EXERCICE 1

Programmer la fonction `chercheIndex`, prenant en paramètre un tableau d'entiers `tab` (type `list`) et un entier `n`, et qui renvoie un tuple de tous les indices (index) de `n` dans `tab`. Si `n` n'est pas présent dans `tab`, la fonction renvoie un tuple vide.

Exemples :

```
>>> chercheIndex([5, 3, 12, 7, 9, 6, 18],1)
()
>>> chercheIndex([2, 4, 18, 9, 11, 5, 7],2)
(0, )
>>> chercheIndex([2, 3, 5, 2, 4, 11, 19],2)
(0, 3)
```

EXERCICE 2 (4 points)

On veut écrire un module dans lequel sera implémentée une classe '`Point`'.

Une première méthode appelée 'constructeur ' permet de définir les attributs des instances de cette classe, à savoir les coordonnées `x` et `y` sous le forme de deux entiers signés.

Une deuxième méthode '`distance`' permet de calculer le distance euclidienne entre deux instances de la classe `Point`.

On rappelle que la distance entre deux points du plan de coordonnées `(x ; y)` et `(x' ; y')` est donnée par la formule : $d = \sqrt{(x-x')^2 + (y-y')^2}$

Enfin une troisième méthode '`minDist`' permet de sélectionner dans un tableau contenant plusieurs instances de la classe `Point`, celles dont la distance est la plus courte avec une instance de référence.

```
p0, p1, p2, p3 = Point(), Point(7, 9), Point(2, 3), Point(-4, 3)
p4, p5, p6, p7 = Point(-7, 0), Point(-8, 3), Point(-2, 3), Point(-1, 9)
p8, p9, p10, p11 = Point(2, -3), Point(8, -3), Point(2, 5), Point(2, 9)

tab1 = [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11]
p0.minDist(tab1) == (p2, p6, p8)
Out[3]: True

tab2 = [p0, p1, p2, p3, p4, p5, p6, p7, p8, p10, p11]
p9.minDist(tab2) == (p8, )
Out[4]: True

tab3 = [p0, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11]
p1.minDist(tab3) == (p11, )
Out[5]: True
```

Dans ce module on commence par importer la fonction racine carrée (`sqrt`) du module `math` de Python.

Compléter les lignes 25, 26, 43, 65, 66, 67, 70 et 71 de l'implémentation de la classe `Point`.