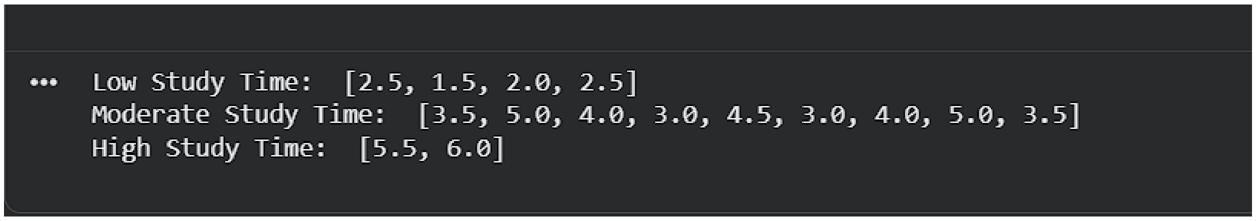
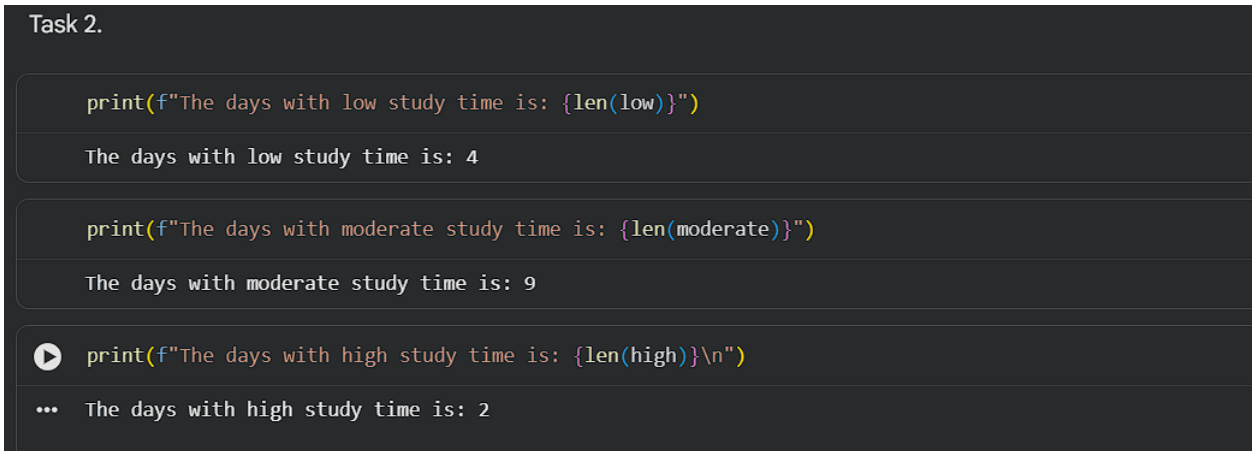


```
Low Study Time:  [2.5, 1.5, 2.0, 2.5]
Moderate Study Time:  [3.5, 5.0, 4.0, 3.0, 4.5, 3.0, 4.0, 5.0, 3.5]
High Study Time:  [5.5, 6.0]
```



Task 2.

```python
print(f"The days with low study time is: {len(low)}")
```

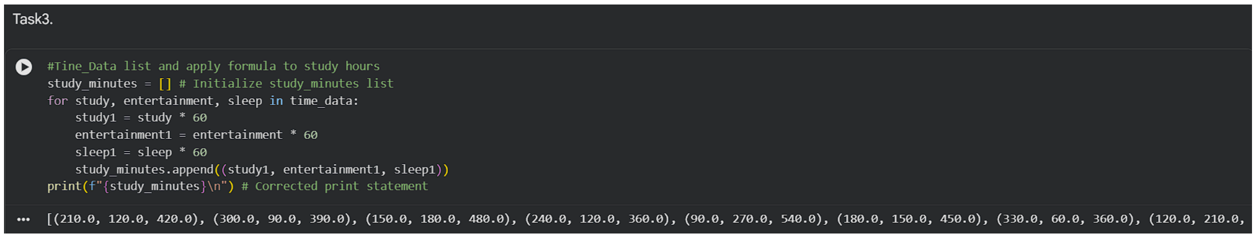
```
The days with low study time is: 4
```

```python
print(f"The days with moderate study time is: {len(moderate)}")
```

```
The days with moderate study time is: 9
```

```python
print(f"The days with high study time is: {len(high)}\n")
```

