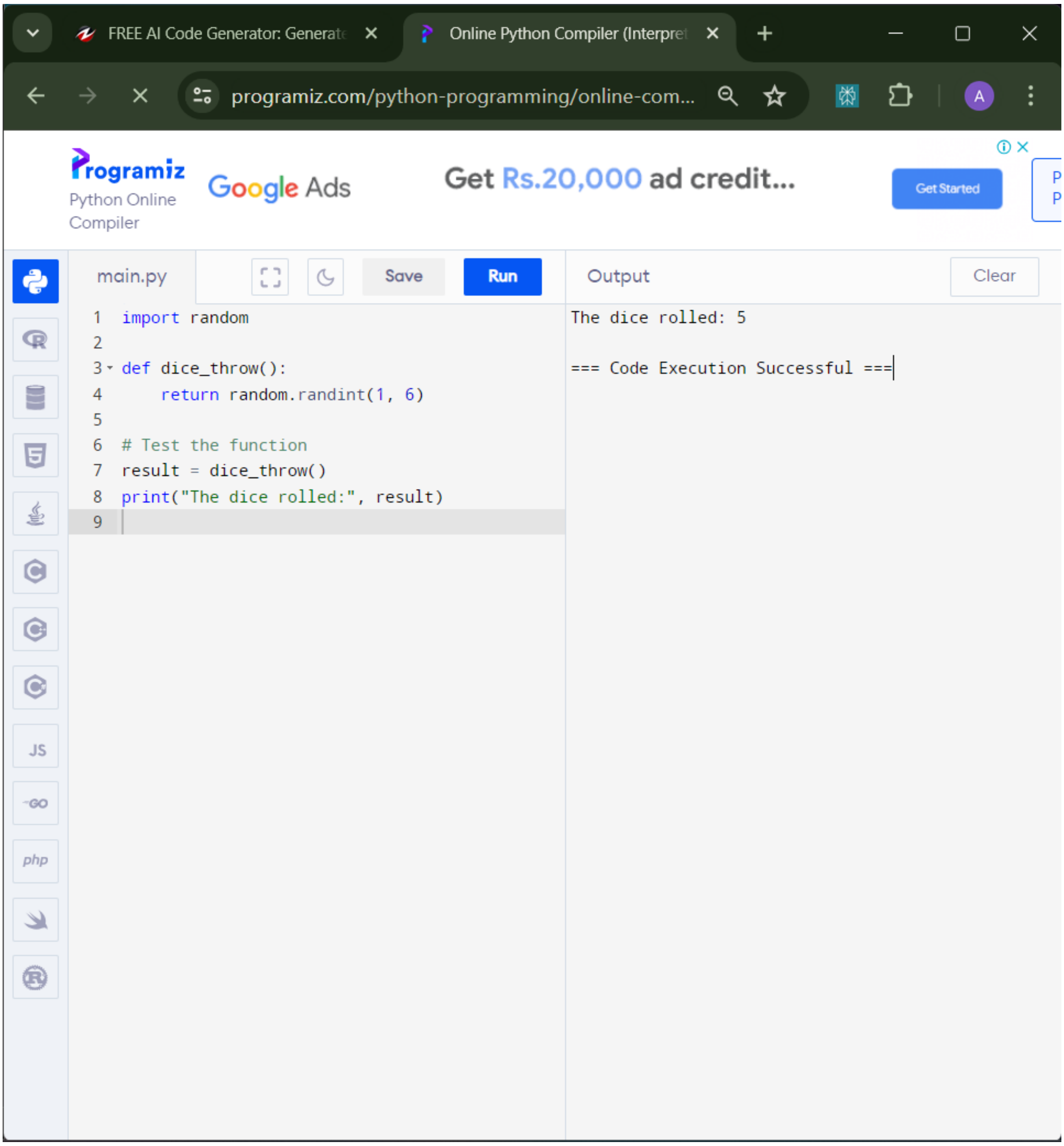
1.Dice throw problem:-

Code:-

```
import random


def dice_throw():
    return random.randint(1, 6)


# Test the function
result = dice_throw()
print("The dice rolled:", result)
```

