1. 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"]) → []
Ans:

In [2]:
```
def filter_list(string):
    output = []
    for i in string:
        if type(i) == int:
            output.append(i)
    print(f'filter_list({string}) → {output}')
filter_list([1, 2, 3, 'a', 'b', 4])
filter_list(['A', 0, 'Edabit', 1729, 'Python', '1729'])
filter_list(['Nothing', 'here'])
```

```
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']) → []
```

2. 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]
Ans:

In [3]:
```
def add_indexes(string):
    output = []
    for i in range(len(string)):
        output.append(i+string[i])
    print(f'add_indexes({string}) → [{output}]')
add_indexes([0, 0, 0, 0, 0])
add_indexes([1, 2, 3, 4, 5])
add_indexes([5, 4, 3, 2, 1])
```

```
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]]
```

3. 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
Ans:

In [4]:
```
import math
def cube_volume(height, radius):
    output = ((math.pi)*pow(radius,2))*(height/3)
    print(f'cone_volume({height}, {radius}) ⟶ {output}')
cube_volume(3,2)
cube_volume(15,6)
cube_volume(18,0)
```

```
cone_volume(3, 2) ⟶ 12.566370614359172
cone_volume(15, 6) ⟶ 565.4866776461628
cone_volume(18, 0) ⟶ 0.0
```

4. 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.

Ans:

In [5]:
```
def triangle(n):
    Output = n*((n+1)/2)
    print(f'triangle({n}) ⟶ {Output}')
triangle(1)
triangle(6)
triangle(215)
```

```
triangle(1) ⟶ 1.0
triangle(6) ⟶ 21.0
triangle(215) ⟶ 23220.0
```

5. 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

Ans:

In [6]:
```
def missing_num(string):
    for i in range(1,11):
        if i not in string:
            print(f'missing_num({string}) ⟶ {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])
```

```
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
```