

Introduction to Programming in Python

Building a Computer: Representing Information

Outline

- ① Non-negative Integers
- ② Binary Arithmetic
- ③ Negative Integers
- ④ Real Numbers
- ⑤ Characters
- ⑥ Strings
- ⑦ Structured Information

Non-negative Integers

Non-negative Integers

The number of digits needed to represent a non-negative (ie, unsigned) integer x in any base b is $\lceil \log_b(x + 1) \rceil$

Non-negative Integers

The number of digits needed to represent a non-negative (ie, unsigned) integer x in any base b is $\lceil \log_b(x + 1) \rceil$

The range of unsigned integers that can be represented using n base- b digits is $[0, b^n - 1]$

Non-negative Integers

The number of digits needed to represent a non-negative (ie, unsigned) integer x in any base b is $\lceil \log_b(x + 1) \rceil$

The range of unsigned integers that can be represented using n base- b digits is $[0, b^n - 1]$

An unsigned integer x can be expanded in any base b as the polynomial

$$x = c_{n-1}b^{n-1} + c_{n-2}b^{n-2} + \cdots + c_1b^1 + c_0b^0,$$

where the coefficients $c_i \in \{0, 1, \dots, b - 1\}$

Non-negative Integers

The number of digits needed to represent a non-negative (ie, unsigned) integer x in any base b is $\lceil \log_b(x + 1) \rceil$

The range of unsigned integers that can be represented using n base- b digits is $[0, b^n - 1]$

An unsigned integer x can be expanded in any base b as the polynomial

$$x = c_{n-1}b^{n-1} + c_{n-2}b^{n-2} + \cdots + c_1b^1 + c_0b^0,$$

where the coefficients $c_i \in \{0, 1, \dots, b - 1\}$

We say $c_{n-1}c_{n-2} \dots c_1c_0$ is the base- b representation of x , and write it as $c_{n-1}c_{n-2} \dots c_1c_0_b = x_{10}$

Non-negative Integers

Non-negative Integers

Example (representing 173 in the decimal system; $b = 10$ and $c_i \in \{0, 1, \dots, 9\}$)

Non-negative Integers

Example (representing 173 in the decimal system; $b = 10$ and $c_i \in \{0, 1, \dots, 9\}$)

- The number of decimal digits needed to represent 173 is $\lceil \log_{10}(173 + 1) \rceil = 3$

Non-negative Integers

Example (representing 173 in the decimal system; $b = 10$ and $c_i \in \{0, 1, \dots, 9\}$)

- The number of decimal digits needed to represent 173 is $\lceil \log_{10}(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 decimal digits is $[0, 10^3 - 1] = [0, 999]$

Non-negative Integers

Example (representing 173 in the decimal system; $b = 10$ and $c_i \in \{0, 1, \dots, 9\}$)

- The number of decimal digits needed to represent 173 is $\lceil \log_{10}(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 decimal digits is $[0, 10^3 - 1] = [0, 999]$
- $173 = 1 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$

Non-negative Integers

Example (representing 173 in the decimal system; $b = 10$ and $c_i \in \{0, 1, \dots, 9\}$)

- The number of decimal digits needed to represent 173 is $\lceil \log_{10}(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 decimal digits is $[0, 10^3 - 1] = [0, 999]$
- $173 = 1 \cdot 10^2 + 7 \cdot 10^1 + 3 \cdot 10^0$
- Therefore, $173_{10} = 173_{10}$

Non-negative Integers

Non-negative Integers

Example (representing 173 in the binary system; $b = 2$ and $c_i \in \{0, 1\}$)

Non-negative Integers

Example (representing 173 in the binary system; $b = 2$ and $c_i \in \{0, 1\}$)

- The number of binary digits (aka bits) needed to represent 173 is $\lceil \log_2(173 + 1) \rceil = 8$

Non-negative Integers

Example (representing 173 in the binary system; $b = 2$ and $c_i \in \{0, 1\}$)

- The number of binary digits (aka bits) needed to represent 173 is $\lceil \log_2(173 + 1) \rceil = 8$
- The range of unsigned integers that can be represented using 8 bits is $[0, 2^8 - 1] = [0, 255]$

Non-negative Integers

Example (representing 173 in the binary system; $b = 2$ and $c_i \in \{0, 1\}$)

- The number of binary digits (aka bits) needed to represent 173 is $\lceil \log_2(173 + 1) \rceil = 8$
- The range of unsigned integers that can be represented using 8 bits is $[0, 2^8 - 1] = [0, 255]$
- $173 = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

Non-negative Integers

Example (representing 173 in the binary system; $b = 2$ and $c_i \in \{0, 1\}$)

- The number of binary digits (aka bits) needed to represent 173 is $\lceil \log_2(173 + 1) \rceil = 8$
- The range of unsigned integers that can be represented using 8 bits is $[0, 2^8 - 1] = [0, 255]$
- $173 = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
- Therefore, $10101101_2 = 173_{10}$

