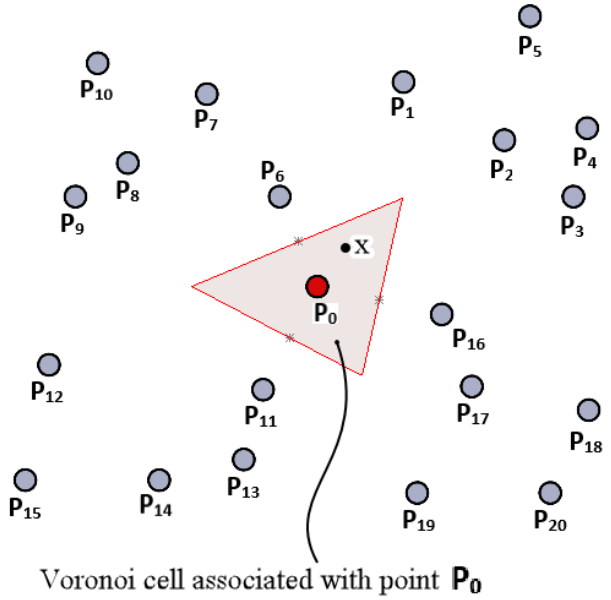
Voronoi Diagram

In mathematics, a Voronoi diagram is a partition of a plane into regions close to each of a given set of objects. It can be classified also as a tessellation. In the simplest case, these objects are just finitely many points in the plane (called seeds, sites, or generators). For each seed there is a corresponding region, called a **Voronoi cell**, consisting of all points of the plane closer to that seed than to any other. (*from Wikipedia*).

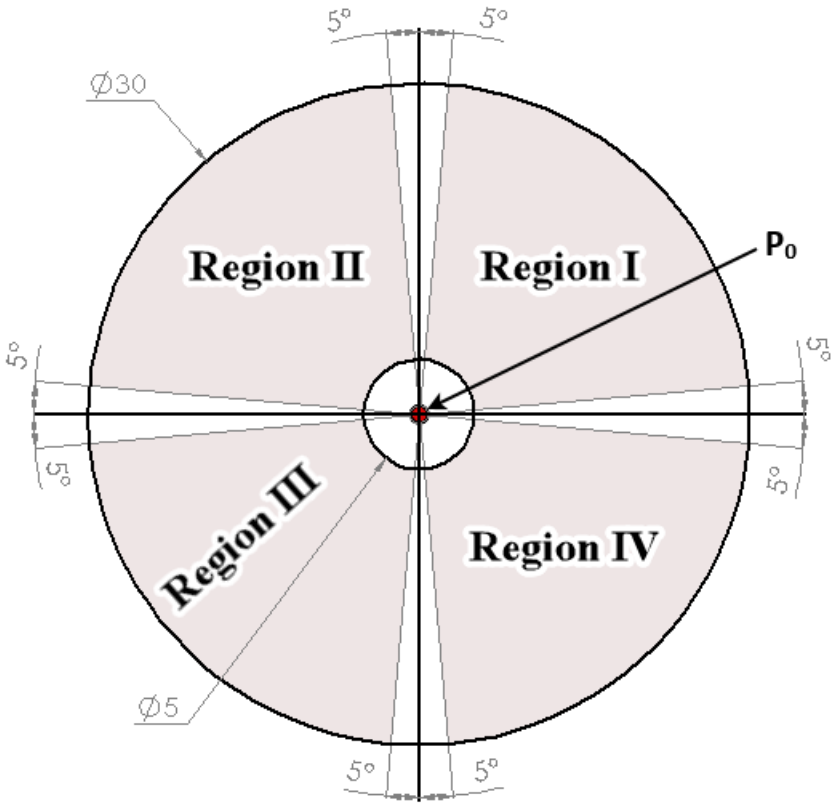
For example, in the figure below, any generic point x in the **Voronoi cell** (shaded area), which is associated with point $\mathbf{P_0}$, is closer to $\mathbf{P_0}$ than any other points $\mathbf{P_1}$-$\mathbf{P_{20}}$.



