1	
7	1	
3	1	
1	1	MSB

Divide by 2	Remainder	
339	1	LSB
169	1	
84	0	
42	0	
21	1	
10	0	
5	1	
2	0	
1	1	MSB

Divide by 2	Remainder	
75	1	LSB
37	1	
18	0	
9	1	
4	0	
2	0	
1	1	MSB

56

This is what you should have gotten.

Hexadecimal numbers

- We also use a **hexadecimal** (or hex) notation where the arithmetic is base 16.
- *What are the valid digits for hexadecimal numbers?*

57

Lastly, we will see one more numeral system called hexadecimal or hex for short which is base 16.

Think

- We also use a **hexadecimal** (or hex) notation where the arithmetic is base 16.
- *What are the valid digits for hexadecimal numbers?*

58

What are the valid digits for hex numbers?

Think

- We also use a **hexadecimal** (or hex) notation where the arithmetic is base 16.
- *What are the valid digits for hexadecimal numbers?*
  - One idea: 0, 1, 2, 3, ..., 15
  - *What's the problem here?*

59

Maybe you thought 0, …, 15. But what's the problem here? That's right, how would we distinguish between say 14 and 1 and 4 next to each other?

Hexadecimal numbers

- We also use a **hexadecimal** (or hex) notation where the arithmetic is base 16.
- *What are the valid digits for hexadecimal numbers?*
  - 0, 1, 2, ..., 9, **A, B, C, D, E, F**
  - *Instead of **10, 11, 12, 13, 14, 15***

60

Instead, we will do something different. We will use the 10 unique digits 0, …, 9, and then we will also use the first six letters of the English alphabet. We could have really used any other six unique characters but working with letters is convenient.

Think

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

- *What number is* $2A7_{16}$?

61

OK, let's think, what's the hex number 2A7?

Think

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

- *What number is* $2A7_{16}$?

$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
2	A	7

62

We will fill out our table with base 16 now

Think

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

- *What number is* $2A7_{16}$?

$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
2	A	7

- $2\text{A}7_{16} = 2\times16^2 + 10\times16^1 + 7\times16^0$

63

So $2\text{A}7_{16} = 2\times16^2 + 10\times16^1 + 7\times16^0$

Think

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

- *What number is* $2A7_{16}$?

$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
2	A	7

- $2A7_{16} = 2\times16^2 + 10\times16^1 + 7\times16^0 =$
- $= 2\times256 + 10\times16 + 7 = 679_{10}$

64

which is equal to $2\times256 + 10\times16 + 7 = 679_{10}$

Think

- $h_{n-1} \dots h_2\, h_1 h_0$ represents the integer
$$h_{n-1}16^{n-1} + \cdots + h_2 16^2 + h_1 16^1 + h_0 16^0$$
- *If we had 3 digits, how many hex numbers could we represent?*

Had we had 3 digits, how many hex numbers could we represent

Think

- $h_{n-1} \dots h_2\, h_1 h_0$ represents the integer

$$h_{n-1}16^{n-1} + \cdots + h_2 16^2 + h_1 16^1 + h_0 16^0$$

- *If we had 3 digits, how many hex numbers could we represent?*
  - 4096 numbers: $0, 1, 2, 3, \dots, 16^3 - 1 = 0, 1, 2, 3, \dots, 4095$

66

Assuming positive numbers only, the range would be 0, 12, 3, 16^3-1, that is 4096 numbers

Think

- $h_{n-1} \ldots h_2\, h_1 h_0$ represents the integer
$$h_{n-1}16^{n-1} + \cdots + h_2 16^2 + h_1 16^1 + h_0 16^0$$
- *If we had 3 digits, how many hex numbers could we represent?*
  - 4096 numbers: $0, 1, 2, 3, \ldots, 16^3 - 1 = 0, 1, 2, 3, \ldots, 4095$
- *If we had n digits, how many hex numbers could we represent?*

67

What if we had n digits?

