1. Create a function that takes a number as an argument and returns True or False depending on whether the number is symmetrical or not. A number is symmetrical when it is the same as its reverse.

Examples:
is_symmetrical(7227) ➞ True
is_symmetrical(12567) ➞ False
is_symmetrical(44444444) ➞ True
is_symmetrical(9939) ➞ False
is_symmetrical(1112111) ➞ True

Ans:

In [18]:
```
def is_symmetrical(n):
    if str(n) == str(n)[::-1]:
        print(f'is_symmetrical{n} ➞ {True}')
    else:
        print(f'is_symmetrical{n} ➞ {False}')
is_symmetrical(7227)
is_symmetrical(12567)
is_symmetrical(44444444)
is_symmetrical(9939)
is_symmetrical(1112111)
```

```
is_symmetrical7227 ➞ True
is_symmetrical12567 ➞ False
is_symmetrical44444444 ➞ True
is_symmetrical9939 ➞ False
is_symmetrical1112111 ➞ True
```

2. Given a string of numbers separated by a comma and space, return the product of the numbers.

Examples:
multiply_nums("2, 3") ➞ 6
multiply_nums("1, 2, 3, 4") ➞ 24
multiply_nums("54, 75, 453, 0") ➞ 0
multiply_nums("10, -2") ➞ -20

Ans:

In [20]:
```
def multiply_nums(string):
    output = string.replace(' ','').split(',')
    product = 1
    for i in output:
        product *= int(i)
    print(f'multiply_nums{string} ➞ {product}')
multiply_nums("2, 3")
multiply_nums("1, 2, 3, 4")
multiply_nums("54, 75, 453, 0")
multiply_nums("10, -2")
```

```
multiply_nums2, 3 ➞ 6
multiply_nums1, 2, 3, 4 ➞ 24
multiply_nums54, 75, 453, 0 ➞ 0
multiply_nums10, -2 ➞ -20
```

3. Create a function that squares every digit of a number.

Examples:
square_digits(9119) ➞ 811181
square_digits(2483) ➞ 416649

square_digits(3212) ⟶ 9414

Notes:
The function receives an integer and must return an integer.
Ans:

In [23]:
```
def square_digits(n):
    string = [str(int(i)**2) for i in str(n)]
    output = ''.join(string)
    print(f'square_digits{n} ⟶ {int(output)}')

square_digits(9119)
square_digits(2483)
square_digits(3212)
```

```
square_digits9119 ⟶ 811181
square_digits2483 ⟶ 416649
square_digits3212 ⟶ 9414
```

4. Create a function that sorts a list and removes all duplicate items from it.

Examples:
setify([1, 3, 3, 5, 5]) ⟶ [1, 3, 5]
setify([4, 4, 4, 4]) ⟶ [4]
setify([5, 7, 8, 9, 10, 15]) ⟶ [5, 7, 8, 9, 10, 15]
setify([3, 3, 3, 2, 1]) ⟶ [1, 2, 3]

Ans:

In [24]:
```
def setify(string):
    output = sorted(set(string))
    print(f'setify{string} ⟶ {output}')
setify([1, 3, 3, 5, 5])
setify([4, 4, 4, 4])
setify([5, 7, 8, 9, 10, 15])
setify([3, 3, 3, 2, 1])
```

```
setify[1, 3, 3, 5, 5] ⟶ [1, 3, 5]
setify[4, 4, 4, 4] ⟶ [4]
setify[5, 7, 8, 9, 10, 15] ⟶ [5, 7, 8, 9, 10, 15]
setify[3, 3, 3, 2, 1] ⟶ [1, 2, 3]
```

5. Create a function that returns the mean of all digits.

Examples:
mean(42) ⟶ 3
mean(12345) ⟶ 3
mean(666) ⟶ 6

Notes:
1.The mean of all digits is the sum of digits / how many digits there are (e.g. mean of digits in 512 is (5+1+2)/3(number of digits) = 8/3=2).
2.The mean will always be an integer.

Ans:

In [27]:

```
def mean(n):
    string = [int(i) for i in str(n)]
    output = sum(string)/len(str(n))
    print(f'Mean({n})  ➞ {int(output)}')

mean(42)
mean(12345)
mean(666)
```

```
Mean(42)  ➞ 3
Mean(12345)  ➞ 3
Mean(666)  ➞ 6
```