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!!\_\_ "

ANS: def double\_char(string):

doubled\_string = ""

for char in string:

doubled\_string += char \* 2

return doubled\_string

# Test cases

print(double\_char("String")) # Output: "SSttrriinngg"

print(double\_char("Hello World!")) # Output: "HHeelllloo WWoorrlldd!!"

print(double\_char("1234!\_ ")) # Output: "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):

if type(arg) is bool:

return not arg

else:

return "boolean expected"

# Test cases

print(reverse(True)) # Output: False

print(reverse(False)) # Output: True

print(reverse(0)) # Output: "boolean expected"

print(reverse(None)) # Output: "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: To calculate the thickness of the paper after folding it 'n' number of times, we need to keep multiplying the initial thickness (0.5mm) by 2 for each fold. After that, we need to convert the final thickness from millimeters to meters (1m = 1000mm).

Here's the function in Python:

def num\_layers(n):

if n < 0:

return "Number of folds should be a positive integer."

initial\_thickness\_mm = 0.5

final\_thickness\_mm = initial\_thickness\_mm \* (2 \*\* n)

final\_thickness\_m = final\_thickness\_mm / 1000

return "{:.3f}m".format(final\_thickness\_m)

# Test cases

print(num\_layers(1)) # Output: "0.001m"

print(num\_layers(4)) # Output: "0.008m"

print(num\_layers(21)) # Output: "1048.576m"

The function calculates the final thickness in millimeters by multiplying the initial thickness (0.5mm) by 2 raised to the power of 'n' (`2 \*\* n`). It then converts the final thickness from millimeters to meters and formats the output string to show three decimal places.

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(s):

return [index for index, char in enumerate(s) if char.isupper()]

# Test cases

print(index\_of\_caps("eDaBiT")) # Output: [1, 3, 5]

print(index\_of\_caps("eQuINoX")) # Output: [1, 3, 4, 6]

print(index\_of\_caps("determine"))# Output: []

print(index\_of\_caps("STRIKE")) # Output: [0, 1, 2, 3, 4, 5]

print(index\_of\_caps("sUn")) # Output: [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):

return [num for num in range(1, n + 1) if num % 2 == 0]

# Test cases

print(find\_even\_nums(8)) # Output: [2, 4, 6, 8]

print(find\_even\_nums(4)) # Output: [2, 4]

print(find\_even\_nums(2)) # Output: [2]