1. Implement a class iterator to flatten a nested list of lists of integers. Each list element is either an integer or a list. There can be many levels of nested lists in lists. The class initializes with a nested list. It also has two methods:
1. next() returns an integer in the order of appearance.
2. hasNext() returns True / False regarding if all integers have been retrieved or not.
Write the Class implementation for three required methods.
Examples
ni, actual = NestedIterator([[1, 1], 2, [1, 1]]), []
while ni.hasNext():
actual.append(ni.next())
actual → [1, 1, 2, 1, 1]
ni, actual = NestedIterator([1, [4, [6]]]), []
while ni.hasNext():
actual.append(ni.next())
actual → [1, 4, 6]
ni, actual = NestedIterator([[[]], []]), []
while ni.hasNext():
actual.append(ni.next())
actual → []

Ans:

In [4]:

```
class NestedIterator:
    def __init__(self,in_list):
        self.list = in_list
        self.flatten_list = []
        self.test(self.list)

    def test(self,in_list):
        for ele in in_list:
            if isinstance(ele,int):
                self.flatten_list.append(ele)
            else:
                self.test(ele)

    def hasNext(self):
        return True if len(self.flatten_list) > 0 else False

    def next(self):
        return self.flatten_list.pop(0)

ni, actual = NestedIterator([[1, 1], 2, [1, 1]]), []
while ni.hasNext():
    actual.append(ni.next())
print(f'actual → {actual}')

ni, actual = NestedIterator([1, [4, [6]]]), []
while ni.hasNext():
    actual.append(ni.next())
print(f'actual → {actual}')

ni, actual = NestedIterator([[[]], []]), []
while ni.hasNext():
    actual.append(ni.next())
print(f'actual → {actual}')
```

```
actual → [1, 1, 2, 1, 1]
actual → [1, 4, 6]
actual → []
```

2. Given a 3x3 matrix of a completed tic-tac-toe game, create a function that returns whether the game is a win for "X", "O", or a "Draw", where "X" and "O" represent themselves on the matrix, and "E" represents an empty spot.

Examples:

```
tic_tac_toe([
  ["X", "O", "X"],
  ["O", "X",  "O"],
  ["O", "X",  "X"]
]) → "X"

tic_tac_toe([
  ["O", "O", "O"],
  ["O", "X", "X"],
  ["E", "X", "X"]
]) → "O"

tic_tac_toe([
  ["X", "X", "O"],
  ["O", "O", "X"],
  ["X", "X", "O"]
]) → "Draw"
```

Ans:

In [1]:

```
def tic_tac_toe(in_list):
    output = None
    #horizontal match
    for ele in in_list:
        if len(list(set(ele))) == 1:
            output = list(set(ele))[0]
            break
    #vertical match
    if output == None:
        for i in range(len(in_list)):
            temp = []
            for j in range(len(in_list)): temp.append(in_list[j][i])
            if len(list(set(temp))) == 1: output = list(set(temp))[0]
    #diagonal match
    if output == None:
        temp = []
        for ele in [0,1,2]:
            temp.append(in_list[ele][ele])
        if len(list(set(temp))) == 1: output = list(set(temp))[0]
    #reverse diagonal match
    if output == None:
        temp = []
        for i in [0,1,2]:
            for j in [0,1,2]:
                if sum([i,j]) == 2: temp.append(in_list[i][j])
        if len(list(set(temp))) == 1: output = list(set(temp))[0]
    #Draw Condition
    if output == None: output = 'Draw'
    print(f'tic_tac_toe({in_list}) → "{output}"')

```

```
tic_tac_toe([["X", "O", "X"],["O", "X", "O"],["O", "X", "X"]])
tic_tac_toe([["O", "O", "O"],["O","X", "X"],["E", "X", "X"]])
tic_tac_toe([["X", "X", "O"],["O", "O", "X"],["X", "X", "O"]])
tic_tac_toe([["X", "X", "O"],["X", "O", "X"],["X", "O", "O"]])
tic_tac_toe([["O", "O", "X"],["O", "X", "O"],["X", "O", "O"]])
```

```
tic_tac_toe([['X', 'O', 'X'], ['O', 'X', 'O'], ['O', 'X', 'X']]) → "X"
tic_tac_toe([['O', 'O', 'O'], ['O', 'X', 'X'], ['E', 'X', 'X']]) → "O"
tic_tac_toe([['X', 'X', 'O'], ['O', 'O', 'X'], ['X', 'X', 'O']]) → "Draw"
tic_tac_toe([['X', 'X', 'O'], ['X', 'O', 'X'], ['X', 'O', 'O']]) → "X"
tic_tac_toe([['O', 'O', 'X'], ['O', 'X', 'O'], ['X', 'O', 'O']]) → "X"
```

3. Your computer might have been infected by a virus! Create a function that finds the viruses in files and removes them from your computer.

Examples:
remove_virus("PC Files: spotifysetup.exe, virus.exe, dog.jpg") → "PC Files: spotifysetup.exe, dog.jpg"
remove_virus("PC Files: antivirus.exe, cat.pdf, lethalmalware.exe, dangerousvirus.exe ") → "PC Files: antivirus.exe, cat.pdf" remove_virus("PC Files: notvirus.exe, funnycat.gif") → "PC Files: notvirus.exe, funnycat.gif")

Ans:

In [3]:

```
def remove_virus(string):
    import re
    in_list = [x.strip() for x in re.split(", ",string)]
    output = []
    for ele in in_list:
        if ele not in ['virus.exe','dangerousvirus.exe','lethalmalware.exe']:
            output.append(ele)
    print(f'remove_virus({string}) ⟶ "{", ".join(output)}"')
remove_virus("PC Files: spotifysetup.exe, virus.exe, dog.jpg")
remove_virus("PC Files: antivirus.exe, cat.pdf, lethalmalware.exe, dangerousvirus.exe ")
remove_virus("PC Files: notvirus.exe, funnycat.gif")
```

```
remove_virus(PC Files: spotifysetup.exe, virus.exe, dog.jpg) ⟶ "PC Files: spotifysetup.exe, dog.jpg"
remove_virus(PC Files: antivirus.exe, cat.pdf, lethalmalware.exe, dangerousvirus.exe ) ⟶ "PC Files: antivirus.exe, cat.pdf"
remove_virus(PC Files: notvirus.exe, funnycat.gif) ⟶ "PC Files: notvirus.exe, funnycat.gif"
```

4. In a video game, a meteor will fall toward the main character's home planet. Given the meteor's trajectory as a string in the form y = mx + b and the character's position as a tuple of (x, y), return True if the meteor will hit the character and False if it will not.

Examples:
will_hit("y = 2x - 5", (0, 0)) ⟶ False
will_hit("y = -4x + 6", (1, 2)) ⟶ True
will_hit("y = 2x + 6", (3, 2)) ⟶ False

Ans:

In [28]:

```
def will_hit(string,pos):
    import re
    in_list = string.split(" ")
    b = int(in_list[3]+in_list[4])
    m = int(re.findall(r'-?\d+', in_list[2])[0])
    output = False
    if pos[1] == m*pos[0] + b:
        output = True
    print(f'will_hit({string,pos}) ⟶ {output}')
will_hit("y = 2x - 5", (0, 0))
will_hit("y = -4x + 6", (1, 2))
will_hit("y = 2x + 6", (3, 2))
```

```
will_hit(('y = 2x - 5', (0, 0))) ⟶ False
will_hit(('y = -4x + 6', (1, 2))) ⟶ True
will_hit(('y = 2x + 6', (3, 2))) ⟶ False
```