Programiz
Python Online Compiler


Programiz PRO >

main.py | Share | Run

```python
def is_palindromic(s: str) -> bool:

    return s == s[::-1]

def first_palindromic_string(words: list[str]) -> str:
    for word in words:
        if is_palindromic(word):
            return word
    return ""


words = ["abc", "car", "ada", "racecar", "cool"]
print(first_palindromic_string(words))
```

Output | Clear

```
ada

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

Programiz
Python Online Compiler



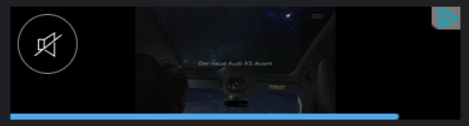
main.py | Share | Run

```python
def brute_force_string_matching(text, pattern):
    text_length = len(text)
    pattern_length = len(pattern)

    for i in range(text_length - pattern_length + 1):
        match = True
        for j in range(pattern_length):
            if text[i + j] != pattern[j]:
                match = False
                break

        if match:
            return i
    return -1


text = "Look at that Red rose"
pattern = "at"
index = brute_force_string_matching(text, pattern)
if index != -1:
    print(f"Pattern '{pattern}' found at index {index}.")
else:
    print(f"Pattern '{pattern}' not found.")

```

Output | Clear

```
Pattern 'at' found at index 5.

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

Programiz
Python Online Compiler



Programiz PRO >

main.py | Share | Run | Output | Clear

```python
import numpy as np

def strassen_multiply(A, B):
    n = len(A)

    if n == 1:
        return A * B

    mid = n // 2
    A11 = A[:mid, :mid]
    A12 = A[:mid, mid:]
    A21 = A[mid:, :mid]
    A22 = A[mid:, mid:]
    B11 = B[:mid, :mid]
    B12 = B[:mid, mid:]
    B21 = B[mid:, :mid]
    B22 = B[mid:, mid:]

    M1 = strassen_multiply(A11 + A22, B11 + B22)
    M2 = strassen_multiply(A21 + A22, B11)
    M3 = strassen_multiply(A11, B12 - B22)
    M4 = strassen_multiply(A22, B21 - B11)
    M5 = strassen_multiply(A11 + A12, B22)
    M6 = strassen_multiply(A21 - A11, B11 + B12)
    M7 = strassen_multiply(A12 - A22, B21 + B22)

    C11 = M1 + M4 - M5 + M7
    C12 = M3 + M5
    C21 = M2 + M4
```

```
[[ 8.  8.  8.  8.]
 [16. 16. 16. 16.]
 [24. 24. 24. 24.]
 [16. 16. 16. 16.]]

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

main.py

Share Run

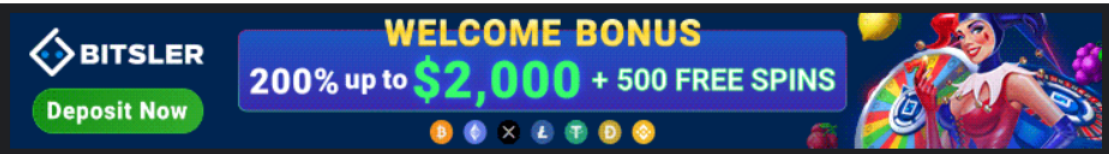
```
24      M6 = strassen_multiply(A21 - A11, B11 + B12)
25      M7 = strassen_multiply(A12 - A22, B21 + B22)
26
27      C11 = M1 + M4 - M5 + M7
28      C12 = M3 + M5
29      C21 = M2 + M4
30      C22 = M1 - M2 + M3 + M6
31
32      C = np.zeros((n, n))
33      C[:mid, :mid] = C11
34      C[:mid, mid:] = C12
35      C[mid:, :mid] = C21
36      C[mid:, mid:] = C22
37
38      return C
39
40  A = np.array([[1, 1, 1, 1],
41                [2, 2, 2, 2],
42                [3, 3, 3, 3],
43                [2, 2, 2, 2]])
44
45  B = np.array([[1, 1, 1, 1],
46                [2, 2, 2, 2],
47                [3, 3, 3, 3],
48                [2, 2, 2, 2]])
49
50  result = strassen_multiply(A, B)
51  print(result)
52
```

Output

Clear

```
[[ 8.  8.  8.  8.]
 [16. 16. 16. 16.]
 [24. 24. 24. 24.]
 [16. 16. 16. 16.]]

