Number Systems

A

Objectives

In this appendix, you'll:

- Learn basic number systems concepts, such as base, positional value and symbol value.
- Learn how to work with numbers represented in the binary, octal and hexadecimal number systems.
- Abbreviate binary numbers as octal numbers or hexadecimal numbers.
- Convert octal numbers and hexadecimal numbers to binary numbers.
- Convert back and forth between decimal numbers and their binary, octal and hexadecimal equivalents.
- Format binary, octal and hexadecimal numbers as strings.

A.1 Introduction

In this appendix, we introduce the binary, octal and hexadecimal number systems. When we write an integer such as 227 or –63 in a Python program, the number is assumed to be in the decimal (base 10) number system. The digits in the decimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The lowest digit is 0 and the highest digit is 9—one less than the base of 10.

Binary Numbers

Internally, computers use the binary (base 2) number system, which has only two digits—0 and 1. Its lowest digit is 0 and its highest digit is 1—one less than the base of 2. As we'll see, binary numbers tend to be much longer than their decimal equivalents. It's cumbersome to work with binary numbers. So two other number systems—the octal number system (base 8) and the hexadecimal number system (base 16)—are popular primarily because they make it convenient to abbreviate binary numbers.

Octal Numbers

In the octal number system, the digits range from 0 to 7. Because both the binary number system and the octal number system have fewer digits than the decimal number system, their digits are the same as the corresponding digits in decimal.

Hexadecimal Numbers

The hexadecimal number system poses a problem because it requires 16 digits—a lowest digit of 0 and a highest digit with a value equivalent to decimal 15 (one less than the base of 16). Hexadecimal numbers use the letters A through F to represent the hexadecimal digits corresponding to decimal values 10 through 15. Thus in hexadecimal we can have numbers like 876 consisting solely of decimal-like digits, numbers like 8A55F consisting of digits and letters, and numbers like FFE consisting solely of letters.

Binary, Octal, Decimal and Hexadecimal Digits

The following table summarizes the digits of the binary, octal, decimal and hexadecimal number systems:

Binary digit	Octal digit	Decimal digit	Hexadecimal digit
0	0	0	0
1	1	1	1
	2	2	2
	3	3	3
	4	4	4
	5	5	5
	6	6	6
	7	7	7
		8	8
		9	9
			A (decimal value of 10)
			B (decimal value of 11)
			C (decimal value of 12)
			D (decimal value of 13)
			E (decimal value of 14)
			F (decimal value of 15)

The following table compares the binary, octal, decimal and hexadecimal number systems:

Attribute	Binary	Octal	Decimal	Hexadecimal
Base	2	8	10	16
Lowest digit	0	0	0	0
Highest digit	1	7	9	F

Decimal Positional Values

Each number system uses **positional notation**—each position in which a digit is written has a different positional value. For example, in the decimal number 937, we say that the 7 is written in the ones position, the 3 is written in the tens position and the 9 is written in the hundreds position. Each of these positions is a power of the base (in this case, base 10) and that these powers begin at 0 and increase by 1 as we move left in the number:

Positional values in the decimal number system			
Decimal digit	9	3	7
Position name	Hundreds	Tens	Ones
Positional value	100	10	1
Positional value as a power of the base (10)	10^2	10^1	10^0

For longer decimal numbers, the next positions to the left would be the thousands position (10 to the 3rd power), the ten-thousands position (10 to the 4th power), the hundred-thousands position (10 to the 5th power), the millions position (10 to the 6th power), the ten-millions position (10 to the 7th power) and so on.

Binary Positional Values

In the binary number 101, the rightmost 1 is written in the ones position, the 0 is written in the twos position and the leftmost 1 is written in the fours position. Each position is a power of base 2. These powers begin at 0 and increase by 1 as we move left in the number. So, $101 = (1 * 2^2) + (0 * 2^1) + (1 * 2^0) = 4 + 0 + 1 = 5$:

Positional values in the binary number system			
Binary digit	1	0	1
Position name	Fours	Twos	Ones
Positional value	4	2	1
Positional value as a power of the base (2)	2^2	2^1	2^0

For longer binary numbers, the next positions to the left would be the eights position (2^3), the sixteens position (2^4), the thirty-twos position (2^5), the sixty-fours position (2^6) and so on.

Octal Positional Values

In the octal number 425, we say that the 5 is written in the ones position, the 2 is written in the eights position and the 4 is written in the sixty-fours position. Each of these positions is a power of base 8 and that these powers begin at 0 and increase by 1 as we move left in the number:

Positional values in the octal number system			
Decimal digit	4	2	5
Position name	Sixty-fours	Eights	Ones
Positional value	64	8	1
Positional value as a power of the base (8)	8^2	8^1	8^0

For longer octal numbers, the next positions to the left would be the five-hundred-and-twelves position (8^3), the four-thousand-and-ninety-sixes position (8^4), the thirty-two-thousand-seven-hundred-and-sixty-eights position (8^5) and so on.

