**Question1**

**Create a function that takes a string and returns a string in which each character is repeated once.**

**Examples**

**double\_char("String") ➞ "SSttrriinngg"**

**double\_char("Hello World!") ➞ "HHeellllooWWoorrlldd!!"**

**double\_char("1234!\_ ") ➞ "11223344!!\_\_ "**

**ANS:-**

**def double(str):**

**return ''.join([c+c for c in str])**

**print(double('akash'))**

**aakkaasshh**

**print(double('String'))**

**SSttrriinngg**

**print(double('Hello World!'))**

**HHeelllloo WWoorrlldd!!**

**print(double('1234!\_ '))**

**11223344!!\_\_**

**Question2**

**Create a function that reverses a boolean value and returns the string "boolean expected" if another variable type is given.**

### Examples

**reverse(True) ➞ False**

**reverse(False) ➞ True**

**reverse(0) ➞ "boolean expected"**

**reverse(None) ➞ "boolean expected"**

**ANS:-**

**def reverse(arg=None):**

**return not arg if type(arg) == bool else "boolean expected"**

**print(reverse(True)) *# False***

**print(reverse(False)) *# True***

**print(reverse(0)) *# "boolean expected"***

**print(reverse(None)) *# "boolean expected"***

**False**

**True**

**boolean expected**

**boolean expected**

**Question3**

**Create a function that returns the thickness (in meters) of a piece of paper after folding it n number of times. The paper starts off with a thickness of 0.5mm.**

### Examples

**num\_layers(1) ➞ "0.001m"**

**# Paper folded once is 1mm (equal to 0.001m)**

**num\_layers(4) ➞ "0.008m"**

**# Paper folded 4 times is 8mm (equal to 0.008m)**

**num\_layers(21) ➞ "1048.576m"**

**# Paper folded 21 times is 1048576mm (equal to 1048.576m)**

**ANS:-**

**def num\_layers(n):**

**thickness = 0.5**

**for \_ in range(n):**

**thickness \*= 2**

**return str(thickness / 1000)+'m' *# for meters***

**print(num\_layers(1))**

**print(num\_layers(4))**

**print(num\_layers(21))**

**0.001m**

**0.008m**

**1048.576m**

**Question4**

**Create a function that takes a single string as argument and returns an ordered list containing the indices of all capital letters in the string.**

### Examples

**index\_of\_caps("eDaBiT") ➞ [1, 3, 5]**

**index\_of\_caps("eQuINoX") ➞ [1, 3, 4, 6]**

**index\_of\_caps("determine") ➞ []**

**index\_of\_caps("STRIKE") ➞ [0, 1, 2, 3, 4, 5]**

**index\_of\_caps("sUn") ➞ [1]**

**ANS:-**

**def index\_of\_caps(word):**

**indices = []**

**for i in range(len(word)):**

**if word[i].isupper():**

**indices.append(i)**

**return indices**

**print(index\_of\_caps('BhaNu'))**

**print(index\_of\_caps('eDaBiT'))**

**print(index\_of\_caps('eQuINoX'))**

**print(index\_of\_caps('determine'))**

**print(index\_of\_caps('STRIKE'))**

**print(index\_of\_caps('sUn'))**

**[0, 3]**

**[1, 3, 5]**

**[1, 3, 4, 6]**

**[]**

**[0, 1, 2, 3, 4, 5]**

**[1]**

**Question5**

**Using list comprehensions, create a function that finds all even numbers from 1 to the given number.**

### Examples

**find\_even\_nums(8) ➞ [2, 4, 6, 8]**

**find\_even\_nums(4) ➞ [2, 4]**

**find\_even\_nums(2) ➞ [2]**

**ANS:-**

**def find\_even\_nums(n):**

**even =[x for x in range(2,n+1) if x % 2 == 0]**

**return even**

**n = int(input('Enter a number : '))**

**find\_even\_nums(n)**

**Enter a number : 10**

**[2, 4, 6, 8, 10]**

**find\_even\_nums(8)**

**[2, 4, 6, 8]**

**find\_even\_nums(4)**

**[2, 4]**

**find\_even\_nums(2)**

**[2]**