How numbers, images and texts are represented in the computer

Christophe@pallier.org

Sept. 2015

Contents

Representation of integers 1
Representation of text 4
How to read a text file in Python 5
Representation of images 6
Black and white bitmaps 7
Grey level pictures **8**
Colored bitmaps 9

There are 10 kinds of people: those who count in binary and the others

Computers represent all objects as series of 0 and 1, also known as bits (for “binary digits”).

Representation of integers

Just like a number can be written in base 10, it can be written in base 2:

E.g.

```
12 : 10  + 2  = 8 + 4 = 2**3 + 2**1 : 1010
33 : 30  + 3  = 32 + 1 = 2**5 + 1 : 100001

1 :   1
2 :  10
3 :  11
```

```
4 : 100
5 : 101
6 : 110
7 : 111
...
```

To learn more about how integer numbers can be represented in binary format, see http://csunplugged.org/binary-numbers

(1) Convert (manually) into decimal the following binary numbers:

- 101
- 1000
- 1011
- 11111111

Answer: 5, 8, 11, 255

(2) Write a function that, given the binary representation of a number as a string of ‘0’ and ‘1’, returns its value as a integer.

```python
def todec(s):
    n = 0
    for i in s:
        n = n * 2 + int(i)
    return n

for i in ['101', '1000', '1011', '11111111']:
    print(todec(i))
```

(3) Now we will go in the other direction: Our aim is to write a program that, given a number (in decimal), computes its binary representation.

If you have an idea how to program it, please proceed. If not, we propose that you follow the following steps:

(4) Study the program below. Execute it with various values of the variable *num*. Do you understand the last line? Do you see a limitation of this program?

```python
num = 143
d3 = int(num/1000) % 10 # thousands
d2 = int(num/100)  % 10 # hundreds
d1 = int(num/10)   % 10 # dec
d0 =  num % 10
print(str(d3) + str(d2) + str(d1) + str(d0))
```

(5) Adapt the above program to print the binary representation of num

```
num = 17
b0 = num % 2
b1 = int(num/2) % 2
b2 = int(num/4) % 2
b3 = int(num/8) % 2
b4 = int(num/16)  % 2
b5 = int(num/32)  % 2
b6 = int(num/64)  % 2
b7 = int(num/128) % 2
b8 = int(num/256) % 2
print(str(b8) + str(b7) +  str(b6) + str(b5) + str(b4) + str(b3) + str(b2) + str(b1) + str(b
```

(6) Modify the above program to print the binary representations of every number between 0 and 255.

```
def tobin(num):
    b8 = int(num/256) % 2
    b7 = int(num/128) % 2
    b6 = int(num/64)  % 2
    b5 = int(num/32)  % 2
    b4 = int(num/16)  % 2
    b3 = int(num/8) % 2
    b2 = int(num/4) % 2
    b1 = int(num/2) % 2
    b0 = num % 2
    return (str(b8) + str(b7) +  str(b6) + str(b5) + str(b4) + str(b3) + str(b2) + str(b1) +

for n in range(256):
    print(n, tobin(n))
```

(7) (Advanced) Write an improved version that uses a loop and does not have a limitation in size.

```
def binary(n):
    if n==0:
        return "0"
    s = ""
    while n > 0:
        b = str(n % 2)
        s = b + s
        n = n / 2
    return s
```

(8) Study the following code. Do you understand why it works?

```
def binary(num):
    if num == 0:
        return "0"
    if num == 1:
        return "1"
    return(binary(int(num /2)) + binary(num % 2))

print(binary(1234))
```

Answer: It is a recursive function which calls itself. See http://en.wikipedia.org/wiki/Recursion_%28computer_science%29

Remark: measures of memory size

- 1 byte = 8 bits
- 1 Kilobyte (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 kbytes = 1048576 bytes
- 1 Gigabytes (GB) = 1024 Mbyes
- Terabyte, Petabyte, Exabyte…

Exercice (advanced): Write a function that return the hexadecimal representation (base 16) of a number.

