Daily Coding Problem #138

Problem

This problem was asked by Google.

Find the minimum number of coins required to make n cents.

You can use standard American denominations, that is, 1¢, 5¢, 10¢, and 25¢.

For example, given `n = 16`, return 3 since we can make it with a 10¢, a 5¢, and a 1¢.

Solution

If we think about this problem recursively, we can assume we already have a `minimum_coin(n)` function and do the following:

- Calculate the minimum number of coins to make `n - denomination` for each denomination if it's valid
- Return the min of these, plus one (to account for adding that coin).

Then, we can figure out the base cases:

- If n is 0, then we can make that with 0 coins.
- If n is one of the denominations `1, 5, 10, 25`, then return 1.

Here's the code:

```
DENOMINATIONS = [1, 5, 10, 25]

def minimum_coins(n):
    if n == 0:
        return 0
    elif n in DENOMINATIONS:
        return 1
    else:
        return min(1 + minimum_coins(n - d) for d in DENOMINATIONS if n - d >=
0)
```

This will run really slowly -- in exponential time, since for each call we're making up to 4 calls recursively and only decrementing our input by a constant value. To improve performance, we can use dynamic programming to keep an of all of our previous results. Now we can just look up values in the array instead of repeating subcomputations.

```
DENOMINATIONS = [1, 5, 10, 25]

def minimum_coins_dp(n):
    cache = [0 for _ in range(n + 1)]

    for d in DENOMINATIONS:
        if d < len(cache):
            cache[d] = 1

    for i in range(1, n + 1):
        cache[i] = min(1 + cache[i - d] for d in DENOMINATIONS if i - d >= 0)
```

```
    return cache[n]
```

Now this runs in O(n) time and space.
