1. Let $A$ be a language that is downward self-reducible. Given an input $x$, we simulate the polynomial-time computation that (with queries) decides $A$, and recursively compute the answer to each query as it is made. Since the recursive calls are all on strings shorter than $|x|$, we will eventually reach the base case in which we query strings of length 1. The program we are describing will simply have the answers to these (constant number of) length-1 queries hard-coded. The depth of the recursion is at most $|x|$, and at each level of recursion, we need to remember the state which requires space at most $\text{poly}(|x|)$. This last point holds because the basic computation runs in polynomial time, and hence polynomial space. Thus the overall procedure runs in $\text{PSPACE}$.

2. Assume both inequalities fail to hold. Then we have $L = P$ and $P = \text{PSPACE}$ which together imply $L = \text{PSPACE}$. But we know these two classes are different, by the Space Hierarchy Theorem. Thus one of the inequalities must hold.

3. Let $f(x)$ and $g(x)$ be logspace reductions. On input $x$ we want to compute $f(g(x))$ in logspace. We cannot compute $g(x)$ and then evaluate $f$ on it, because we don’t have enough storage space to write-down the intermediate result $g(x)$, which may be polynomially long in the input length $|x|$. Let $M_f$ and $M_g$ be Turing Machines that compute $f$ and $g$ in logspace, respectively. We build a new Turing Machine that simulates $M_f$, and whenever $M_f$ would have read the $i$-th bit of its input, we pause (remembering $M_f$’s state in $O(\log |x|)$ bits), and simulate $M_g$ on input $x$ (ignoring what it would have written on its output tape) until it outputs the $i$-th bit of $g(x)$. At this point have the necessary bit to continue simulating $M_f$. At every point in this simulation we need only remember the state of one of the two machines (requiring space $O(\log |x|)$), while running the other, so the total space is $O(\log |x|)$ as desired.

If a $P$-complete language $A$ is in $L$, then to compute any language $B \in P$, we can compose the reduction $f$ from $B$ to $A$ with a logspace computation deciding $A$ in the same manner as described above to obtain a logspace computation deciding $B$. Thus $P \subseteq L$, and we already know that $L \subseteq P$, so $P = L$.

4. We need to show that under the assumption $L = P$, we have $\text{EXP} \subseteq \text{PSPACE}$. Given a language $A \in \text{EXP}$ decidable by a Turing Machine $M$ running in time $2^{\text{poly}(|x|)}$, define the language

$$\text{PAD}_A = \{x\#^N : x \in A, N = 2^{\text{poly}(|x|)}\}$$

We produce a machine $M'$ that decides $\text{PAD}_A$ as follows: we read the input up to the first $\#$ remembering $x$, check that there are exactly $N = 2^{\text{poly}(|x|)}$ $\#$s following $x$, and then simulate $M$ on input $x$. Machine $M'$ runs in time $2^{O(\text{poly}(|x|))}$ which is polynomial in its input length of
Thus $M'$ runs in polynomial time, and $\text{PAD}_A$ is in $\text{P}$. Since $L = \text{P}$, there is a Turing Machine $M''$ that decides $\text{PAD}_A$ in log space.

Now, we describe how to decide $A$ in polynomial space. Define $f(x) = \#^N$, where $N = 2^{|x|^k}$. It is clear that $f$ is computable in polynomial space, because we just need a counter that can count up to $N$. Now, given input $x$, we can decide if $x \in A$ in polynomial space by composing $f$ with the Turing Machine $M''$ that runs in $O(\log N) = \text{poly}(|x|)$ space, using the space-efficient composition from the previous problem.

5. For any language $A$, define a padded version $\text{PAD}_A = \{x\#^{n^2-|x|} : x \in A\}$.

Note that if $A \in \text{SPACE}(n^2)$, then $\text{PAD}_A \in \text{SPACE}(O(n))$. This is because we can decide $\text{PAD}_A$ by first scanning the the tape until we encounter the first $\#$. Say this happens at position $n$. Then we check that the rest of the input is exactly $n^2 - n$ additional $\#$ symbols. All of this requires only $O(n)$ space. Now we return to the beginning of the string, and simulate the machine that decides $A$ in space $n^2$, treating $\#$ symbols as blanks. Measured as a function of the input length $n^2$, the running time of this simulation is linear.

On the other hand, if $\text{PAD}_A \in \text{P}$, then $A \in \text{P}$. To decide if an input $x$ is in $A$, we simply produce the string $x\#^{n^2-|x|}$ (in polynomial time in $|x|$), and then simulate the polynomial-time machine deciding $\text{PAD}_A$.

Now, select $A$ to be a language in $\text{SPACE}(n^2)$ but not in $\text{SPACE}(O(n))$. Such a language exists by the Space Hierarchy Theorem. Suppose for the purpose of contradiction that $\text{SPACE}(O(n)) = \text{P}$. As argued above, $A \in \text{SPACE}(n^2)$ implies $\text{PAD}_A \in \text{SPACE}(O(n))$. By assumption $\text{SPACE}(O(n)) \subseteq \text{P}$. Then as argued above $\text{PAD}_A \in \text{P}$ implies $A \in \text{P}$. Finally by assumption, $\text{P} \subseteq \text{SPACE}(O(n))$, and so we conclude $A \in \text{SPACE}(O(n))$, a contradiction.