Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

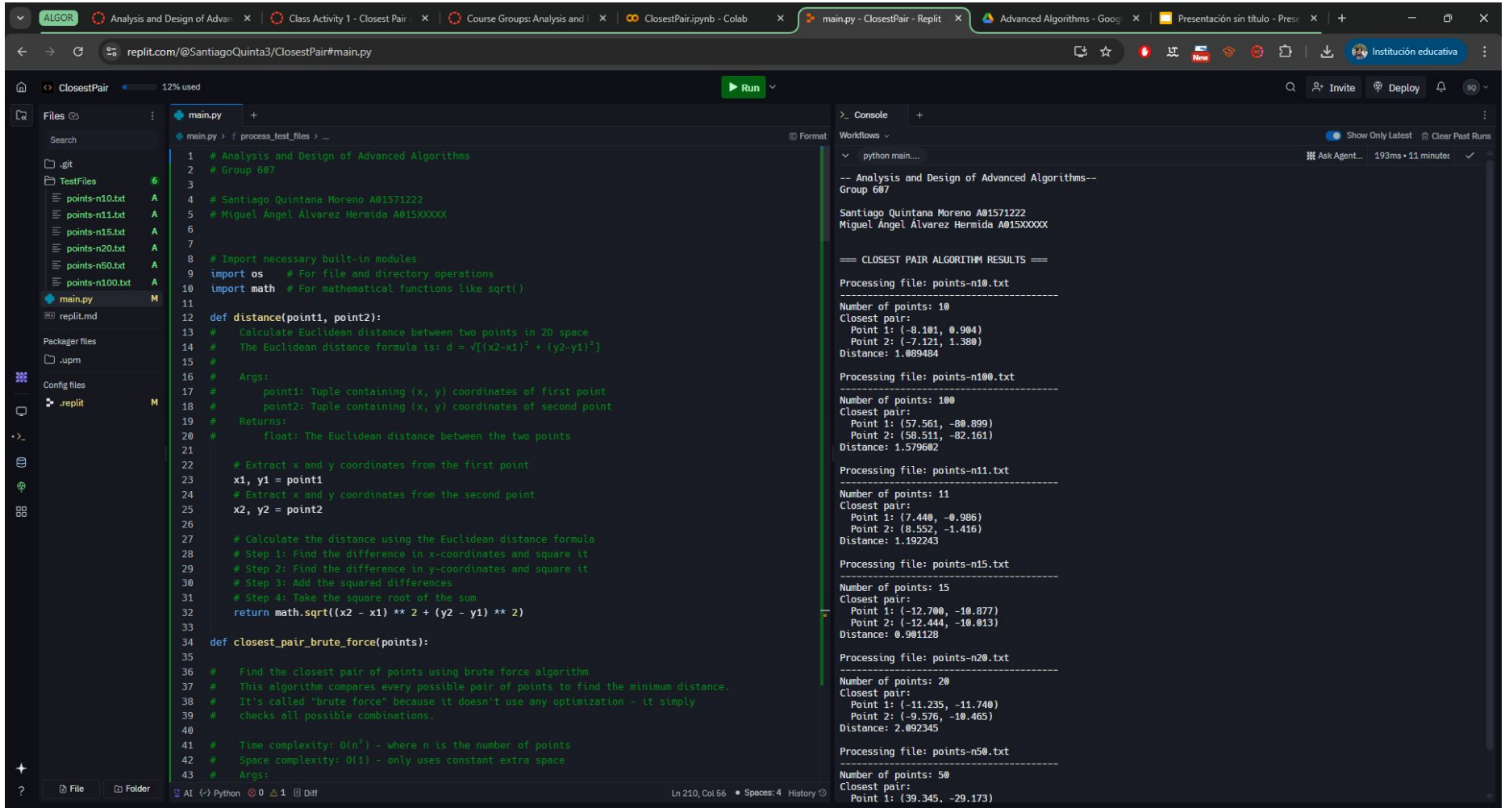
Class Activity 1:
Closest Pair and String Matching, by Brute Force

