Hash Map or Hash Table

Hash Tables, also known as Hash Maps are a type of data structure which can be used to store or retrieve data in **O(1)**. Like the other data structures, their working is also complex. In abstract, they are like python dictionaries. For better understanding, let us look at a LeetCode problem.

Problem 202. Happy Number

In this problem, we'll be given an integer *n* and we have to keep adding the square of its digits. And keep doing this task with the resulting integer till we get result as 1. This process might result in an endless loop.

Example – 19

$1^2 + 9^2 = 82$
$8^2 + 2^2 = 68$
$6^2 + 8^2 = 100$
$1^2 + 0^2 + 0^2 = 1$

<u>Logic –</u> We will keep mapping the integer to its square of digits in our Hash Table until the integer doesn't exist in our Hash Table or its sum results in 1.

If it does exist already, it means this is an endless loop.
Cause we already added the square of its digits and if the sum again goes to the original integer, then it's simply a loop.

<u>In the end, we will return True if we get sum as 1 otherwise False after the detection of loop.</u>

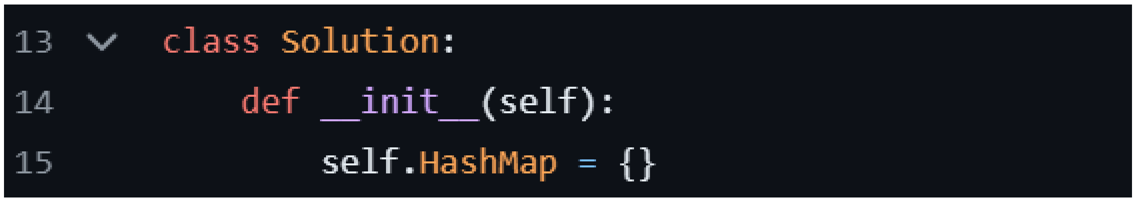
Our Hash Table will look like this for a particular example.

{ 19 : 82
82 : 68
68 : 100 }

It will stop mapping to 68 : 100 cause sum of square of digits in 100 is 1.

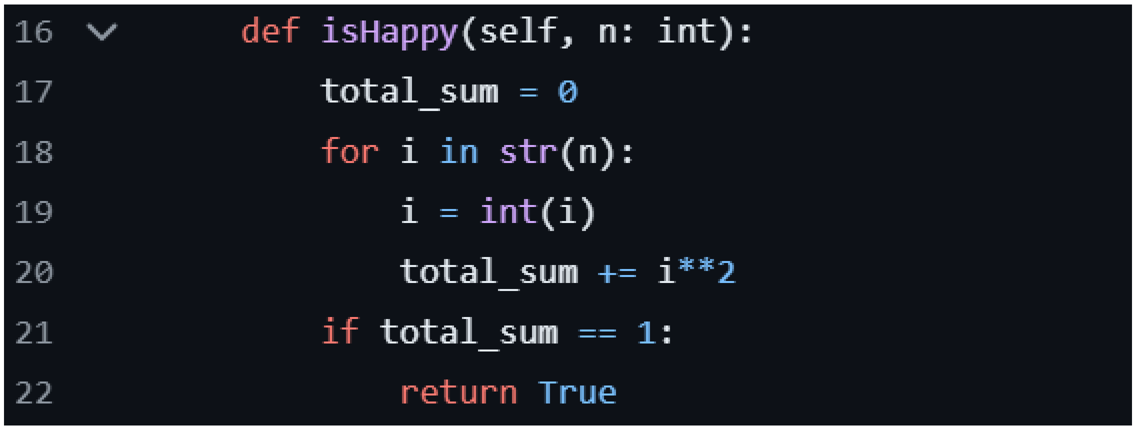
Now, let's look at the code for practical implementation.

Code -



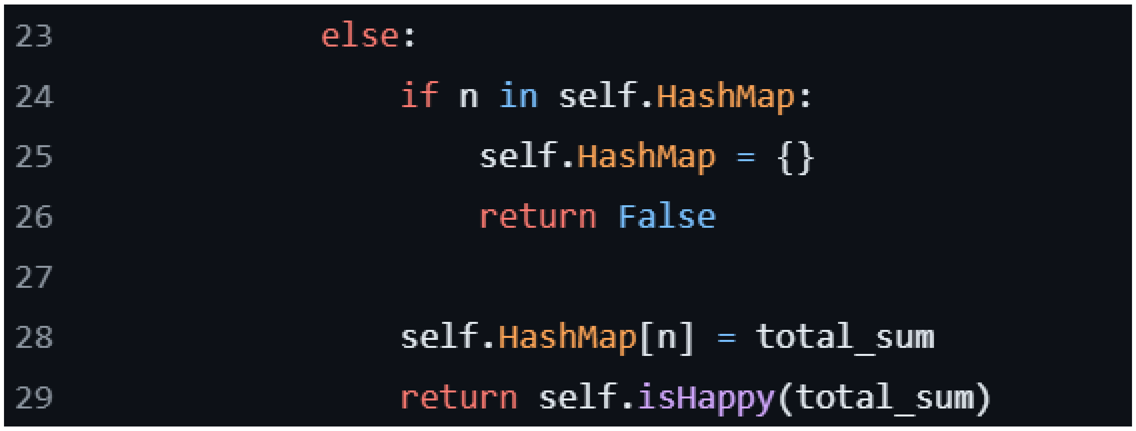
```
13  class Solution:
14      def __init__(self):
15          self.HashMap = {}
```

- We will begin by initializing our class and Hash Table.



```
16      def isHappy(self, n: int):
17          total_sum = 0
18          for i in str(n):
19              i = int(i)
20              total_sum += i**2
21          if total_sum == 1:
22              return True
```

- Then, we will create our function to write the logic which we've discussed above.
- In line 17, total_sum represents the sum of squares of digits of the integer.
- Then, we will use a for loop to iter over every digit of n as string. And in line 19 and 20, we will convert the digit to integer and add up the it's square to total_sum.
- And then we will check if total_sum is 1 or not. If yes, then we will return the function with True.



```
23            else:
24                if n in self.HashMap:
25                    self.HashMap = {}
26                    return False
27
28                self.HashMap[n] = total_sum
29                return self.isHappy(total_sum)
```

- What if total_sum is not 1? In that case, in line 24 we will check if the integer already exists in Hash Table or not. If it exists, then we will empty set Hash Table to empty in line 25 for testing other data sets in the same class and return the function with False in line 26.
  Lines 24 to 26 are used for the infinite loop detection.

- Now, if total_sum is not 1 and it doesn't already exist in the Hash Table, then we will map the integer n to its corresponding sum in line 28.
- And at the end, we will call this function recursively by replacing the integer n with total_sum in line 29 to do the same process with total_sum like we did with the integer until it returns True or False.