#### 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

I/P:

**def** is\_symmetrical(in\_num):

**if** str(in\_num) **==** str(in\_num)[::**-**1]:

print(f'{in\_num} ➞ {**True**}')

**else**:

print(f'{in\_num} ➞ {**False**}')

is\_symmetrical(7227)

is\_symmetrical(12567)

is\_symmetrical(44444444)

is\_symmetrical(9939)

is\_symmetrical(1112111)

O/P:

7227 ➞ True

12567 ➞ False

44444444 ➞ True

9939 ➞ False

1112111 ➞ True

#### 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

I/P:

**def** multiply\_nums(in\_string):

out\_string **=** in\_string**.**replace(' ','')**.**split(',')

out\_num **=** 1

**for** ele **in** out\_string:

out\_num **\*=** int(ele)

print(f'{in\_string} ➞ {out\_num}')

multiply\_nums("2, 3")

multiply\_nums("1, 2, 3, 4")

multiply\_nums("54, 75, 453, 0")

multiply\_nums("10, -2")

O/P:

2, 3 ➞ 6

1, 2, 3, 4 ➞ 24

54, 75, 453, 0 ➞ 0

10, -2 ➞ -20

#### 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.

I/P:

**def** square\_digits(in\_num):

in\_list **=** [str(int(ele)**\*\***2) **for** ele **in** str(in\_num)]

out\_list **=** ''**.**join(in\_list)

print(f'{in\_num} ➞ {int(out\_list)}')

square\_digits(9119)

square\_digits(2483)

square\_digits(3212)

O/P:

9119 ➞ 811181

2483 ➞ 416649

3212 ➞ 9414

#### 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]

I/P:

**def** setify(in\_list):

out\_list **=** sorted(set(in\_list))

print(f'{in\_list} ➞ {out\_list}')

setify([1, 3, 3, 5, 5])

setify([4, 4, 4, 4])

setify([5, 7, 8, 9, 10, 15])

setify([3, 3, 3, 2, 1])

O/P:

[1, 3, 3, 5, 5] ➞ [1, 3, 5]

[4, 4, 4, 4] ➞ [4]

[5, 7, 8, 9, 10, 15] ➞ [5, 7, 8, 9, 10, 15]

[3, 3, 3, 2, 1] ➞ [1, 2, 3]

#### 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.

I/P:

**def** mean(in\_num):

in\_list **=** [int(ele) **for** ele **in** str(in\_num)]

out\_num **=** sum(in\_list)**/**len(str(in\_num))

print(f'Mean of {in\_num} ➞ {out\_num:.0f}')

mean(42)

mean(12345)

mean(666)

O/P:

Mean of 42 ➞ 3

Mean of 12345 ➞ 3

Mean of 666 ➞ 6