```
import math

def find_nth_root(x, power, epsilon=0.0001):
    """
    Finds the nth root of a number using bisection method.

    Parameters:
    - x (float): The number to find the root of.
    - power (int): The root to compute (e.g., 2 for square root).
    - epsilon (float): Acceptable error margin.

    Returns:
    - float: Approximate nth root or None for invalid input.
    """
    if power <= 0:
        raise ValueError("Power must be a positive integer.")
    if x < 0 and power % 2 == 0:
        raise ValueError("Cannot find even-powered root of a negative number.")

    low = min(-1.0, x)
    high = max(1.0, x)
```

```
    ans = (high + low) / 2.0

    while abs(ans ** power - x) >= epsilon:
        if ans ** power < x:
            low = ans
        else:
            high = ans
        ans = (high + low) / 2.0

    return ans


def calculate_euclidean_distance(point1,
point2):
    """
    Calculates the Euclidean distance between
two 2D points.

    Parameters:
    - point1 (tuple): Coordinates (x1, y1)
    - point2 (tuple): Coordinates (x2, y2)

    Returns:
    - float: Euclidean distance
    """
    x1, y1 = point1
```

```
    x1, y1 = point1
    x2, y2 = point2
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)


# Example usage
if __name__ == "__main__":
    print("Nth Root Example:")
    try:
        x = float(input("Enter a number: "))
        p = int(input("Enter the root power: "))
        print(f"The {p}th root of {x} is
approximately {find_nth_root(x, p):.5f}")
    except ValueError as e:
        print(e)

    print("\nEuclidean Distance Example:")
    pt1 = (float(input("Enter x1: ")),
float(input("Enter y1: ")))
    pt2 = (float(input("Enter x2: ")),
float(input("Enter y2: ")))
    distance =
calculate_euclidean_distance(pt1, pt2)
    print(f"The distance between {pt1} and {pt2}
is {distance:.5f}")
```