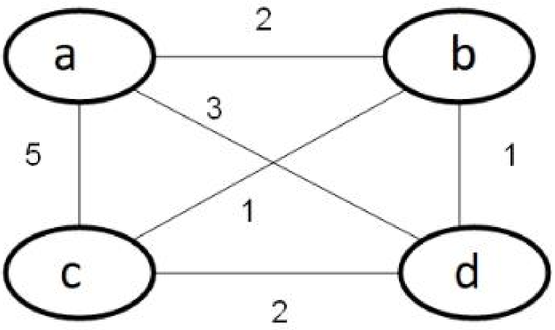
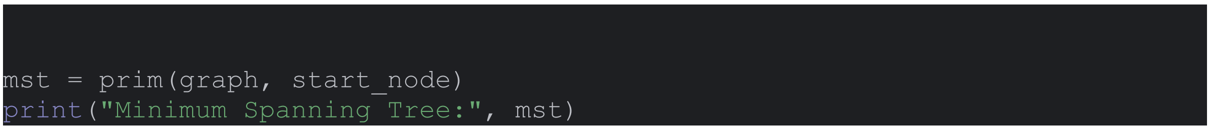
ASSIGNMENT

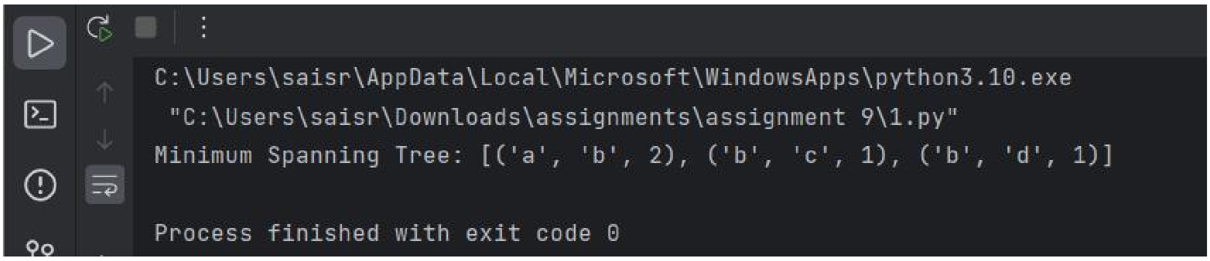
1. Apply Prim’s algorithm to solve the minimum spanning tree for the given graph. Also compute the total cost of all edges.



```
import heapq


def prim(graph, start_node):

    mst = []
    visited = set([start_node])

    edges = [(cost, start_node, to) for to, cost in
graph[start_node].items()]
    heapq.heapify(edges)


    while edges:
        cost, frm, to = heapq.heappop(edges)

        if to in visited:
            continue


        mst.append((frm, to, cost))
        visited.add(to)


        for to_next, cost_next in graph[to].items():
            if to_next not in visited:
                heapq.heappush(edges, (cost_next, to, to_next))

    return mst


graph = {
    'a': {'c': 5, 'b': 2},
    'b': {'a': 3, 'c': 1, 'd': 1},
    'c': {'a': 5, 'b': 1},
    'd': {'b': 1}
}


start_node = 'a'
```

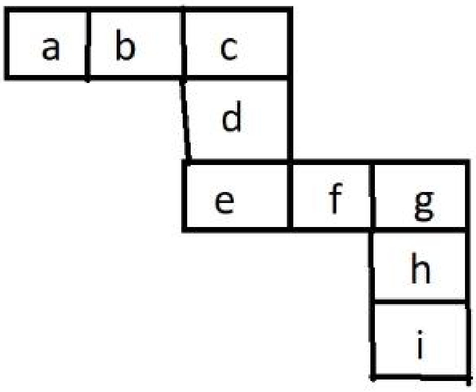


```
mst = prim(graph, start_node)
print("Minimum Spanning Tree:", mst)
```

