#### Create a function that takes a list of strings and integers, and filters out the list so that it returns a list of integers only.

**Examples:**  
filter\_list([1, 2, 3, "a", "b", 4]) ➞ [1, 2, 3, 4]  
filter\_list(["A", 0, "Edabit", 1729, "Python", "1729"]) ➞ [0, 1729]  
filter\_list(["Nothing", "here"]) ➞ []

I/P:

def filter\_list(in\_list):

out\_list **=** []

**for** ele **in** in\_list:

**if** type(ele) **==** int:

out\_list**.**append(ele)

print(f'Output ➞ {out\_list}')

filter\_list([1, 2, 3, "a", "b", 4])

filter\_list(["A", 0, "Edabit", 1729, "Python", "1729"])

filter\_list(["Nothing", "here"])

O/P:

Output ➞ [1, 2, 3, 4]

Output ➞ [0, 1729]

Output ➞ []

#### Given a list of numbers, create a function which returns the list but with each element's index in the list added to itself. This means you add 0 to the number at index 0, add 1 to the number at index 1, etc...

**Examples:**  
add\_indexes([0, 0, 0, 0, 0]) ➞ [0, 1, 2, 3, 4]  
add\_indexes([1, 2, 3, 4, 5]) ➞ [1, 3, 5, 7, 9]  
add\_indexes([5, 4, 3, 2, 1]) ➞ [5, 5, 5, 5, 5]

I/P:

**def** add\_indexes(in\_list):

out\_list **=** []

**for** ele **in** range(len(in\_list)):

out\_list**.**append(ele**+**in\_list[ele])

print(f'{in\_list} ➞ {out\_list}')

add\_indexes([0, 0, 0, 0, 0])

add\_indexes([1, 2, 3, 4, 5])

add\_indexes([5, 4, 3, 2, 1])

O/P:

[0, 0, 0, 0, 0] ➞ [0, 1, 2, 3, 4]

[1, 2, 3, 4, 5] ➞ [1, 3, 5, 7, 9]

[5, 4, 3, 2, 1] ➞ [5, 5, 5, 5, 5]

#### Create a function that takes the height and radius of a cone as arguments and returns the volume of the cone rounded to the nearest hundredth. See the resources tab for the formula.



**Examples:**  
cone\_volume(3, 2) ➞ 12.57  
cone\_volume(15, 6) ➞ 565.49  
cone\_volume(18, 0) ➞ 0

I/P:

**import** math

**def** cube\_volume(height, radius):

output **=** ((math**.**pi)**\***pow(radius,2))**\***(height**/**3)

print(f'Output ➞ {output:.2f}')

cube\_volume(3,2)

cube\_volume(15,6)

cube\_volume(18,0)

O/P:

Output ➞ 12.57

Output ➞ 565.49

Output ➞ 0.00

#### This Triangular Number Sequence is generated from a pattern of dots that form a triangle.

The first 5 numbers of the sequence, or dots, are: 1, 3, 6, 10, 15  
This means that the first triangle has just one dot, the second one has three dots, the third one has 6 dots and so on. Write a function that gives the number of dots with its corresponding triangle number of the sequence.

**Examples:**  
triangle(1) ➞ 1  
triangle(6) ➞ 21  
triangle(215) ➞ 23220

I/P:

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

print(f'Output ➞ {int((in\_num)**\***((in\_num**+**1)**/**2))}')

triangle(1)

triangle(6)

triangle(215)

O/P:

Output ➞ 1

Output ➞ 21

Output ➞ 23220

#### Create a function that takes a list of numbers between 1 and 10 (excluding one number) and returns the missing number.

**Examples:**  
missing\_num([1, 2, 3, 4, 6, 7, 8, 9, 10]) ➞ 5  
missing\_num([7, 2, 3, 6, 5, 9, 1, 4, 8]) ➞ 10  
missing\_num([10, 5, 1, 2, 4, 6, 8, 3, 9]) ➞ 7

I/P:

**def** missing\_num(in\_list):

**for** i **in** range(1,11):

**if** i **not** **in** in\_list:

print(f'{in\_list} ➞ {i}')

missing\_num([1, 2, 3, 4, 6, 7, 8, 9, 10])

missing\_num([7, 2, 3, 6, 5, 9, 1, 4, 8])

missing\_num([10, 5, 1, 2, 4, 6, 8, 3, 9])

O/P:

[1, 2, 3, 4, 6, 7, 8, 9, 10] ➞ 5

[7, 2, 3, 6, 5, 9, 1, 4, 8] ➞ 10

[10, 5, 1, 2, 4, 6, 8, 3, 9] ➞ 7