Output:-


FREE AI Code Generator: Generate
Online Python Compiler (Interpret
programiz.com/python-programming/online-com...
Programiz
Python Online Compiler
main.py
Save
Run
Output
Clear

```
import random

def dice_throw():
    return random.randint(1, 6)

# Test the function
result = dice_throw()
print("The dice rolled:", result)
```

```
The dice rolled: 5

=== Code Execution Successful ===
```

2. Sum subset problem:-

Code:-

```
def sum_subset(arr, target):
    if target == 0:
        return True
    if not arr or target < 0:
        return False
    return sum_subset(arr[1:], target - arr[0]) or sum_subset(arr[1:], target)
```

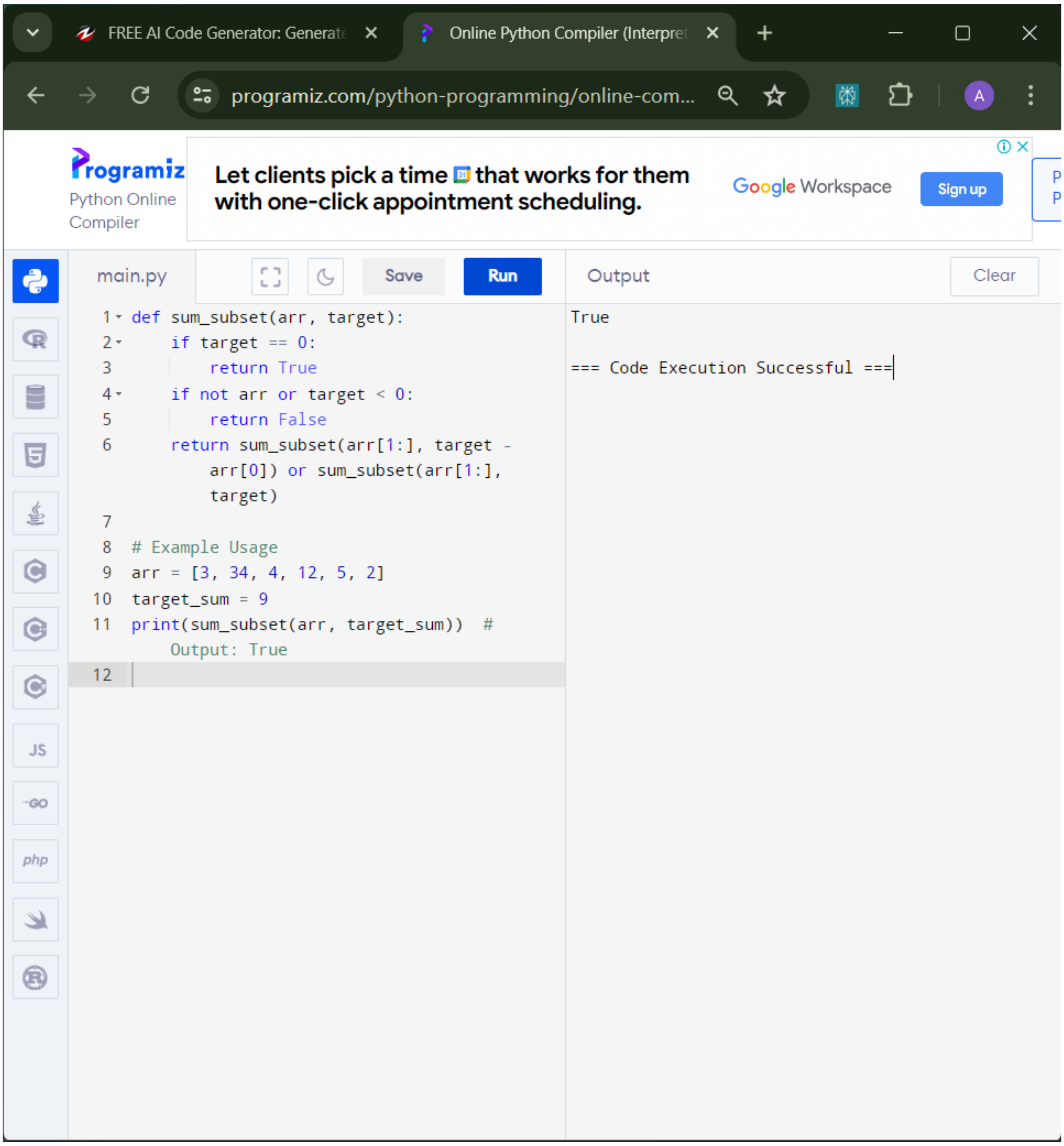
# Example Usage

arr = [3, 34, 4, 12, 5, 2]

