# Python Assignment day-2

**NAME:- Sushant Kumar Singh**
**Email:- ssushant886@gmail.com**

---

## Qustion:- 1

**Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.**
**This is a famous Google interview question, also being asked by many other companies now a days.**
**Consider the following dictionary**
**{ i, like, sam, sung, samsung, mobile, ice,**
**cream, icecream, man, go, mango}**
**Input: ilike**
**Output: Yes**
**The string can be segmented as "i like".**
**Input: ilikesamsung**
**Output: Yes**
**The string can be segmented as "i like samsung"**
**or "i like sam sung".**

## Answer:-

```
def word_break(word, dictionary):
    size = len(word)
    return size == 0 or any(word[:i] in dictionary and word_break(word[i:size], dictionary) for i in range(1, size + 1))

dictionary = set(["mobile", "samsung", "sam", "sung", "man", "mango", "icecream", "and", "go", "i", "like", "ice", "cream"])

# Sample input cases given as in qustion number 1
print("Yes" if word_break("ilikesamsung", dictionary) else "No")
print("Yes" if word_break("iiiiiiii", dictionary) else "No")
print("Yes" if word_break("", dictionary) else "No")
print("Yes" if word_break("ilikelikeimangoiii", dictionary) else "No")
print("Yes" if word_break("samsungandmango", dictionary) else "No")
print("Yes" if word_break("samsungandmangok", dictionary) else "No")
```

## Question-2

**A number can always be represented as a sum of squares of other numbers. Note that 1 is a square and we can always break a number as (1*1 + 1*1 + 1*1 + …). Given a number n, find the minimum number of squares that sum to X.**
**Examples :**
**Input: n = 100**
**Output: 1**
**Explanation:**
**100 can be written as 102. Note that 100 can also be written as 52 + 52 + 52 + 52, but this representation requires 4 squares.**
**Input: n = 6**
**Output: 3**

## Answers:-

```
def get_min_squares(n):
   if n <= 3:
      return n

   return min(1 + get_min_squares(n - x * x) for x in range(1, n + 1) if x * x <= n)

user_input = int(input("Enter a number: "))

result = get_min_squares(user_input)
print(f"Minimum number of squares to sum to {user_input}: {result}")
```

## Question-3

**Given a number N, the task is to check if it is divisible by 7 or not.**
**Note: You are not allowed to use the modulo operator, floating point arithmetic is also not allowed.**
**Naive approach: A simple method is repeated subtraction. Following is another interesting method.**
**Divisibility by 7 can be checked by a recursive method. A number of the form 10a + b is divisible by 7 if and only if a – 2b is divisible by 7. In other words, subtract twice the last digit from the number formed by the remaining digits. Continue to do this until a small number.**
**Example: the number 371: 37 – (2×1) = 37 – 2 = 35; 3 – (2 × 5) = 3 – 10 = -7; thus, since -7 is divisible by 7, 371 is divisible by 7.**

## Answer:-

```
def is_divisible_by_7(num):
    if num < 0:
        return is_divisible_by_7(-num)

    if num == 0 or num == 7:
        return True

    if num < 10:
        return False

    return is_divisible_by_7(num // 10 - 2 * (num - num // 10 * 10))



user_input = int(input("Enter a number: "))


if is_divisible_by_7(user_input):
    print("Divisible")
else:
    print("Not Divisible")
```

## Question-4

**Find the n'th term in Look-and-say (Or Count and Say) Sequence. The look-and-say sequence is the sequence of the below integers:**
**1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, …**
**How is the above sequence generated?**
**n'th term is generated by reading (n-1)'th term.**
**The first term is "1"**
**Second term is "11", generated by reading first term as "One 1"**
**(There is one 1 in previous term)**
**Third term is "21", generated by reading second term as "Two 1"**
**Fourth term is "1211", generated by reading third term as "One 2 One 1"**
**and so on**
**Input: n = 3, Output: 21 // Input: n = 5 ,Output: 111221**

## Answer:-

```
def generator(s):
    ans = ""
    tempCount = {}

    for i in range(len(s) + 1):
        if i == len(s) or s[i] not in tempCount and i > 0:
            ans += str(tempCount[s[i - 1]]) + s[i - 1]
            tempCount.clear()
        if i == len(s):
            tempCount[None] = 1
        else:
            if s[i] in tempCount:
                tempCount[s[i]] += 1
            else:
                tempCount[s[i]] = 1
    return ans

def count_and_say(n):
    res = "1"
    for i in range(1, n):
        res = generator(res)
    return res
user_input = int(input("Enter a value for N: "))
print(count_and_say(user_input))
```