DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run

```
def dice_throw(num_sides, num_dice, target):
    dp = [[0 for _ in range(target + 1)] for _ in range(num_dice + 1)]
    dp[0][0] = 1
    for dice in range(1, num_dice + 1):
        for current_sum in range(1, target + 1):
            dp[dice][current_sum] = 0
            for roll in range(1, num_sides + 1):
                if current_sum - roll >= 0:
                    dp[dice][current_sum] += dp[dice - 1][current_sum
                        - roll]
    return dp[num_dice][target]
num_sides = 6
num_dice = 3
target = 8
result = dice_throw(num_sides, num_dice, target)
print(f"Number of ways to get a sum of {target} with {num_dice} dice:
    {result}")

```

Output | Clear

```
Number of ways to get a sum of 8 with 3 dice: 21

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


30°C Partly sunny | Search | ENG IN | 16:16 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

Programiz PRO >

main.py | Share | Run | Output | Clear

```python
def assembly_line(time_line1, time_line2, transfer_line1, transfer_line2, entry1, entry2, exit1, exit2):
    n = len(time_line1)
    dp1 = [0] * n
    dp2 = [0] * n
    dp1[0] = entry1 + time_line1[0]
    dp2[0] = entry2 + time_line2[0]
    for i in range(1, n):
        dp1[i] = min(dp1[i-1] + time_line1[i], dp2[i-1] + transfer_line1[i-1] + time_line1[i])
        dp2[i] = min(dp2[i-1] + time_line2[i], dp1[i-1] + transfer_line2[i-1] + time_line2[i])
    final_time1 = dp1[n-1] + exit1
    final_time2 = dp2[n-1] + exit2
    return min(final_time1, final_time2)
time_line1 = [4, 5, 3, 2]
time_line2 = [2, 10, 1, 4]
transfer_line1 = [0, 7, 4, 5]
transfer_line2 = [0, 9, 2, 8]
entry1, entry2 = 10, 12
exit1, exit2 = 18, 7
result = assembly_line(time_line1, time_line2, transfer_line1, transfer_line2, entry1, entry2, exit1, exit2)
print(f"Minimum time to process the product: {result}")

```

```
Minimum time to process the product: 35

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

30°C Partly sunny | Search | ENG IN | 16:18 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run | Output | Clear

```
def assembly_line_3_lines(time_line1, time_line2, time_line3,
    transfer_line1, transfer_line2, transfer_line3, entry1, entry2,
    entry3, exit1, exit2, exit3):
    n = len(time_line1)
    dp1 = [0] * n
    dp2 = [0] * n
    dp3 = [0] * n
    dp1[0] = entry1 + time_line1[0]
    dp2[0] = entry2 + time_line2[0]
    dp3[0] = entry3 + time_line3[0]
    for i in range(1, n):
        dp1[i] = min(dp1[i-1] + time_line1[i], dp2[i-1] +
            transfer_line1[i-1] + time_line1[i], dp3[i-1] +
            transfer_line1[i-1] + time_line1[i])
        dp2[i] = min(dp2[i-1] + time_line2[i], dp1[i-1] +
            transfer_line2[i-1] + time_line2[i], dp3[i-1] +
            transfer_line2[i-1] + time_line2[i])
        dp3[i] = min(dp3[i-1] + time_line3[i], dp1[i-1] +
            transfer_line3[i-1] + time_line3[i], dp2[i-1] +
            transfer_line3[i-1] + time_line3[i])
    final_time1 = dp1[n-1] + exit1
    final_time2 = dp2[n-1] + exit2
    final_time3 = dp3[n-1] + exit3
    return min(final_time1, final_time2, final_time3)
time_line1 = [4, 5, 3, 2]
time_line2 = [2, 10, 1, 4]
time_line3 = [3, 8, 2, 6]
```

```
Minimum time to process the product: 35

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


30°C Partly sunny | Search | ENG IN | 16:21 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

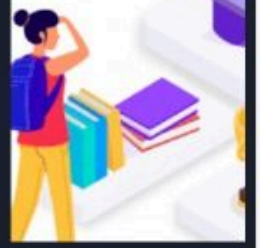
https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run | Output | Clear