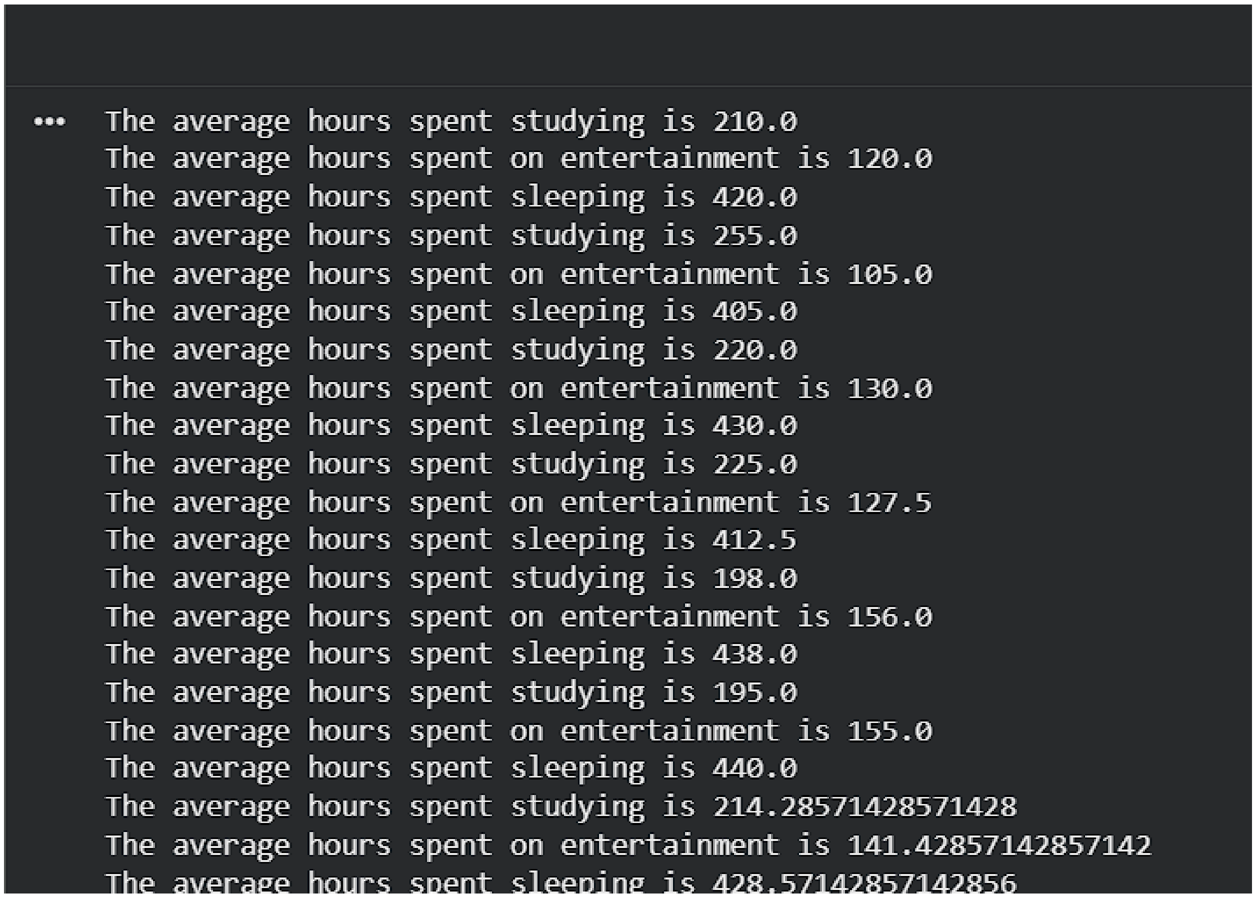
```
The days with high study time is: 2
```



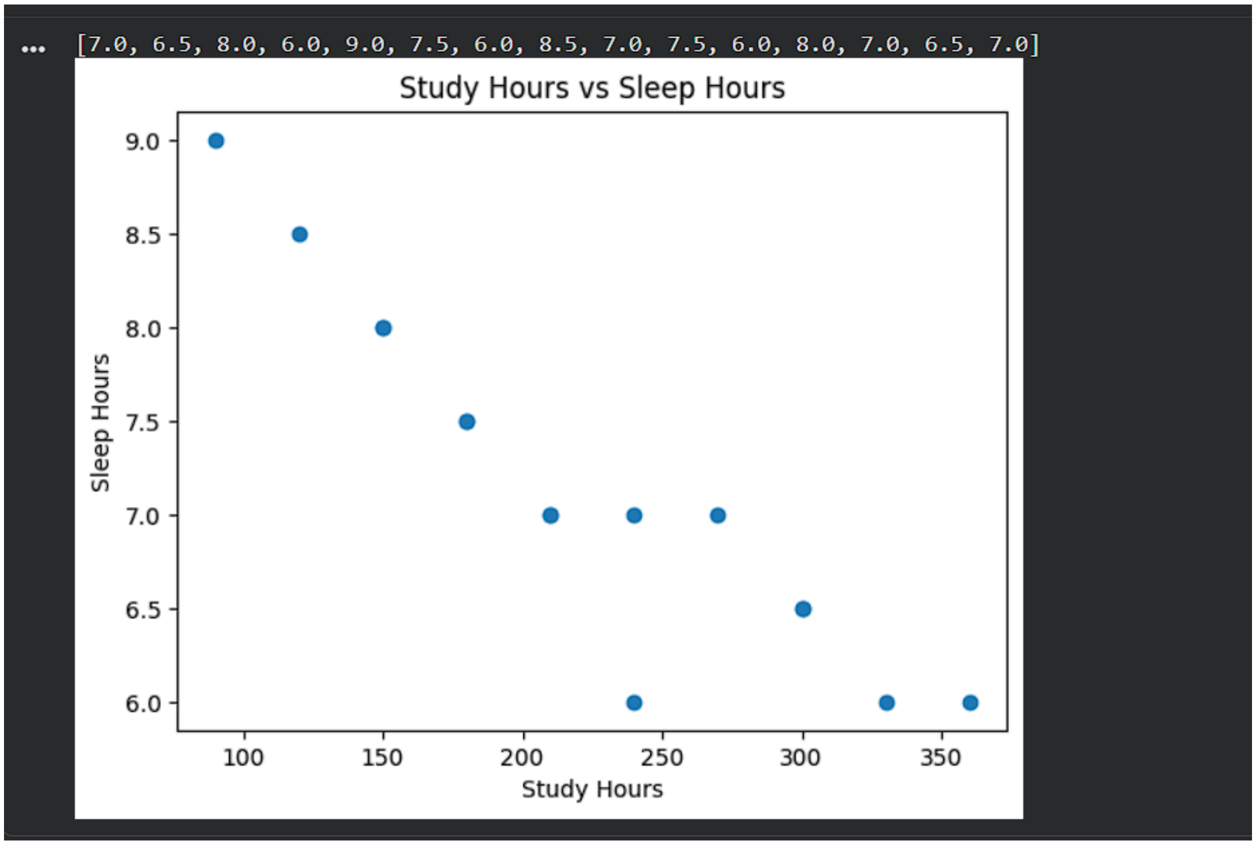
Task3.

```python
#Tine_Data list and apply formula to study hours
study_minutes = [] # Initialize study_minutes list
for study, entertainment, sleep in time_data:
    study1 = study * 60
    entertainment1 = entertainment * 60
    sleep1 = sleep * 60
    study_minutes.append((study1, entertainment1, sleep1))
print(f"{study_minutes}\n") # Corrected print statement
```

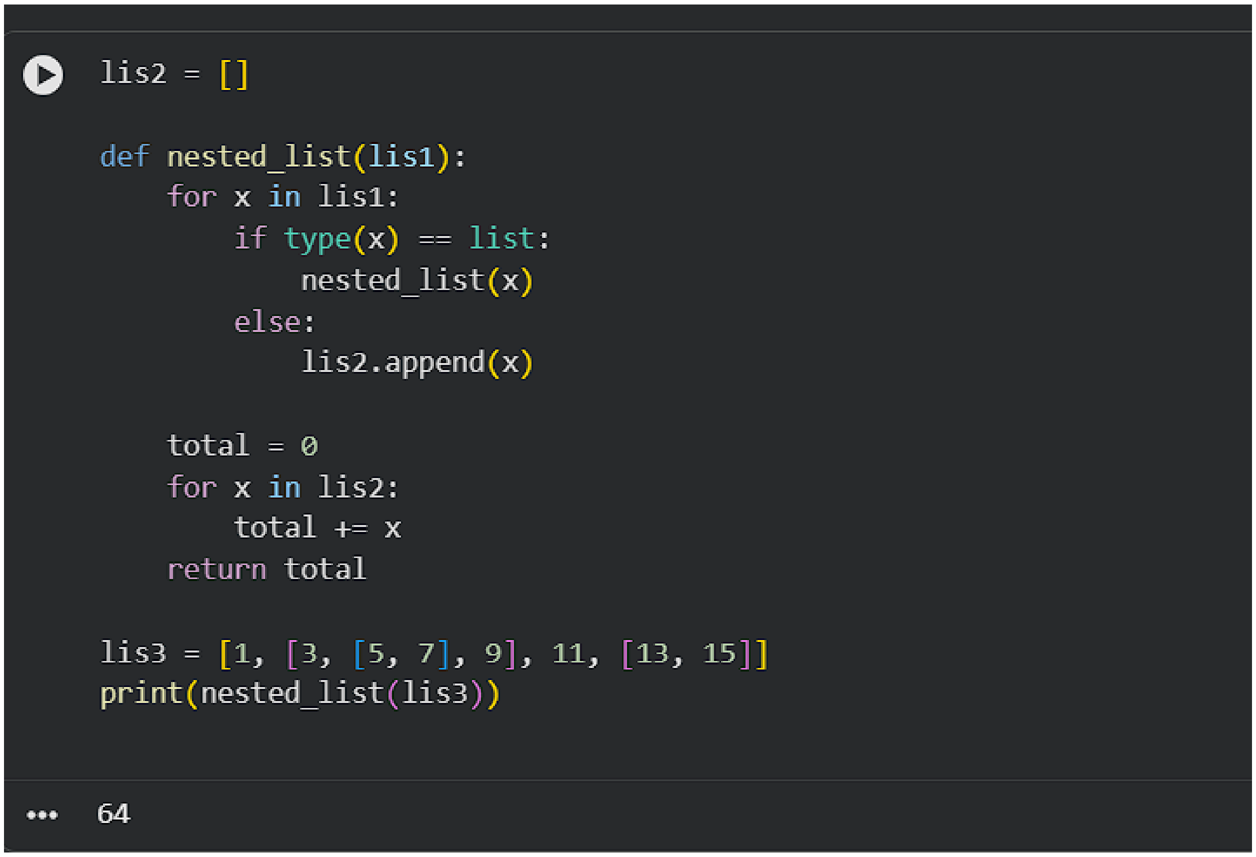
```
