The Pumping Lemma:

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, s=xyz, satisfying the following conditions:

1. For each $i \geq 0$, $xy^iz \in A$,
2. $|y|>0$, and
3. $|xy| \leq p$.

The Pumping Lemma <u>erroneous</u> usage example:

The language $L=\{0^{2n} \mid n \geq 0\}$ is obviously regular – for example, it matches the regular expression $(00)^*$. But the following pumping lemma argument seems to show it's not regular. What's gone wrong?

Proof that $L=\{0^{2n} \mid n \geq 0\}$ is not regular.

Proof by contradiction. Assume L is regular. Then, the pumping lemma holds. That is, there exists pumping length $p \geq 1$ such that any string $s \in L$ with $|s|>p$ can be written as s=xyz such that:

1. $xy^iz \in L$ for all $i \geq 0$
2. $|y|>0$
3. $|xy| \leq p$.

Let $s=0^{2p}$. Clearly, $|s|>p$. Then, we can write it as s=xyz, where $x=\varepsilon$, y=0, and $z=0^{2p-1}$. This satisfies conditions 2 and 3. However, taking i=0, we get $xy^iz=\varepsilon 0^{2p-1}=0^{2p-1}$, which is not in L because its length is odd. Therefore, the language is not regular.

Explanation: To show that a language isn't regular, you need to show that ***every*** decomposition into xyz that satisfies properties 2 and 3 fails to satisfy property 1. It's not enough to just show that one decomposition doesn't work.

For the above example, the following decomposition works:

$s=0^{2p} = 000^{2(p-1)}=\varepsilon\ 00\ 0^{2(p-1)}$ ($x=\varepsilon$, $y=00$, $z=0^{2(p-1)}$).

We have $|xy| = |\varepsilon\ 00| = 2 \leq p$ (p should be ≥ 2; if not, we can always take p=p+1),

$|y| = |00| > 0$, and $xy^iz = (00)^i 0^{2(p-1)} \in L$.

To understand why the pumping lemma is the way it is, it helps to think about the proof. If a language is regular, it is accepted by some DFA. That DFA has some number of states: call it p. By the pigeonhole principle, whenever that DFA reads a string longer than p, it must visit some state twice: say state q. Now, x is the part of the input read upto (and including) the first visit to q, y is the part read after the first visit and upto and including the second (which must be at least one character) and z is the rest. But now you can see that xz must be accepted: x takes you from the start state to q and z takes you from q to an accepting state. Likewise, xy^iz must be accepted for any positive i, since each repetition of y takes you from q back to q. Note that the decomposition of the input into x, y and z is entirely determined by the automaton which is, in turn, determined (but not uniquely) by the language. So you don't get to choose the decomposition: if the language is regular, *some* decomposition exists; to show that a language is not regular, you must show that *every* decomposition fails.