Integers: 0, 1, 2, -3, -20
Numbers w/ finite decimal part: $\frac{1}{10} = 0.1$, $\frac{1}{2} = 0.5$
Numbers w/ infinite decimal part: $\frac{1}{3} = 0.333333...$

For some fractions, whether their representation is finite or infinite will depend on whether the representation chosen is binary (base-2) or decimal (base-10).

16 bits should be able to represent 0-65,535 but Integers are -1, 0, +

2's complement method

<table>
<thead>
<tr>
<th>decimal value</th>
<th>binary</th>
<th>inverse</th>
<th>adjusted result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>0100</td>
<td>1011</td>
<td>1100 = -4</td>
</tr>
</tbody>
</table>

To convert from normal binary to 2's digits complement, simple take your number, invert every number and add 1.

To convert back, do the same

\[
1100 \rightarrow 0011 \\
+1 \\
0100 = 4 \quad \therefore 1100 = -4
\]

To subtract numbers, use 2's comp + add.

\[
13 - 4 = 0100 \rightarrow 1011 + 1100 \\
\rightarrow 1100\quad\text{Since carry out}
\]

\[
1101 \\
+1 \\
1100 \\
\uparrow \\
1001 = 9
\]
Decimal numbers or real numbers: Whole number part + a decimal fraction part.

Example: 21.625

In Base-10

<table>
<thead>
<tr>
<th></th>
<th>10^3</th>
<th>10^2</th>
<th>10^1</th>
<th>10^0</th>
<th>10^-1</th>
<th>10^-2</th>
<th>10^-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
2 \times 10^1 = 20
\]
\[
1 \times 10^0 = 1
\]
\[
6 \times 10^{-1} = 0.6 \quad \text{move 1 decimal place or } \frac{6}{10}
\]
\[
2 \times 10^{-2} = 0.02 \quad \text{move 2 decimal places or } \frac{2}{100}
\]
\[
5 \times 10^{-3} = 0.005 \quad \text{move 3 decimal places or } \frac{5}{1000}
\]

\[
21.625 = (00010101,101)_2
\]

\[
\begin{array}{c}
\frac{.625}{2} = .101 \\
\frac{1.250}{2} = .625 \\
\frac{0.500}{2} = .250 \\
\frac{1.000}{2} = .500 \\
\end{array}
\]

All 0's finished. 20,

\[
21.625 = (00010101,101)_2
\]
There will always be values whose representations require an infinite number of digits. The best we can do is to store some finite subset of numbers that will serve as reasonable approximations to the ones we've left out. This is referred to as limited precision.

Example \( (0.1)_b = (0.0001100110011)_2 \)

To maximize the precision possible using a fixed number of bits of storage, real numbers are stored in the computer employing floating point representation. Represented by 2 parts - the normalized fraction part, called the mantissa, and the exponent to correct for placing the value in normalized form.