Hexadecimal Positional Values

In the hexadecimal number 3DA, we say that the A is written in the ones position, the D is written in the sixteens position and the 3 is written in the two-hundred-and-fifty-sixes position. Each of these positions is a power of base 16 and that these powers begin at 0 and increase by 1 as we move left in the number:

Positional values in the hexadecimal number system			
Decimal digit	3	D	A
Position name	Two-hundred-and-fifty-sixes	Sixteens	Ones
Positional value	256	16	1
Positional value as a power of the base (16)	16^2	16^1	16^0

For longer hexadecimal numbers, the next positions to the left would be the four-thousand-and-ninety-sixes position (16^3), the sixty-five-thousand-five-hundred-and-thirty-sixes position (16^4) and so on.

A.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers

The main use for octal and hexadecimal numbers in computing is for abbreviating lengthy binary representations. The following table highlights the fact that lengthy binary numbers can be expressed concisely in number systems with higher bases than the binary number system:

Decimal number	Binary representation	Octal representation	Hexadecimal representation
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

A particularly important relationship that both the octal number system and the hexadecimal number system have to the binary system is that the bases of octal and hexadecimal (8 and 16 respectively) are powers of the base of the binary number system (base 2). Consider the following 12-digit binary number and its octal and hexadecimal equivalents:

Binary number	Octal equivalent	Hexadecimal equivalent
100011010001	4321	8D1

To see how the binary number converts easily to octal, simply break the 12-digit binary number into groups of three consecutive bits each and write those groups over the corresponding digits of the octal number as follows:

```
100    011    010    001
4      3      2      1
```

The octal digit under each group of three bits corresponds precisely to the octal equivalent of that 3-digit binary number.

Similarly, in converting from binary to hexadecimal, you break the 12-digit binary number into groups of four consecutive bits each and write those groups over the corresponding digits of the hexadecimal number as follows:

```
1000   1101   0001
8      D      1
```

Notice that the hexadecimal digit you wrote under each group of four bits corresponds precisely to the hexadecimal equivalent of that 4-digit binary number.

A.3 Converting Octal and Hexadecimal Numbers to Binary Numbers

In the previous section, we saw how to convert binary numbers to their octal and hexadecimal equivalents by forming groups of binary digits and simply rewriting them as their equivalent octal digit values or hexadecimal digit values. This process may be used in reverse to produce the binary equivalent of a given octal or hexadecimal number.

For example, the octal number 653 is converted to binary simply by writing the 6 as its 3-digit binary equivalent 110, the 5 as its 3-digit binary equivalent 101 and the 3 as its 3-digit binary equivalent 011 to form the 9-digit binary number 110101011.

The hexadecimal number FAD5 is converted to binary simply by writing the F as its 4-digit binary equivalent 1111, the A as its 4-digit binary equivalent 1010, the D as its 4-digit binary equivalent 1101 and the 5 as its 4-digit binary equivalent 0101 to form the 16-digit 1111101011010101.

A.4 Converting from Binary, Octal or Hexadecimal to Decimal

We're accustomed to working in decimal, and therefore it's often convenient to convert a binary, octal, or hexadecimal number to decimal to get a sense of what the number is "really" worth. To convert a number to decimal from another base, multiply the decimal equivalent of each digit by its positional value and sum these products. For example, let's convert the binary number 110101 to its decimal equivalent, 53:

Converting a binary number to decimal						
Postional values:	32	16	8	4	2	1
Symbol values:	1	1	0	1	0	1
Products:	1*32=32	1*16=16	0*8=0	1*4=4	0*2=0	1*1=1
Sum:	= 32 + 16 + 0 + 4 + 0s + 1 = 53					

To convert octal 7614 to its decimal equivalent, 3980, we use the same technique, this time using appropriate octal positional values:

Converting an octal number to decimal				
Positional values:	512	64	8	1
Symbol values:	7	6	1	4
Products	7*512=3584	6*64=384	1*8=8	4*1=4
Sum:	= 3584 + 384 + 8 + 4 = 3980			

To convert hexadecimal AD3B to its decimal equivalent, 44347, we use the same technique, this time using appropriate hexadecimal positional values:

Converting a hexadecimal number to decimal				
Postional values:	4096	256	16	1
Symbol values:	A (=10)	D (=13)	3	B
Products	10*4096=40960	13*256=3328	3*16=48	B*1=11
Sum:	= 40960 + 3328 + 48 + 11 = 44347			

A.5 Converting from Decimal to Binary, Octal or Hexadecimal

Converting from decimal to binary, octal, or hexadecimal also follows the conventions in the preceding section.