```
        dp1[i] = min(dp1[i-1] + time_line1[i], dp2[i-1] +
            transfer_line1[i-1] + time_line1[i], dp3[i-1] +
            transfer_line1[i-1] + time_line1[i])
        dp2[i] = min(dp2[i-1] + time_line2[i], dp1[i-1] +
            transfer_line2[i-1] + time_line2[i], dp3[i-1] +
            transfer_line2[i-1] + time_line2[i])
        dp3[i] = min(dp3[i-1] + time_line3[i], dp1[i-1] +
            transfer_line3[i-1] + time_line3[i], dp2[i-1] +
            transfer_line3[i-1] + time_line3[i])
    final_time1 = dp1[n-1] + exit1
    final_time2 = dp2[n-1] + exit2
    final_time3 = dp3[n-1] + exit3
    return min(final_time1, final_time2, final_time3)
time_line1 = [4, 5, 3, 2]
time_line2 = [2, 10, 1, 4]
time_line3 = [3, 8, 2, 6]
transfer_line1 = [0, 7, 4, 5]
transfer_line2 = [0, 9, 2, 8]
transfer_line3 = [0, 6, 3, 9]
entry1, entry2, entry3 = 10, 12, 14
exit1, exit2, exit3 = 18, 7, 9
result = assembly_line_3_lines(time_line1, time_line2, time_line3,
    transfer_line1, transfer_line2, transfer_line3, entry1, entry2,
    entry3, exit1, exit2, exit3)
print(f"Minimum time to process the product: {result}")

```

```
Minimum time to process the product: 35

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

30°C Partly sunny | Search | ENG IN | 16:21 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

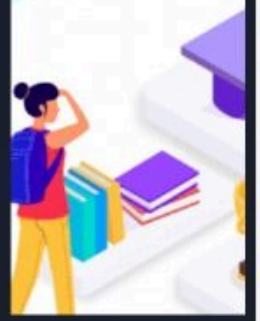
Programiz
Python Online Compiler

main.py | Share | Run

```python
def min_path_sum(matrix):
    if not matrix or not matrix[0]:
        return 0
    m, n = len(matrix), len(matrix[0])
    dp = [[0 for _ in range(n)] for _ in range(m)]
    dp[0][0] = matrix[0][0]
    for j in range(1, n):
        dp[0][j] = dp[0][j-1] + matrix[0][j]
    for i in range(1, m):
        dp[i][0] = dp[i-1][0] + matrix[i][0]
    for i in range(1, m):
        for j in range(1, n):
            dp[i][j] = matrix[i][j] + min(dp[i-1][j], dp[i][j-1])
    return dp[m-1][n-1]
matrix = [
    [1, 3, 1],
    [1, 5, 1],
    [4, 2, 1]
]
result = min_path_sum(matrix)
print(f"The minimum path sum is: {result}")

```

Output | Clear

```
The minimum path sum is: 7

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

30°C Partly sunny | Search | ENG IN | 16:23 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run | Output | Clear

```python
import math
def tsp(dist):
    n = len(dist)
    dp = [[math.inf] * n for _ in range(1 << n)]
    dp[1][0] = 0
    for mask in range(1 << n):
        for i in range(n):
            if not (mask & (1 << i)):
                continue
            for j in range(n):
                if mask & (1 << j):
                    continue
                next_mask = mask | (1 << j)
                dp[next_mask][j] = min(dp[next_mask][j], dp[mask][i]
                    + dist[i][j])
    final_result = math.inf
    for i in range(1, n):
        final_result = min(final_result, dp[(1 << n) - 1][i] +
            dist[i][0])
    return final_result
dist = [
    [0, 10, 15, 20, 25],
    [10, 0, 35, 25, 30],
    [15, 35, 0, 30, 20],
    [20, 25, 30, 0, 15],
    [25, 30, 20, 15, 0]
result = tsp(dist)
```