Group: 607
Team #6
Dr. Katie Brodhead

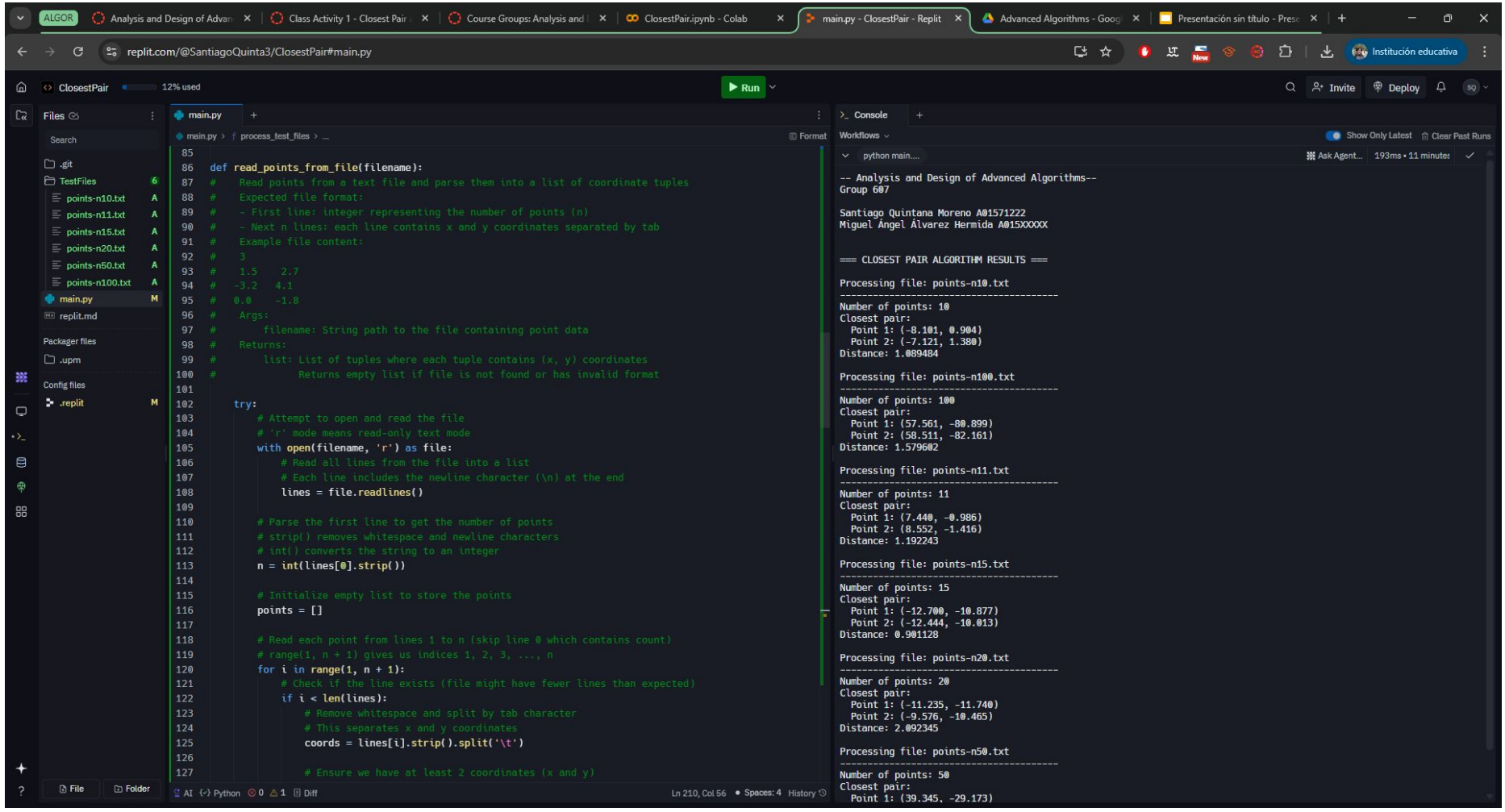
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

