La notion de fonction, en seconde générale

A. La notion de fonction en seconde générale. (Inspiré des nombreux exercices d'algorithmiques figurant dans le manuel Pixel de chez Bordas, auxquels j'ajoute toujours une activité graphique).

Les algorithmes sont élaborés et mise en œuvre d'abord en lignes de commandes, et une fois au point les lignes de commandes sont intégrées dans une ou plusieurs fonctions. On compare les différents aspects des fonctions en programmation, avec les fonctions mathématiques. La représentation graphique confronte les élèves à la nécessité de raisonner sur les points et leurs coordonnées.

A.1. Fonctions affines (fichier professeur : FonctionAffinePosi2nde0.py)

Les coefficients sont les arguments de la fonction, qui est définie dans le corps du programme.

A.1.a. Représentation graphique du signe d'une fonction affine **f(x)** = **mx** + **p** sur **[a ; b]**. (`fonction Python SigneAffine3(…)`, lignes 48 à 67 du fichier FonctionAffinePosi2nde0.py)

L'objectif est de différencier les points d'ordonnées négatives de ceux d'ordonnées positives, soit par des couleurs, soit par des marques. Il faut d'abord créer les listes des valeurs des abscisses et des ordonnées, ce qui montre la nécessité de bien décrire le domaine de définition. On crée ensuite une liste qui contient les signes des ordonnées et enfin une liste contenant les couleurs (par exemple bleue pour les ordonnées négatives, verte pour les ordonnées positives) à appliquer au points selon le signe de leur ordonnée.

A.1.b. Représentation graphique de la position de la droite représentative de la fonction affine **f** sur **[a ; b]** par rapport à la droite d'équation **y** = **k**, réel. (`fonction Python PosikAffine4(…)`, lignes 80 à 99 du fichier FonctionAffinePosi2nde0.py)

L'objectif est de représenter différemment les points d'ordonnées inférieures à **k** de ceux d'ordonnées supérieures à **k**, soit par des couleurs, soit par des marques. C'est une simple petite modification par rapport au programme du **A.1.a.** On créera un nouveau fichier.

A.1.c. Soit **f** et **g** deux fonctions affines définies sur l'intervalle **[a ; b]**. (`fonction Python PositAffine5(...)` lignes 112 à 135 du fichier FonctionAffinePosi2nde0.py)

L'objectif est d'illustrer les positions relatives des deux droites **Df** et **Dg** représentatives des fonctions affine **f** et **g**.

Il s'agit donc de représenter différemment les points de **Dg** d'ordonnées inférieures à ceux de **Df** de même abscisse, de ceux de **Dg** d'ordonnées supérieures à ceux de **Df** de même abscisse, soit par des couleurs, soit par des marques. On peut facilement modifier le programme du **A.1.a.** pour obtenir les résultats demandés.

☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ **PROLONGEMENTS** ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺

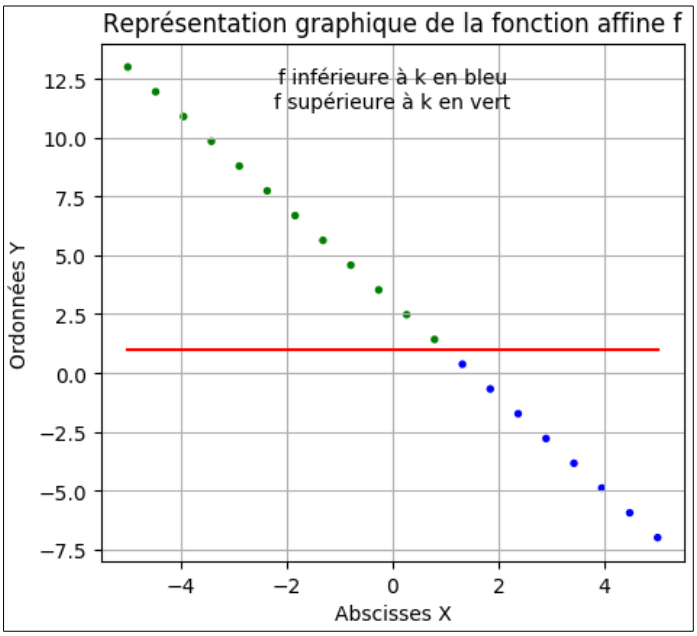
A.2. Fonctions quelconques (fichier professeur : FonctionGenePosi2nde0.py)