Think

- $h_{n-1} \dots h_2\, h_1 h_0$ represents the integer
$$h_{n-1}16^{n-1} + \cdots + h_2 16^2 + h_1 16^1 + h_0 16^0$$
- *If we had 3 digits, how many hex numbers could we represent?*
  - 4096 numbers: $0, 1, 2, 3, \dots, 16^3 - 1 = 0, 1, 2, 3, \dots, 4095$
- *If we had n digits, how many hex numbers could we represent?*
  - 16^n numbers: $0, 1, 2, 3, \dots, 16^{\mathrm{n}} - 1$

68

I hope it's obvious we would have 16^n numbers.

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

Hexadecimal numbers

- $h_n h_{n-1} \ldots h_2\, h_1 h_0$ represents the integer $h_n 16^n + h_{n-1} 16^{n-1} + \ldots + h_2\, 16^2 + h_1 16^1 + h_0 16^0$
- For example, the hex number $FACE_{16}$ can be written as:

$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$
15	10	12	14

  - $15 \times 16^3 + 10 \times 16^2 + 12 \times 16^1 + 14 \times 16^0$
  - This is equal to the decimal representation $61440 + 2560 + 192 + 14 = 64206_{10}$
- Hex numbers are sometimes preceded by 0x to be distinguished from decimal numbers, e.g., 0x15 is 15_{16} which is equal to 21_{10} and not 15_{10}.

69

Writing long binary numbers can be both tedious and prone to errors. For this reason, computer scientists also like working in base 16 or with hexadecimal/hex notation (we saw that a group of 4 bits can represent 2^4=16 possibilities). Hexadecimal numbers use the digits 0 to 9 along with the letters A to F which are used for 10 to 15, respectively.

Equivalence across number systems

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

70

This table shows the first 16 numbers as represented in the three systems we have seen so far.

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

PRACTICE TIME

- Convert the following hexadecimal numbers to their decimal representation.
  - $32A_{16}$
  - $D1CE_{16}$
  - $1B7F_{16}$
  - 47_{16}

Let's practice by converting the hexadecimal numbers to decimal.

ANSWER

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

- Convert the following hexadecimal numbers to their decimal representation.
  - $32A_{16} = 3\times16^2 + 2\times16^1 + 10\times16^0 = 810_{10}$
  - $D1CE_{16} = 13\times16^3 + 1\times16^2 + 12\times16^1 + 14\times16^0 = 53710_{10}$
  - $1B7F_{16} = 1\times16^3 + 11\times16^2 + 7\times16^1 + 15\times16^0 = 7039_{10}$
  - $47_{16} = 4\times16^1 + 7\times16^0 = 71_{10}$

72

Is this what you got?

Decimal to hexadecimal conversion

- Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
- Example: Convert the decimal number 333_{10} to hexadecimal.

Divide by 16	**Remainder**
333	

LSB

MSB

73

Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
Let's see an example of converting the decimal number 333 to hex. We start by dividing it by 16.

Decimal to hexadecimal conversion

- Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
- Example: Convert the decimal number 333_{10} to hexadecimal.
  - $\frac{333}{16} = 20$ with a remainder of $13_{10} = D_{16}$ which goes in the 1's column.

Divide by 16	Remainder	
333	$13_{10} = D_{16}$	LSB
20		
		MSB

		1
		D

74

333/16=20 with a remainder of 13 (in decimal) or D in hex. So we put D in the 1s column. We next divide 20 by 16.

Decimal to hexadecimal conversion

- Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
- Example: Convert the decimal number 333_{10} to hexadecimal.
  - $\frac{333}{16} = 20$ with a remainder of $13_{10} = D_{16}$ which goes in the 1's column.
  - $\frac{20}{16} = 1$ with a remainder of 4, so 4 goes in the 16's column.

Divide by 16	Remainder	
333	$13_{10} = D_{16}$	LSB
20	$4_{10} = 4_{16}$	
1		MSB

	16	1
	4	D

75

20/16= 1 with a remainder of 4 (same in decimal and hex), so 4 goes in the 16's column. We next divide 1 by 16.

Decimal to hexadecimal conversion

- Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
- Example: Convert the decimal number 333_{10} to hexadecimal.
  - $\frac{333}{16} = 20$ with a remainder of $13_{10} = D_{16}$ which goes in the 1's column.
  - $\frac{20}{16} = 1$ with a remainder of 4, so 4 goes in the 16's column.
  - $\frac{1}{16} = 0$ with a remainder of 1, so there is a 1 going to the 256s column.

