Programming_Assingment20

Question1

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']) ➞ []

In [1]:
```
lst = [1, 2, 3, 'a', 'b', 4]
def filter_list(lst):
    intLst = []
    for i in lst:
        if type(i) == int:
            intLst.append(i)
    return intLst
```

In [2]:
```
filter_list([1, 2, 3, 'a', 'b', 4])
```

Out[2]:
```
[1, 2, 3, 4]
```

In [3]:
```
filter_list(['A', 0, 'Edabit', 1729, 'Python', '1729'])
```

Out[3]:
```
[0, 1729]
```

In [4]:
```
filter_list(['Nothing', 'here'])
```

Out[4]:
```
[]
```

Question2

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]

In [5]:

```
def add_indexes(lst):
    ind = 0
    index = []
    for i in lst:
        index.append(lst.index(i,ind) + i)
        ind+=1
    return index
```

In [6]:

```
add_indexes([0, 0, 0, 0, 0])
```

Out[6]:

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

In [7]:

```
add_indexes([1, 2, 3, 4, 5])
```

Out[7]:

```
[1, 3, 5, 7, 9]
```

In [8]:

```
add_indexes([5, 4, 3, 2, 1])
```

Out[8]:

```
[5, 5, 5, 5, 5]
```

Question3

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

In [9]:

```
import math
pi = math.pi

# Function to calculate Volume of Cone
def cone_volume(r, h):
    return round((1 / 3) * pi * r * r * h)


# Driver Code
radius = float(5)
height = float(12)

print( "Volume Of Cone : ", cone_volume(radius, height) )
```

```
Volume Of Cone :  314
```

In [10]:

```
cone_volume(3, 2)
```

Out[10]:

```
19
```

In [11]:

```
cone_volume(15, 6)
```

Out[11]:

```
1414
```

In [12]:

```
cone_volume(18, 0)
```

Out[12]:

```
0
```

Question4

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

A Rule

We can make a "Rule" so we can calculate any triangular number.

First, rearrange the dots like this:

triangular numbers 1 to 5

Then double the number of dots, and form them into a rectangle:

triangular numbers when doubled become n by n+1 rectangles

Now it is easy to work out how many dots: just multiply n by n+1

Dots in rectangle = n(n+1)

But remember we doubled the number of dots, so

Dots in triangle = n(n+1)/2

We can use xn to mean "dots in triangle n", so we get the rule:

Rule: xn = n(n+1)/2

In [13]:

```
def triangle(n):
    return n*(n+1)*0.5

n = int(input('Enter the trinalge number :'))
print("The {}th triangle has {} dots ".format(n,int(triangle(n))))
```

```
Enter the trinalge number :234
The 234th triangle has 27495 dots
```

In [14]:

```
triangle(215)
```

Out[14]:

```
23220.0
```

In [15]:

```
 triangle(1)
```

Out[15]:

```
1.0
```

Question5

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

In [16]:

```
def missing_num(lst):
    total = sum([x for x in range(11)])
    sum_Of_list = sum(lst)
    return total - sum_Of_list

print(missing_num([1, 2, 3, 4, 6, 7, 8, 9, 10]))
```

```
5
```

In [17]:

```
missing_num([7, 2, 3, 6, 5, 9, 1, 4, 8])
```

Out[17]:

```
10
```

In [18]:

```
missing_num([10, 5, 1, 2, 4, 6, 8, 3, 9])
```

Out[18]:

```
7
```