**Question 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:-**

**def is\_symmetrical(num):**

**currentDigit = reversedDigit = 0**

**remainingNum = num**

**while(remainingNum != 0):**

**currentDigit = remainingNum % 10**

**reversedDigit = reversedDigit \* 10 + currentDigit**

**print('Reveresed Digit :',reversedDigit)**

**remainingNum = remainingNum // 10**

**if reversedDigit == num:**

**print('Num {} is symmetrical'.format(num))**

**else:**

**print('Num {} is not symmetrical'.format(num))**

**is\_symmetrical(7227)**

**Reveresed Digit : 7**

**Reveresed Digit : 72**

**Reveresed Digit : 722**

**Reveresed Digit : 7227**

**Num 7227 is symmetrical**

**is\_symmetrical(12567)**

**Reveresed Digit : 7**

**Reveresed Digit : 76**

**Reveresed Digit : 765**

**Reveresed Digit : 7652**

**Reveresed Digit : 76521**

**Num 12567 is not symmetrical**

**is\_symmetrical(44444444)**

**Reveresed Digit : 4**

**Reveresed Digit : 44**

**Reveresed Digit : 444**

**Reveresed Digit : 4444**

**Reveresed Digit : 44444**

**Reveresed Digit : 444444**

**Reveresed Digit : 4444444**

**Reveresed Digit : 44444444**

**Num 44444444 is symmetrical**

**is\_symmetrical(9939)**

**Reveresed Digit : 9**

**Reveresed Digit : 93**

**Reveresed Digit : 939**

**Reveresed Digit : 9399**

**Num 9939 is not symmetrical**

**is\_symmetrical(1112111)**

**Reveresed Digit : 1**

**Reveresed Digit : 11**

**Reveresed Digit : 111**

**Reveresed Digit : 1112**

**Reveresed Digit : 11121**

**Reveresed Digit : 111211**

**Reveresed Digit : 1112111**

**Num 1112111 is symmetrical**

**Question 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:-**

**def multiply\_nums(s):**

**s = s.replace(' ', "")**

**s = s.split(',')**

**sum = 1**

**for i in s:**

**sum = sum \* int(i)**

**return sum**

**multiply\_nums('2, 3')**

**6**

**multiply\_nums('1, 2, 3, 4')**

**24**

**multiply\_nums('54, 75, 453, 0')**

**0**

**multiply\_nums('10, -2')**

**-20**

**Question 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:-**

**def square\_digits(num):**

**z = ''.join(str(int(i)\*\*2) for i in str(num))**

**return int(z)**

**square\_digits(9119)**

**811181**

**square\_digits(2483)**

**416649**

**square\_digits(3212)**

**9414**

**Question 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:-**

**def setify(lst):**

**return list(set(lst))**

**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]**

**Question 5**

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

### Examples

**mean(42) ➞ 3**

**mean(12345) ➞ 3**

**mean(666) ➞ 6**

### Notes

* **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).**
* **The mean will always be an integer.**

**ANS:-**

**def mean(n):**

**N = len(str(n))**

**sum = mean = 0**

**for digit in str(n):**

**sum += int(digit)**

**return int(sum/N)**

**mean(42)**

**3**

**mean(12345)**

**3**

**mean(666)**

**6**