In this term project, you will find the Voronoi cell associated with a point located on the origin. You should accomplish this with the following steps:

- Generate randomly distributed points (see below).
- Determine the Voronoi cell associated with the origin (see below)
- Plot the generated points and the Voronoi cell (*mathplotlib* module)
- See the end of the document for the allowed modules, data types and functions.
- Do not use module, data type and functions which are not allowed.

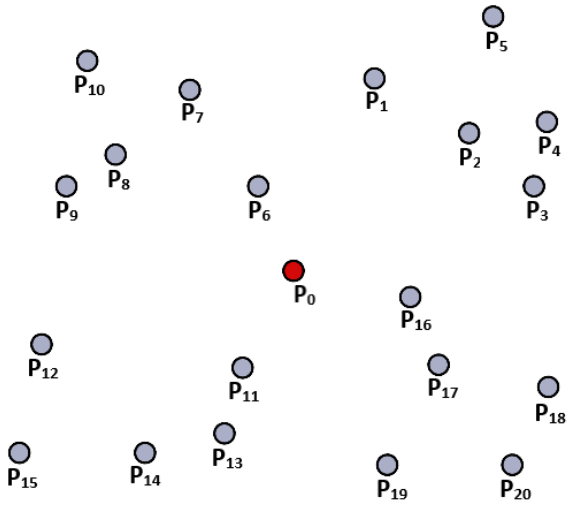
Generate randomly distributed points in plane: Divide the neighborhood of point $\mathbf{P_0}$ located at the origin into four regions as shown and generate five randomly distributed points in each region.



Hint: You can use polar coordinates to generate points. As an example, let's generate a point in region I.

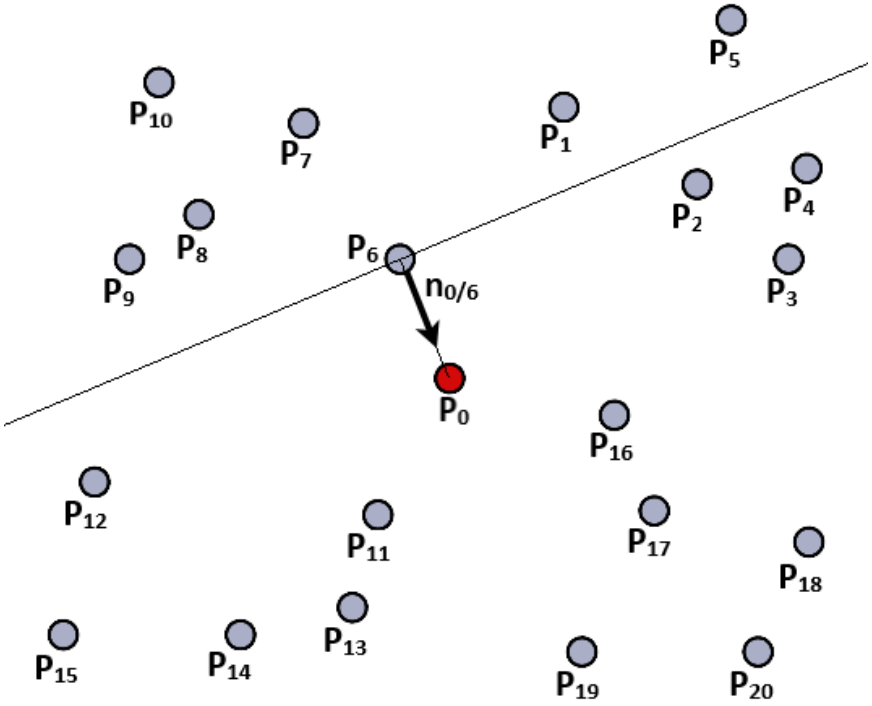
- By using random *module* in *Python*, generate a number between 2.5 and 15. This number will correspond to the radius **r**.
- By using random *module* in *Python*, generate a number between 5 and 85. This number will correspond to the angle θ.
- Find **x**- and **y**-coordinates as $\mathbf{x}=\mathbf{r}\cdot\cos\theta$ and $\mathbf{y}=\mathbf{r}\cdot\sin\theta$

