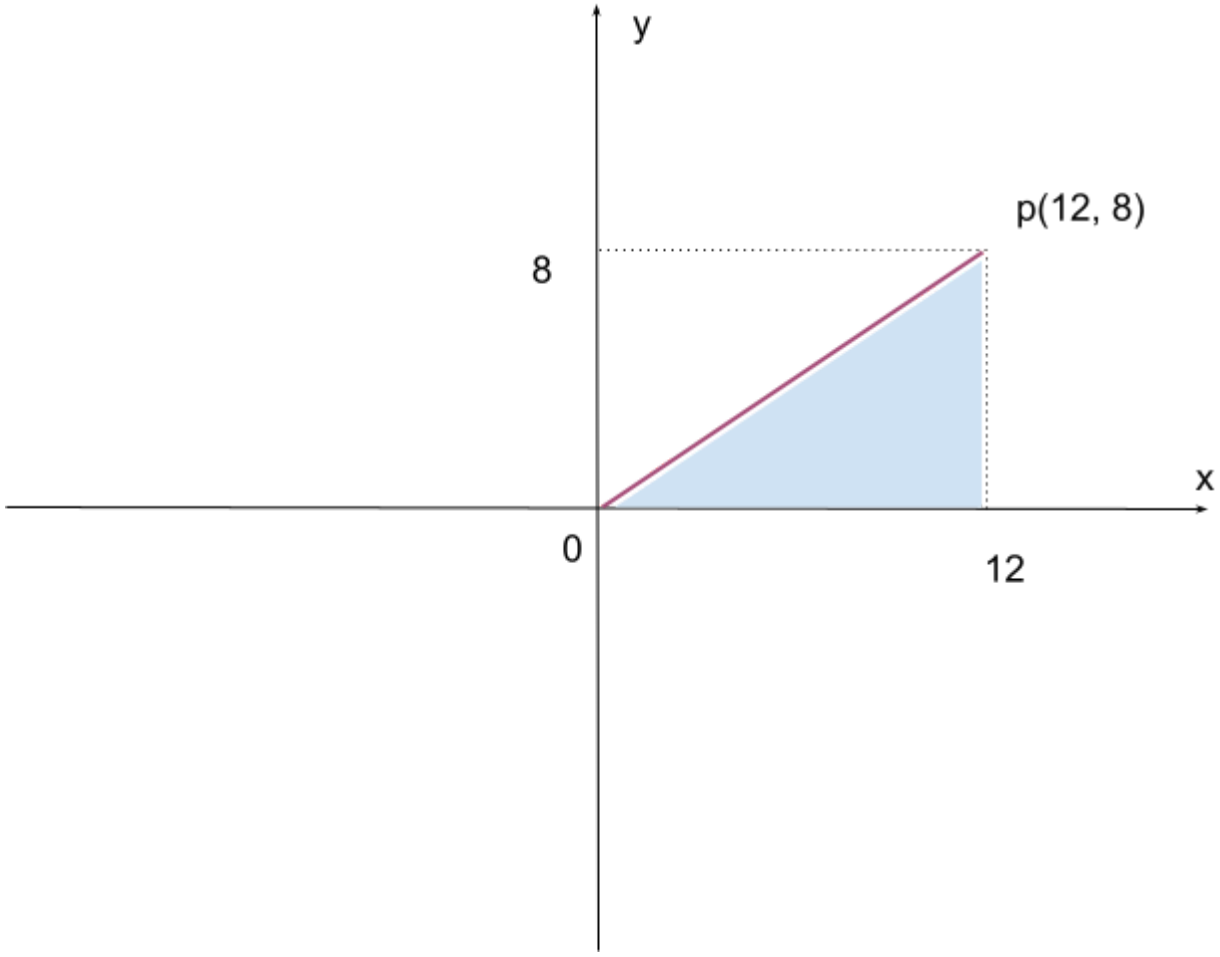
1. Haideti sa ne amintim putina geometrie..



Avem cele doua axe de coordonate x și y. Un punct p din plan are doua coordonate, cate una pe fiecare dintre axe. De exemplu punctul **p** din imagine este la coordonatele 12, 8.