To go further:

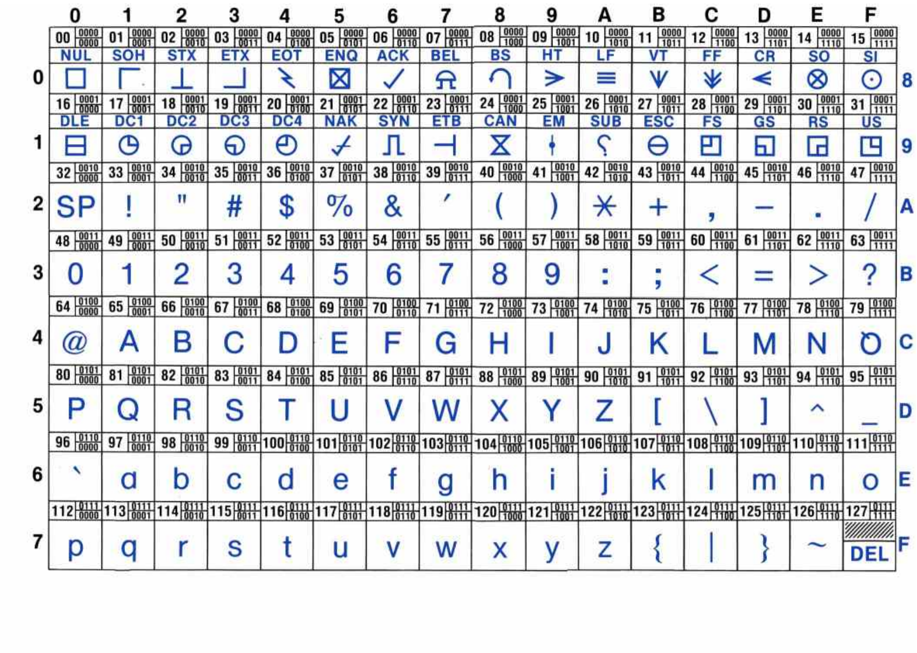
- If you want to know how negative integer numbers are represented, see http://en.wikipedia.org/wiki/Two%27s_complement
- Execute 0.1 + 0.2 on the python command line. If you are surprised, read What Every Programmer Should Know About Floating-Point Arithmetic

Representation of text

A text file is nothing but a sequences of characters (a word document is not a text file).

For a long time, characters were encoded using ASCII code.

