TP_pente_d_une_droite

February 3, 2021

1 TP - Pentes et fonctions affines avec Python

Objectif mathématique : Pouvoir déterminer la pente et l'expression d'une fonction affine à partir de la donnée de deux points de la droite représentant cette fonction

Objectif algorithmique : utilisation de la notion de fonction et de l'instruction conditionnelle en `Python`

1.0.1 A. Pente.

1) Rappel

Soit $A(x_A; y_A)$ et $B(x_B; y_B)$, la pente de la droite (AB) représentant une fonction affine $f(x) = mx + p$ est le nombre $m = \frac{y_B - y_A}{x_B - x_A}$ avec $x_A \neq x_B$ Écrire dans la cellule ci-dessous la pente de la droite (AB) avec $A(2; -1)$ et $B(4; 0)$

1.0.2 2) Fonction en Python

Une fonction est une suite d'instructions portant un nom, effectuant une certaine tâche, etutilisantzéro, un ou plusieurs arguments ou paramètres. Elle permet de découper le problème étudié en sous-problèmes et d'éviter ainsi la répétition d'instructions. Une fois qu'elle est définie, on peut l'utiliser autant de fois que nécessaire tout au long de l'exécution de l'algorithme.

Syntaxe en Python:

```
def nom_de_la_fonction(argument1, argument2, ...):
    instructions
    return(resultat1, resultat2, ...)
```

- le mot **`def`** est obligatoire c'est un mot du language.
- les **`:`** à la fin de la ligne de éfinition de la fonction sont obligatoires, ils marquent le début des instructions à exécuter dans la fonction.
- la **tabulation** (les espaces au début de chaque ligne de la fonction) est aussi obligatoire, elle est définie automatiquement par l'éditeur et doit être respectée.
- L'instruction **`return`** interrompt le programme dès qu'elle s'est exécutée et permet de renvoyer une ou plusieurs valeurs de tous types

a. Compléter la fonction ci-dessous pour qu'elle retourne la pente de la droite (AB) avec $A(x_A; y_A)$ et $B(x_B; y_B)$ puis cliquer sur le bouton **`Exécuter`** ou **`Run`** dans la barre des boutons

[ ]:
```
def pente(xA, yA, xB, yB):
    m= ....
    return m
```

Taper dans la cellule ci-dessous `pente(2,-1,4,0)` puis cliquer à nouveau sur le bouton **Exécuter** ou **Run** dans la barre des boutons. Comparer le résultat obtenu avec celui trouvé dans la question 1)

[ ]:
```

```

Dans la cellule ci-dessous, tester encore une fois la fonction `pente()` avec $A(2;-1)$ et $B(2;0)$

[ ]:
```

```

Que se passe-t-il ? Donner une explication.

1.0.3 3) L'instruction conditionnelle en Python

La fonction pente() définie précédemment génère une erreur lorsque les points A et B ont les mêmes abscisses.

l'instruction conditionnelle permet de choisir ce que l'on veut exécuter en fonction d'une condition. La syntaxe est :

```
if condition :
    instruction à exécuter si la condition est vraie
else:
    instruction à exécuter si la condition est fausse
```

- Les mots **`if`** et **`else`** sont des mots du langage ce qui signifie **si** et **alors**
- la tabulation a le même sens que celle présentée dans la définition d'une fonction
- **condition** est une expression de test, comme : >- `a < b` pour tester si `a` est plus que `b` >- `a <= b` pour tester si `a` est plus ou égale à `b` >- `a == b` pour tester si `a` est égale à `b` >- `a != b` pour tester si `a` est différent de `b` Dans la cellule ci-dessous, compléter la nouvelle fonction pente()

[ ]:
```
def pente(xA, yA, xB, yB):
    if ... == ... :
       return("impossible")
    else:
         m=...
    return m
```

Tester cette nouvelle version avec les points $A(2;-1)$ et $B(2;0)$ et aussi avec les points $A(1;3)$ et $B(4;-3)$

[ ]:
```

```

1.0.4 4) Pour allet plus loin

Lorsqu'on connait le nombre **m** et les coordonnées d'un point $A(x_A; y_A)$ de la droite on peut déterminer le nombre **p** appelé ordonnée à l'origine avec $p = y_A - mx_A$.

Compléter et tester la fonction **`ordonnee_origine(m,xA,xB)`** qui retourne comme son nom l'indique, la valeur du nombre p.

[ ]:
```
def ordonnee_origine(m, xA, yA):
    p = ....
    return ....
```

Compléter et tester la fonction suivante nommée **fonction_affine** prend en paramètres les coordonnées de 2 points et retourne si c'est possible les nombres **m** et **p**.

[ ]:
```
def fonction_affine(xA, yA, xB, yB):
    m = ......
    if m == .... :
        return("il n'y a pas de fonction affine avec ces deux points")
    else:
        p = ....
        return (m,p)
```