Non-negative Integers

Non-negative Integers

Example (representing 173 in the octal system; $b = 8$ and $c_i \in \{0, 1, \dots, 7\}$)

Non-negative Integers

Example (representing 173 in the octal system; $b = 8$ and $c_i \in \{0, 1, \dots, 7\}$)

- The number of octal digits needed to represent 173 is $\lceil \log_8(173 + 1) \rceil = 3$

Non-negative Integers

Example (representing 173 in the octal system; $b = 8$ and $c_i \in \{0, 1, \dots, 7\}$)

- The number of octal digits needed to represent 173 is $\lceil \log_8(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 octal digits is $[0, 8^3 - 1] = [0, 511]$

Non-negative Integers

Example (representing 173 in the octal system; $b = 8$ and $c_i \in \{0, 1, \dots, 7\}$)

- The number of octal digits needed to represent 173 is $\lceil \log_8(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 octal digits is $[0, 8^3 - 1] = [0, 511]$
- $173 = 2 \cdot 8^2 + 5 \cdot 8^1 + 5 \cdot 8^0$

Non-negative Integers

Example (representing 173 in the octal system; $b = 8$ and $c_i \in \{0, 1, \dots, 7\}$)

- The number of octal digits needed to represent 173 is $\lceil \log_8(173 + 1) \rceil = 3$
- The range of unsigned integers that can be represented using 3 octal digits is $[0, 8^3 - 1] = [0, 511]$
- $173 = 2 \cdot 8^2 + 5 \cdot 8^1 + 5 \cdot 8^0$
- Therefore, $255_8 = 173_{10}$

Non-negative Integers

Non-negative Integers

Example (representing 173 in the hexadecimal system; $b = 16$ and $c_i \in \{0, 1, \dots, 9, A, B, C, D, E, F\}$)

Non-negative Integers

Example (representing 173 in the hexadecimal system; $b = 16$ and $c_i \in \{0, 1, \dots, 9, A, B, C, D, E, F\}$)

- The number of hexadecimal digits needed to represent 173 is $\lceil \log_{16}(173 + 1) \rceil = 2$

Non-negative Integers

Example (representing 173 in the hexadecimal system; $b = 16$ and $c_i \in \{0, 1, \dots, 9, A, B, C, D, E, F\}$)

- The number of hexadecimal digits needed to represent 173 is $\lceil \log_{16}(173 + 1) \rceil = 2$
- The range of unsigned integers that can be represented using 2 hexadecimal digits is $[0, 16^2 - 1] = [0, 255]$

Non-negative Integers

Example (representing 173 in the hexadecimal system; $b = 16$ and $c_i \in \{0, 1, \dots, 9, A, B, C, D, E, F\}$)

- The number of hexadecimal digits needed to represent 173 is $\lceil \log_{16}(173 + 1) \rceil = 2$
- The range of unsigned integers that can be represented using 2 hexadecimal digits is $[0, 16^2 - 1] = [0, 255]$
- $173 = A \cdot 16^1 + D \cdot 16^0$

Non-negative Integers

Example (representing 173 in the hexadecimal system; $b = 16$ and $c_i \in \{0, 1, \dots, 9, A, B, C, D, E, F\}$)

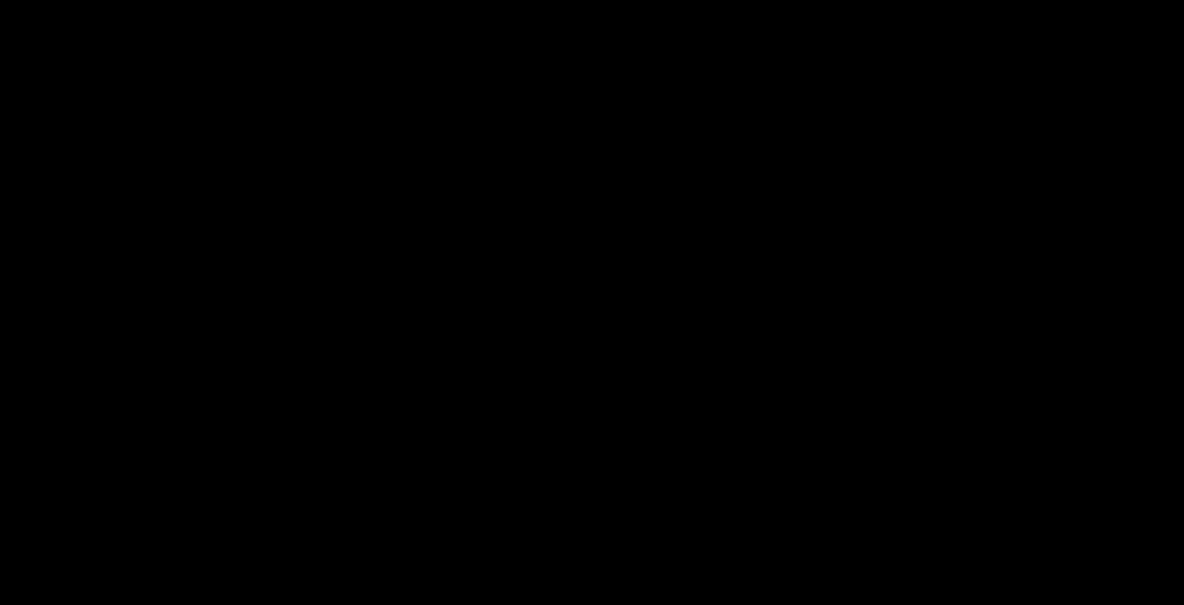
- The number of hexadecimal digits needed to represent 173 is $\lceil \log_{16}(173 + 1) \rceil = 2$
- The range of unsigned integers that can be represented using 2 hexadecimal digits is $[0, 16^2 - 1] = [0, 255]$
- $173 = A \cdot 16^1 + D \cdot 16^0$
- Therefore, $AD_{16} = 173_{10}$