Randomly generated points:

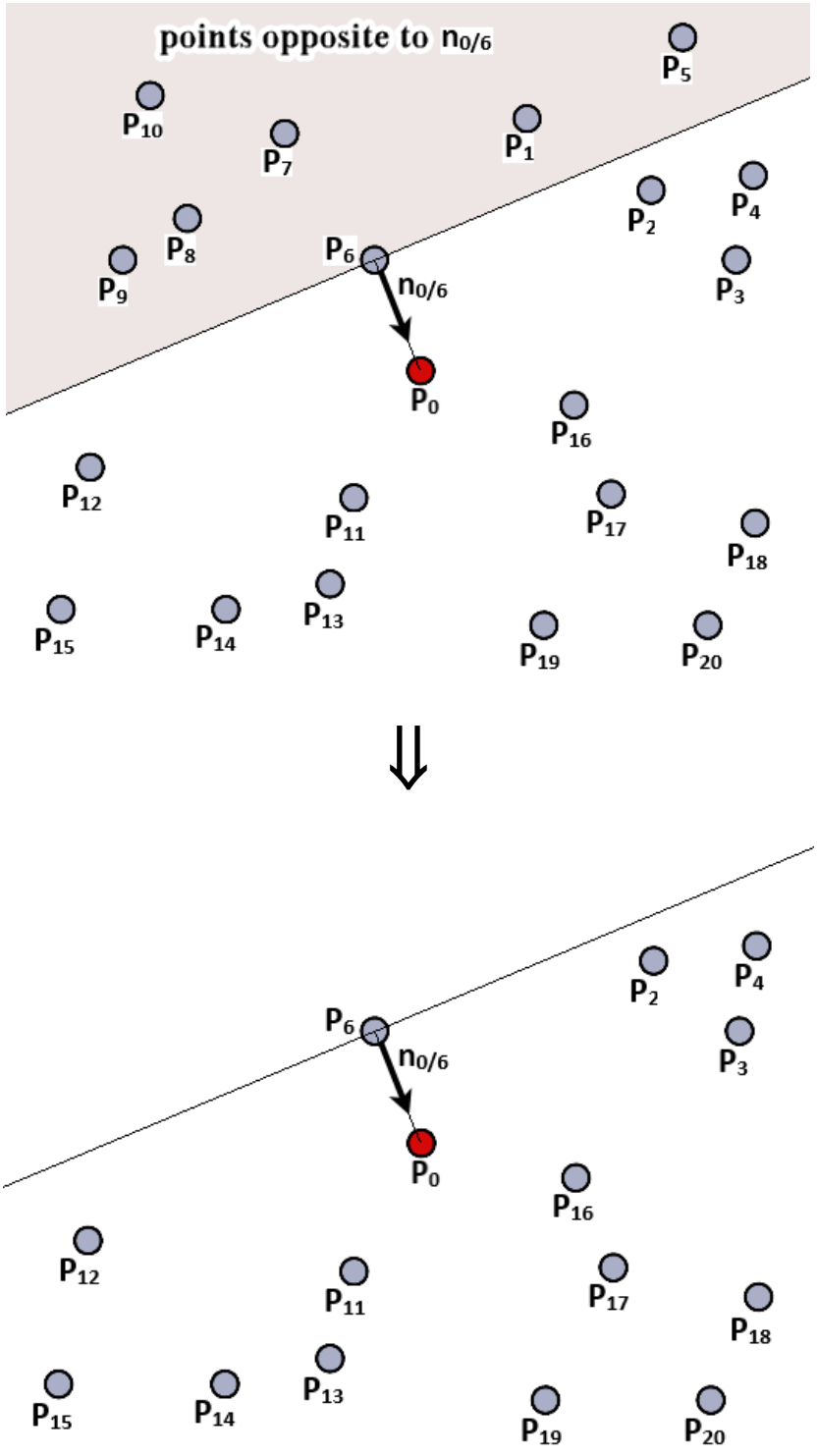


Steps to generate Voronoi cell associated with point P_0 (the origin):

