```python
import numpy as np
import matplotlib.pyplot as plt
from utils import *


''' Génère en utilisant la récursivité une liste de tous les chemins possibles, commençant par 0.
  Pour chaque point, on ajoute tous les chemins commençant par ce point -> appel à get_all_path sur l'ensemble des points privé du point choisi
  fp permet de fixer le premier point à 0
  Cette fonction calcule en même temps la longueur de chaque chemin
  -> renvoie la liste des couples (path_i, length_i), avec i variant de 1 à (n-1)! (car nombre de permutations sur l'ensemble [1, n-1]'''
def get_all_paths(inds, dists, points, fp = False):
  n = len(inds)
  if(n == 0): return [([], 0)]

  paths = []
  for p in [inds[0]] if fp else inds:
    n_point = list(inds)
    del n_point[n_point.index(p)]

    n_paths = get_all_paths(n_point, dists, points)
    for path in n_paths:
      if(path[0] == []): # Cas limite de la récursion
        d = dists[p, 0]
        paths.append([[p], d])
      else:
        d = dists[p, path[0][0]]
        paths.append([[p] + path[0], d + path[1]])
  return paths

''' Enfin, fonction qui sélectionne le chemin le plus court, par un simple appel à min() '''
def get_best_path(inds, dists, points):
        paths = get_all_paths(inds, dists, points, True)
        best_path = min(paths, key = lambda p: p[1]) # p[1] correction à l'élément length de (path, length)

        return best_path

points = [(-0.464, -0.48), (-0.106, -0.155), (-0.484, 0.2225), (0.12, -0.66), (0.308, 0.275), (-0.004, 0.5), (0.51, -0.2525), (-0.4, -0.1825), (0.03, -0.135)]
#points = gen_points(9)

dists = create_distance_matrix(points)

best_path = get_best_path(list(range(9)), dists, points)

display_path(best_path[0], points)
plot_points(points)

plt.title("Meilleur chemin - méthode exhaustive")

plt.show()

```