Definiti o clasa **Point** a carei constructor sa aiba 2 parametri numere (intregi sau float). Acesti doi parametri reprezinta cele doua coordonate ale punctului pe cele doua axe.
Mai definiti o metoda **get_distance** în clasa nou creata, metoda care va returna distanta de la punctul nostru la originea axei de coordonate (adica la punctul de coordonate 0, 0). Ne amintim teorema lui Pitagora pentru calculul distantei. Putem sa ne imaginam triunghiul dreptunghic albastru, în care cele doua catete sunt de dimensiuni 12 respectiv 8 și trebuie sa calculam ipotenuza, care de fapt este chiar distanta de la punct la origine.
Distanta poate fi un numar float asa ca as vrea ca functia get_distance sa returneze numarul rotunjit la 2 zecimale (vedeti functia average unde am mai folosit rotunjirea)

Cititi de la tastatura doua valori pentru cele doua coordonate ale unui punct. Instantiati (creati) obiectul **Point** și afisati distanta la origine folosind metoda creata. Exemplu de output:

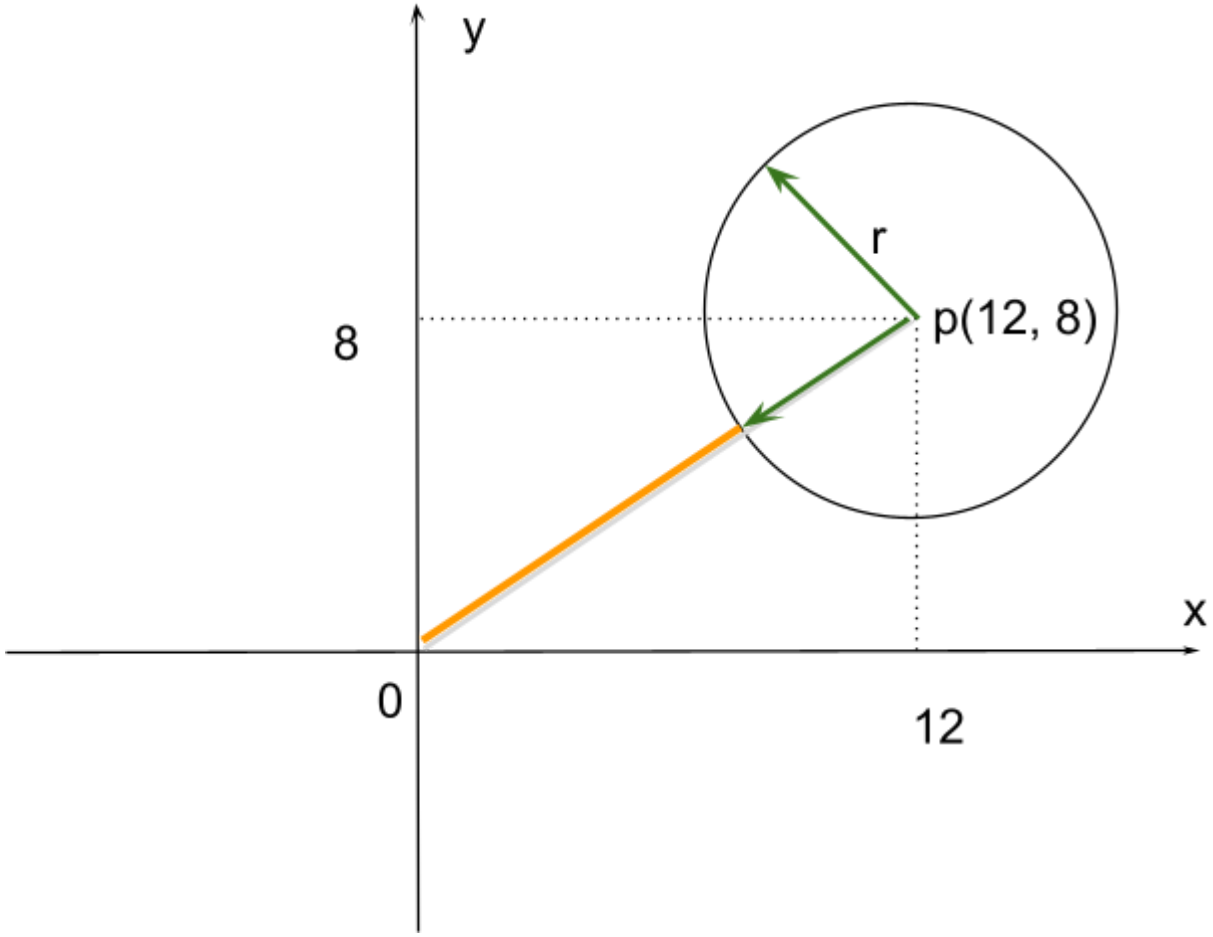
```
Introdu valoarea coordonatei x: 3
Introdu valoarea coordonatei y: 4
Distanta de la punct la origine este 5.0
```