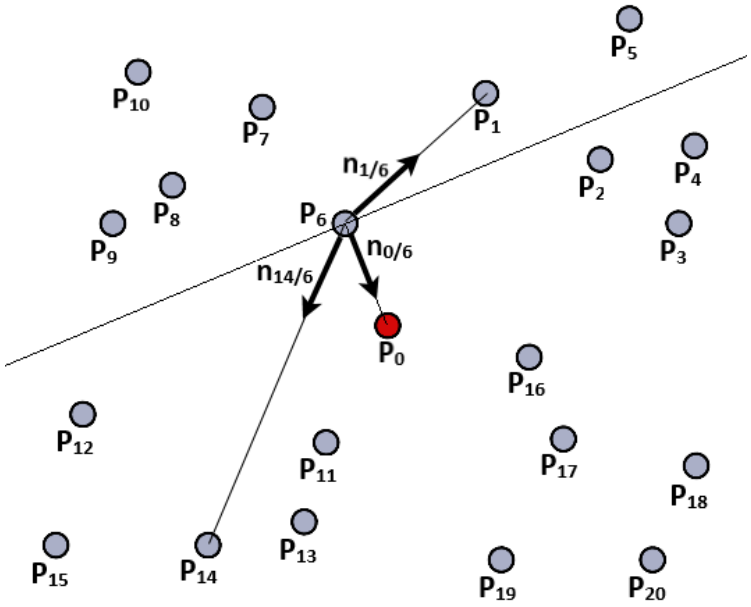
- Identify the closest point to $\mathbf{P_0}$. In our example $\mathbf{P_6}$ is the closest point.
- Draw a line from $\mathbf{P_0}$ to $\mathbf{P_6}$.
- Find the unit vector $\mathbf{n_{0/6}}$ from $\mathbf{P_6}$ to $\mathbf{P_0}$.
- Draw a line passing through point $\mathbf{P_6}$ and perpendicular to $\mathbf{n_{0/6}}$.



- Remove points facing opposite to $\mathbf{n_{0/6}}$. In our example $\mathbf{P_1}$, $\mathbf{P_5}$, $\mathbf{P_7}$, $\mathbf{P_8}$, $\mathbf{P_9}$ and $\mathbf{P_{10}}$ are removed.

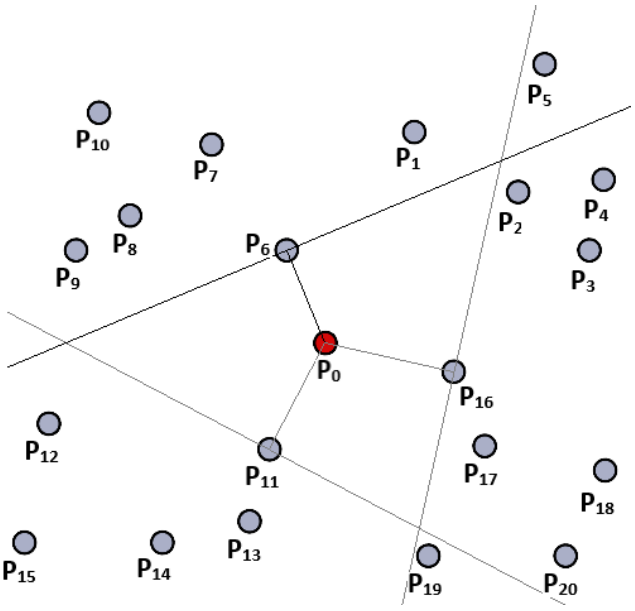


- You can use dot product to decide which points will be removed:
  - Find the unit vectors from $\mathbf{P_6}$ to points, say $\mathbf{P_1}$ and $\mathbf{P_{14}}$: $\mathbf{n_{1/6}}$ and $\mathbf{n_{14/6}}$.
  - Since the dot product between unit vectors is the cosine of the angle between vectors, if the angle is greater than 90^o, then the dot product will result in a negative number.
  - $\mathbf{n_{0/6}} \cdot \mathbf{n_{1/6}} < 0.0$ and $\mathbf{n_{0/6}} \cdot \mathbf{n_{14/6}} > 0.0$

