K.Gowthami

192224229

CSA0876

DAY -01

1st

```
1. import math
2.
3. num = 5
4.
5. # Calculate the cube of the number
6. cube = num ** 3
7.
8. # Calculate the square root of the number
9. sqrt = math.sqrt(num)
10.
11. print(f"Cube of {num}: {cube}")
12. print(f"Square root of {num}: {sqrt}")
```

2nd

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next


class Solution:
    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:
        dummy = cur = ListNode()
        carry = 0
        while l1 or l2 or carry:
            x = l1.val if l1 else 0
            y = l2.val if l2 else 0
            sum = carry + x + y
            carry = sum // 10
            cur.next = ListNode(sum % 10)
```

```
            cur = cur.next
            l1 = l1.next if l1 else None
            l2 = l2.next if l2 else None
        return dummy.next
```

3rd:

```
def find_LSD_and_MSD(n: int) -> tuple:
    # Convert the integer to a string to easily access digits
    str_n = str(abs(n))

    # Find the least significant digit (LSD)
    LSD = int(str_n[-1])

    # Find the most significant digit (MSD)
    MSD = int(str_n[0])

    return LSD, MSD
```

4th:

```
def lengthOfLongestSubstring(s: str) -> int:
    chars = set()
    left = 0
    result = 0
    for right in range(len(s)):
        while s[right] in chars:
            chars.remove(s[left])
            left += 1
        chars.add(s[right])
        result = max(result, right - left + 1)
    return result
```

5th:

```
def find_LSD_and_MSD(n: int) -> tuple:
    # Convert the integer to a string to easily access digits
    str_n = str(abs(n))

    # Find the least significant digit (LSD)
    LSD = int(str_n[-1])

    # Find the most significant digit (MSD)
    MSD = int(str_n[0])

    return LSD, MSD
```

6th:

```
def longestPalindrome(s: str) -> str:
    def expand_around_center(left: int, right: int) -> str:
        while left >= 0 and right < len(s) and s[left] == s[right]:
            left -= 1
            right += 1
        return s[left + 1:right]

    longest = ""
    for i in range(len(s)):
        palindrome1 = expand_around_center(i, i)  # Odd length palindrome
        palindrome2 = expand_around_center(i, i + 1)  # Even length palindrome
        longest = max(longest, palindrome1, palindrome2, key=len)

    return longest
```

7th:

```
from datetime import datetime
```

```
def check_anniversary(year, month, day):
    anniversary = datetime(year, month, day)
    if anniversary.year % 4 == 0 and (anniversary.year % 100 != 0 or anniversary.year % 400 == 0):
        # Leap year, print next anniversary
        next_anniversary = anniversary.replace(year=anniversary.year + 4)
        print(f"Next anniversary: {next_anniversary.date()}")
    else:
        # Not leap year, print previous anniversary
        previous_anniversary = anniversary.replace(year=anniversary.year - 4)
        print(f"Previous anniversary: {previous_anniversary.date()}")
```

8th:

```
def reverse(x: int) -> int:
    is_negative = x < 0
    x = abs(x)
    reversed_x = 0
    while x > 0:
        reversed_x = reversed_x * 10 + x % 10
        x = x // 10
    reversed_x = -reversed_x if is_negative else reversed_x
    return reversed_x if -2**31 <= reversed_x <= 2**31 - 1 else 0
print(reverse(123))
```

9th:

```
# Input
my_list = [5, 2, 9, 1, 7, 3]

# Sort the list in ascending order
sorted_list = sorted(my_list)

# Output
print(sorted_list)
```

10th:

```
# Input
words = ["hello", "world", "abc", "def", "ghi", "jkl"]

# Sort the words in alphabetical order
sorted_words = sorted(words)

# Output
print(sorted_words)
```

11th:

```
# Initialize counters
uppercase_count = 0
lowercase_count = 0

# Read characters until '*' is encountered
print("Enter characters (enter '*' to stop):")
while True:
    char = input()
    if char == '*':
        break
    if char.isupper():
        uppercase_count += 1
    elif char.islower():
        lowercase_count += 1

# Display output
print(f"Uppercase letters: {uppercase_count}")
print(f"Lowercase letters: {lowercase_count}")
```

12th;

```
# Input
my_list = [1, 2, 2, 3, 4, 4, 5, 6, 6]

# Remove duplicates using a set
unique_list = list(set(my_list))

# Output
print(unique_list)
```

13th:

```
# Input
N = int(input("Enter the number of values: "))
values = []
for i in range(N):
        value = float(input(f"Enter value {i+1}: "))
        values.append(value)

# Calculation
sum = sum(values)

# Output
print(f"Sum: {sum}")
```

14th:

```
# Input
string = input("Enter a string: ")

# Initialize counters
vowels = 0
consonants = 0
```

```
# Iterate through the string
for char in string:
        # Check if the character is a vowel
        if char.lower() in ['a', 'e', 'i', 'o', 'u']:
                vowels += 1
        # Check if the character is a consonant
        elif char.isalpha():
                consonants += 1

# Output
print(f"Vowels: {vowels}")
print(f"Consonants: {consonants}")
```

DAY -02

1st:

```
input_list = [4, 1, 3, 2, 4, 5, 2, 3, 6, 7, 7, 8, 9, 1]

unique_list = list(set(input_list))

ascending_order = sorted(unique_list)

descending_order = sorted(unique_list, reverse=True)

# Print the results
print("Original List with Duplicates:")
print(input_list)
```

```
print("\nList after Removing Duplicates:")
print(unique_list)

print("\nList Sorted in Ascending Order:")
print(ascending_order)

print("\nList Sorted in Descending Order:")
print(descending_order)
```

2nd:

```
def nth_max(nums, n):
    """
    Returns the nth maximum number in a list.
    """
    return sorted(nums, reverse=True)[n-1]

def nth_min(nums, n):
    """
    Returns the nth minimum number in a list.
    """
    return sorted(nums)[n-1]

# Example usage
nums = [12, 45, 7, 23, 56, 89, 34]
print("3rd maximum number:", nth_max(nums, 3))
print("2nd minimum number:", nth_min(nums, 2))
```

3rd:

```
sum_of_squares = sum(x**2 for x in nums)

nums = [1, 2, 3, 4, 5]
```

```
sum_of_squares = sum(x**2 for x in nums)
print(sum_of_squares)  # Output: 55
```

This is equivalent to:

```
sum_of_squares = sum([x**2 for x in nums])
```

4th:

```
def count_occurrences(nums):
    return {x: nums.count(x) for x in set(nums)}

nums = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
print(count_occurrences(nums))
```

This code uses a dictionary comprehension to count the occurrence of each element in the list. The set(nums) function is used to get unique elements in the list, and the count method is used to count the occurrence of each element.

Output:

```
{1: 1, 2: 2, 3: 3, 4: 4}
```

This shows that the number 1 occurs once, the number 2 occurs twice, the number 3 occurs thrice, and the number 4 occurs four times in the list.

Alternatively, you can use the Counter class from the collections module:

```
from collections import Counter

nums = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
print(Counter(nums)).
```