Decimal-to-Binary Conversions

Suppose we wish to convert decimal 57 to binary. We begin by writing the positional values of the columns right to left until we reach a column whose positional value is greater than the decimal number. We don't need that column, so we discard it. Thus, we first write:

Positional values:	64	32	16	8	4	2	1

Then we discard the column with positional value 64, leaving:

Positional values:	32	16	8	4	2	1

Next we work from the leftmost column to the right. We divide 32 into 57 and observe that there's one 32 in 57 with a remainder of 25, so we write 1 in the 32 column. We divide 16 into 25 and observe that there's one 16 in 25 with a remainder of 9 and write

1 in the 16 column. We divide 8 into 9 and observe that there's one 8 in 9 with a remainder of 1. The next two columns each produce quotients of 0 when their positional values are divided into 1, so we write 0s in the 4 and 2 columns. Finally, 1 into 1 is 1, so we write 1 in the 1 column. This yields:

Positional values:	32	16	8	4	2	1
Symbol values:	1	1	1	0	0	1

and thus decimal 57 is equivalent to binary 111001.

Decimal-to-Octal Conversions

To convert decimal 103 to octal, we begin by writing the positional values of the columns until we reach a column whose positional value is greater than the decimal number. We do not need that column, so we discard it. Thus, we first write:

Positional values:	512	64	8	1

Then we discard the column with positional value 512, yielding:

Positional values:	64	8	1

Next we work from the leftmost column to the right. We divide 64 into 103 and observe that there's one 64 in 103 with a remainder of 39, so we write 1 in the 64 column. We divide 8 into 39 and observe that there are four 8s in 39 with a remainder of 7 and write 4 in the 8 column. Finally, we divide 1 into 7 and observe that there are seven 1s in 7 with no remainder, so we write 7 in the 1 column. This yields:

Positional values:	64	8	1
Symbol values:	1	4	7

and thus decimal 103 is equivalent to octal 147.

Decimal-to-Hexadecimal Conversions

To convert decimal 375 to hexadecimal, we begin by writing the positional values of the columns until we reach a column whose positional value is greater than the decimal number. We do not need that column, so we discard it. Thus, we first write:

Positional values:	4096	256	16	1

Then we discard the column with positional value 4096, yielding:

Positional values:	256	16	1

Next we work from the leftmost column to the right. We divide 256 into 375 and observe that there's one 256 in 375 with a remainder of 119, so we write 1 in the 256 column. We divide 16 into 119 and observe that there are seven 16s in 119 with a remainder of 7 and write 7 in the 16 column. Finally, we divide 1 into 7 and observe that there are seven 1s in 7 with no remainder, so we write 7 in the 1 column. This yields:

Positional values:	256	16	1
Symbol values:	1	7	7

and thus decimal 375 is equivalent to hexadecimal 177.

A.6 Converting Integers to Strings with Built-in Functions bin, oct and hex

You can convert values to binary, octal and hexadecimal strings with the built-in functions **bin**, **oct** and **hex** respectively.

```
In [1]: bin(100)
Out[1]: '0b1100100'

In [2]: oct(100)
Out[2]: '0o144'

In [3]: hex(100)
Out[3]: '0x64'
```

A.7 Binary, Octal and Hexadecimal String Formatting

Integers

You can format integers as strings in binary (**b**), octal (**o**) or hexadecimal (**x** or **X**) format:

```
In [1]: f'{200:b} {200:o} {200:x} {200:X}'
Out[1]: '11001000 310 c8 C8'
```

The format specifiers x and X each format an integer in hexadecimal format, but x uses lowercase a through f and X uses uppercase A through F.

Grouping Digits

As you've seen, you can format numbers with thousands separators by using a comma (,):

```
In [2]: f'{12345678:,d}'
Out[2]: '12,345,678'

In [3]: f'{123456.78:,.2f}'
Out[3]: '123,456.78'
```

To make binary, octal and hexadecimal values more readable, you can insert underscores to group every four digits of the value starting from the rightmost digit:

```
In [4]: f'{1000000000:_b}'
Out[4]: '11_1011_1001_1010_1100_1010_0000_0000'

In [5]: f'{1000000000:_o}'
Out[5]: '73_4654_5000'

In [6]: f'{1000000000:_x}'
Out[6]: '3b9a_ca00'
```

Including the Numeric Base for Binary, Octal and Hexadecimal Values

You can use # to format binary, octal and hexadecimal values preceded by '0b', '0o' or '0x' to indicate the numeric value's base number system (2, 8 or 16, respectively):

```
In [7]: f'{1000000000:#_b}'
Out[7]: '0b11_1011_1001_1010_1100_1010_0000_0000'

In [8]: f'{1000000000:#_o}'
Out[8]: '0o73_4654_5000'

In [9]: f'{1000000000:#_x}'
Out[9]: '0x3b9a_ca00'
```