**Pattern Recognition**

Pattern: Similarity between problems.

Patten Recognition: Finding similarities among small, decomposed problems.

The same problem-solving solution can be used wherever a pattern exists.

The More patterns we find, the easier and quicker the problem-solving.

Pattern Recognition

1. Find Common elements.

2. Find Common differences.

3. Find Individual Elements

4. Describe the Identified Patterns.

5. Make prediction based on Patterns.

**Iterative Accumulation**

Basically, add the final value of each iteration.

There are 3 things to identify:

* result variable
* for loop
* target of iteration

eg1.

1 + 1/2 + 1/3 + … + 1/10

it1 = 1

it2 = 1/2

…

It10 = 1/10

n = 10

result = 0

for i in range(1,n+1):

result += 1/i

print(result)

eg2.

1/(1\*2) + 1/(2\*3) + 1/(3\*4) + … + 1/(10\*11)

It1 = 1/(1\*2)

It2 = 1/(2\*3)

…

It10 = 1/(10\*11)

n = 10

result = 0

for i in range(1,n+1):

result += 1/(i\*(i+1)) or 1/(i\*\*2 + i)

print(result)

eg3.

1/9 + 2/8 + … + 9/1

It1 = 1/9

…

It10 = 9/1

for I in range (1, n+1):

result += i/(10-i)

eg4.

3/5 + 4/6 + … +12/14

It1: 3/5

It2: 4/6

…

It10: 12/14

for i in range(1,n+1):

result += (i+2)/(i+4)

eg5. Add and Subtract

1 - 2 + 3 - 4 +…+ 9 – 10

It1: 1

It2: -2

It3: 3

…

It9: 9

It10: -10

for i in range(1,n+1):

if i%2 == 0:

i = -i

result += i [or just: result -= i \* (-1\*\*i)]

eg.6

+0/1

-1/2

…

+8/9

-9/10

for i in range(1,n+1):

result += (-1\*\*i)((i-1)/i)