[(210.0, 120.0, 420.0), (300.0, 90.0, 390.0), (150.0, 180.0, 480.0), (240.0, 120.0, 360.0), (90.0, 270.0, 540.0), (180.0, 150.0, 450.0), (330.0, 60.0, 360.0), (120.0, 210.0,
```



```
The average hours spent studying is 210.0
The average hours spent on entertainment is 120.0
The average hours spent sleeping is 420.0
The average hours spent studying is 255.0
The average hours spent on entertainment is 105.0
The average hours spent sleeping is 405.0
The average hours spent studying is 220.0
The average hours spent on entertainment is 130.0
The average hours spent sleeping is 430.0
The average hours spent studying is 225.0
The average hours spent on entertainment is 127.5
The average hours spent sleeping is 412.5
The average hours spent studying is 198.0
The average hours spent on entertainment is 156.0
The average hours spent sleeping is 438.0
The average hours spent studying is 195.0
The average hours spent on entertainment is 155.0
The average hours spent sleeping is 440.0
The average hours spent studying is 214.28571428571428
The average hours spent on entertainment is 141.42857142857142
The average hours spent sleeping is 428.57142857142856
```

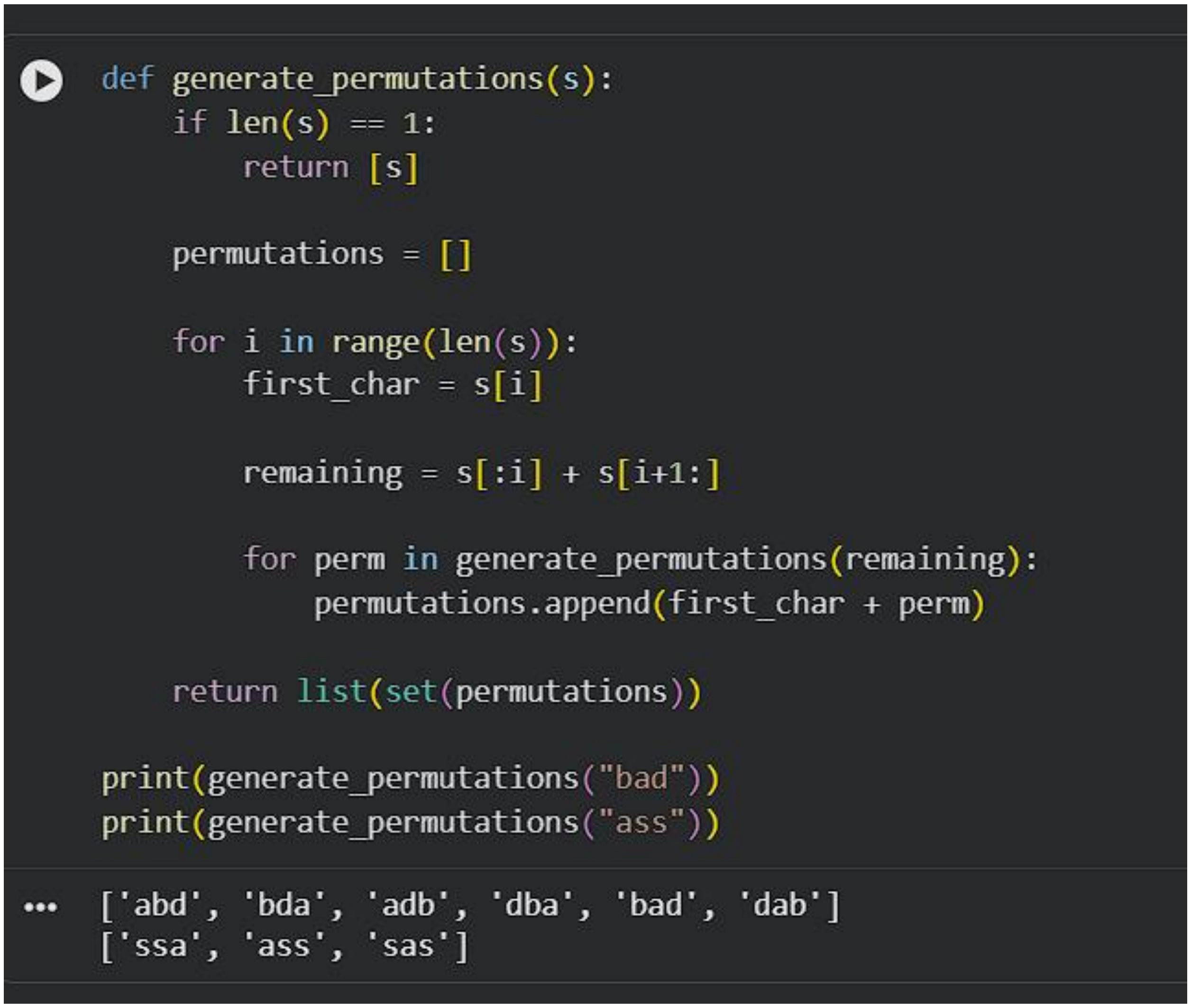


```
[7.0, 6.5, 8.0, 6.0, 9.0, 7.5, 6.0, 8.5, 7.0, 7.5, 6.0, 8.0, 7.0, 6.5, 7.0]