A.2.a. et **A.2.b.** Reprendre, respectivement, les exercices **1.b.**, **1.c.**, en utilisant les fonctions quelconques fournies dans les énoncés. (`fonctions Python PositFgene2(…)` lignes 87 à 104 et `PositFgene4(…)` lignes 161 à 180 du fichier FonctionGenePosi2nde0.py)

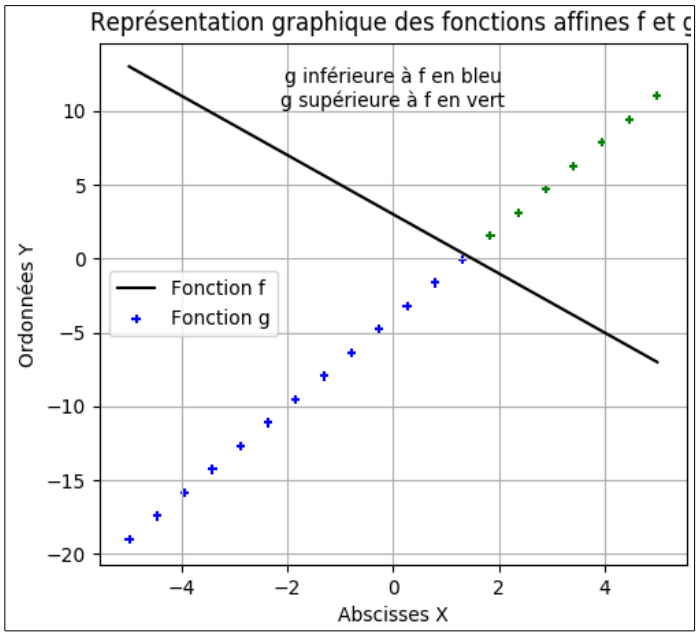
Il est plus simple, dans un premier temps, de définir les fonctions en dehors du programme principal. Les adaptations des programmes du **A.1**. sont très simples à faire.

Quelques exemples d'utilisation des fonctions Python.

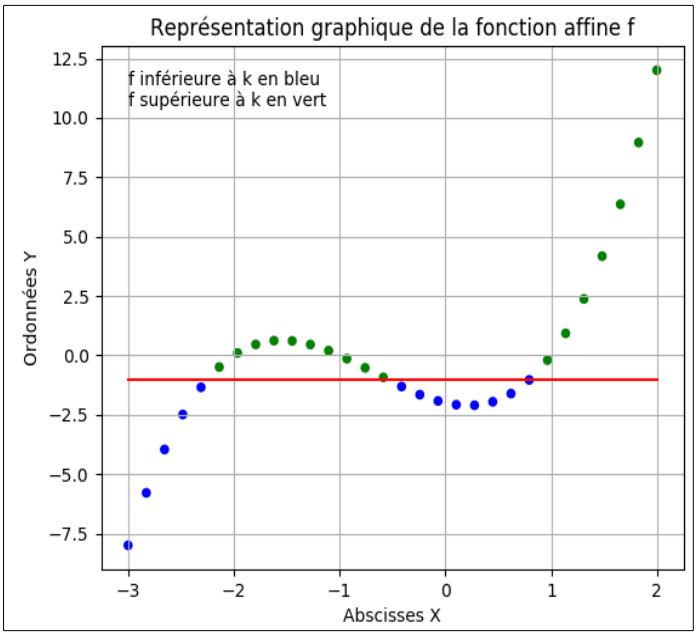
```
PosikAffine4(k = 1, m = -2, p = 3,
     a = -5, b = 5, nbpoints = 20)
```



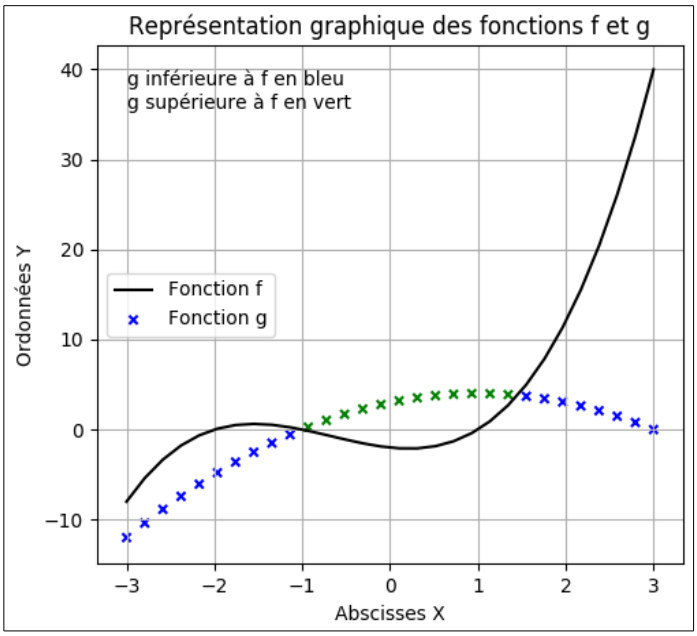

```
PositAffine5(mf = -2, pf = 3, mg = 3,
pg = -4, a = -5, b = 5, nbpoints = 20)
```




```
def f(x):
    return (x**3 + 2 * x**2 - x - 2)
a = -3 ; b = 2 ; k = -1
nbpoints = 30
PositFgene2()
```




```
def f(x):
    return (x**3 + 2 * x**2 - x - 2)
def g(x):
    return (- x**2 + 2 * x + 3)
a = -3 ; b = 3 ; nbpoints = 30
PositFgene4()
```



