```
from math import degrees, radians, atan2, sin, cos, acos
import numpy as np

A1 = 13.5
A2 = 10.5


def analytical(x, y):
   pq2 = degrees(acos(((x**2)+(y**2)-(A1**2)-(A2**2))/(2*A1*A2)))
   pq1 = degrees(atan2(y, x)-atan2((A2*sin(radians(pq2))), (A1+(A2*cos(radians(pq2))))))

   nq2 = -pq2
   nq1 = degrees(atan2(y, x)-atan2((A2*sin(radians(nq2))), (A1+(A2*cos(radians(nq2))))))

   print('+ q2\'s q1: ', pq1)
   print('Positive q2: ', pq2)
   print('- q2\'s q1: ', nq1)
   print('Negative q2: ', nq2)
   print()


def forward(joints):
   x = (A1 * cos(radians(joints[0]))) + (A2 * cos(radians(joints[0]+joints[1])))
   y = (A1 * sin(radians(joints[0]))) + (A2 * sin(radians(joints[0]+joints[1])))
   return [x, y]


def numerical(x, y):
   target = np.array([x, y])
   joints = [90, 0]
   start_pos = np.array(forward(joints))
   error = np.linalg.norm(target - start_pos)

   while error > 0.1:
      for i in range(2):
         joints_add = joints[:]
         joints_add[i] += 0.5
         pos_add = np.array(forward(joints_add))
         err_add = np.linalg.norm(target - pos_add)

         joints_sub = joints[:]
         joints_sub[i] -= 0.5
         pos_sub = np.array(forward(joints_sub))
         err_sub = np.linalg.norm(target - pos_sub)

         if err_add >= err_sub:
            joints[i] = joints_sub[i]
            error = err_sub
```

```
            elif err_add < err_sub:
                joints[i] = joints_add[i]
                error = err_add

        print("q1 and q2: ", joints)
        print()


analytical(0, 24)
analytical(5, 10)
analytical(-5, 10)
analytical(5, -10)
analytical(-5, -10)
numerical(0, 24)
numerical(5, 10)
numerical(-5, 10)
numerical(5, -10)
numerical(-5, -10)
```