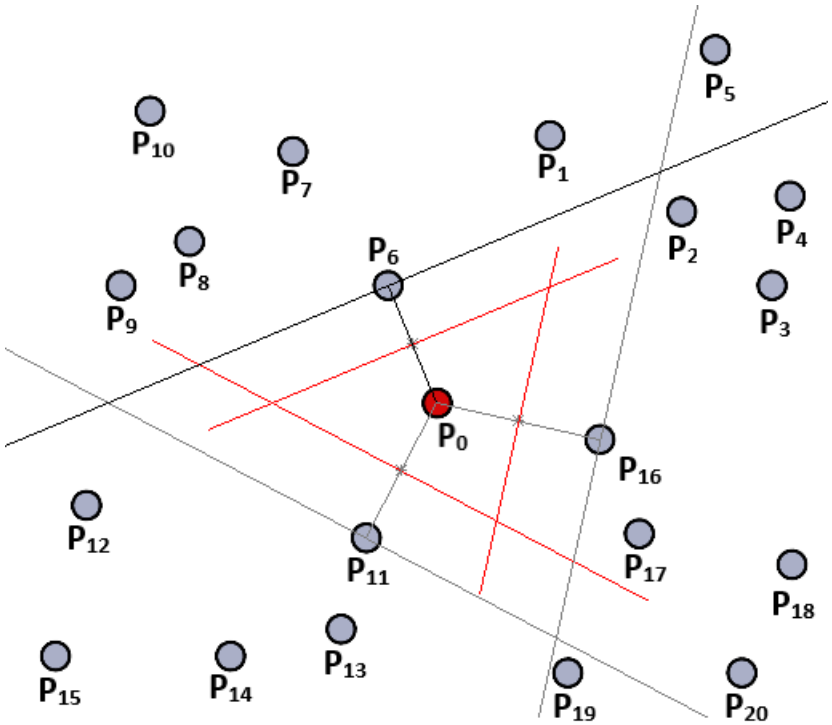


- Repeat the above steps for the remaining points, i.e.,
  - Find the closest point to $\mathbf{P_0}$.
  - Find the unit vector and draw the line.
  - Remove the points facing opposite to the unit vector.
  - Repeat these steps until no points are left to remove.

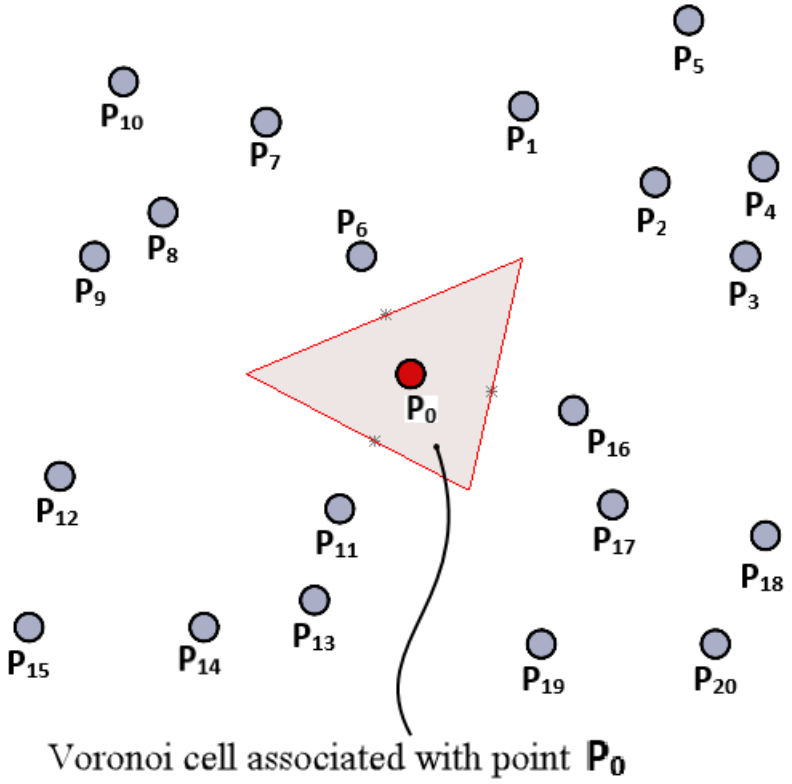
- In our example, points **P_6**, **P_{11}** and **P_{16}** are selected until no points are left to remove.