Non-negative Integers

Non-negative Integers

Dec	Bin	Oct	Hex
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

Binary Arithmetic



Binary Arithmetic

Example (addition in binary)

			1	1	1	1				
		1	0	1	0	1	1	0	1	173
+	0	0	1	1	0	1	0	0		52
	1	1	1	0	0	0	0	1		225

Negative Integers

Negative Integers

Two's complement method to compute $-x$

Negative Integers

Two's complement method to compute $-x$

- Represent x in binary

Negative Integers

Two's complement method to compute $-x$

- Represent x in binary
- Flip the bits of the result

Negative Integers

Two's complement method to compute $-x$

- Represent x in binary
- Flip the bits of the result
- Add 1 to the result

Negative Integers

Negative Integers

Example (-83 on an 8-bit computer)

Negative Integers

Example (-83 on an 8-bit computer)

- Represent 83 in binary as 01010011

Negative Integers

Example (-83 on an 8-bit computer)

- Represent 83 in binary as 01010011
- Flip the bits of the result to obtain 10101100

Negative Integers

Example (-83 on an 8-bit computer)

- Represent 83 in binary as 01010011
- Flip the bits of the result to obtain 10101100
- Add 1 to the result to obtain 10101101

Negative Integers

Example (-83 on an 8-bit computer)

- Represent 83 in binary as 01010011
- Flip the bits of the result to obtain 10101100
- Add 1 to the result to obtain 10101101

Note: just like how $83 + (-83) = 0$, we have $01010011 + 10101101 = 10000000$

Negative Integers

Negative Integers

The range of signed integers that can be represented using n bits is $[-2^{n-1}, 2^{n-1} - 1]$

Negative Integers

The range of signed integers that can be represented using n bits is $[-2^{n-1}, 2^{n-1} - 1]$

Example (signed 4-bit integers)

Dec	Bin
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Real Numbers

Real Numbers

sign (1 bit)	offset-binary exponent (5 bits)	binary fraction (10 bits)
--------------	---------------------------------	---------------------------

IEEE 754 Half-precision Format

Real Numbers

sign (1 bit)	offset-binary exponent (5 bits)	binary fraction (10 bits)
--------------	---------------------------------	---------------------------

IEEE 754 Half-precision Format

Example

Real Numbers

sign (1 bit)	offset-binary exponent (5 bits)	binary fraction (10 bits)
--------------	---------------------------------	---------------------------

IEEE 754 Half-precision Format

Example

$$- \underline{1} \underline{01001} \underline{1010000000} = -2^{9-15} \times 1.101_2 = -2^{-6}(1 + 2^{-1} + 2^{-3}) = -0.025390625_{10}$$

Real Numbers

sign (1 bit)	offset-binary exponent (5 bits)	binary fraction (10 bits)
--------------	---------------------------------	---------------------------

IEEE 754 Half-precision Format

Example

$$- \textcolor{red}{1} \textcolor{blue}{01001} \textcolor{blue}{1010000000} = -2^{\textcolor{red}{9}-15} \times 1.\textcolor{blue}{101}_2 = -2^{-6}(1 + 2^{-1} + 2^{-3}) = -0.025390625_{10}$$

$$- \textcolor{red}{0} \textcolor{blue}{10101} \textcolor{blue}{1001000100} = +2^{\textcolor{red}{21}-15} \times 1.\textcolor{blue}{10010001}_2 = 2^6(1 + 2^{-1} + 2^{-4} + 2^{-8}) = 100.25_{10}$$

Characters

Characters

American Standard Code for Information Interchange (ASCII) defines 8-bit character encodings