Nom Prénom :

TP_ALGO_2_1 : (A.1.a.) Signes d'une fonction affine – Position des points de la droite par rapport à l'axe des abscisses. (Fiche élève 1/3)

1° Exécuter les lignes de commandes ci-dessous à la main et noter les résultats au crayon papier.

2° Dans Spyder ouvrir le fichier script Python **FonctionAffinePosi2nde0_Elev.py** et exécuter chacune des lignes de commande figurant dans le tableau ci-dessous (entre les **lignes 13 et 32**). Comparer les résultats affichés avec les résultats obtenus en faisant les calculs à la main. Corriger si nécessaire.

```
"Définition de la fonction f : f(x) = mx + p ;"
"du domaine de définition [a ; b] et"
"    du nombre de points nbpoints servant à tracer la représentation graphique"
import numpy as np
m = - 2 ; p = 3 ; a = - 4 ; b = 4 ; nbpoints = 9

def f(x):
    return m * x + p                    #     f(x) = ...
```

```
"Calcul des abscisses et des ordonnées des nbpoints points"

x = np.linspace(a, b, nbpoints)

Résultats :                         [-4       -3       ...       ...       ...       ...       ...       ...       4]

y = f(x)

Résultats :                         [11        9       ...       ...       ...       ...       ...       ...      -5]
```

```
y < 0

Résultats :                        [False    False     ...       ...       ...       ...       ...       ...     True]

signy = (y < 0) * 1

Résultats :                          [0        0       ...       ...       ...       ...       ...       ...       1]
              0        1
couleur = ['green', 'blue']         #    couleur[0] = 'green' ; couleur[1] = 'blue'

scy = [couleur[i] for i in signy]

Résultats :                       ['green'  'green'    ...       ...       ...       ...       ...       ...    'blue']
```

Nom Prénom :

TP_ALGO_2_1 : (A.1.a.) Signes d'une fonction affine – Position de points de la droite par rapport à l'axe des abscisses. (Fiche élève 2/3)

1° Décrire par une phrase ce que fait chacune des commandes du tableau suivant. Noter les résultats au crayon papier.

2° Sélectionner toutes les lignes des commandes graphiques (**lignes 34 à 43** du fichier élève) et lancer leur exécution. Comparer avec votre graphique.

```
from matplotlib import pyplot as plt
plt.figure()
plt.scatter(x, y, s = 50, marker = 'o', color = scy,
            edgecolors = 'none')
Description :



plt.plot(x, [0] * nbpoints, 'r-')
Description :



plt.title('Représentation graphique de la fonction affine f')
Description :


plt.xlabel('Abscisses X')
Description :


plt.ylabel('Ordonnées Y')
Description :


plt.text(min(x), max(y), 'f négative en bleu\nf positive en vert',
         verticalalignment = 'top', horizontalalignment = 'left')
Description :



plt.grid()
Description :

plt.show()
Description :
```

Nuage de points de f : f(x) = –2x + 3 ; Df = [–4 ; 4]
Bleus pour y < 0, vert pour y > 0.