Divide by 16	Remainder	
333	$13_{10} = D_{16}$	LSB
20	$4_{10} = 4_{16}$	
1	$1_{10} = 1_{16}$	MSB

256	16	1
1	4	D

76

$\frac{1}{16} = 0$ with a remainder of 1 (same in decimal and hex), so there is a 1 going to the 256s column.

Decimal to hexadecimal conversion

- Similarly to converting from decimal to binary, we will now repeatedly divide by 16 and the remainder is converted to hexadecimal and goes to the corresponding column from right to left.
- Example: Convert the decimal number 333_{10} to hexadecimal.
  - $\frac{333}{16} = 20$ with a remainder of $13_{10} = D_{16}$ which goes in the 1's column.
  - $\frac{20}{16} = 1$ with a remainder of 4, so 4 goes in the 16's column.
  - $\frac{1}{16} = 0$ with a remainder of 1, so there is a 1 going to the 256s column.
  - Putting this all together, $333_{10} = 14D_{16}$.

Divide by 16	Remainder	
333	$13_{10} = D_{16}$	LSB
20	$4_{10} = 4_{16}$	
1	$1_{10} = 1_{16}$	MSB

256	16	1
1	4	D

77

Putting this all together, $333_{10} = 14D_{16}$.

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

PRACTICE TIME

- Convert the following decimal numbers to their hexadecimal representation.
  - 2546_{10}
  - 1457_{10}
  - 269_{10}
  - 47_{10}

78

Let's practice by converting the following decimal numbers to their hex representation.

A, B, C, D, E, F
10, 11, 12, 13, 14, 15

ANSWER

- Convert the following decimal numbers to their hexadecimal representation.
  - $2546_{10} = 9F2$ $1457_{10} = 5B1$ $269_{10} = 10D$ $47_{10} = 2F$

Divide by 16	Remainder
2546	$2_{10} = 2_{16}$
159	$15_{10} = F_{16}$
9	$9_{10} = 9_{16}$

Divide by 16	Remainder
1457	$1_{10} = 1_{16}$
91	$11_{10} = B_{16}$
5	$5_{10} = 5_{16}$

Divide by 16	Remainder
269	$13_{10} = D_{16}$
16	$0_{10} = 0_{16}$
1	$1_{10} = 1_{16}$

Divide by 16	Remainder
47	$15_{10} = F_{16}$
2	$2_{10} = 2_{16}$

LSB

MSB

79

Is this what you got?

Binary to hexadecimal conversion

110011100010000_2

break into 4 bit groups

$0110\ 0111\ 0001\ 0000_2$

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

To convert from binary to hex, we read the binary digits in groups of four

Binary to hexadecimal conversion

110011100010000_2

$0110 \ 0111 \ 0001 \ 0000_2$

break into 4 bit groups

convert each 4 bit group into a digit

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

81

and convert each group

Binary to hexadecimal conversion

110011100010000_2

break into 4 bit groups

$0110\ 0111\ 0001\ 0000_2$

convert each 4 bit group into a digit

$6\ 7\ 1\ 0_{16}$

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

82

group to its corresponding hexadecimal number. If needed, prepend leading 0s to make the number of bits a multiple of 4.

PRACTICE TIME

- Convert the following binary numbers to their hexadecimal representation.
  - 10010011_2
  - 101100110101_2
  - 11111011101110010_2

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Can you give it a try?

ANSWER

- Convert the following binary numbers to their hexadecimal representation.
  - $1001\ 0011_2 = 93_{16}$
  - $1011\ 0011\ 0101_2 = B35_{16}$
  - $11111011101110010_2 = 0001\ 1111\ 0111\ 0111\ 0010 = 1F772_{16}$

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Hexadecimal to binary conversion

- We replace each hex digit with the four binary bits that correspond to its value.
- Example: Convert the hexadecimal number $CAFE_{16}$ to its binary representation.
  - $C_{16} = 1100_2$
  - $A_{16} = 1010_2$
  - $F_{16} = 1111_2$
  - $E_{16} = 1110_2$
  - Putting this all together, $CAFE_{16} = 1100101011111110_2$.