```
The minimum path sum is: 7

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


30°C Partly sunny | Search | ENG IN | 16:25 18-10-2024

DAY-7[1].pdf | python online compiler - Search | Online Python Compiler (Interpre | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

main.py | Share | Run | Output | Clear

```
    dp[1][0] = 0
    for mask in range(1 << n):
        for i in range(n):
            if not (mask & (1 << i)):
                continue
            for j in range(n):
                if mask & (1 << j):
                    continue
                next_mask = mask | (1 << j)
                dp[next_mask][j] = min(dp[next_mask][j], dp[mask][i]
                    + dist[i][j])
    final_result = math.inf
    for i in range(1, n):
        final_result = min(final_result, dp[(1 << n) - 1][i] +
            dist[i][0])
    return final_result
dist = [
    [0, 10, 15, 20, 25],
    [10, 0, 35, 25, 30],
    [15, 35, 0, 30, 20],
    [20, 25, 30, 0, 15],
    [25, 30, 20, 15, 0]
result = tsp(dist)
print(f"The minimum traveling cost for TSP with 5 cities is:
    {result}")

```

```
The minimum path sum is: 7

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


30°C Partly sunny | Search | ENG IN | 16:26 18-10-2024

Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz

Python Online Compiler

Programiz PRO >

main.py

Share

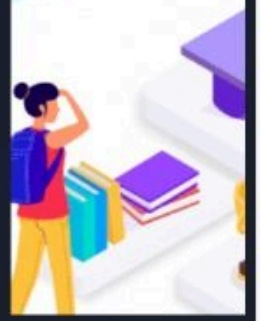
Run

Output

Clear

```python
def expand_around_center(s, left, right):
    while left >= 0 and right < len(s) and s[left] == s[right]:
        left -= 1
        right += 1
    return s[left + 1:right]
def longest_palindrome(s):
    if len(s) == 0:
        return ""
    longest_pal = ""
    for i in range(len(s)):
        odd_pal = expand_around_center(s, i, i)
        even_pal = expand_around_center(s, i, i + 1)
        longest_pal = max(longest_pal, odd_pal, even_pal, key=len)
    return longest_pal
s = "babad"
result = longest_palindrome(s)
print(f"The longest palindromic substring is: '{result}'")

```

```
The longest palindromic substring is: 'bab'

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

30°C Partly sunny

Search

ENG IN

16:28 18-10-2024

Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz

Python Online Compiler

Programiz PRO >

main.py

Share

Run

```python
def length_of_longest_substring(s: str) -> int:
    char_index = {}
    left = 0
    max_length = 0
    for right in range(len(s)):
        current_char = s[right]
        if current_char in char_index and char_index[current_char] >= left:
            left = char_index[current_char] + 1
        char_index[current_char] = right
        max_length = max(max_length, right - left + 1)
    return max_length
s = "abcabcbb"
result = length_of_longest_substring(s)
print(f"The length of the longest substring without repeating characters is: {result}")

```

Output

Clear

```
The length of the longest substring without repeating characters is: 3

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


30°C Partly sunny

Search

ENG IN

16:29 18-10-2024

Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz

Python Online Compiler

Programiz PRO >

main.py

Share

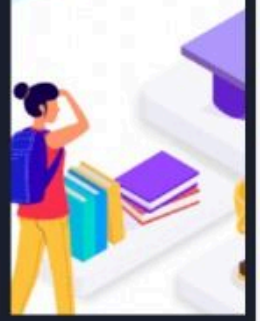
Run