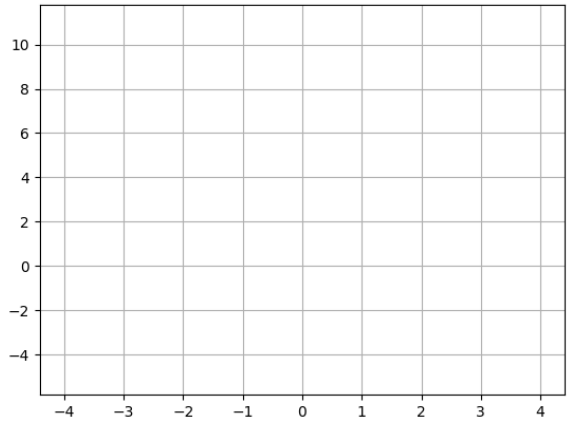


Tableau des signes de f :

x	–4		4
f(x) = –2x + 3			

Comment faire pour être plus précis au niveau du changement de signe ? Compléter puis utiliser la fonction Python SigneAffine3(…) de la page suivante pour compléter le tableau

x	–4		4
f(x) = –2x + 3			

Nom Prénom :

TP_ALGO_2_1 : (A.1.a.) Signes d'une fonction affine – Position de points de la droite par rapport à l'axe des abscisses. (Fiche élève 3/3)

1° Recopier toutes les lignes de commandes vues dans les deux tableaux précédents pour compléter la fonction `SigneAffine3(...)` ci-dessous.

2° Dans le fichier **FonctionAffinePosi2nde0_Elev.py**, copier-coller les lignes de commandes aux **lignes 53 à 66** de la fonction `SigneAffine3(...)`

3° Utiliser la fonction Python `SigneAffine3(...)` pour étudier le signe de quelques fonctions affines en complétant les tableaux de la colonne de droite.

```
Réunir toutes les lignes de commande dans une seule fonction
   sans oublier l'indentation

def SigneAffine3(m = – 2, p = 3, a = – 4, b = 4,
                 nbpoints = 200):
    import numpy as np
    import matplotlib.pyplot as plt
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
```

En utilisant la fonction Python `SigneAffine3(...)` compléter le tableau des signes des fonctions suivantes :

g(x) = 3x + 2 ; Dg = [–3 ; 2]

x	–3		2
g(x) = 3x + 2			

---

h(x) = 4x – 3 ; Dh = [–4 ; 0]

x	–4	0
h(x) = 4x – 3		

---

k(x) = –2x – 4 ; Dk = [–5 ; –2]

x	–5	–2
k(x) = –2x – 4		

---

p(x) = – x / 2 ; Dp = [–2 ; 3]

x	–2		3
p(x) = – x / 2			

Nom Prénom :

TP_ALGO_2_2 : (A.1.b.) Position de points de la droite Df représentative de la fonction f, par rapport à la droite d'équation y = k.
Résolution graphique des inéquation f(x) < k et f(x) > k. (Fiche élève 1/1)

1° Quelles modifications doit-on apporter à la fonction Python `SigneAffine3(...)` pour que les points de la droite **Df**, situés au dessus de la droite d'équation **y** = **k**, soient de couleur différents de ceux situés au dessous de la droite d'équation **y** = **k** ?

2° Compléter les **lignes 83 à 89** du code de cette nouvelle fonction qui sera nommée `PosikAffine4(...)`. L'utiliser pour compléter les tableaux ci-dessous.

```
"Points de f inférieure à k en bleu,"
"    points de f supérieure à k en vert"
def PosikAffine4(k = 1, m = – 2, p = 3, a = – 4, b = 4,
                 nbpoints = 200):
    import numpy as np
    import matplotlib.pyplot as plt
....
....
....
....
....
....
....
    plt.figure()
    plt.scatter(x, y, s = 15, marker = 'o', color = scy,
                edgecolors = 'none')
    plt.plot(x, [k] * nbpoints, 'r-')
    plt.title('Représentation graphique de la fonction affine f')
    plt.xlabel('Abscisses X')
    plt.ylabel('Ordonnées Y')
    plt.text((min(x) + max(x)) / 2, max(y),
        'f inférieure à k en bleu\nf supérieure à k en vert',
        verticalalignment = 'top', horizontalalignment = 'center')
    plt.grid()
    plt.show()
```

Utiliser la fonction Python `PosikAffine4(...)` pour placer f < k et f > k dans les tableaux suivants :

g(x) = 3x + 2 ; Dg = [–3 ; 2]
g < 4 ; g > 4

x	–3	2
g(x) = 3x + 2		

---

h(x) = 4x – 3 ; Dh = [–4 ; 0]
h < – 6 ; h > – 6

x	–4	0
h(x) = 4x – 3		

---

p(x) = –2x – 4 ; Dp = [–2 ; 3]
p < – 1 ; p > – 1

x	– 2	3
p(x) = – x / 2		

Nom Prénom :

TP_ALGO_2_3 : (A.1.c.) Position de points de la droite Dg représentative de la fonction g, par rapport à la droite Df représentative de la fonction f
Résolution graphique des inéquations f(x) < g(x) et f(x) > g(x) (Fiche élève 1/1)

1° La fonction **f** est représentée par une droite noire **Df**. Quelles modifications doit-on apporter à la fonction Python `SigneAffine4(...)` pour que des points de **Dg** situés au dessous de **Df** soient de couleur bleue et que des points de **Dg** situés au dessus de **Df** soient de couleur verte ?

