Euler Problem #4 - Largest Palindrome Product of 3 digit numbers

- Generalize algorithm:

1. largest palindromic product of 2 n-digit numbers ⟶ start with this
2. largest palindromic product of 2 n-digit numbers where n_1 and n_2 are the lengths of num1 and num2 respectively but n_1 and n_2 are not necessarily the same } expand to this

Given we need to multiply over pairs using a two pointer strategy:

- Have a pointer to the highest n-digit number and another one to the lowest n-digit number

High pointer = $10^n - 1$

low pointer = 10^{n-1}

Pseudocode:

```
Function largest palindrome(number length):

    max = 0 --> global variable to update & track largest palindrome
    low, high --> values for lowest & highest numbers using function parameter to determine length

    For i in range(start at high, end at low, step -1):
        For j in range(start at i, end low, step -1):
            product = i x j
            If product is less than max:
                break --> because as j decreases it will only produce smaller products
            If product is palindrome and greater than max:
                max = product

    return max
```

having both pointers start at high is the best option because:
1. It optimizes search for largest products first, contrary to starting pointers at low
2. Avoids checking duplicate pairs like 100x900 and 900x100 if one started low and then high

```
Function is Palindrome(number):
    convert to iterable object
    get length

    For i in range(length//2): --> use of length//2 to check each half of number to ensure symmetry
        If index not equal to          works for all length nums 12321 as checks first and last 2
        length-index-1:                works for even length nums 1221
            return False               --> given 0-based indexing this finds the corresponding mirror
    return True
```