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

Programiz
Python Online Compiler



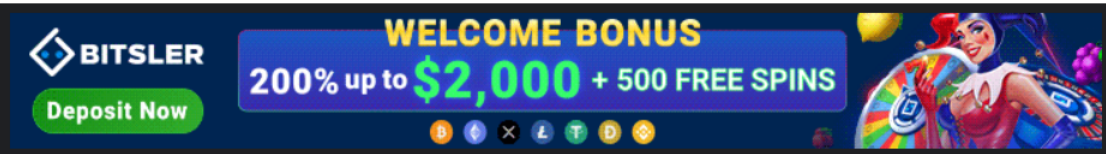

Programiz PRO >

main.py | Share | Run | Output | Clear

```python
def schedule_tasks(jobs, deadlines, profits):
    n = len(jobs)

    job_list = [(jobs[i], deadlines[i], profits[i]) for i in range(n)]

    job_list.sort(key=lambda x: x[2], reverse=True)

    max_deadline = max(deadlines)
    slots = [-1] * (max_deadline + 1)
    total_profit = 0

    for job, deadline, profit in job_list:

        for slot in range(deadline, 0, -1):
            if slots[slot] == -1:

                slots[slot] = job
                total_profit += profit
                break


    scheduled_jobs = [slot for slot in slots if slot != -1]

    print("Scheduled Jobs:", scheduled_jobs)
    print("Total Profit:", total_profit)


jobs = ['J1', 'J2', 'J3', 'J4', 'J5']
deadlines = [2, 2, 1, 3, 4]
```

```
Scheduled Jobs: ['J3', 'J2', 'J4', 'J5']
Total Profit: 280

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

Programiz
Python Online Compiler




Programiz PRO >

main.py | Share | Run

```
    job_list.sort(key=lambda x: x[2], reverse=True)

    max_deadline = max(deadlines)
    slots = [-1] * (max_deadline + 1)
    total_profit = 0

    for job, deadline, profit in job_list:

        for slot in range(deadline, 0, -1):
            if slots[slot] == -1:

                slots[slot] = job
                total_profit += profit
                break


    scheduled_jobs = [slot for slot in slots if slot != -1]

    print("Scheduled Jobs:", scheduled_jobs)
    print("Total Profit:", total_profit)


jobs = ['J1', 'J2', 'J3', 'J4', 'J5']
deadlines = [2, 2, 1, 3, 4]
profits = [20, 60, 40, 100, 80]

schedule_tasks(jobs, deadlines, profits)
```

Output | Clear

```
Scheduled Jobs: ['J3', 'J2', 'J4', 'J5']
Total Profit: 280

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

WhatsApp | TEAM 4 - C | WhatsApp | chat gpt - | ChatGPT | ChatGPT | python co | Online Pyt | Online Pyt | (22) Whats

programiz.com/python-programming/online-compiler/

Gmail YouTube Maps

Programiz
Python Online Compiler

Programiz PRO >

main.py

Share Run

Output Clear

```python
def find_max_and_min(arr, low, high):

    if low == high:
        return arr[low], arr[low]

    if low + 1 == high:
        if arr[low] < arr[high]:
            return arr[high], arr[low]
        else:
            return arr[low], arr[high]

    mid = (low + high) // 2
    max1, min1 = find_max_and_min(arr, low, mid)
    max2, min2 = find_max_and_min(arr, mid + 1, high)

    return max(max1, max2), min(min1, min2)

arr = [31, 22, 12, -7, 75, -6, 17, 47, 60]
max_val, min_val = find_max_and_min(arr, 0, len(arr) - 1)
print(f"Maximum: {max_val}, Minimum: {min_val}")