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

To go from hex to binary, we replace each hex digit with the four binary bits that correspond to its value.

PRACTICE TIME

- Convert the following hexadecimal numbers to their binary representation.
  - $\mathrm{F00D}_{16}$
  - $2B3BAD_{16}$
  - $\mathrm{DEADC0DE}_{16}$

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

86

Let's try it

ANSWER

- Convert the following hexadecimal numbers to their binary representation.
  - $\mathrm{F00D}_{16} = 1111000000001101_2$
  - $2B3BAD_{16} = 001010110011101110101101_2$
  - $\mathrm{DEADC0DE}_{16} = 11011110101011011100000011011110_2$

Hexadecimal digit	Binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

87

Fun, huh? And that's it for today folks. Time to practice on your own at home!

Practice Problems

88

Practice Problems – Problem 1

Assuming x = 6, y = 2, z = 12. What would the following Python expressions evaluate to?

- x == 10 or x < 5
- x != 7 and x > 3
- not (x >= 5 and y <= 2)
- (x < 0) or (x >= 0)
- not (x != x)
- x > 1 and (z / x) != 2
- x <= 0 or (z / x) == 1
- x == 4 and y == 4

Practice Problems – Problem 2

- Rewrite the following code to make it cleaner and avoid the use of nested if statements:

```
def get_discount(member, purchase_amount):
	if member == True:
		if purchase_amount > 100:
			return "Discount applied"
		else:
			return "No discount"
	else:
		if purchase_amount > 200:
			return "Discount applied"
		else:
			return "No discount"

print(get_discount(True, 120)) # Expected: "Discount applied"
print(get_discount(True, 80)) # Expected: "No discount"
print(get_discount(False, 220)) # Expected: "Discount applied"
print(get_discount(False, 50)) # Expected: "No discount"
```


Can you write the following function in a more condensed way? I have provided you the following print statements and what you should expect them to produce

Practice Problems – Problem 3

- For each row, fill in the missing entries by converting the given number from its representation to every other equivalent representation.

Binary	Decimal	Hexadecimal
	39_{10}	
100111010_2		
		$BADE_{16}$
	47_{10}	
101100111_2		
		$C0FFEE_{16}$

Practice Problem 1 – ANSWER

Assuming x = 6, y = 2, z = 12. What would the following Python expressions evaluate to?

- x == 10 or x < 5 → 6 == 10 or 6 < 5 → False or False → **False**
- x != 7 and x > 3 → 6 != 7 and 6 > 3 → True and True → **True**
- not (x >= 5 and y <= 2) → not (6 >= 5 and 2 <= 2) → not (True and True) → not True → **False**
- (x < 0) or (x >= 0) → 6 < 0 or 6 >= 0 → False or True → **True** (always true, no matter what x is)
- not (x != x) → not False → **True**
- x > 1 and (z / x) != 2 → 6 > 1 and (12 / 6 != 2) → True and False → **False**
- x <= 0 or (z / x) == 1 → 6 <= 0 or (12 / 6 == 1) → False or False → **False**
- x == 4 and y == 4 → 6 == 4 and 2 == 4 → False and False → **False**

Practice Problem 2 - ANSWER

- Rewrite the following code to make it cleaner and avoid the use of nested if statements:

```
def get_discount(member, purchase_amount):
	if (purchase_amount > 200) or (member and purchase_amount > 100):
		return "Discount applied"
	else:
		return "No discount"

print(get_discount(True, 120)) # Expected: "Discount applied"
print(get_discount(True, 80)) # Expected: "No discount"
print(get_discount(False, 220)) # Expected: "Discount applied"
print(get_discount(False, 50)) # Expected: "No discount"
```

93

Here is a condensed version that accomplishes the same thing

Practice Problem 3 - ANSWER

- For each row, fill in the missing entries by converting the given number from its representation to every other equivalent representation.

Binary	Decimal	Hexadecimal
100111_2	39_{10}	27_{16}
100111010_2	314_{10}	$13A_{16}$
1011101011011110_2	47838_{10}	$BADE_{16}$
101111_2	47_{10}	$2F_{16}$
101100111_2	359_{10}	167_{16}
$110000001111111111101110_2$	12648430_{10}	$C0FFEE_{16}$