(9) lookup the ASCII representation of your first name in the table and use the chr function of Python to print it.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	00 0000 0000 NUL □	01 0000 0001 SOH ⌜	02 0000 0010 STX ⊥	03 0000 0011 ETX ⌋	04 0000 0100 EOT ⌁	05 0000 0101 ENQ ⊠	06 0000 0110 ACK ✓	07 0000 0111 BEL ⍾	08 0000 1000 BS ⤺	09 0000 1001 HT ⪫	10 0000 1010 LF ≡	11 0000 1011 VT ⩛	12 0000 1100 FF ↡	13 0000 1101 CR ⪪	14 0000 1110 SO ⊗	15 0000 1111 SI ⊙	8
1	16 0001 0000 DLE ⊟	17 0001 0001 DC1 ◷	18 0001 0010 DC2 ◶	19 0001 0011 DC3 ◵	20 0001 0100 DC4 ◴	21 0001 0101 NAK ⍻	22 0001 0110 SYN ⎍	23 0001 0111 ETB ⊣	24 0001 1000 CAN ⧖	25 0001 1001 EM ⍿	26 0001 1010 SUB ␦	27 0001 1011 ESC ⊖	28 0001 1100 FS ◰	29 0001 1101 GS ◱	30 0001 1110 RS ◲	31 0001 1111 US ◳	9
2	32 0010 0000 SP	33 0010 0001 !	34 0010 0010 "	35 0010 0011 #	36 0010 0100 $	37 0010 0101 %	38 0010 0110 &	39 0010 0111 '	40 0010 1000 (	41 0010 1001)	42 0010 1010 *	43 0010 1011 +	44 0010 1100 ,	45 0010 1101 -	46 0010 1110 .	47 0010 1111 /	A
3	48 0011 0000 0	49 0011 0001 1	50 0011 0010 2	51 0011 0011 3	52 0011 0100 4	53 0011 0101 5	54 0011 0110 6	55 0011 0111 7	56 0011 1000 8	57 0011 1001 9	58 0011 1010 :	59 0011 1011 ;	60 0011 1100 <	61 0011 1101 =	62 0011 1110 >	63 0011 1111 ?	B
4	64 0100 0000 @	65 0100 0001 A	66 0100 0010 B	67 0100 0011 C	68 0100 0100 D	69 0100 0101 E	70 0100 0110 F	71 0100 0111 G	72 0100 1000 H	73 0100 1001 I	74 0100 1010 J	75 0100 1011 K	76 0100 1100 L	77 0100 1101 M	78 0100 1110 N	79 0100 1111 O	C
5	80 0101 0000 P	81 0101 0001 Q	82 0101 0010 R	83 0101 0011 S	84 0101 0100 T	85 0101 0101 U	86 0101 0110 V	87 0101 0111 W	88 0101 1000 X	89 0101 1001 Y	90 0101 1010 Z	91 0101 1011 [	92 0101 1100 \	93 0101 1101]	94 0101 1110 ^	95 0101 1111 _	D
6	96 0110 0000 `	97 0110 0001 a	98 0110 0010 b	99 0110 0011 c	100 0110 0100 d	101 0110 0101 e	102 0110 0110 f	103 0110 0111 g	104 0110 1000 h	105 0110 1001 i	106 0110 1010 j	107 0110 1011 k	108 0110 1100 l	109 0110 1101 m	110 0110 1110 n	111 0110 1111 o	E
7	112 0111 0000 p	113 0111 0001 q	114 0111 0010 r	115 0111 0011 s	116 0111 0100 t	117 0111 0101 u	118 0111 0110 v	119 0111 0111 w	120 0111 1000 x	121 0111 1001 y	122 0111 1010 z	123 0111 1011 {	124 0111 1100 |	125 0111 1101 }	126 0111 1110 ~	127 0111 1111 ▨ DEL	F

Figure 1: ascii table

For example, if you name is 'ZOE', you would type:

```
print(chr(90)+chr(79)+chr(69))
```

Remark: ASCII codes use one byte per characters. This is fine for English, but cannot cover all the caracters of all alphabets. It cannot even encode french accented letters. Unicode was invented that associate a unique 2 bytes number to each character of any human script. It is possible to write text files using these number, but more economic to encode the most common letters with one byte, and keep the compatibility with ASCII (UTF-8).

How to read a text file in Python

Download Alice in Wonderland

```
f = file('alice.txt')
o = f.read()
print(o)
lines = o.split("\n")
print(lines)
```

(10) Write a program that counts the number of lines, and number of words in alice.txt (we suppose that words are separated by spaces).

```
f = file('alice.txt')
o = f.read()
print(o)
lines = o.split("n")

nlines = len(lines)

nw = 0
for l in lines:
    nw += len(l.split(" "))

print(nlines)
print(nw)
```

(11) Write a program that detects if a text file contains the word ‘NSA’

```
def spot_nsa(filename):
    f = file(filename)
    o = f.read()
    lines = o.split("n")
    found = False
    for l in lines:
        if "NSA" in l.split(" "):
            found = True
            break
    return found
```

Representation of images

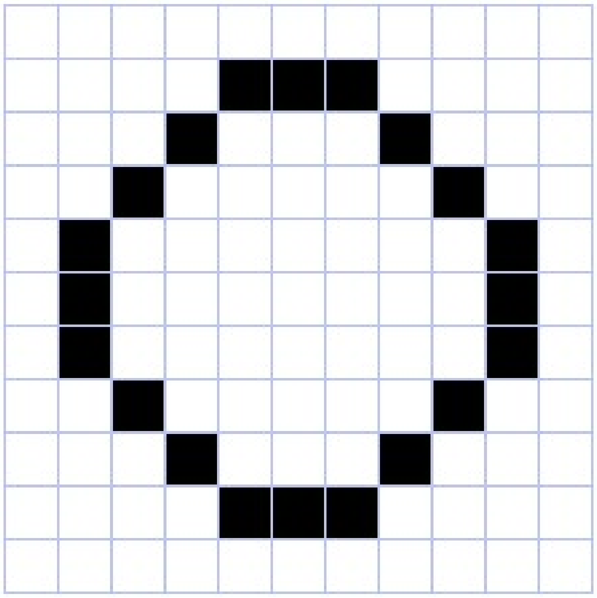
Images can be stored either:

- as bitmaps, that is a two dimensional arrays of dots (formats: bmp, png, gif, jpeg. . .)
- as vectorized formats, the image contain instruction for drawing objects (eps, pdf, svg, . . .).

Here we are just going to manipulate bitmaps.

Black and white bitmaps

Each dot (pixel) is either '0' (black) or '1' (white).



(12) What is the size in kilobytes of a 1024x768 pixels images ?

Answer: 1024*768/8/1024=96 KB

(13) Execute the follwoing code (it requires the modules numpy and matplotlib).

```
import numpy as np
import matplotlib.pyplot as plt

a = np.array([[0, 0, 0, 0, 0, 0, 0],
              [0, 0, 1, 1, 1, 0, 0],
              [0, 0, 1, 1, 1, 0, 0],
              [0, 0, 1, 1, 1, 0, 0],
              [0, 0, 1, 1, 1, 0, 0],
              [0, 0, 1, 1, 1, 0, 0],
              [0, 0, 0, 0, 0, 0, 0]])
plt.imshow(a, cmap=plt.cm.gray, interpolation='nearest')
plt.show()
```

Numpy's arrays are a new type of object. There are similar to lists, but optimised for mathematical computations. Notably, they can be multidimensional (i.e. you can use a[i,j] notation). You can learn more about arrays in the documents http://www.pallier.org/cours/AIP2013/python4science.pdf and http://wiki.scipy.org/Tentative_NumPy_Tutorial.

(14) Exercice:

1. Create a cross.
2. create a 200x200 bitmap:
    1. add a diagonal
    2. make two crosses imitating the British Flag

```
a = np.zeros((200,200))
for i in range(200):
    a[i, i] = 1
plt.imshow(a, cmap=plt.cm.gray, interpolation='nearest')
plt.show()
a[0:200:2,] = 1
plt.imshow(a, cmap=plt.cm.gray, interpolation='nearest')
plt.show()
```

Grey level pictures

Each dot is now associated to an integer value, e.g. ranging from 0 to 255 for 8-bits codes, coding for a grey level (smaller=darker). Each dot needs one byte.

How large is the file for a 1024x768 image pixels with 256 grey levels?

The following code displays an image:

```
import scipy.misc
l = scipy.misc.lena()
plt.imshow(l,  cmap=plt.cm.gray)
plt.show()
```

This code runs a low pass (averaging) filter on it:

```
import scipy.ndimage
bl = scipy.ndimage.gaussian_filter(l, 3)
plt.imshow(bl,  cmap=plt.cm.gray)
plt.show()
```

Edge detector. It is easy to implement an edge detector with a neural network. See https://courses.cit.cornell.edu/bionb2220/UnderstandingLateralInhibition.html.

Using the ndimage.convolve function, apply the following filters to the image and diplay the results.

```
kernel1 = np.array([[-1, -1, -1],
                   [-1,  8, -1],
                   [-1, -1, -1]])


kernel2 = np.array([[-1, -1, -1, -1, -1],
                   [-1,  1,  2,  1, -1],
                   [-1,  2,  4,  2, -1],
                   [-1,  1,  2,  1, -1],
                   [-1, -1, -1, -1, -1]])
```

More manipulations are available at http://scipy-lectures.github.io/advanced/image_processing/.

Colored bitmaps

Each dot is now associated to three bytes, representing the Red, Gree and Blue intensities (see http://www.colorpicker.com/).

How large is the file for a 1024x768 RGB image?

Exercice: What are the RGB triplets for BLACK, WHITE, RED, YELLOW?