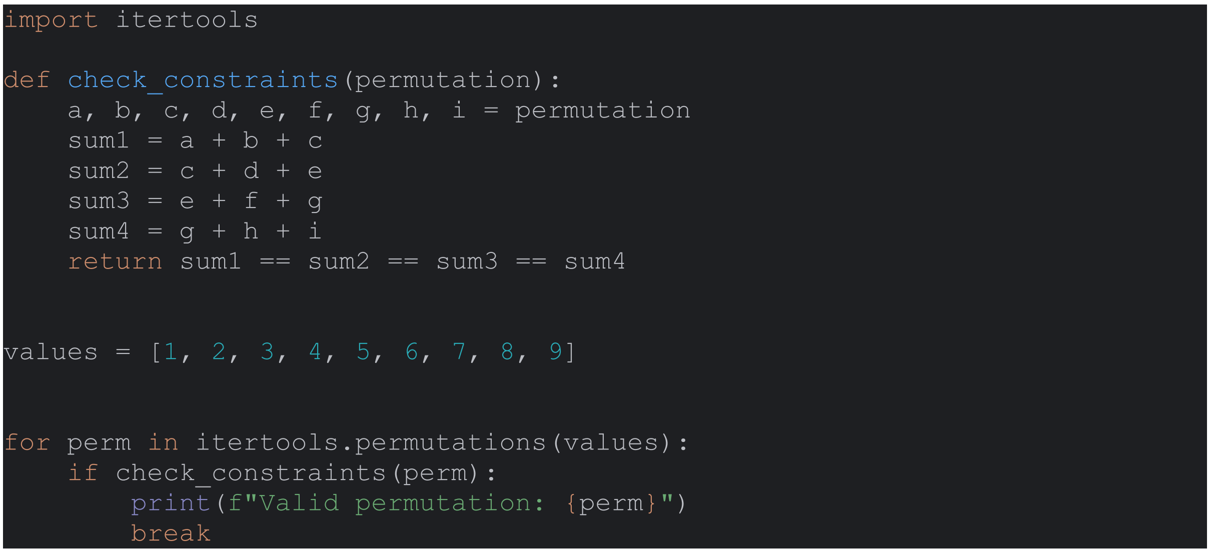


```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment 9\1.py"
Minimum Spanning Tree: [('a', 'b', 2), ('b', 'c', 1), ('b', 'd', 1)]

Process finished with exit code 0
```

2. To Compute the sum of Subsets for the following graph and then satisfy the given constraints.
   Set S { } =(a, b, c, d, e, f, g, h, i) Values used are V{i}=(1, 2, 3……….9)

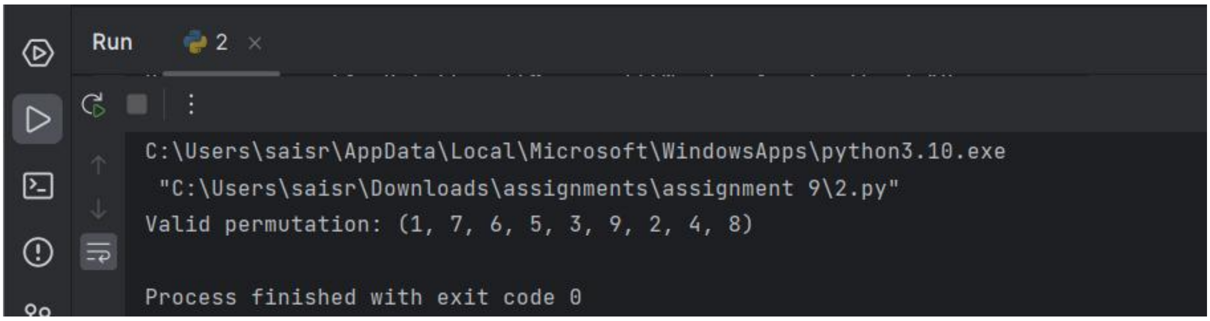


Used all values Only one time
Constraints hold such as
a + b + c = c +d + e = e + f + g = g + h + i



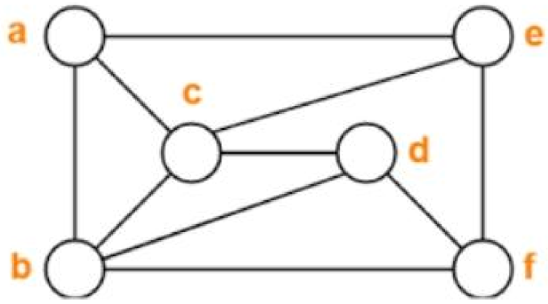
```
import itertools

def check_constraints(permutation):
    a, b, c, d, e, f, g, h, i = permutation
    sum1 = a + b + c
    sum2 = c + d + e
    sum3 = e + f + g
    sum4 = g + h + i
    return sum1 == sum2 == sum3 == sum4


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


for perm in itertools.permutations(values):
    if check_constraints(perm):
        print(f"Valid permutation: {perm}")
        break
```

```
else:
    print("No valid permutation found.")
```

