BINARY

When you save a number, or some text on your computer, what happens to it? How does your computer store it? How does it do math, for example, adding 3 + 4 ?

 In memory, or on your hard drive, how is the number 7 or 2, or the letter A represented to the electronics in your computer? In your computer's memory, or CPU, all of the data that's being operated on is represented as pulses of electricity.

We humans use base 10. There are 10 different numbers in our world, the digits 0-9, that we can use to describe any number. To replicate that on a computer, the computer hardware will need to tell the difference between 10 different things - as this is electronics, the difference between 10 different voltages.  Turns out that it's quite complex to store lots of different values. If a computer had to differentiate between 10 different voltages to represent the numbers 0-9, it's quite tricky to tell the difference between two similar voltages, such as the difference between 6 and 7. The system would need to do a lot of error checking, and that slows things down, and makes the hardware much more complex and more expensive.

So computer designers decided to represent all of the data a computer uses as binary numbers. Binary is base 2. In binary, there are only two numbers: 0 and 1. If hardware only has to deal with two voltages - on and off - it's much more reliable and much easier to tell the difference. If the power supply wobbles and the computer gets 0.9v instead of 1v for a 1 value, it's still able to recognize that as a 1.

- Numbers like 32, 64, 128... are powers of two, or round numbers in binary. That's why so many things in the world of computing use these numbers.

Binary is everywhere.

* Magnetic hard drives are divided into tiny magnetic domains, one for each bit. Each bit can have its magnetic direction set to one of two directions. One direction is considered to be 0, the other direction represents 1.  A read/write head for the drive can set, or read, the magnetic direction of any bit.
* Solid state drives, flash drives, cellphone storage etc. store 1s and 0s in memory cells... they are pretty complicated! But they still store 0s and 1s. <https://en.wikipedia.org/wiki/Flash_memory>
* Optical disc have very small shiny, and non-shiny parts of the tracks - called lands and pits, representing 1 and 0. The disc starts shiny, and to write, a laser burns pits where a 0 should be written. When reading, a laser shines on the disc, and if the light is reflected from a shiny land, that's a 1; if it doesn't reflect from a pit, that's a 0.
* Networks transmit electrical pulse for 1, and no pulse for 0

And then what do you do with letters? You know that letters are encoded as ASCII (in the beginning) and Unicode (now) which are numerical values.

Python supports writing number literals in binary. Simply preface a number with 0b and Python will treat it as a binary value.

It also has a bin() function to return a string representing a number in binary. Example:

fourty\_five = 0b101101   # The binary number 101101 is 45 in decimal  
print(fourty\_five) # prints 45. Numbers are printed in decimal, regardless of base  
  
twelve = 0b1100 # 12  
three = 0b11 # 3 in decimal  
  
sum = twelve + three  
print(sum) # Prints 15  
  
# The bin() function  
print(bin(sum)) # The string '0b1111'  
  
binval = bin(45) # What's 45 in binary?  
print(binval) # The STRING '0b101101'  
print(type(binval))

**Converting decimal to binary**

**An algorithm to convert decimal number to binary string**

def binary(decimal):  
 ''' Convert a decimal number to a binary string '''  
  
 bin\_str = ''  
 place = 1  
  
 while place \* 2 <= decimal :  
 place \*= 2  
  
 while place >= 1:  
 if decimal >= place:  
 bin\_str = bin\_str + '1'  
 else:  
 bin\_str = bin\_str + '0'  
  
 if decimal >= place:  
 decimal -= place  
  
 place = place / 2  
  
 return bin\_str

**An algorithm to convert binary string to decimal number**

# Binary string to decimal number. This considers all binary numbers to be positive.   
# There are MANY other algorithms for doing this. Can you think of another way?  
  
def decimal(binary\_str):  
   
 place = 1; # The 1s place, the 2s place, the 4s place...  
 dec = 0  
 for bit in binary\_str[::-1]: # Loop from end of string to the start  
 print(dec, place)  
 if (bit == '1'): # If the digit is a 1, add on the place value  
 dec += place  
 place \*= 2  
  
 return dec

More reading:  <http://www.math.grin.edu/~rebelsky/Courses/152/97F/Readings/student-binary>

Negative numbers in binary: if your computer hardware can only represent 0 and 1 - no other symbols - how do you represent negative numbers? Various ways, but two's complement is the most common: <https://en.wikipedia.org/wiki/Two's_complement>