```

```
Maximum: 75, Minimum: -7

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

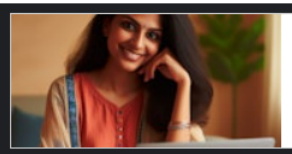
34°C Mostly cloudy

Search

ENG IN

14:18 27-08-2024

Programiz
Python Online Compiler



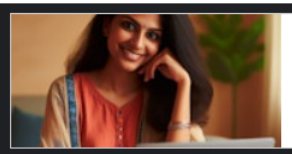


main.py | Share | Run | Output | Clear

```python
import math

def calculate_distance(point1, point2):
    """Calculate Euclidean distance between two points."""
    return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)

def find_closest_pair(points):
    """Find the closest pair of points and the minimum distance using brute force."""
    n = len(points)
    if n < 2:
        raise ValueError("At least two points are required to find the closest pair.")

    min_distance = float('inf')
    closest_pair = (None, None)

    for i in range(n):
        for j in range(i + 1, n):
            distance = calculate_distance(points[i], points[j])
            if distance < min_distance:
                min_distance = distance
                closest_pair = (points[i], points[j])

    return closest_pair, min_distance

points = [(1, 2), (4, 5), (7, 8), (3, 1)]
closest_pair, min_distance = find_closest_pair(points)

```

```
Closest pair: (1, 2) - (3, 1)
Minimum distance: 2.23606797749979

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

Programiz
Python Online Compiler





main.py

Share Run

```
4       Calculate Euclidean distance between two points.
5       return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)
6
7   def find_closest_pair(points):
8       """Find the closest pair of points and the minimum distance using brute
            force."""
9       n = len(points)
10      if n < 2:
11          raise ValueError("At least two points are required to find the closest
                pair.")
12
13      min_distance = float('inf')
14      closest_pair = (None, None)
15
16      for i in range(n):
17          for j in range(i + 1, n):
18              distance = calculate_distance(points[i], points[j])
19              if distance < min_distance:
20                  min_distance = distance
21                  closest_pair = (points[i], points[j])
22
23      return closest_pair, min_distance
24
25  points = [(1, 2), (4, 5), (7, 8), (3, 1)]
26  closest_pair, min_distance = find_closest_pair(points)
27
28  print(f"Closest pair: {closest_pair[0]} - {closest_pair[1]}")
29  print(f"Minimum distance: {min_distance}")
30
```

Output Clear

```
Closest pair: (1, 2) - (3, 1)
Minimum distance: 2.23606797749979

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

(25) WhatsApp | TEAM 4 - Google Docs | ChatGPT | Online Python Compiler and Vis

codechef.com/python-online-compiler

Gmail YouTube Maps

Python3

```python
def can_segment_string(s, dictionary):

    dp = [False] * (len(s) + 1)
    dp[0] = True


    word_set = set(dictionary)


    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)]

dictionary = ["mobile", "Samsung", "sam", "sung", "man", "mango", "icecream",
    "and", "go", "I", "like"]
input_string = "samsungandmango"

result = can_segment_string(input_string, dictionary)
print(result)
```

Run | Visualize Code

Enter Input here

If your code takes input, add it in the above box before running.

Output

Status : Successfully executed

Time: 0.0100 secs

Memory: 8.332 Mb

Your Output

True

History

python online compiler

"python code runner" 2 days ago

**Online Python Compiler** and Visualizer
codechef.com/**python-online-compiler**

**Online Python Compiler**
online-python.com/online_python_compiler

**Online Python Compiler - online** editor
onlinegdb.com/online_python_compiler

python code runner - Yahoo India Search R...
in.search.yahoo.com/search?fr=mcafee&type=E...

"online python compiler" 2 days ago

**online python compiler** - Yahoo India Sear...
in.search.yahoo.com/search?fr=mcafee&type=E...

"python online compiler" 31 days ago

**python online compiler** - Yahoo India Sear...
in.search.yahoo.com/search?fr=mcafee&type=E...

Online HTML Editor
programiz.com/html/online-compiler/

Search

ENG IN 10:16 29-08-2024

(25) WhatsApp | TEAM 4 - Google Docs | ChatGPT | Online Python Compiler and Vis

codechef.com/python-online-compiler

Gmail YouTube Maps

# Online Python Compiler and Visualizer

Learning Python? Check out our complete Python roadmap

Python3