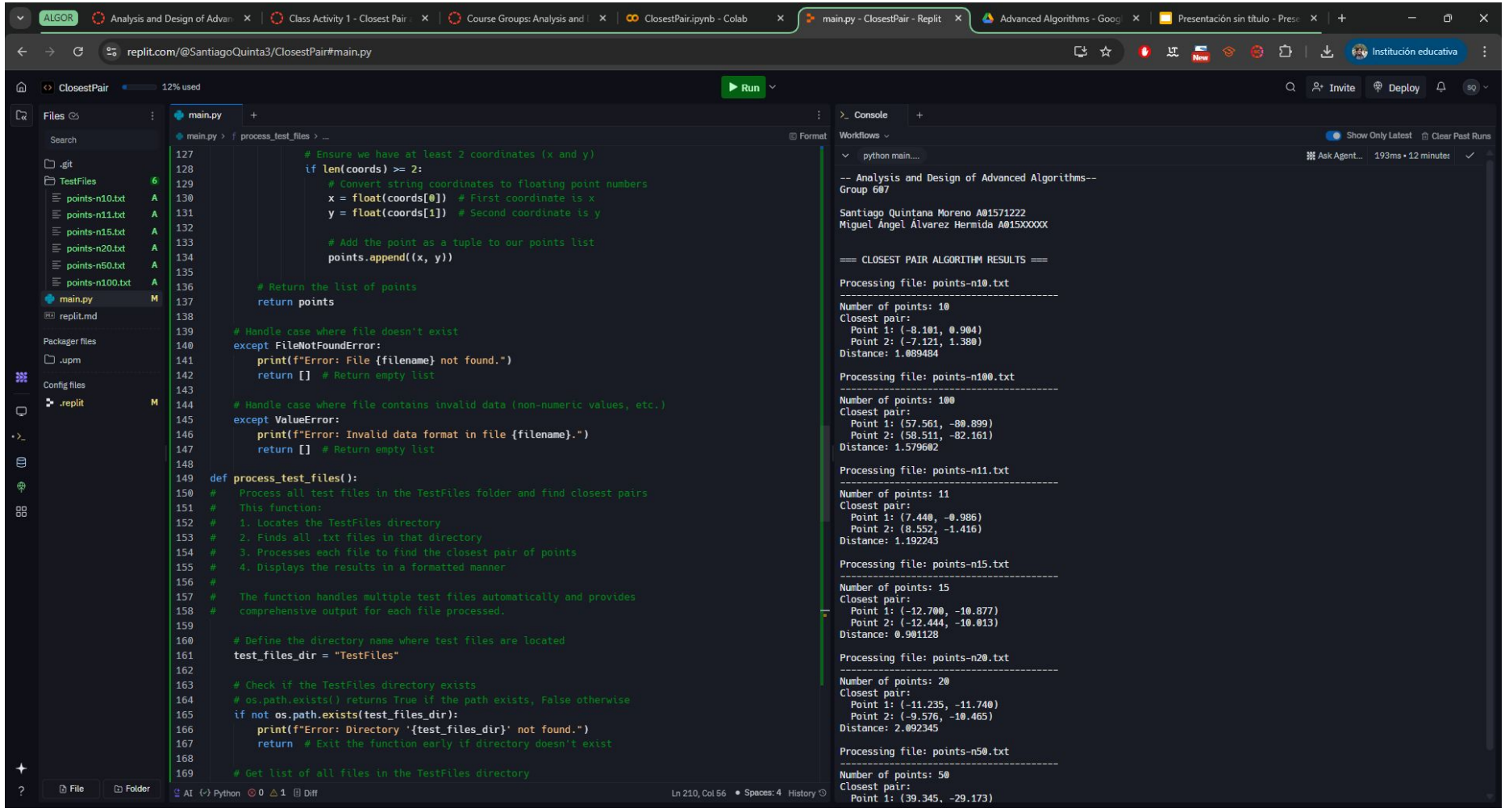
Closest Pair

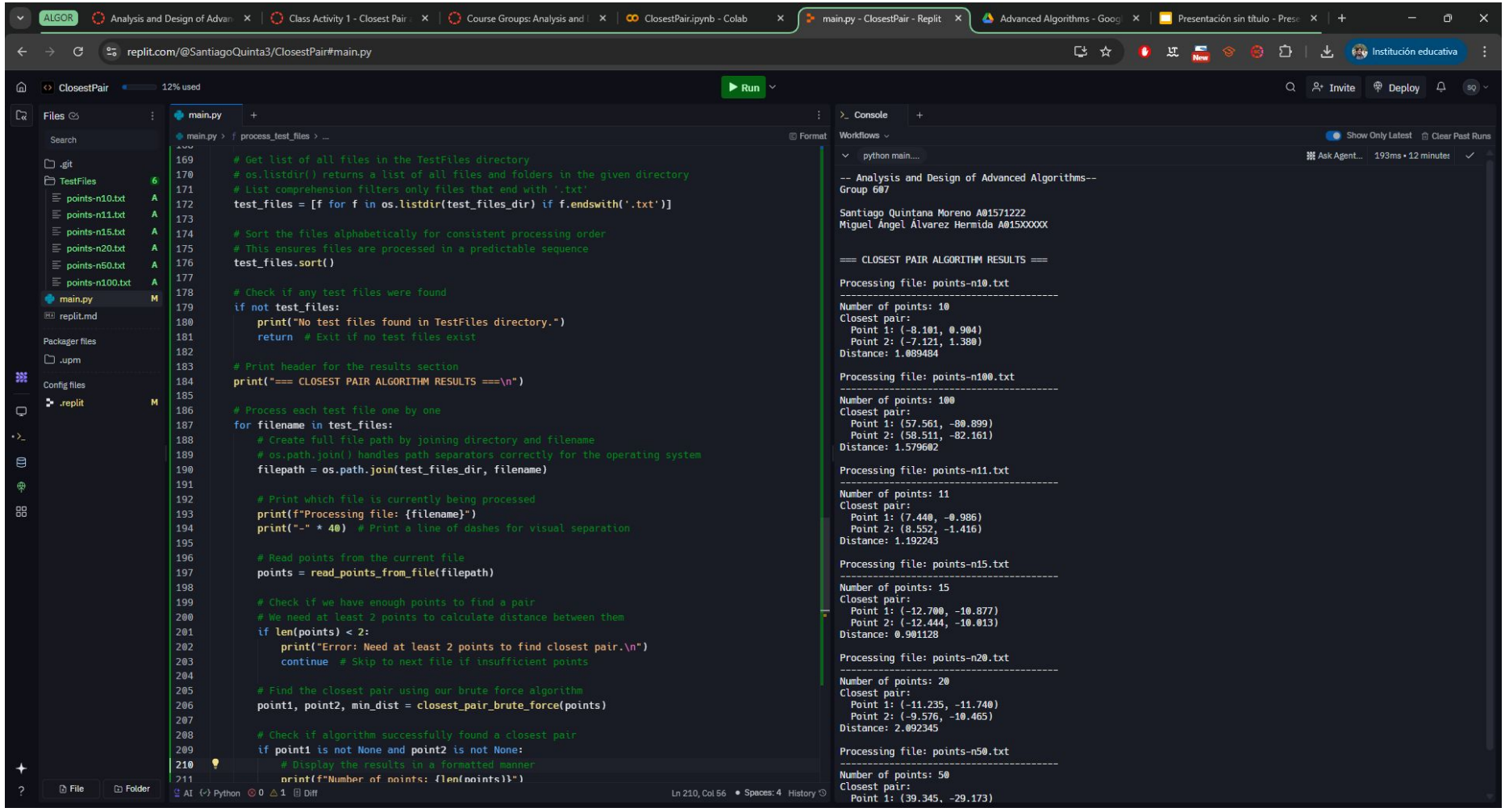
https://colab.research.google.com/drive/1n-mzcIrjgDiGZT3DACrtkdRCA6fHvvuv?usp=sharing

