```
#25.Find Missing Numbers
def missing_number(arr,n):
    expected_sum=n*(n+1)//2 #here to find the expect sum usin length
of array
    actual_sum=sum(arr)
    return expected_sum-actual_sum
arr=[1,2,3,4,5,6,7]
n=10
print(f"missing number:{missing_number(arr,n)}")
```

```
missing number:27
```

```
#26.Check Balanced Parentheses
def is_balanced_parantheses(s):
    stack=[]
    mapping={')':'(','}':'{',']':'['}#maps the brackets and find them
available or not
    for char in s:
        if char in mapping.values():
            stack.append(char)
        elif char in mapping.keys():
            if not stack or mapping[char]!=stack.pop():
                return False
    return not stack
s="[{)}"
print(f"Is balanced?{is_balanced_parantheses(s)}")
```

```
Is balanced?False
```

```
#27.Longest Word in a Sentence
def longest_word(sentence):
    words=sentence.split() # splits the sentence and find the maxmimum
length word using max function
    return max(words,key=len)

sentence="I got an internship in Mainflow Services and Technologies"
print(f"Longest word:{longest_word(sentence)}")
```

```
Longest word:Technologies
```

```
#28.Count words in a sentence
def count_words(sentence):
    words=sentence.split() ## splits the sentence and find the length
word using len function
    return len(words)
sentence="I got an internship in Mainflow Services and Technologies"
print(f"Word count:{count_words(sentence)}")
```

```
Word count:9
```

```
#29.Check Pythagorean Triplet
def is_pythagorean_triplet(a,b,c):
    a,b,c=sorted([a,b,c])
    return a*2+b**2==c**2 # checks the given number is follow
pythagorous theorem or not

print(f"Is Pythagorean Triplet?{is_pythagorean_triplet(4,7,2)}")
```

```
Is Pythagorean Triplet?False
```

```
#30.bubble Sort
def bubble_sort(arr):
    n=len(arr)
    for i in range(n):
        for j in range(0,n-i-1):
            if arr[j]>arr[j+1]:  #sorting the array by comparing the
elements
                arr[j],arr[j+1]=arr[j+1],arr[j]
    return arr

arr=[55,54,76,20,43,30]
print(f"Sorted Array:{bubble_sort(arr)}")
```

```
Sorted Array:[20, 30, 43, 54, 55, 76]
```

```
#31.Binary Search
def binary_search(arr,target): #it search the given element using
binary search technique
    low,high=0,len(arr)-1
    while low <=high:
        mid=(low+high)//2
        if arr[mid]==target:
            return mid
        elif arr[mid]<target:
            low=mid+1
        else:
            high=mid-1
    return -1

arr=[10,15,20,25,30,35,40,45,50]
target=25
print(f"Index of {target}:{binary_search(arr,target)}")
```

```
Index of 25:3
```

```
#32.Find Subarray with given sum
def subarray_with_sum(arr,target): #in this given it checks the sum of
ekements present in the array and give those elements as output
    current_sum=0
    start=0
    for end in range(len(arr)):
        current_sum+=arr[end]
        while current_sum>target:
            current_sum-=arr[start]
            start+=1
        if current_sum==target:
            return start,end
    return -1

arr=[3,5,7,9,6,1]
target=15
print(f"Subarray indices:{subarray_with_sum(arr,target)}")
```

```
Subarray indices:(0, 2)
```

```
#4.Log Analysis System
def analyze_logs(file_path):
    ip_count={}
    with open(file_path,'r') as file:
        for line in file:
            parts=line.split()
            if len(parts)>0:
                ip=parts[0]
                ip_count[ip]=ip_count.get(ip,0)+1
    sorted_ips=sorted(ip_count.items(),key=lambda x:x[1],reverse=True)
    return sorted_ips[:5]

with open("server_logs.txt",'w') as file:
    file.write("192.168.1.01 - -[2/feb/2025] GET /index.html 200\n")
    file.write("172.168.1.1 - -[2/feb/2025] GET /about.html 420\n")
    file.write("198.128.1.7 - -[2/feb/2025] GET /contact.html 257\n")
    file.write("112.229.1.9 - -[2/feb/2025] GET /invest.html 490\n")
    file.write("203.321.1.4 - -[2/feb/2025] GET /india.html 260\n")
#it analyzes the given logs in the file And provide top things
print("Top Ip addresses:")
print(analyze_logs("server_logs.txt"))
```

```
Top Ip addresses:
[('192.168.1.01', 1), ('172.168.1.1', 1), ('198.128.1.7', 1),
('112.229.1.9', 1), ('203.321.1.4', 1)]
```