```python
def word_break(s: str, wordDict: list) -> bool:
    word_set = set(wordDict)
    dp = [False] * (len(s) + 1)
    dp[0] = True
    for i in range(1, len(s) + 1):
        for j in range(i):
            if dp[j] and s[j:i] in word_set:
                dp[i] = True
                break
    return dp[len(s)]
s = "leetcode"
wordDict = ["leet", "code"]
result = word_break(s, wordDict)
print(f"Can the string '{s}' be segmented? {result}")

```

Output

Clear

```
Can the string 'leetcode' be segmented? True

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

30°C Partly sunny

Search

ENG IN

16:31 18-10-2024

Online Python Compiler (Interpre

https://www.programiz.com/python-programming/online-compiler/

Programiz

Python Online Compiler

Programiz PRO >

main.py

Share

Run

```
def word_break(s: str, wordDict: list) -> bool:
    word_set = set(wordDict)
    dp = [False] * (len(s) + 1)
    dp[0] = True
    for i in range(1, len(s) + 1):
        for j in range(i):
            if dp[j] and s[j:i] in word_set:
                dp[i] = True
                break
    return dp[len(s)]
s = "ilikesamsung"
wordDict = ["i", "like", "sam", "sung", "samsung", "mobile", "ice",
    "cream", "icecream", "man", "go", "mango"]
result = word_break(s, wordDict)
print(f"Can the string '{s}' be segmented? {result}")

```

Output

Clear

```
Can the string 'ilikesamsung' be segmented? True

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


30°C Partly sunny

Search

ENG IN

16:33 18-10-2024

Online Python Compiler (Interpre | ChatGPT | OpenAI | ChatGPT

https://www.programiz.com/python-programming/online-compiler/

Programiz

Python Online Compiler

Programiz PRO >

main.py

Share | Run

```python
def fullJustify(words, maxWidth):
    res = []
    cur_line = []
    num_of_letters = 0
    for word in words:
        if num_of_letters + len(word) + len(cur_line) > maxWidth:
            for i in range(maxWidth - num_of_letters):
                cur_line[i % (len(cur_line) - 1 or 1)] += ' '
            res.append(''.join(cur_line))
            cur_line, num_of_letters = [], 0
        cur_line.append(word)
        num_of_letters += len(word)
    res.append(' '.join(cur_line).ljust(maxWidth))
    return res
words = ["This", "is", "an", "example", "of", "text", "justification
    ."]
maxWidth = 16
result = fullJustify(words, maxWidth)
for line in result:
    print(f'"{line}"')

```

Output | Clear

```
"This    is    an"
"example  of text"
"justification.  "

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


30°C Partly sunny

Search

ENG IN

16:35 18-10-2024

Online Python Compiler (Interpre | ChatGPT | OpenAI | ChatGPT | DAY-7[1].pdf

https://www.programiz.com/python-programming/online-compiler/

Programiz
Python Online Compiler

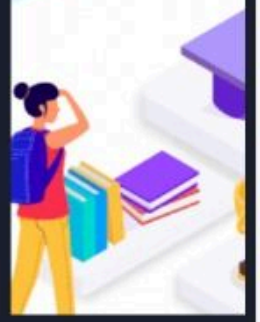
Programiz PRO >

main.py | Share | Run

```
class WordFilter:
    def __init__(self, words):
        self.words = words
        self.prefix_suffix_map = {}
        for index, word in enumerate(words):
            for i in range(len(word) + 1):
                for j in range(len(word) + 1):
                    prefix = word[:i]
                    suffix = word[j:]
                    key = (prefix, suffix)
                    self.prefix_suffix_map[key] = index
    def f(self, pref, suff):
        return self.prefix_suffix_map.get((pref, suff), -1)
words = ["apple", "banana", "app", "apricot"]
word_filter = WordFilter(words)
print(word_filter.f("ap", "le"))
print(word_filter.f("app", ""))
print(word_filter.f("ban", "na"))
print(word_filter.f("ap", "cot"))

```

Output | Clear

```
0
2
1
3

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

30°C Partly sunny | Search | ENG IN | 16:38 18-10-2024