Power of Two

Question:

https://leetcode.com/problems/power-of-two/submissions/

Given an integer `n`, return `true` if it is a power of two. Otherwise, return `false`.

An integer `n` is a power of two, if there exists an integer `x` such that $n == 2^x$.

Example 1:

```
Input: n = 1
Output: true
Explanation: 2^0 = 1
```

Example 2:

```
Input: n = 16
Output: true
Explanation: 2^4 = 16
```

Approach 1:

My first approach was to keep dividing the number by 2 either until the number becomes 1 or the number becomes odd.

Let's take the number to 24.

After the first division, the number becomes 12.

After the second division, the number becomes 6.

After the third division, the number becomes 3, which is odd and False is returned.

Solution 1:

```
def isPowerOfTwo(self, n: int) -> bool:
        if n<=0:
                return False
        while(n>1):
                if n%2!=0:
                        return False
                n=n//2
        return True
```

Approach 2:

If we find the base of a number using the log of base 2 then we can directly check if the number is a perfect power of 2 or not.
Ex: 16
log2(16) = 4.0 = 4
log2(15) = 3.9 = 3
(2^4) = 16
(2^3) = 8
and by comparing them to their original numbers, we can find if the number is a perfect square or not.

Solution 2:

```
def isPowerOfTwo(self, n: int) -> bool:
        if n<=0:
            return False
        a=int(m.log2(n))
        return 2**a == n
```

Approach 3:

Let's take an example to understand this approach…

N = 8 = 1000

N-1 = 7 =0111

```
      1000
      0111
  &   — — —
      0000
```

All powers of 2 have this pattern: 1000,100,10,10000

So by subtracting 1 and taking and of n&(n-1) returns 0 for perfect powers of 2.

Solution 3:

```
def isPowerOfTwo(self, n: int) -> bool:
        return n>0 and n&(n-1)==0
```