```python
def find_subsets_with_sum(nums, target_sum):
    def backtrack(start, current_sum, path):
        if current_sum == target_sum:
            print(path)
            return
        if current_sum > target_sum or start >= len(nums):
            return

        path.append(nums[start])
        backtrack(start + 1, current_sum + nums[start], path)
        path.pop()
        backtrack(start + 1, current_sum, path)

    backtrack(0, 0, [])

nums1 = [3, 34, 4, 12, 5, 2]
target_sum1 = 30
find_subsets_with_sum(nums1, target_sum1)
```

Run | Visualize Code

Ctrl⇧↵ / ⌘⇧↵

t here

If your code takes input, add it in the above box before running.

Output

Status : Successfully executed

Time: 0.0200 secs

Memory: 8.256 Mb

No details to show

History

python online compiler

"python code runner" 2 days ago

- **Online Python Compiler** and Visualizer — codechef.com/**python-online-compiler**
- **Online Python Compiler** — online-python.com/online_python_compiler
- **Online Python Compiler - online** editor — onlinegdb.com/online_python_compiler
- python code runner - Yahoo India Search R... — in.search.yahoo.com/search?fr=mcafee&type=E...

"online python compiler" 2 days ago

- **online python compiler** - Yahoo India Sear... — in.search.yahoo.com/search?fr=mcafee&type=E...

"python online compiler" 31 days ago

- **python online compiler** - Yahoo India Sear... — in.search.yahoo.com/search?fr=mcafee&type=E...
- Online HTML Editor — programiz.com/html/online-compiler/

High winds In 1 hour

Search

ENG IN

10:22 29-08-2024

(22) WhatsApp | TEAM 4 - Google Docs | ChatGPT | Online R Compiler - Programiz

programiz.com/r/online-compiler/

Gmail YouTube Maps

Programiz
R Online Compiler


Programiz PRO >

main.r | Share | Run | Output | Clear

```r
import math

def euclidean_distance(point1, point2):

    dx=point1[0]-point2[0]
    dy=point1[1]-point2[1]
    return math.sqrt(dx * dx+dy*dy)

def closest_pair_of_points(points):
    min_distance=float('inf')
    closest_pair=(None,None)

    num_points=len(points)

    for i in range(num_points):
        for j in range(i+1,num_points):
            distance=euclidean_distance(points[i],points[j])
            if distance<min_distance:
                min_distance=distance
                closest_pair=(points[i],points[j])

    return closest_pair,min_distance

points = [(1,2),(4,5),(7,8),(3,1)]
closest_pair,min_distance=closest_pair_of_points(points)

print(f"Closest pair: {closest_pair[0]} - {closest_pair[1]}")
print(f"Minimum distance: {min_distance}")
```

```
output: P3, P4, P6, P5, P7, P1
```

Search

ENG IN

20:46 29-08-2024

(22) WhatsApp | TEAM 4 - Google Docs | ChatGPT | Online R Compiler - Programiz

programiz.com/r/online-compiler/

Gmail YouTube Maps

Programiz
R Online Compiler


Programiz PRO >

main.r | Share | Run | Output | Clear

```r
import math

def calculate_distance(point1, point2):
    dx = point1[0] - point2[0]
    dy = point1[1] - point2[1]
    return math.sqrt(dx * dx + dy * dy)

def find_closest_pair(points):
    min_distance = float('inf')
    closest_points = (None, None)

    num_points = len(points)

    for i in range(num_points):
        for j in range(i + 1, num_points):
            distance = calculate_distance(points[i], points[j])
            if distance < min_distance:
                min_distance = distance
                closest_points = (points[i], points[j])

    return closest_points, min_distance

points = [(1, 2), (4, 5), (7, 8), (3, 1)]
pair, distance = find_closest_pair(points)
print(f"Closest pair: {pair[0]} - {pair[1]}")
print(f"Minimum distance: {distance}")
```

```
Output:

      Closest pair: (1, 2) - (3, 1) Minimum distance: 1.4142135623730951
```

Search

ENG IN

20:40 29-08-2024