**Défi 1 :**

A = (-1.0, -2.0)

A désigne un 2-uplet(un couple)  
A[0] est l’abscisse du point A et A[1] est son ordonnée.

Complétez le programme ci-dessous en construisant la liste des ordonnées des points 'y'

x = [A[0], B[0], C[0], D[0]] # liste des abscisse des 4 points

# ‘y’, liste des ordonnées des 4 points

Construire la fonction distance qui prend en paramètres les couples de coordonnées de deux points et renvoie la distance entre ces 2 points.

from math import sqrt   
# importation de sqrt() permettant de calculer la racine carrée

def distance (E,F):

""" tuple \* tuple -> float

Préconditions: E et F sont des couples de flottants représentants  
 les coordonnées des points  
 Renvoie la distance entre E et F

"""

xe = E [0] # xe prend la valeur de l'abscisse de E

ye = E [1] # ye prend la valeur de l'ordonnée de E

# faites la même chose pour les coordonnées de F, xf et yf

xf = \_\_\_\_\_\_\_\_

yf = \_\_\_\_\_\_\_\_

Return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Quels indices nous permettent de penser que ABCD est sans doute un carré ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Défi 2 :** Jeu de fléchettes

Affichez le repère avec les 10 points de la liste liste\_points  
Note: Il faut créer les listes ‘x’ contenant les abscisses des points et la liste ‘y’ des ordonnées.

import matplotlib.pyplot as plt # importation des bibliothèques

import matplotlib.image as mpimg \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_# fonction d'affichage 'plot'

plt.axis([-50,150,-50,150]) # paramétres des dimensions des axes.

plt.grid()

cible = mpimg.imread('img/cible.png') # importation de l'image (cible)

imgplot = plt.imshow(cible) # affichage de la cible

plt.show()

Donnez le programme qui renvoie le nombre de fléchettes dans la cible.

def dans\_la\_cible(ma\_liste):

""" list -> int

ma\_liste est la liste des positions des fléchettes

Renvoie le nombre de fléchettes à une distance d <= à 50

"""

compteur = 0 # initialisation du compteur

for i in range(10): # boucle sur les 10 lancés

P = ma\_liste [i] # P récupère les coordonnées du ième point

O = (50,50) # O est l'origine (le centre de la cible)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_# si c'est vrai, on ajoute 1 au compteur

return compteur

**Défis du tireur d’élite :** Que faut-il modifier pour tirer systématiquement dans la cible ? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Comment tirer dans un cercle de centre O et rayon 30 ?