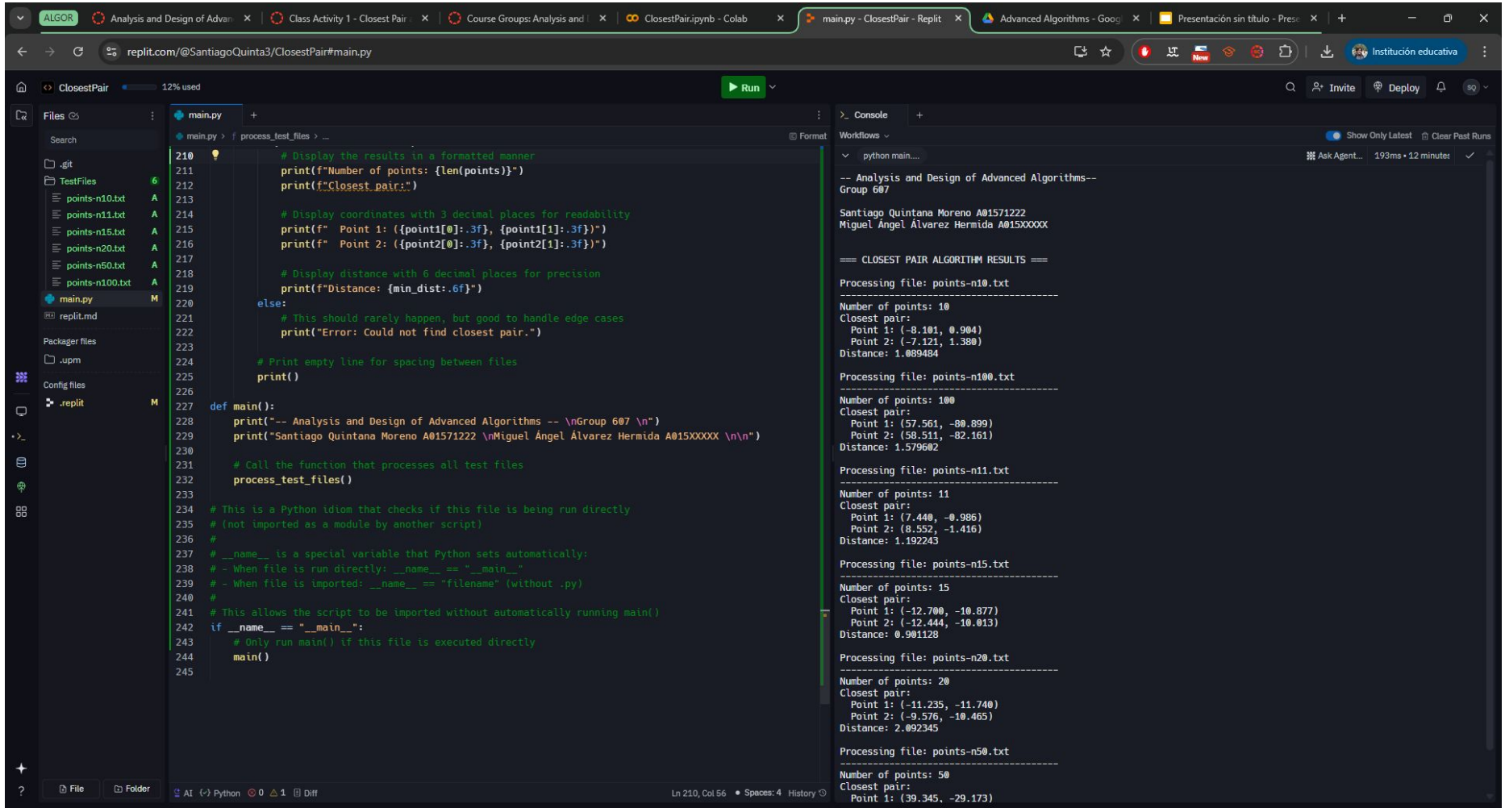

ALGOR
Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles
points-n10.txt
points-n11.txt
points-n15.txt
points-n20.txt
points-n50.txt
points-n100.txt
main.py
replit.md
Packager files
.upm
Config files
.replit
main.py
main.py › process_test_files › ...
Format
1 # Analysis and Design of Advanced Algorithms
2 # Group 607
3
4 # Santiago Quintana Moreno A01571222
5 # Miguel Ángel Álvarez Hermida A015XXXXX
6
7
8 # Import necessary built-in modules
9 import os # For file and directory operations
10 import math # For mathematical functions like sqrt()
11
12 def distance(point1, point2):
13 # Calculate Euclidean distance between two points in 2D space
14 # The Euclidean distance formula is: d = √[(x2-x1)² + (y2-y1)²]
15 #
16 # Args:
17 # point1: Tuple containing (x, y) coordinates of first point
18 # point2: Tuple containing (x, y) coordinates of second point
19 # Returns:
20 # float: The Euclidean distance between the two points
21
22 # Extract x and y coordinates from the first point
23 x1, y1 = point1
24 # Extract x and y coordinates from the second point
25 x2, y2 = point2
26
27 # Calculate the distance using the Euclidean distance formula
28 # Step 1: Find the difference in x-coordinates and square it
29 # Step 2: Find the difference in y-coordinates and square it
30 # Step 3: Add the squared differences
31 # Step 4: Take the square root of the sum
32 return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
33
34 def closest_pair_brute_force(points):
35
36 # Find the closest pair of points using brute force algorithm
37 # This algorithm compares every possible pair of points to find the minimum distance.
38 # It's called "brute force" because it doesn't use any optimization - it simply
39 # checks all possible combinations.
40
41 # Time complexity: O(n²) - where n is the number of points
42 # Space complexity: O(1) - only uses constant extra space
43 # Args:
AI
Python
0
1
Diff
Ln 210, Col 56
Spaces: 4
History
File
Folder
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 11 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)


ALGOR
Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles 6
points-n10.txt A
points-n11.txt A
points-n15.txt A
points-n20.txt A
points-n50.txt A
points-n100.txt A
main.py M
replit.md
Packager files
.upm
Config files
.replit M
File
Folder
main.py
main.py › process_test_files › ...
Format
43 # Args:
44 # points: List of tuples, where each tuple contains (x, y) coordinates
45 # Returns:
46 # tuple: (point1, point2, min_distance) where:
47 # - point1: First point of the closest pair
48 # - point2: Second point of the closest pair
49 # - min_distance: The distance between the closest pair
50
51 # Get the total number of points in the list
52 n = len(points)
53
54 # Edge case: If we have fewer than 2 points, we can't find a pair
55 if n < 2:
56 return None, None, float('inf')
57
58 # Initialize minimum distance to infinity (largest possible value)
59 # This ensures any real distance will be smaller
60 min_distance = float('inf')
61
62 # Initialize the closest pair as empty tuple
63 # This will store the two points that are closest to each other
64 closest_pair = (None, None)
65
66 # Outer loop: iterate through each point as the first point of a pair
67 # We go from 0 to n-1 (all points except the last one)
68 for i in range(n):
69 # Inner loop: iterate through remaining points as the second point of a pair
70 # We start from i+1 to avoid comparing a point with itself
71 # and to avoid duplicate comparisons (comparing point A with B and B with A)
72 for j in range(i + 1, n):
73 # Calculate the distance between current pair of points
74 dist = distance(points[i], points[j])
75
76 # If this distance is smaller than our current minimum
77 if dist < min_distance:
78 # Update the minimum distance
79 min_distance = dist
80 # Update the closest pair with current points
81 closest_pair = (points[i], points[j])
82
83 # Return the closest pair and their distance
84 return closest_pair[0], closest_pair[1], min_distance
85
AI
Python
0 1
Diff
Ln 210, Col 56
Spaces: 4
History
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 11 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)