2. Creati un modul numit **geometry.py** și mutati acolo clasa **Point** de la problema anterioara. Refaceti programul anterior importand acest modul (cu totul sau doar clasa Point) și folosind clasa din modul. la problemele urmatoare vom folosi acest modul pentru a ne defini alte clase sau a modifica pe **Point** și întotdeauna importam modulul.

3. Introduceti de la tastatura coordonatele a doua obiecte de tip **Point** și instantiati-le (**p1** si **p2**). În functie de distanta fiecaruia la originea axelor afisati una dintre variantele:

```
p1 este mai departe de centru decat p2
p2 este mai departe de centru decat p1
p1 și p2 sunt la aceeasi distanta fata de centru
```

4. Definim o clasa **Circle** (cerc) care extinde (în modulul anterior) clasa **Point**. Un cerc este definit de doua coordonate **x** și **y** precum și de o **raza**, care este tot un numar. Asadar constructorul clasei Circle va avea un parametru în plus care va fi valoarea razei. Incercati sa refolositi pe cat posibil constructorul definit în **Point** atunci cand definiti constructorul lui **Circle**. Clasa **Circle** va avea doua metode:
- **get_area** care va returna valoarea ariei cercului. Va las asa gasiti voi formula ariei (suprafetei) unui cerc în functie de raza sa
- **get_distance** care va returna distanta de la cerc la origine. Distanta de la cerc la origine o definim ca fiind cea de la circumferinta cercului la origine, nu de la centrul sau. Asadar ca sa o calculam ar trebui sa calculam distanta de la centrul cercului la origine și apoi sa scadem raza(vedeti linia portocalie din figura de mai jos). Implementati aceasta metoda și daca este posibil incercati sa refolositi **get_distance** definit în clasa de baza Point pentru calculul distantei de la cerc la origine. Cititi de la tastatura datele unui cerc, coordonate și raza și afisati aria cercului și distanta la origine. Output-ul sa fie cat mai explicit.


y
r
p(12, 8)
8
x
0
12

5. Avem o lista de puncte ca și în exemplul de mai jos:

```
p1 = Point(4, 8.5)
p2 = Point(3, 4)
p3 = Point(-5, 6)
l = [p1, p2, p3]
```

O astfel de lista poate avea oricat de multe puncte. Sortati crescator lista dupa criteriul distantei de la punct la origine. La inceput vor fi punctele aflate mai aproape de origine mergand spre capatul listei vor fi cele mai indepartate. Incercati sa refolositi cat mai mult din ce ati definit deja în clasa Point.

Ca sa verificati ca sunt sortate bine punctele, parcurgeti lista și afisati distanta pentru fiecare dintre puncte și observati daca intr-adevar sunt aranjate corespunzator.

Incercati cat mai multe variante de rezolvare:
- va creati voi algoritmul
- incercati sa folositi acel parametru key al lui sorted
- incercati sa folositi o functie lambda.

Incercati și varianta în care în lista sunt și puncte și cercuri. E vreo diferenta de implementare a codului?