target_sum = 9

print(sum_subset(arr, target_sum)) # Output: True

Output:-



FREE AI Code Generator: Generat
Online Python Compiler (Interpret
programiz.com/python-programming/online-com...
Programiz
Python Online Compiler
main.py
Save
Run
Output
Clear

```
def sum_subset(arr, target):
    if target == 0:
        return True
    if not arr or target < 0:
        return False
    return sum_subset(arr[1:], target - arr[0]) or sum_subset(arr[1:], target)

# Example Usage
arr = [3, 34, 4, 12, 5, 2]
target_sum = 9
print(sum_subset(arr, target_sum))  # Output: True
```

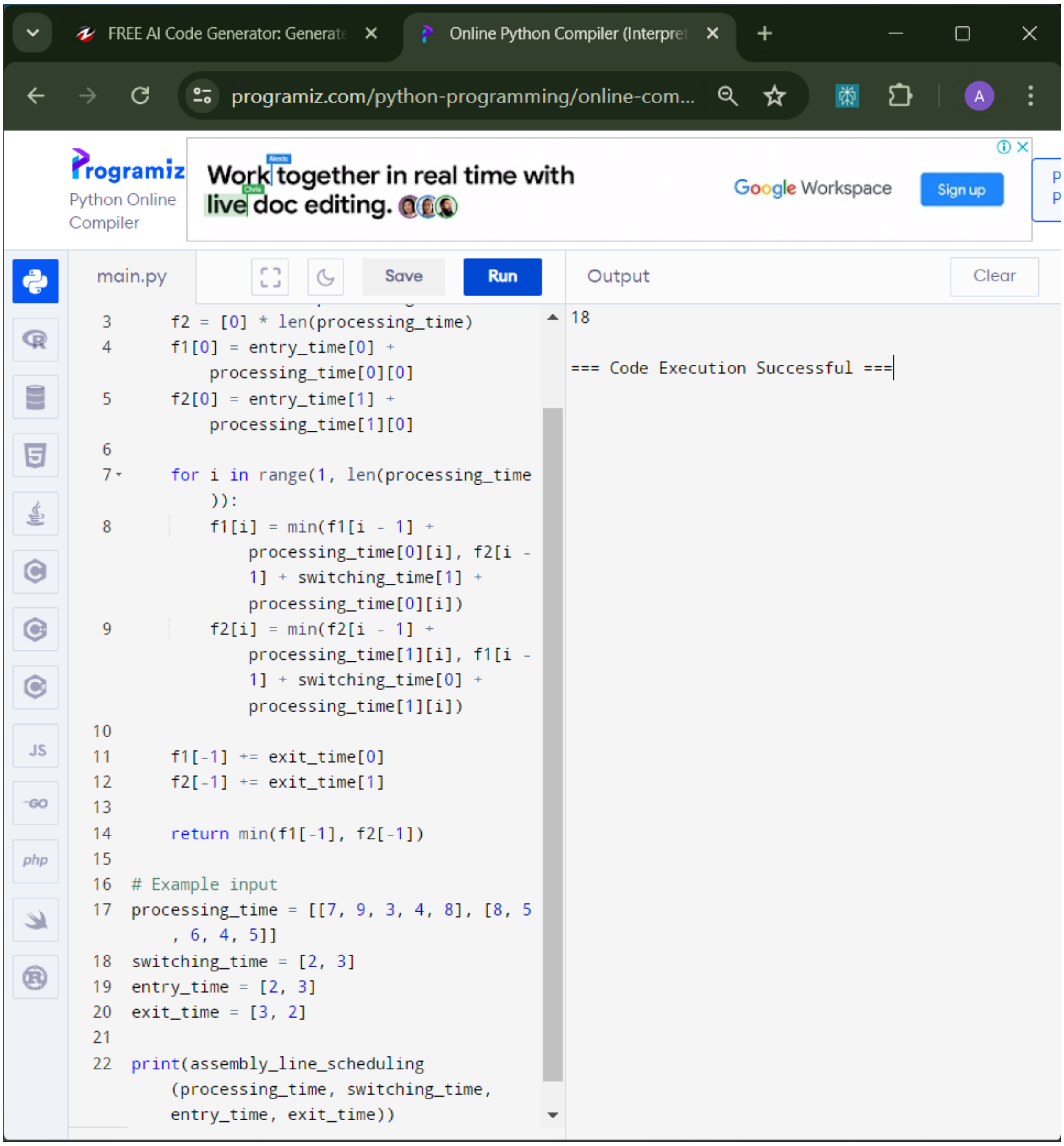
```
True

=== Code Execution Successful ===
```

