1.

```
for fizzbuzz in range(1, 100):

    # Number divisible by 15,(divisible
    # by both 3 & 5), print 'FizzBuzz'
    # in place of the number
    if fizzbuzz % 15 == 0:
        print("FizzBuzz")
        continue

    # Number divisible by 3, print 'Fizz'
    # in place of the number
    elif fizzbuzz % 3 == 0:
        print("Fizz")
        continue

    # Number divisible by 5,
    # print 'Buzz' in
    # place of the number
    elif fizzbuzz % 5 == 0:
        print("Buzz")
        continue

    # Print numbers
    print(fizzbuzz)
```

2.

```
total=int(input("Enter the total users"))
staff_users=int(input("Enter the staff users"))
non_teaching=staff_users/3
student_users=total-(staff_users+non_teaching)
n=int(student_users)
print(n)
```

3.

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: list[int]) -> list[int]:
        result = []
        for num in nums:
            count = 0
            for comp in nums:
                if num > comp:
                    count += 1
            result.append(count)
        return result
ob=Solution()
print(ob.smallerNumbersThanCurrent([6,5,4,8]))
```

4.

```
class Solution(object):
   def isPalindrome(self, s):
      """
      :type s: str
      :rtype: bool
      """
      x = ""
      diff = ord('a') - ord('A')
      for i in s:
         if ord(i)>=ord('a') and ord(i)<=ord('z') or ord(i)>=ord("0") and ord(i)<=ord("9"):
            x+=i
         elif ord(i)>=ord('A') and ord(i)<=ord('Z'):
            i = chr(diff+ord(i))
            x+=i
      #print(s)
      #print(x)
      return x == x[::-1]
ob1 = Solution()
print(ob1.isPalindrome("A Man, a Plan, a Canal: Panama"))
```

5.

```
# Returns minimum number of jumps to reach arr[n-1] from arr[0]
def minJumps(arr, n):
    # The number of jumps needed to reach the starting index is 0
    if (n <= 1):
        return 0

    # Return -1 if not possible to jump
    if (arr[0] == 0):
        return -1

    # initialization
    # stores all time the maximal reachable index in the array
    maxReach = arr[0]
    # stores the amount of steps we can still take
    step = arr[0]
    # stores the amount of jumps necessary to reach that maximal reachable position
    jump = 1

    # Start traversing array

    for i in range(1, n):
        # Check if we have reached the end of the array
        if (i == n - 1):
            return jump

        # updating maxReach
        maxReach = max(maxReach, i + arr[i])

        # we use a step to get to the current index
        step -= 1;

        # If no further steps left
        if (step == 0):
            # we must have used a jump
            jump += 1

            # Check if the current index / position or lesser index
            # is the maximum reach point from the previous indexes
            if (i >= maxReach):
                return -1

            # re-initialize the steps to the amount
            # of steps to reach maxReach from position i.
            step = maxReach - i;
    return -1


# Driver program to test above function
arr = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]

size = len(arr)

# Calling the minJumps function
print("Minimum number of jumps to reach end is % d " % minJumps(arr, size))
```

6.

```
def removeChar(s, c):
    # find total no. of
    # occurrence of character
    counts = s.count(c)

    # convert into list
    # of characters
    s = list(s)

    # keep looping until
    # counts become 0
    while counts:
        # remove character
        # from the list
        s.remove(c)

        # decremented by one
        counts -= 1

    # join all remaining characters
    # of the list with empty string
    s = ''.join(s)

    print(s)


# Driver code
if __name__ == '__main__':
    s = "hello world"
    removeChar(s, 'l')
```

7.

```
def countstrings(n, start):
    if n == 0:
        return 1
    count = 0
    for i in range(start, 5):
        count += countstrings(n - 1, i)
    return count
def countVowelStrings(n):
    #  char arr[5]={'a','e','i','o','u'};
    # starting from index 0 add the vowels to strings
    return countstrings(n, 0)
n = 1
print(countVowelStrings(n))
```

8.

```
def value(r):
    if (r == 'I'):
        return 1
    if (r == 'V'):
        return 5
    if (r == 'X'):
        return 10
    if (r == 'L'):
        return 50
    if (r == 'C'):
        return 100
    if (r == 'D'):
        return 500
    if (r == 'M'):
        return 1000
    return -1


def romanToDecimal(str):
    res = 0
    i = 0

    while (i < len(str)):

        # Getting value of symbol s[i]
        s1 = value(str[i])

        if (i + 1 < len(str)):

            # Getting value of symbol s[i + 1]
            s2 = value(str[i + 1])

            # Comparing both values
            if (s1 >= s2):

                # Value of current symbol is greater
                # or equal to the next symbol
                res = res + s1
                i = i + 1
            else:

                # Value of current symbol is greater
                # or equal to the next symbol
                res = res + s2 - s1
                i = i + 2
        else:
            res = res + s1
            i = i + 1

    return res


# Driver code
print("Integer form of Roman Numeral is"),
print(romanToDecimal("LVIII"))
```

9.

```
month = input("input the month (e.g. January, February etc.): ")
day = int(input("input the day: "))

if month in ('January', 'February', 'March'):
    season = 'winter'
elif month in ('April', 'May', 'June'):
    season = 'summer'
elif month in ('July', 'August', 'September'):
    season = 'spring'
else:
    season = 'autumn'

if (month == 'June') and (day > 20):
    season = 'spring'
elif (month == 'March') and (day > 19):
    season = 'summer'
elif (month == 'September') and (day > 21):
    season = 'autumn'
elif (month == 'December') and (day > 20):
    season = 'winter'

print("Season is",season)
```

10.

```
from random import sample

# initialize list
test_list = ['Python', 'Programs', 'are', 'very', 'difficult']

# printing original list
print("The original list : " + str(test_list))

# Scramble strings in list
# using list comprehension + sample() + join()
res = [''.join(sample(ele, len(ele))) for ele in test_list]

# printing result
print("Scrambled strings in lists are : " + str(res))
```