1. Create a function that takes a list and string. The function should remove the letters in the string from the list, and return the list.

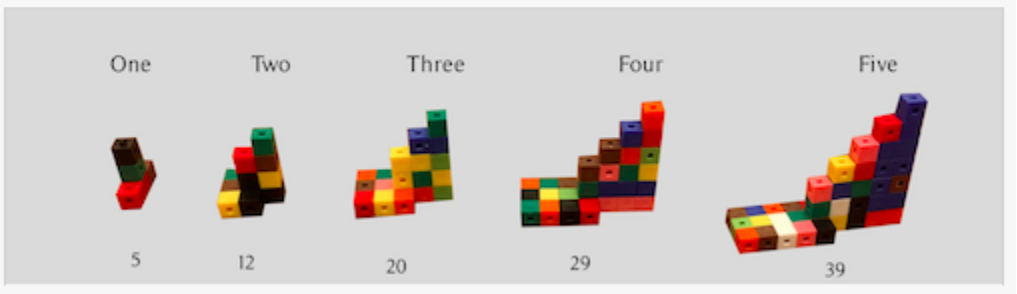
Examples:
remove_letters(["s", "t", "r", "i", "n", "g", "w"], "string") → ["w"]
remove_letters(["b", "b", "l", "l", "g", "n", "o", "a", "w"], "balloon") → ["b", "g", "w"]
remove_letters(["d", "b", "t", "e", "a", "i"], "edabit") → []

Ans:

In [2]:
```
def remove_letters(in_list,in_string):
    stringlist = []
    listcopy = in_list.copy()
    for i in in_string:
        stringlist.append(i)
    for i in stringlist:
        if i in in_list:
            in_list.remove(i)
    print(f'remove_letters({listcopy}, {in_string}) → {in_list}')
remove_letters(["s", "t", "r", "i", "n", "g", "w"], "string")
remove_letters(["b", "b", "l", "l", "g", "n", "o", "a", "w"], "balloon")
remove_letters(["d", "b", "t", "e", "a", "i"], "edabit")
```

```
remove_letters(['s', 't', 'r', 'i', 'n', 'g', 'w'], string) → ['w']
remove_letters(['b', 'b', 'l', 'l', 'g', 'n', 'o', 'a', 'w'], balloon) → ['b', 'g', 'w']
remove_letters(['d', 'b', 't', 'e', 'a', 'i'], edabit) → []
```

2. A block sequence in three dimensions. We can write a formula for this one:



Create a function that takes a number (step) as an argument and returns the amount of blocks in that step.

Examples:
blocks(1) → 5
blocks(5) → 39
blocks(2) → 12

Ans:

In [4]:
```
def blocks(num):
    depth = num*3+(num-1)
    height = [x for x in range(2,num+2)]
    print(f'blocks({num}) → {depth+sum(height)}')
blocks(1)
blocks(5)
blocks(2)
```

```
blocks(1) → 5
blocks(5) → 39
blocks(2) → 12
```

3. Create a function that subtracts one positive integer from another, without using any arithmetic operators such as -, %, /, +, etc.

Examples:
my_sub(5, 9) → 4
my_sub(10, 30) → 20
my_sub(0, 0) → 0
Ans:

In [7]:
```
from operator import sub
def my_sub(num1,num2):
    output = 0
    if num1 >= num2:
        output = sub(num1,num2)
    else:
        output = sub(num2,num1)
    print(f'my_sub{num1,num2} → {output}')
my_sub(5, 9)
my_sub(10, 30)
my_sub(0, 0)
```

```
my_sub(5, 9) → 4
my_sub(10, 30) → 20
my_sub(0, 0) → 0
```

4. Create a function that takes a string containing money in dollars and pounds sterling (seperated by comma) and returns the sum of dollar bills only, as an integer.

For the input string:

Each amount is prefixed by the currency symbol: $ for dollars and £ for pounds.
Thousands are represented by the suffix k. i.e. $4k = $4,000 and £40k = £40,000
Examples:
add_bill("d20,p40,p60,d50") → 20 + 50 = 70
add_bill("p30,d20,p60,d150,p360") → 20 + 150 = 170
add_bill("p30,d2k,p60,d200,p360") → 2 * 1000 + 200 = 2200

Ans:

In [9]:

```python
def add_bill(in_string):
    output = 0
    for i in in_string.split(","):
        if 'd' in i:
            if 'k' in i:
                output += int(i.replace('d','').replace('k',''))*1000
            else:
                output += int(i.replace("d",''))
    print(f'add_bill({in_string}) → {output}')
add_bill("d20,p40,p60,d50")
add_bill("p30,d20,p60,d150,p360")
add_bill("p30,d2k,p60,d200,p360")
```

```
add_bill(d20,p40,p60,d50) → 70
add_bill(p30,d20,p60,d150,p360) → 170
add_bill(p30,d2k,p60,d200,p360) → 2200
```

5. Create a function that flips a horizontal list into a vertical list, and a vertical list into a horizontal list.

In other words, take an 1 x n list (1 row + n columns) and flip it into a n x 1 list (n rows and 1 column), and vice versa.

Examples:
flip_list([1, 2, 3, 4]) → [[1], [2], [3], [4]] # Take a horizontal list and flip it vertical.
flip_list([[5], [6], [9]]) → [5, 6, 9] # Take a vertical list and flip it horizontal.
flip_list([]) → []

Ans:

In [12]:

```python
def flip_list(in_list):
    if len(in_list) > 0:
        output = [ele[0] for ele in in_list] if isinstance(in_list[0],list) else [[ele] for ele in in_list]
    else:
        output = []
    print(f'flip_list({in_list}) → {output}')
flip_list([1,2,3,4])
flip_list([[5],[6],[9]])
flip_list([])
```

```
flip_list([1, 2, 3, 4]) → [[1], [2], [3], [4]]
flip_list([[5], [6], [9]]) → [5, 6, 9]
flip_list([]) → []
```