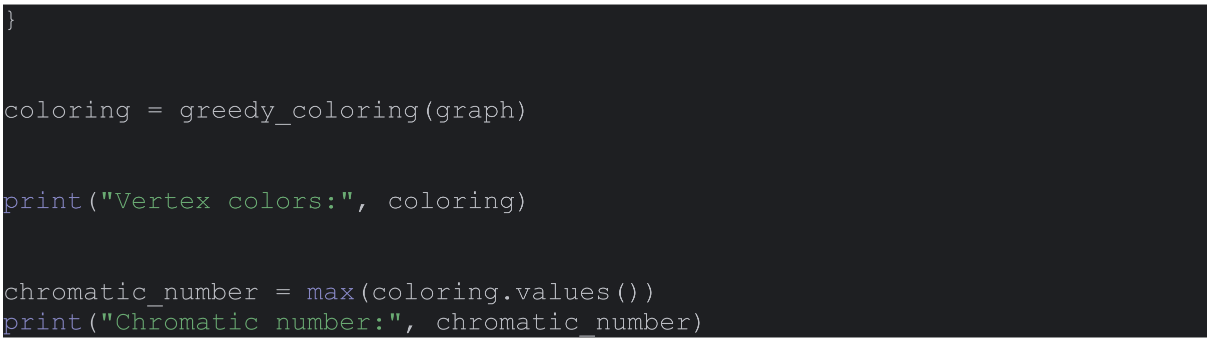


```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment 9\2.py"
Valid permutation: (1, 7, 6, 5, 3, 9, 2, 4, 8)

Process finished with exit code 0
```

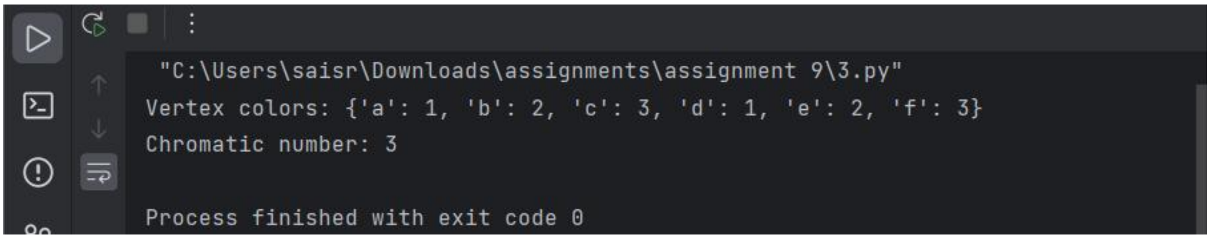
3. Calculate the chromatic no for the following Graph coloring.



```
def greedy_coloring(graph):

    result = {}

    for vertex in graph:

        adjacent_colors = set(result.get(neighbor) for neighbor in graph[vertex])


        color = 1
        while color in adjacent_colors:
            color += 1


        result[vertex] = color

    return result



graph = {
    'a': ['b', 'c', 'e'],
    'b': ['a', 'c', 'd'],
    'c': ['a', 'b', 'd', 'e'],
    'd': ['b', 'c', 'e', 'f'],
    'e': ['a', 'c', 'd', 'f'],
    'f': ['d', 'e']
```

