**Python Programming Basic Assignment-19**

**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!") ➞ "HHeelllloo WWoorrlldd!!"

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

In [3]:

**import** logging **as** lg

lg**.**basicConfig(filename**=**'log\_repeat.log',level**=**lg**.**INFO)

**def** double\_char(string):

new**=**''

**try**:

**for** i **in** string:

new**+=**i**+**i

**except** Exception **as** e:

lg**.**exception(str(e))

lg**.**info(f'Successfull. Value is {new}')

**return** new

In [4]:

double\_char('String')

Out[4]:

'SSttrriinngg'

In [5]:

double\_char("Hello World!")

Out[5]:

'HHeelllloo WWoorrlldd!!'

In [7]:

double\_char("1234!\_ ")

Out[7]:

'11223344!!\_\_ '

In [11]:

lg**.**shutdown()

**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"

In [47]:

**def** reverse(a):

**if** type(a)**==**bool:

**return** **not** a

**else**:

**return** 'boolean expected'

In [48]:

reverse(**False**)

Out[48]:

True

In [49]:

reverse(**True**)

Out[49]:

False

In [50]:

reverse(0)

Out[50]:

'boolean expected'

In [51]:

reverse(**None**)

Out[51]:

'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)**

In [106]:

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

t**=**0.5

**for** i **in** range(n):

t**=**t**\***2

**return** t**/**1000

In [107]:

num\_layers(1)

Out[107]:

0.001

In [108]:

num\_layers(4)

Out[108]:

0.008

In [109]:

num\_layers(21)

Out[109]:

1048.576

**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] m 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]

In [1]:

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

*# l=sorted(l)*

l1**=**[]

**for** i **in** l:

**if** i**.**isupper():

l1**.**append(l**.**index(i))

**return** l1

In [2]:

index\_of\_caps('eDaBiT')

Out[2]:

[1, 3, 5]

In [3]:

index\_of\_caps('eQuINoX')

Out[3]:

[1, 3, 4, 6]

In [4]:

index\_of\_caps('determine')

Out[4]:

[]

In [5]:

index\_of\_caps('STRIKE')

Out[5]:

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

In [6]:

index\_of\_caps('sUn')

Out[6]:

[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]

In [12]:

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

l**=**[i **for** i **in** range(1,num**+**1) **if** i**%2**==0]

**return** l

In [13]:

find\_even\_nums(8)

Out[13]:

[2, 4, 6, 8]

In [14]:

find\_even\_nums(4)

Out[14]:

[2, 4]

In [15]:

find\_even\_nums(2)

Out[15]:

[2]