Example:
Suppose I choose at random one of the 26 letters of the alphabet, and we play the game of “25 questions” in which you must determine which letter I have chosen. I will only answer ‘yes’ or ‘no.’ What is the minimum number of such questions that you must ask in order to guarantee finding the answer? (What form should such questions take? e.g., “Is it A?” “Is it B?” ...or is there some more intelligent way to solve this problem?) The answer to a Yes/No question having equal probabilities conveys one bit worth of information. In the above example with equiprobable states, a result of solving this problem is about -4.7 bits.

Since:

$$I = -\log_2 p_i$$

Note that:

$$\log_2 x = 1.443 \log_e x = 3.322 \log_{10} x$$

Example:
An example illustrating this result: How much information can a student get from a single grade? First, the maximum information occurs if all grades have equal probability (e.g., in a pass/fail class, on average half should pass if we want to maximize the information given by the grade). The maximum information the student gets from a grade will be:

- Pass/Fail : 1 bit.
- A, B, C, D, F : 2.3 bits.
- A+, A-, B+, . . ., D-, F : 3.2 bits.

Thus, using +/- grading gives the students about 0.9 more bits of information per grade than without +/-, and about 2.2 bits per grade more than pass/fail.

Example:
If a source provides us with a sequence chosen from 4 symbols (say A, C, G, T), then the maximum average information per symbol is 2 bits. If

the source provides blocks of 3 of these symbols, then the maximum average information is 6 bits.

Example:
flipping a fair coin once will give us events h and t each with probability 1/2, and thus a single flip of a coin gives us $-\log_2(1/2) = 1$ bit of information (whether it comes up h or t). Flipping a fair coin n times (or, equivalently, flipping n fair coins) gives us $-\log_2(1/2)^n = \log_2(2^n) = n \log_2(2) = n$ bits of information.
We could enumerate a sequence of 25 flips as, for example:

hthhtthhhhthttthhhhthtt

or, using 1 for h and 0 for t, the 25 bits

10110010111010001011101 00

We thus get the nice fact that n flips of a fair coin give us n bits of information and takes n binary digits to specify.