```



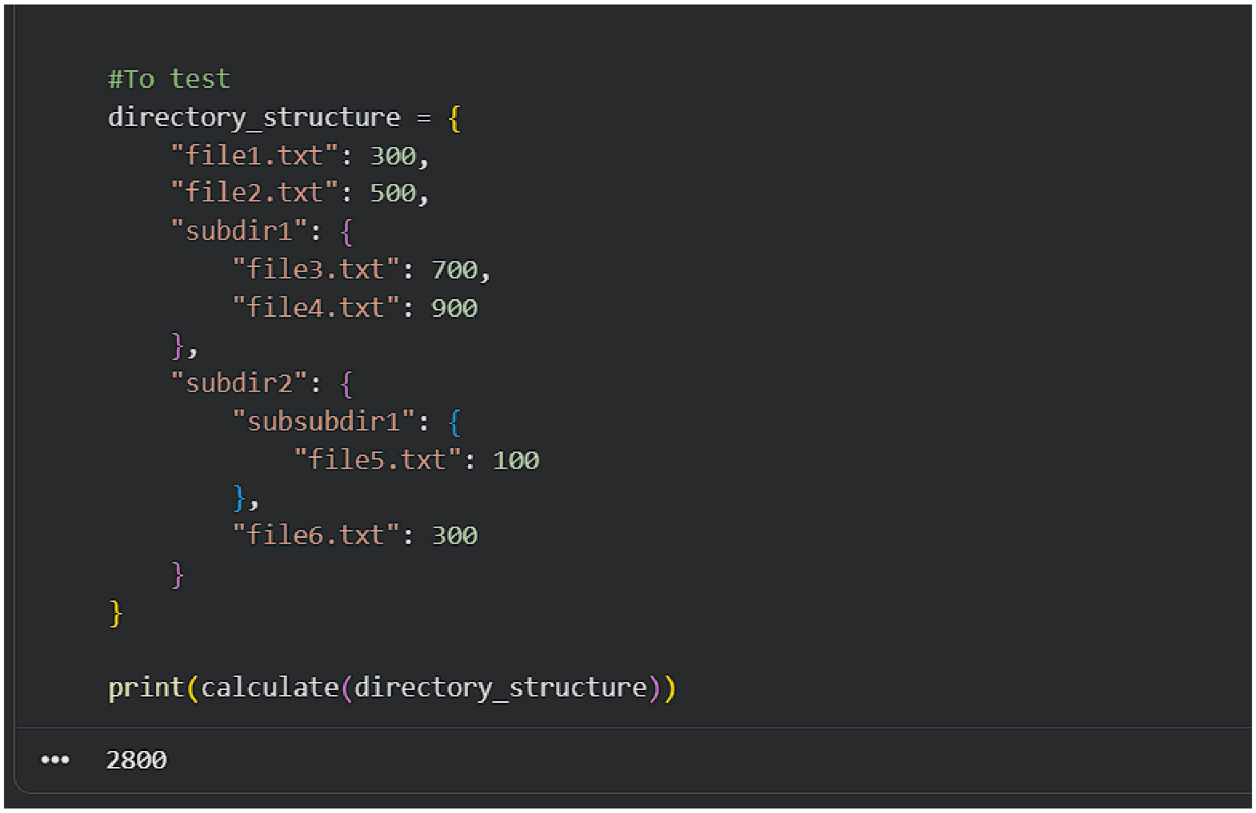
```
lis2 = []

def nested_list(lis1):
    for x in lis1:
        if type(x) == list:
            nested_list(x)
        else:
            lis2.append(x)

    total = 0
    for x in lis2:
        total += x
    return total

lis3 = [1, [3, [5, 7], 9], 11, [13, 15]]
print(nested_list(lis3))
```

```
64
```



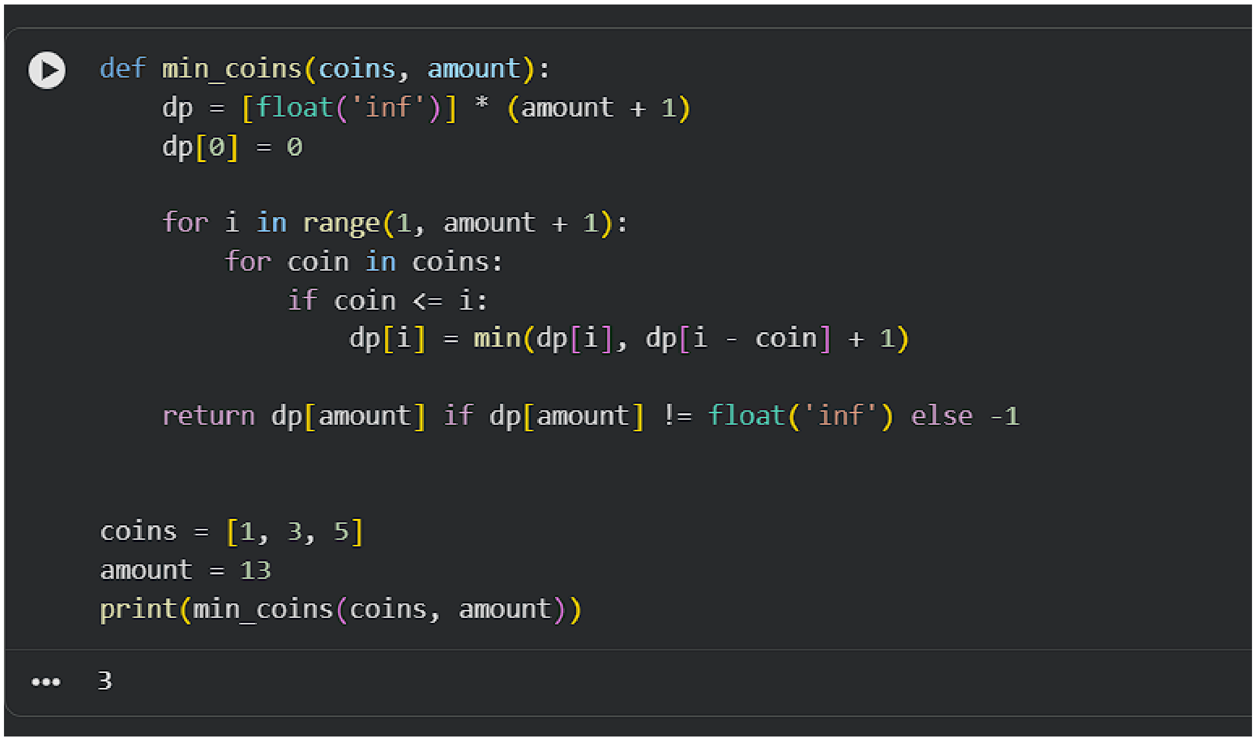
```python
def generate_permutations(s):
    if len(s) == 1:
        return [s]

    permutations = []

    for i in range(len(s)):
        first_char = s[i]