ALGOR
Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles
points-n10.txt
points-n11.txt
points-n15.txt
points-n20.txt
points-n50.txt
points-n100.txt
main.py
replit.md
Packager files
.upm
Config files
.replit
File
Folder
main.py
main.py › process_test_files › ...
Format
def read_points_from_file(filename):
# Read points from a text file and parse them into a list of coordinate tuples
# Expected file format:
# - First line: integer representing the number of points (n)
# - Next n lines: each line contains x and y coordinates separated by tab
# Example file content:
# 3
# 1.5 2.7
# -3.2 4.1
# 0.0 -1.8
# Args:
# filename: String path to the file containing point data
# Returns:
# list: List of tuples where each tuple contains (x, y) coordinates
# Returns empty list if file is not found or has invalid format
try:
# Attempt to open and read the file
# 'r' mode means read-only text mode
with open(filename, 'r') as file:
# Read all lines from the file into a list
# Each line includes the newline character (\n) at the end
lines = file.readlines()
# Parse the first line to get the number of points
# strip() removes whitespace and newline characters
# int() converts the string to an integer
n = int(lines[0].strip())
# Initialize empty list to store the points
points = []
# Read each point from lines 1 to n (skip line 0 which contains count)
# range(1, n + 1) gives us indices 1, 2, 3, ..., n
for i in range(1, n + 1):
# Check if the line exists (file might have fewer lines than expected)
if i < len(lines):
# Remove whitespace and split by tab character
# This separates x and y coordinates
coords = lines[i].strip().split('\t')
# Ensure we have at least 2 coordinates (x and y)
AI
Python
0
1
Diff
Ln 210, Col 56
Spaces: 4
History
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 11 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)


ALGOR
Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles 6
points-n10.txt A
points-n11.txt A
points-n15.txt A
points-n20.txt A
points-n50.txt A
points-n100.txt A
main.py M
replit.md
Packager files
.upm
Config files
.replit M
File
Folder
main.py
main.py › process_test_files › ...
Format
# Ensure we have at least 2 coordinates (x and y)
if len(coords) >= 2:
# Convert string coordinates to floating point numbers
x = float(coords[0]) # First coordinate is x
y = float(coords[1]) # Second coordinate is y
# Add the point as a tuple to our points list
points.append((x, y))
# Return the list of points
return points
Handle case where file doesn't exist
except FileNotFoundError:
print(f"Error: File {filename} not found.")
return [] # Return empty list
Handle case where file contains invalid data (non-numeric values, etc.)
except ValueError:
print(f"Error: Invalid data format in file {filename}.")
return [] # Return empty list
def process_test_files():
# Process all test files in the TestFiles folder and find closest pairs
# This function:
# 1. Locates the TestFiles directory
# 2. Finds all .txt files in that directory
# 3. Processes each file to find the closest pair of points
# 4. Displays the results in a formatted manner
#
# The function handles multiple test files automatically and provides
# comprehensive output for each file processed.
# Define the directory name where test files are located
test_files_dir = "TestFiles"
# Check if the TestFiles directory exists
# os.path.exists() returns True if the path exists, False otherwise
if not os.path.exists(test_files_dir):
print(f"Error: Directory '{test_files_dir}' not found.")
return # Exit the function early if directory doesn't exist
Get list of all files in the TestFiles directory
AI Python 0 1 Diff
Ln 210, Col 56 Spaces: 4 History
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 12 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)


Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles
points-n10.txt
points-n11.txt
points-n15.txt
points-n20.txt
points-n50.txt
points-n100.txt
main.py
replit.md
Packager files
.upm
Config files
.replit
File
Folder
main.py
main.py › process_test_files › ...
Format
# Get list of all files in the TestFiles directory
# os.listdir() returns a list of all files and folders in the given directory
# List comprehension filters only files that end with '.txt'
test_files = [f for f in os.listdir(test_files_dir) if f.endswith('.txt')]
# Sort the files alphabetically for consistent processing order
# This ensures files are processed in a predictable sequence
test_files.sort()
# Check if any test files were found
if not test_files:
print("No test files found in TestFiles directory.")
return # Exit if no test files exist
# Print header for the results section
print("=== CLOSEST PAIR ALGORITHM RESULTS ===\n")
# Process each test file one by one
for filename in test_files:
# Create full file path by joining directory and filename
# os.path.join() handles path separators correctly for the operating system
filepath = os.path.join(test_files_dir, filename)
# Print which file is currently being processed
print(f"Processing file: {filename}")
print("-" * 40) # Print a line of dashes for visual separation
# Read points from the current file
points = read_points_from_file(filepath)
# Check if we have enough points to find a pair
# We need at least 2 points to calculate distance between them
if len(points) < 2:
print("Error: Need at least 2 points to find closest pair.\n")
continue # Skip to next file if insufficient points
Find the closest pair using our brute force algorithm
point1, point2, min_dist = closest_pair_brute_force(points)
# Check if algorithm successfully found a closest pair
if point1 is not None and point2 is not None:
# Display the results in a formatted manner
print(f"Number of points: {len(points)}")
AI
Python
0
1
Diff
Ln 210, Col 56
Spaces: 4
History
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 12 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)