3.Assembly line scheduling:-

Code:-

```
def assembly_line_scheduling(processing_time, switching_time, entry_time, exit_time):
    f1 = [0] * len(processing_time)
    f2 = [0] * len(processing_time)
    f1[0] = entry_time[0] + processing_time[0][0]
    f2[0] = entry_time[1] + processing_time[1][0]

    for i in range(1, len(processing_time)):
        f1[i] = min(f1[i - 1] + processing_time[0][i], f2[i - 1] + switching_time[1] +
processing_time[0][i])
        f2[i] = min(f2[i - 1] + processing_time[1][i], f1[i - 1] + switching_time[0] +
processing_time[1][i])

    f1[-1] += exit_time[0]
    f2[-1] += exit_time[1]

    return min(f1[-1], f2[-1])

# Example input
processing_time = [[7, 9, 3, 4, 8], [8, 5, 6, 4, 5]]
switching_time = [2, 3]
entry_time = [2, 3]
exit_time = [3, 2]

print(assembly_line_scheduling(processing_time, switching_time, entry_time, exit_time))
```

Output;-


FREE AI Code Generator: Generate
Online Python Compiler (Interpret
programiz.com/python-programming/online-com...
Programiz
Python Online Compiler
Work together in real time with live doc editing.
Google Workspace
Sign up
main.py
Save
Run
Output
Clear

```
    f2 = [0] * len(processing_time)
    f1[0] = entry_time[0] + processing_time[0][0]
    f2[0] = entry_time[1] + processing_time[1][0]

    for i in range(1, len(processing_time)):
        f1[i] = min(f1[i - 1] + processing_time[0][i], f2[i - 1] + switching_time[1] + processing_time[0][i])
        f2[i] = min(f2[i - 1] + processing_time[1][i], f1[i - 1] + switching_time[0] + processing_time[1][i])

    f1[-1] += exit_time[0]
    f2[-1] += exit_time[1]

    return min(f1[-1], f2[-1])

# Example input
processing_time = [[7, 9, 3, 4, 8], [8, 5, 6, 4, 5]]
switching_time = [2, 3]
entry_time = [2, 3]
exit_time = [3, 2]

print(assembly_line_scheduling(processing_time, switching_time, entry_time, exit_time))
```

```
18

=== Code Execution Successful ===
```