        remaining = s[:i] + s[i+1:]

        for perm in generate_permutations(remaining):
            permutations.append(first_char + perm)

    return list(set(permutations))

print(generate_permutations("bad"))
print(generate_permutations("ass"))
```

```
['abd', 'bda', 'adb', 'dba', 'bad', 'dab']
['ssa', 'ass', 'sas']
```



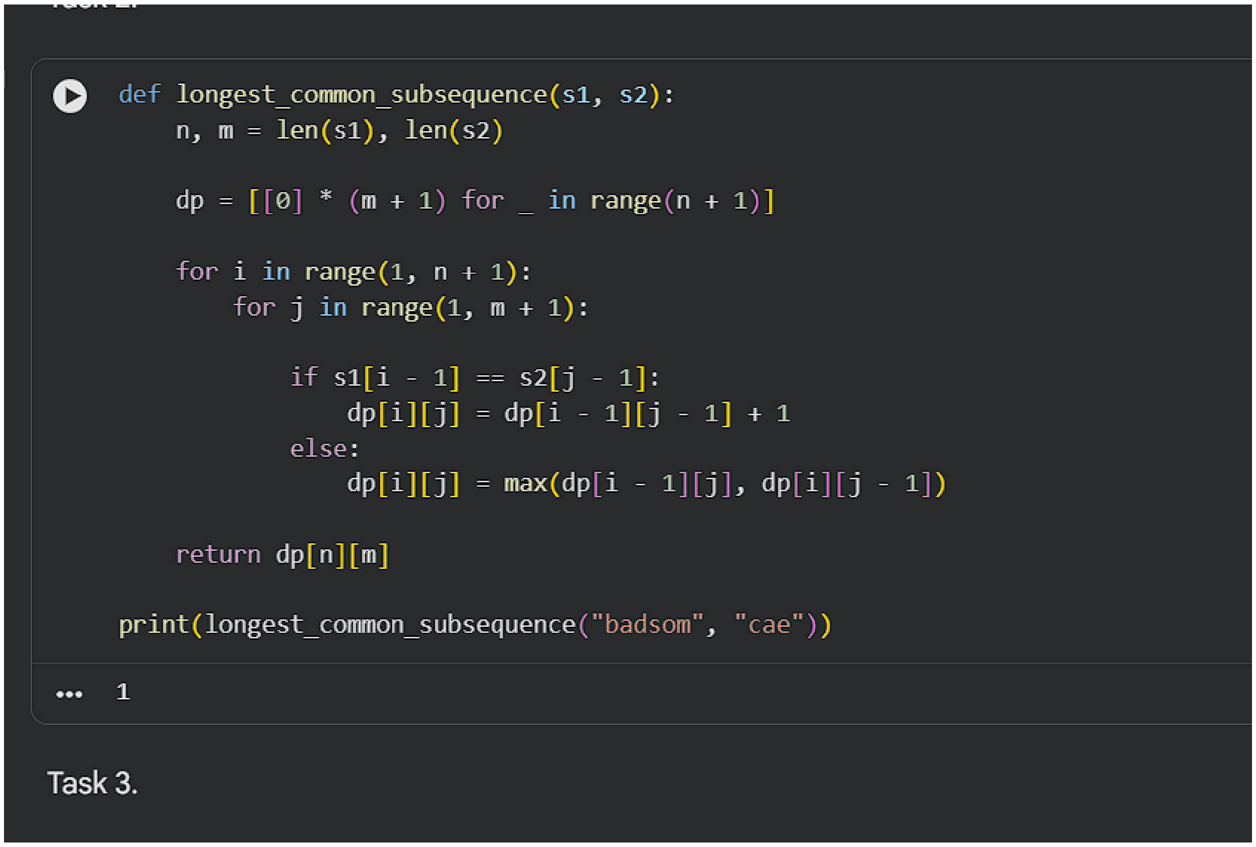
```
#To test
directory_structure = {
    "file1.txt": 300,
    "file2.txt": 500,
    "subdir1": {
        "file3.txt": 700,
        "file4.txt": 900
    },
    "subdir2": {
        "subsubdir1": {
            "file5.txt": 100
        },