ALGOR
Analysis and Design of Advan
Class Activity 1 - Closest Pair
Course Groups: Analysis and
ClosestPair.ipynb - Colab
main.py - ClosestPair - Replit
Advanced Algorithms - Goog
Presentación sin título - Pres
replit.com/@SantiagoQuinta3/ClosestPair#main.py
Institución educativa
ClosestPair
12% used
Run
Invite
Deploy
Files
Search
.git
TestFiles
points-n10.txt
points-n11.txt
points-n15.txt
points-n20.txt
points-n50.txt
points-n100.txt
main.py
replit.md
Packager files
.upm
Config files
.replit
File
Folder
main.py
main.py › process_test_files › ...
Format
# Display the results in a formatted manner
print(f"Number of points: {len(points)}")
print(f"Closest pair:")
# Display coordinates with 3 decimal places for readability
print(f" Point 1: ({point1[0]:.3f}, {point1[1]:.3f})")
print(f" Point 2: ({point2[0]:.3f}, {point2[1]:.3f})")
# Display distance with 6 decimal places for precision
print(f"Distance: {min_dist:.6f}")
else:
# This should rarely happen, but good to handle edge cases
print("Error: Could not find closest pair.")
# Print empty line for spacing between files
print()
def main():
print("-- Analysis and Design of Advanced Algorithms -- \nGroup 607 \n")
print("Santiago Quintana Moreno A01571222 \nMiguel Ángel Álvarez Hermida A015XXXXX \n\n")
# Call the function that processes all test files
process_test_files()
# This is a Python idiom that checks if this file is being run directly
# (not imported as a module by another script)
#
__name__ is a special variable that Python sets automatically:
- When file is run directly: __name__ == "__main__"
- When file is imported: __name__ == "filename" (without .py)
#
# This allows the script to be imported without automatically running main()
if __name__ == "__main__":
# Only run main() if this file is executed directly
main()
AI
Python
0
1
Diff
Ln 210, Col 56
Spaces: 4
History
Console
Workflows
Show Only Latest
Clear Past Runs
python main....
Ask Agent...
193ms • 12 minutes
-- Analysis and Design of Advanced Algorithms--
Group 607
Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida A015XXXXX
=== CLOSEST PAIR ALGORITHM RESULTS ===
Processing file: points-n10.txt
Number of points: 10
Closest pair:
Point 1: (-8.101, 0.904)
Point 2: (-7.121, 1.380)
Distance: 1.089484
Processing file: points-n100.txt
Number of points: 100
Closest pair:
Point 1: (57.561, -80.899)
Point 2: (58.511, -82.161)
Distance: 1.579602
Processing file: points-n11.txt
Number of points: 11
Closest pair:
Point 1: (7.440, -0.986)
Point 2: (8.552, -1.416)
Distance: 1.192243
Processing file: points-n15.txt
Number of points: 15
Closest pair:
Point 1: (-12.700, -10.877)
Point 2: (-12.444, -10.013)
Distance: 0.901128
Processing file: points-n20.txt
Number of points: 20
Closest pair:
Point 1: (-11.235, -11.740)
Point 2: (-9.576, -10.465)
Distance: 2.092345
Processing file: points-n50.txt
Number of points: 50
Closest pair:
Point 1: (39.345, -29.173)

String Matching

https://colab.research.google.com/drive/1iPsbuKRs4kVK1gqeEn0miFX5mvm6B8z-?usp=sharing

```
def stringMatch(text, pattern):
    n = len(text) #Obtains length of the entire inputed text
    m = len(pattern) #Obtains length of the entire inputted pattern
    matches = [] #Empty array used for storing matches as they're found

    for i in range(n - m + 1):
        matchFound = True #for loop used for determining if the patter
has been found
        for j in range(m):
            if text[i + j] != pattern[j]:
                matchFound = False
                break #for loop used for ruling out non-matches
        if matchFound:
            matches.append(i) #if a match has been found, it will be
added to the array

    return matches #returns array with the completed pattern found from
the text
```