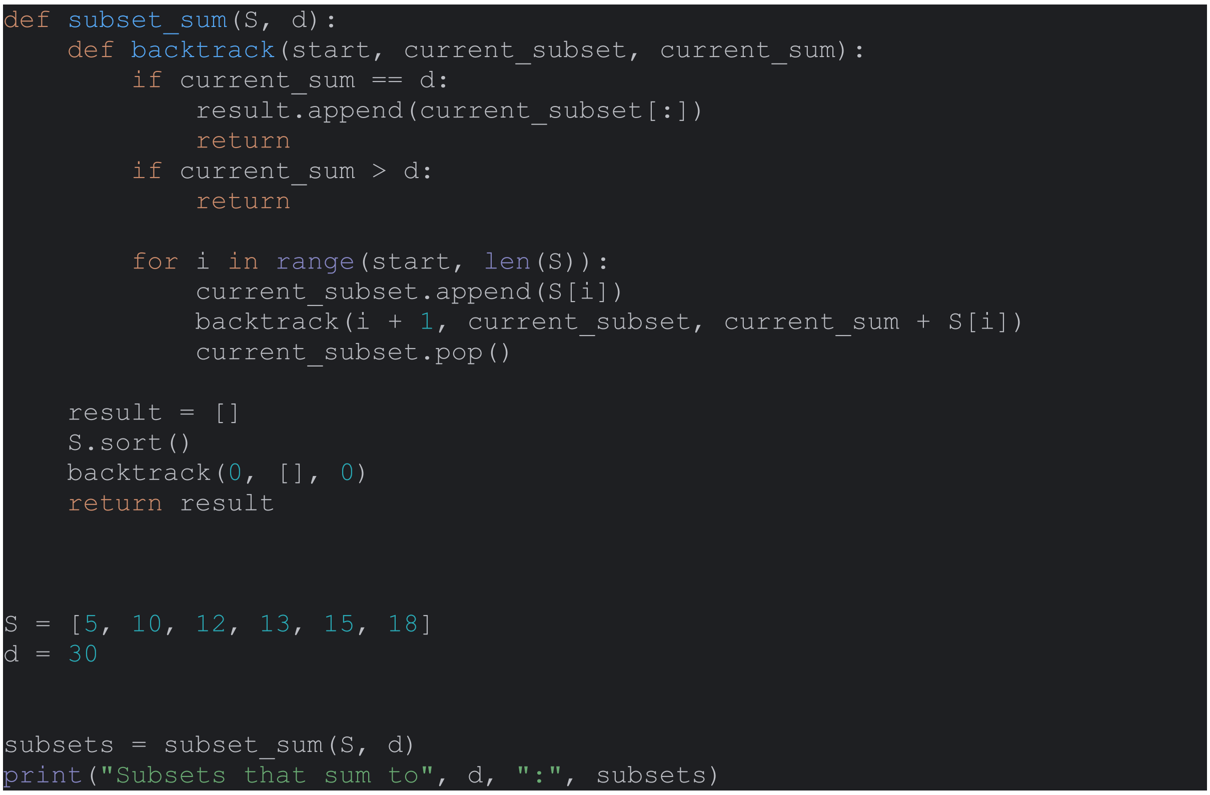


```
}


coloring = greedy_coloring(graph)


print("Vertex colors:", coloring)


chromatic_number = max(coloring.values())
print("Chromatic number:", chromatic_number)
```

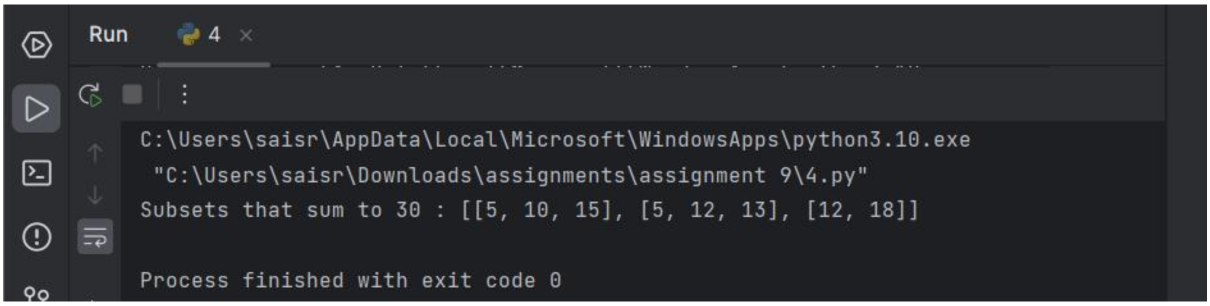


```
 "C:\Users\saisr\Downloads\assignments\assignment 9\3.py"
Vertex colors: {'a': 1, 'b': 2, 'c': 3, 'd': 1, 'e': 2, 'f': 3}
Chromatic number: 3

Process finished with exit code 0
```

4.Consider a set S = (5, 10, 12, 13, 15, 18) and d=30. Solve it for obtaining a sum of subset.



```
def subset_sum(S, d):
    def backtrack(start, current_subset, current_sum):
        if current_sum == d:
            result.append(current_subset[:])
            return
        if current_sum > d:
            return

        for i in range(start, len(S)):
            current_subset.append(S[i])
            backtrack(i + 1, current_subset, current_sum + S[i])
            current_subset.pop()

    result = []
    S.sort()
    backtrack(0, [], 0)
    return result



S = [5, 10, 12, 13, 15, 18]
d = 30


subsets = subset_sum(S, d)
print("Subsets that sum to", d, ":", subsets)
```



```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment 9\4.py"
Subsets that sum to 30 : [[5, 10, 15], [5, 12, 13], [12, 18]]

Process finished with exit code 0
```