        "file6.txt": 300
    }
}

print(calculate(directory_structure))
```

```
2800
```



```
def min_coins(coins, amount):
    dp = [float('inf')] * (amount + 1)
    dp[0] = 0

    for i in range(1, amount + 1):
        for coin in coins:
            if coin <= i:
                dp[i] = min(dp[i], dp[i - coin] + 1)

    return dp[amount] if dp[amount] != float('inf') else -1


coins = [1, 3, 5]
amount = 13
print(min_coins(coins, amount))
```

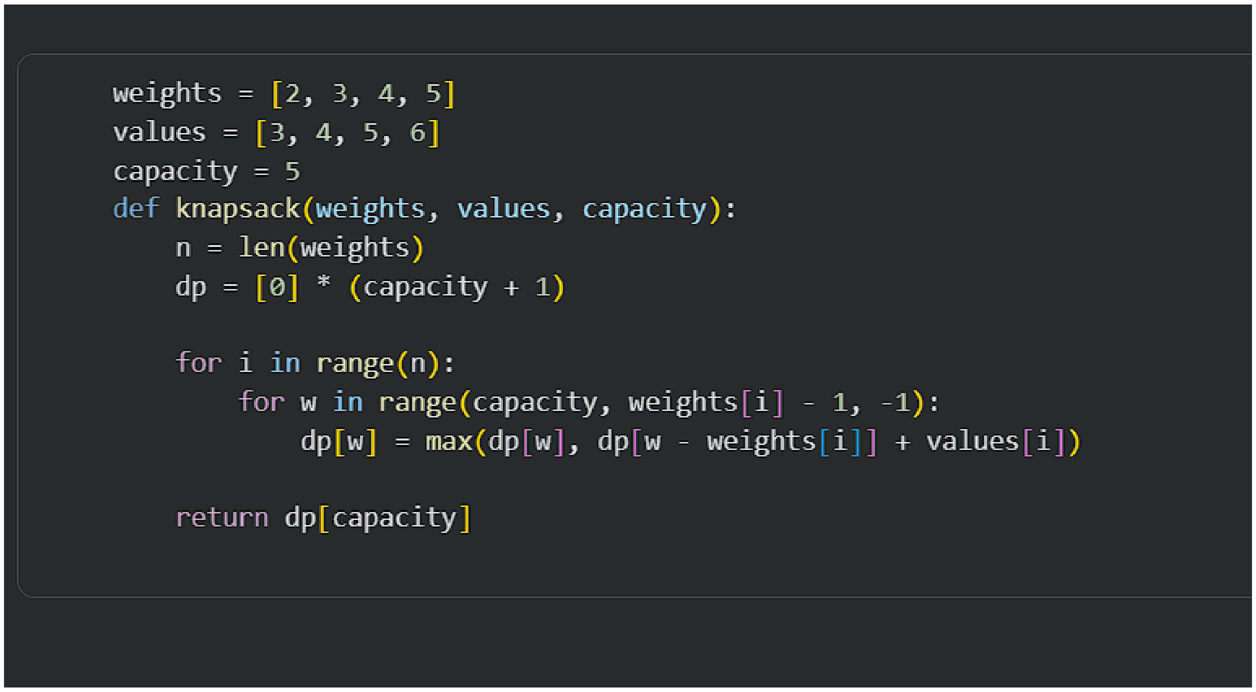
```
3
```



```python
def longest_common_subsequence(s1, s2):
    n, m = len(s1), len(s2)

    dp = [[0] * (m + 1) for _ in range(n + 1)]

    for i in range(1, n + 1):
        for j in range(1, m + 1):

            if s1[i - 1] == s2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    return dp[n][m]

print(longest_common_subsequence("badsom", "cae"))
```

```
1
```

Task 3.



```python
weights = [2, 3, 4, 5]
values = [3, 4, 5, 6]
capacity = 5
def knapsack(weights, values, capacity):
    n = len(weights)
    dp = [0] * (capacity + 1)

    for i in range(n):
        for w in range(capacity, weights[i] - 1, -1):
            dp[w] = max(dp[w], dp[w - weights[i]] + values[i])

    return dp[capacity]
```