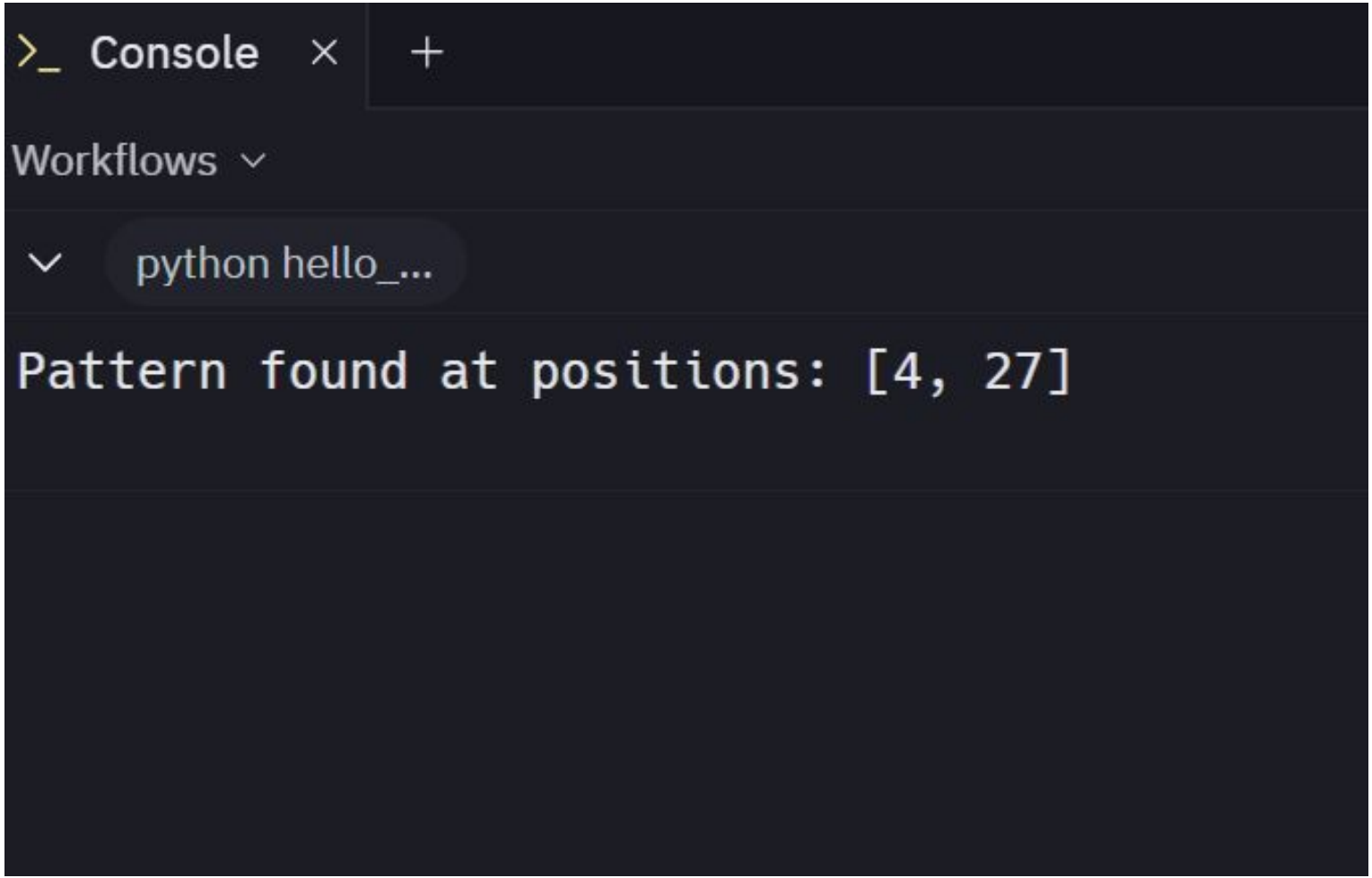
```
def main(): #main function with opens and reads chosen file
    with open("stringMatch.txt", "r", encoding="utf-8") as txt:
        lines = txt.read().splitlines()
        text = lines[0] #whole text is the first line of the file
        pattern = lines[1] #pattern is the second line of the text

    result = stringMatch(text, pattern)

    if result: #if-else statement for displaying the final result,
whether the pattern was found or not
        print(f"Pattern found at positions: {result}")
    else:
        print("Pattern not found")


main()


#O(nm)
#Ω(n)
```


Console
Workflows
python hello_...
Pattern found at positions: [4, 27]