2° Compléter les **lignes 115 à 124** du code de cette nouvelle fonction qui sera nommée `PositAffine5(...)`. L'utiliser et compléter les tableaux ci-dessous.

```
"Points de g inférieure à f en bleu,"
"    points de g supérieure à f en vert"
def PositAffine5(mf = - 2, pf = 3, mg = 3, pg = - 4, a = - 5, b = 5,
                 nbpoints = 200):
    import numpy as np
    import matplotlib.pyplot as plt
....
....
....
....
....
....
....
....
....
....
    plt.figure()
    plt.plot(x, y, ls = '-', color = 'black', label = 'Fonction f')
    plt.scatter(x, z, s = 20, marker = '+', color = scy,
                edgecolors = 'none', label = 'Fonction g')
    plt.title('Représentation graphique des fonctions affines f et g')
    plt.xlabel('Abscisses X')
    plt.ylabel('Ordonnées Y')
    plt.text((min(x) + max(x)) / 2, max(max(y), max(z)),
         'g inférieure à f en bleu\ng supérieure à f en vert',
         verticalalignment = 'top', horizontalalignment = 'center')
    plt.grid()
    plt.legend(loc = 'center left')
    plt.show()
```

Utiliser la fonction Python `PositAffine5(...)` pour placer g < f et g > f dans les tableaux suivants :

f(x) = −2x + 3 ; Df = [−5 ; 5]
g(x) = 3x − 4 ; Dg = [−5 ; 5]
g < f ; g > f

x	− 5	5

---

f(x) = 3x + 2 ; Df = [−3 ; 2]
g(x) = x − 1 ; Dg = [−3 ; 2]
g < f ; g > f

x	− 3	2

NOM Prénom : **PROLONGEMENTS**

TP_ALGO_2_4 : (A.2.a. et b.) Position de points de la courbe Cg représentative de g, par rapport à la courbe Cf représentative de la fonction f
Résolution graphique des inéquations f (x) < g (x) et f(x) > g(x) (Fiche élève 1/1, fichier **FonctionGenePosi2nde0_Elev.py**)

1° La fonction **f** est représentée par une courbe noire **Cf**. Quelles modifications doit-on apporter à la fonction Python `PositAffine5(...)` pour que des points de **Cg** situés au dessous de **Cf** soient de couleur bleue et que des points de **Cg** situés au dessus de **Cf** soient de couleur verte ? Les codes des fonctions **f** et **g** seront écrits en dehors de la fonction principale qui sera nommée `PositFgene4()`.

2° Compléter les **lignes 164 à 169** du code de cette nouvelle fonction `PositFgene4()`. L'exécuter et l'utiliser pour compléter les tableaux ci-dessous.

```
def f(x):
  return (x**2 — 2 * x — 5)  # sur l'intervalle [- 2,5 ; 3,5]
def g(x):
  return (— x**2 + 2 * x + 3)  # sur l'intervalle [- 2,5 ; 3,5]
a = -2,5 ; b = 3,5 ; nbpoints = 100

def PositFgene4():
    import numpy as np
    import matplotlib.pyplot as plt
....
....
....
....
....
....
    plt.figure()
    plt.plot(x, y, 'k-', label = 'Fonction f')
    plt.scatter(x, z, s = 20, marker = 'x', color = scz,
                edgecolors = 'none', label  = 'Fonction g')
    plt.title('Représentation graphique des fonctions f et g')
    plt.xlabel('Abscisses X')
    plt.ylabel('Ordonnées Y')
    plt.text(min(x), max(max(y), max(z)),
      'g inférieure à f en bleu\ng supérieure à f en vert',
      verticalalignment = 'top', horizontalalignment = 'left')
    plt.legend(loc = 'center left')
    plt.grid() ; plt.show()
```

Utiliser la fonction Python PositFgene4() pour placer g < f et g > f dans les tableaux suivants :

f(x) = x² — 2x — 5 ; Df = [— 2,5 ; 3,5]
g(x) = — x² + 2x + 3 ; Dg = [— 2,5 ; 3,5]
g < f ; g > f

x	– 5	5

---

f(x) = 3x + 2 ; Df = [—3 ; 2]
g(x) = x — 1 ; Dg = [—3 ; 2]
g < f ; g > f

x	– 3	2