Characters

American Standard Code for Information Interchange (ASCII) defines 8-bit character encodings

The first 128 ASCII codes

000: \	016: ►	032: □	048: 0	064: @	080: P	096: `	112: p
001: ☉	017: ◀	033: !	049: 1	065: A	081: Q	097: a	113: q
002: ●	018: ‡	034: "	050: 2	066: B	082: R	098: b	114: r
003: ▼	019: ‡	035: #	051: 3	067: C	083: S	099: c	115: s
004: ◆	020: ¶	036: \$	052: 4	068: D	084: T	100: d	116: t
005: ♣	021: §	037: %	053: 5	069: E	085: U	101: e	117: u
006: ♠	022: —	038: &	054: 6	070: F	086: V	102: f	118: v
007: •	023: ‡	039: ’	055: 7	071: G	087: W	103: g	119: w
008: ■	024: †	040: (056: 8	072: H	088: X	104: h	120: x
009: ○	025: ↓	041:)	057: 9	073: I	089: Y	105: i	121: y
010: ☒	026: →	042: *	058: :	074: J	090: Z	106: j	122: z
011: ♀	027: ←	043: +	059: ;	075: K	091: [107: k	123: {
012: ♀	028: ♂	044: ,	060: <	076: L	092: \	108: l	124:
013: ♀	029: ↔	045: -	061: =	077: M	093:]	109: m	125: }
014: ♀	030: ▲	046: .	062: >	078: N	094: ^	110: n	126: ~
015: ♀	031: ▼	047: /	063: ?	079: O	095: _	111: o	127: △

0 – 31 and 127 are control characters and the rest are printable characters

Characters

American Standard Code for Information Interchange (ASCII) defines 8-bit character encodings

The first 128 ASCII codes

000: \	016: ►	032: □	048: 0	064: @	080: P	096: `	112: p
001: ☉	017: ◀	033: !	049: 1	065: A	081: Q	097: a	113: q
002: ●	018: ‡	034: "	050: 2	066: B	082: R	098: b	114: r
003: ▼	019: ‡	035: #	051: 3	067: C	083: S	099: c	115: s
004: ♦	020: ¶	036: \$	052: 4	068: D	084: T	100: d	116: t
005: ♣	021: §	037: %	053: 5	069: E	085: U	101: e	117: u
006: ♠	022: —	038: &	054: 6	070: F	086: V	102: f	118: v
007: •	023: ‡	039: ’	055: 7	071: G	087: W	103: g	119: w
008: ■	024: †	040: (056: 8	072: H	088: X	104: h	120: x
009: ○	025: ↓	041:)	057: 9	073: I	089: Y	105: i	121: y
010: ☒	026: →	042: *	058: :	074: J	090: Z	106: j	122: z
011: ♀	027: ←	043: +	059: ;	075: K	091: [107: k	123: {
012: ♀	028: ♂	044: ,	060: <	076: L	092: \	108: l	124:
013: ♀	029: ↔	045: -	061: =	077: M	093:]	109: m	125: }
014: ♀	030: ▲	046: .	062: >	078: N	094: ^	110: n	126: ~
015: ♀	031: ▼	047: /	063: ?	079: O	095: _	111: o	127: △

0 – 31 and 127 are control characters and the rest are printable characters

The 16-bit Unicode system can represent every character in every known language, with room for more

Strings

Strings

A string is a sequence of characters

Strings

A string is a sequence of characters

It is represented as a sequence of positive integers, with a “length field” at the start specifying the string’s length

Strings

A string is a sequence of characters

It is represented as a sequence of positive integers, with a “length field” at the start specifying the string’s length

Example: the string “Python” is represented in decimal as the sequence

006 080 121 116 104 111 110

and in binary as the sequence

00000110 01010000 01111001 01110100 01101000 01101111 01101110

Structured Information

Structured Information

Any structured information can be represented as a sequence of non-negative integers

Structured Information

Any structured information can be represented as a sequence of non-negative integers

Example (representing pictures, sounds, and movies)

Structured Information

Any structured information can be represented as a sequence of non-negative integers

Example (representing pictures, sounds, and movies)

- A picture as a sequence of triples, each containing the amount of red, green, and blue at a pixel

Structured Information

Any structured information can be represented as a sequence of non-negative integers

Example (representing pictures, sounds, and movies)

- A picture as a sequence of triples, each containing the amount of red, green, and blue at a pixel
- A sound as a temporal sequence of “sound pressure levels”

Structured Information

Any structured information can be represented as a sequence of non-negative integers

Example (representing pictures, sounds, and movies)

- A picture as a sequence of triples, each containing the amount of red, green, and blue at a pixel
- A sound as a temporal sequence of “sound pressure levels”
- A movie as a temporal sequence of pictures (usually 30 per second), along with a matching sound