- To complete the Voronoi cell:
  - Find the midpoints of the lines connecting **P_6**, **P_{11}** and **P_{16}** to **P_0**.
  - Draw parallel lines through midpoints.

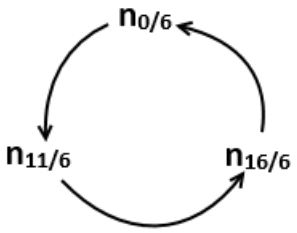


- Find the intersection points of the consecutive lines.
- Remove the extensions to determine the final Voronoi cell.



- To determine the consecutive lines, you can use the two-argument tangent function (*arctan2* function in *numpy* module).
  - Select one of the unit normal vectors and calculate the dot and cross product between the selected unit normal vector and remaining unit vectors. Always use the selected vector as the first vector in the cross product.
  - In our example, there are three-unit normal vectors: $\mathbf{n}_{0/6}$, $\mathbf{n}_{11/6}$ and $\mathbf{n}_{16/6}$.
  - Select $\mathbf{n}_{0/6}$
  - Calculate the dot and cross products:
    $\cos\theta_1 = \mathbf{n}_{0/6} \cdot \mathbf{n}_{11/6}$ and $\sin\theta_1 \mathbf{k} = \mathbf{n}_{0/6} \times \mathbf{n}_{11/6}$
    $\cos\theta_2 = \mathbf{n}_{0/6} \cdot \mathbf{n}_{16/6}$ and $\sin\theta_2 \mathbf{k} = \mathbf{n}_{0/6} \times \mathbf{n}_{16/6}$
    where $\mathbf{k}$ is the unit vector along *z*-axis
  - Calculate angles with *arctan2* function:
    $\theta_1 = \arctan2(\sin\theta_1, \cos\theta_1)$ and $\theta_2 = \arctan2(\sin\theta_2, \cos\theta_2)$

- *arctan2* function gives the angle between $-\pi$ and π. If calculation gives negative value add 2π.
- results are θ_1=129.358° and θ_2=214.446°.
- Order of the lines is $\mathbf{n_{0/6}} \rightarrow \mathbf{n_{11/6}} \rightarrow \mathbf{n_{16/6}}$.
- Cyclic order is



- Cyclic order gives the order of the intersection:
  - Calculate intersection between the lines of which unit normal vectors are $\mathbf{n_{0/6}}$ and $\mathbf{n_{11/6}}$
  - Calculate intersection between the lines of which unit normal vectors are $\mathbf{n_{11/6}}$ and $\mathbf{n_{16/6}}$
  - Calculate intersection between the lines of which unit normal vectors are $\mathbf{n_{16/6}}$ and $\mathbf{n_{0/6}}$

- **All the operations described above (except the final plot) will be performed algebraically in the program.**

Allowed modules, data type and functions.

Modules	Data Types	Functions
standard	Standard data types: integer, float, list etc.	Standard functions and standard operations
numpy	Array, matrix and related operations	Trigonometric functions, matrix inverse
random		Functions to generate random numbers
math		Trigonometric functions
mathplotlib		Plot function

Example of the final Plot (different point set is used):