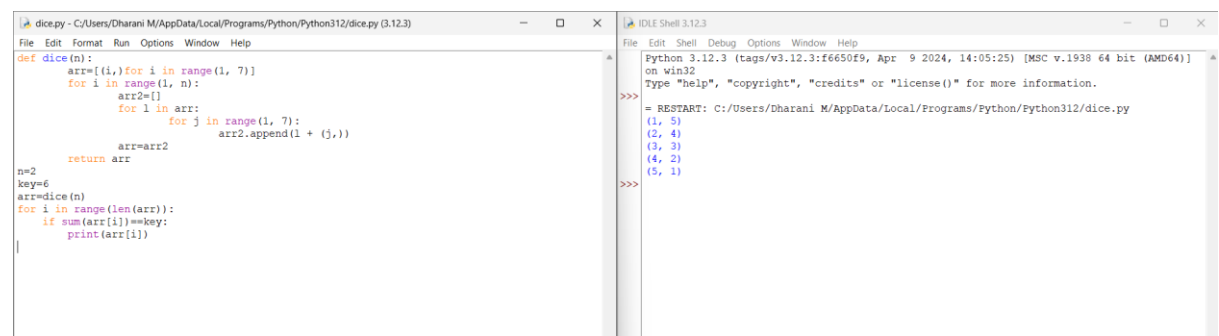


## 1. Dice programming

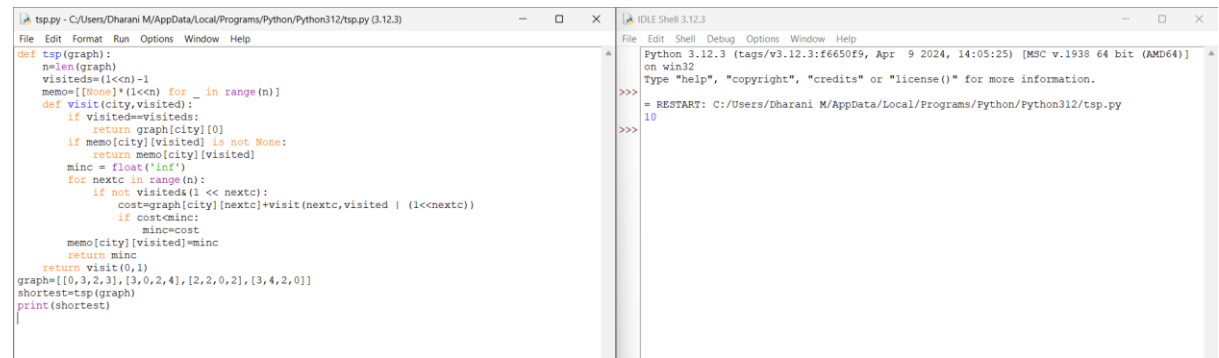


```
def dice(n):
    arr=[i,]for i in range(1, 7)]
    for i in range(1, n):
        arr2=[]
        for l in arr:
            for j in range(1, 7):
                arr2.append(l + (j,))
        arr=arr2
    return arr

n=2
key=6
arr=dice(n)
for i in range(len(arr)):
    if sum(arr[i])==key:
        print(arr[i])
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/dice.py
(1, 5)
(2, 4)
(3, 3)
(4, 2)
(5, 1)
```

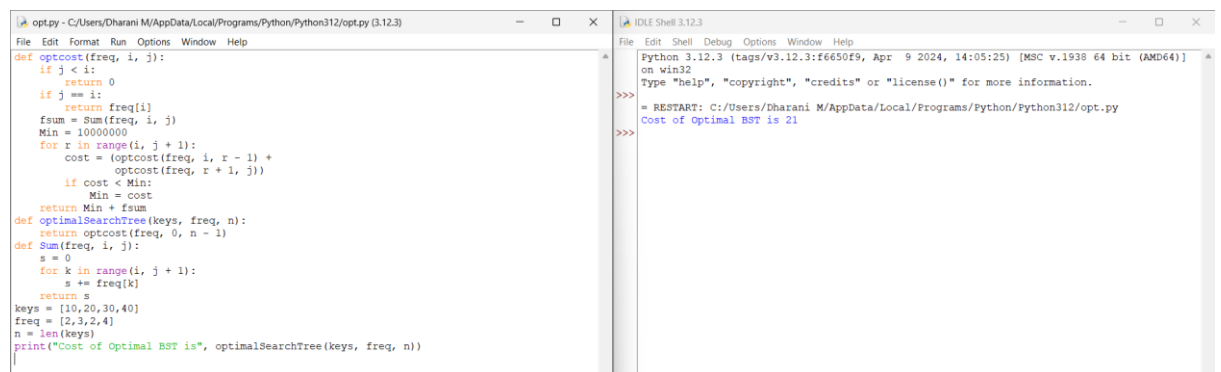
## 2. Tsp code



```
def tsp(graph):
    n=len(graph)
    visited=[l<n-1 for _ in range(n)]
    memo=[None]*(l<n)
    def visit(city,visited):
        if visited==visited:
            return graph[city][0]
        if memo[city][visited] is not None:
            return memo[city][visited]
        minc = float('inf')
        for nextc in range(n):
            if not visited[l << nextc]:
                cost=graph[city][nextc]+visit(nextc,visited | (l<<nextc))
                if cost<minc:
                    minc=cost
        memo[city][visited]=minc
    return minc
    return visit(0,1)
graph=[[0,3,2,3],[3,0,2,4],[2,2,0,2],[3,4,2,0]]
shortest=tsp(graph)
print(shortest)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/tsp.py
10
>>>
```

## 3. Opt code



```
def optcost(freq, i, j):
    if j < i:
        return 0
    if j == i:
        return freq[i]
    fsum = sum(freq, i, j)
    Min = 10000000
    for r in range(i, j + 1):
        cost = (optcost(freq, i, r - 1) +
                optcost(freq, r + 1, j))
        if cost < Min:
            Min = cost
    return Min + fsum
def optimalSearchTree(keys, freq, n):
    return optcost(freq, 0, n - 1)
def Sum(freq, i, j):
    s = 0
    for k in range(i, j + 1):
        s += freq[k]
    return s
keys = [10,20,30,40]
freq = [2,3,2,4]
n = len(keys)
print("Cost of Optimal BST is", optimalSearchTree(keys, freq, n))
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/opt.py
Cost of Optimal BST is 21
>>>
```