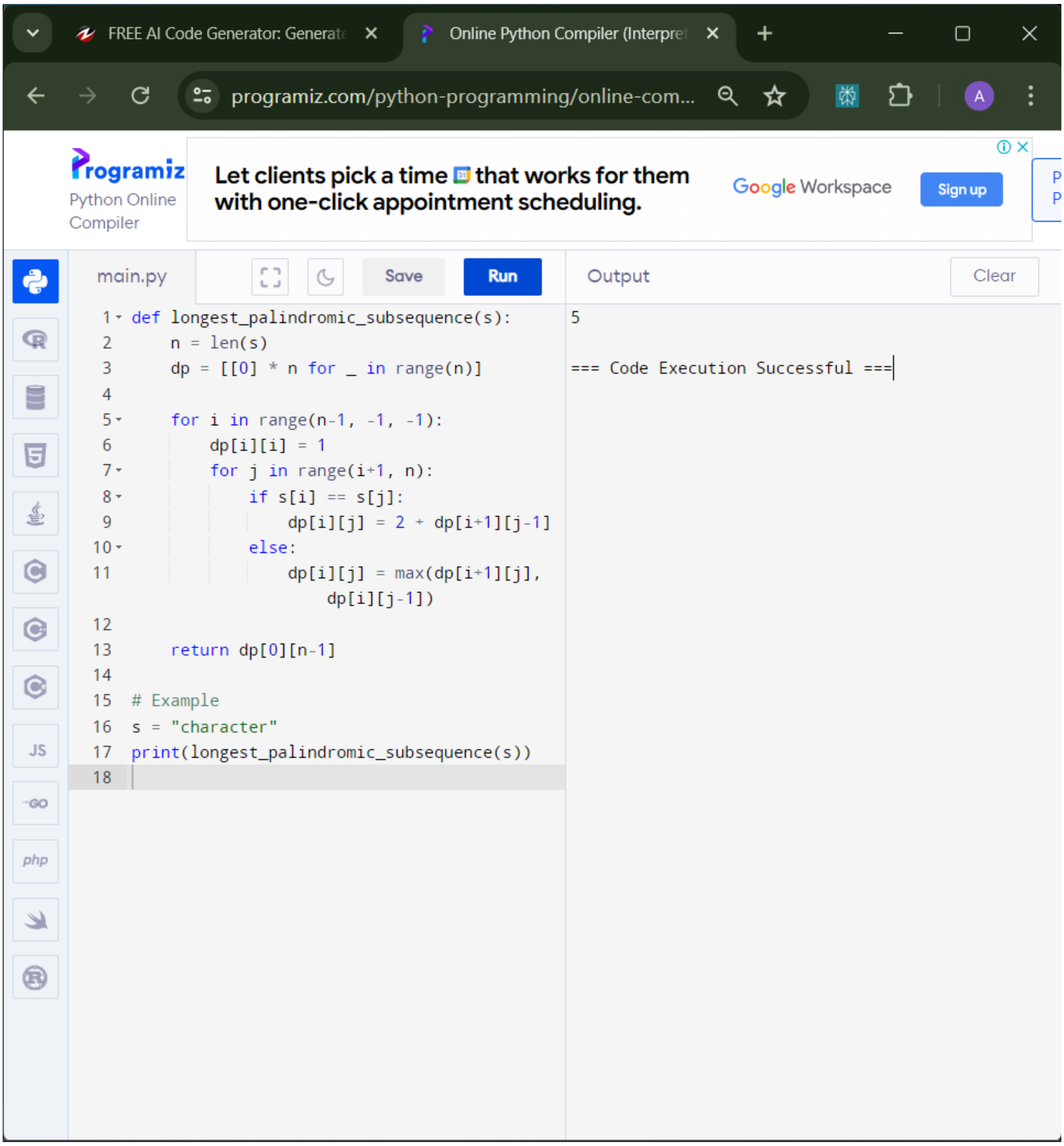
4.Longest palindromic subsequence:-

Code:-

```
def longest_palindromic_subsequence(s):
  n = len(s)
  dp = [[0] * n for _ in range(n)]


  for i in range(n-1, -1, -1):
    dp[i][i] = 1
```

```
        for j in range(i+1, n):
            if s[i] == s[j]:
                dp[i][j] = 2 + dp[i+1][j-1]
            else:
                dp[i][j] = max(dp[i+1][j], dp[i][j-1])

    return dp[0][n-1]

# Example
s = "character"
print(longest_palindromic_subsequence(s))
```

Output:-


FREE AI Code Generator: Generate
Online Python Compiler (Interpret
programiz.com/python-programming/online-com...
Programiz
Python Online Compiler
Let clients pick a time that works for them with one-click appointment scheduling.
Google Workspace
Sign up
main.py
Save
Run
Output
Clear
JS
GO
php

```
def longest_palindromic_subsequence(s):
    n = len(s)
    dp = [[0] * n for _ in range(n)]

    for i in range(n-1, -1, -1):
        dp[i][i] = 1
        for j in range(i+1, n):
            if s[i] == s[j]:
                dp[i][j] = 2 + dp[i+1][j-1]
            else:
                dp[i][j] = max(dp[i+1][j], dp[i][j-1])

    return dp[0][n-1]

# Example
s = "character"
print(longest_palindromic_subsequence(s))
```

```
5

=== Code Execution Successful ===
```