MOMENTS

CODE:

```
import random
from functools import lru_cache
import math


def get_rand_matrix(min_d=2, max_d=10 ** 2, choices=(0, 1)):
    m, n = (random.randint(min_d, max_d) for _ in range(2))
    return tuple(tuple(random.choice(choices) for _ in range(n)) for _ in
range(m))


def get_1_points(graph, val=1, b=1):
    # returns the coordinates of the points having val as the cell value.
    # b is the offset of the b_based_indexing
    m = len(graph)
    n = len(graph[0])   # assuming graph is rectangle.
    for i in range(m):
        for j in range(n):
            if graph[i][j] == val:
                yield (i + b, j + b)


def calc_moments(graph):
    one_points = get_1_points(graph)
    mu = {0:0, 1:0, 2:0}
    for (x, y) in one_points:
        mu[0] += 1
        mu[1] += y
        mu[2] += x
    return mu


def calc_centroids(graph, mu=None):
    if mu is None:
        mu = calc_moments(graph)
    return (mu[2] / mu[0], mu[1] / mu[0])


def calc_central_moments(graph, centroids=None):
    m = len(graph)
    n = len(graph[0])   # assuming graph is rectangle.
    if centroids is None:
        xc, yc = calc_centroids(graph)
    else:
        xc, yc = centroids
    mu_prime = {2:0, 11:0, 20:0}
    for (x, y) in get_1_points(graph):
          mu_prime[20] += (x - xc) ** 2
```

```
            mu_prime[2] += (y - yc) ** 2
            mu_prime[11] += (x - xc) * (y - yc)
    return mu_prime


def calc_principal_angle(graph, mu_prime=None):
    if mu_prime is None:
        mu_prime = calc_central_moments(graph)
    return 0.5 * math.atan2(2 * mu_prime[11], mu_prime[20] - mu_prime[2])


def print_matrix(matrix):
    for row in matrix:
        __import__('builtins').print(row)


if __name__ == '__main__':
    matrix = get_rand_matrix(10, 20)
    print_matrix(matrix)
    moments = calc_moments(matrix)
    centroids = calc_centroids(matrix, moments)
    central_moments = calc_central_moments(matrix, centroids)
    principal_angle = calc_principal_angle(matrix, central_moments)
    print("Moments:", moments)
    print("Centroids:", centroids)
    print("Central Moments:", central_moments)
    print("Central Angle:", principal_angle)
```

OUTPUT:

```
(0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)
(0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1)
(0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1)
(1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1)
(1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1)
(1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1)
(1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0)
(1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0)
(1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0)
(1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0)
(1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)
(0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0)
(0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1)
(1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1)
(0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1)
(1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0)
(0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1)
(1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0)
Moments: {0: 129, 1: 940, 2: 1336}
```

```
Centroids: (10.356589147286822, 7.286821705426356)
Central Moments: {2: 1792.3875968992256, 11: 130.80620155038756, 20:
4559.596899224806}
Central Angle: 0.047129998342276096
```