Fiche n°3 : Matplotlib/numpy et fonctions

On peut bien sûr poser des exercices où l'on demande de tracer la représentation graphique d'une fonction de référence en procédant comme décrit dans la fiche principale.

Proposition d'exercice 1 : Autour des fonctions définies par morceaux

On veut tracer la représentation graphique de la fonction f définie par :

$$f(x) = \begin{cases} x^2 + 1 \text{ si } x < 0 \\ 3x + 1 \text{ si } x \geq 0 \end{cases}$$

1. Compléter la fonction `f` suivante permettant de représenter f en Python.

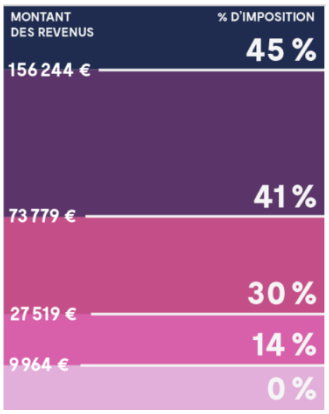
```
def f(x):
    if ..... :
        return .....
    else:
        return .....
```

2. Ecrire un script Python permettant d'afficher cette fonction.

Proposition d'exercice 2 : Calcul de l'impôt brut

Thèmes concernés : pourcentages, fonctions, ...

Un exemple concret, assez intéressant où il est question d'une fonction définie par morceaux est l'exemple des impôts.



La figure ci-contre donne le barème permettant de calculer l'impôt brut sur le revenu en 2019 (en 2020 ça changera je crois).
Le montant de l'impôt brut se calcul ainsi.

Etape 1 : On calcule d'abord l'ensemble des revenus de tout le foyer fiscal que l'on divise par le nombre de parts (1 adulte = 1 part, 1 enfant = ½ part). On note x cette valeur.
Etape 2 : Les 9964 premiers euros ne sont pas imposés. Les 17555€ suivants (27519 – 9964) sont imposés à 14%. Les 46260€ suivants (73779 – 27519) sont imposés à 30% etc..

Etape 3 : Le montant obtenu est le montant de l'impôt par part. On multiplie donc par le nombre de parts.

Exemple : un célibataire a des revenus s'élevant à 35 000€ à l'année. Dans ces 35000€, on ne lui prend rien sur les 9964 premiers euros puis sur les 17555 suivants, on lui prend 14%, soit $17555 \times 0.14 = 2457.7$.
Ensuite, pour aller à 35000, il reste 7481€, on lui prend 30% dessus : $7481 \times 0.30 = 2244.3$.
Bilan : au total, il devra payer $2457.7 + 2244.3 = 4702$ euros d'impôt.

Remarque : le montant obtenu est l'impôt brut. Il y a ensuite des aides pour les faibles revenus permettant de diminuer l'impôt réellement payé (impôt net) : décote, exonération etc…

1. Ecrire une fonction Python `impot(revenu)` qui renvoie l'impôt brut payé en fonction des revenus pour un célibataire (*ça fait travailler les* `if, elif` *etc !!).*
2. Ecrire une fonction permettant d'afficher graphiquement la fonction `impot` dans un repère.

Proposition d'exercice 3 : Tracer la dérivée d'une fonction (sans connaître son expression)

Soit f une fonction définie sur un intervalle $[a,b]$ et dérivable sur cet intervalle.
Notons $x = (x_i)_{1\leq i\leq N}$ une subdivision de l'intervalle $[a;b]$ de pas δ (petit): $x_0 = a, \dots, x_N = b$ et pour tout $i \in [\![0, N-1]\!]$, $x_{i+1} - x_i = \delta$.
Avec Python, il suffit de la créer avec la commande :

```
x = arange(a,b,δ)
```

Pour chaque $i = 0,1,\dots,N-1$, $f'(x_i)$ sera approximativement égal au coefficient directeur de la droite reliant les points de C_f d'abscisses x_i et x_{i+1} (si le pas est assez petit, l'approximation sera bonne puisque x_i et x_{i+1} seront très proches).
On aura donc :

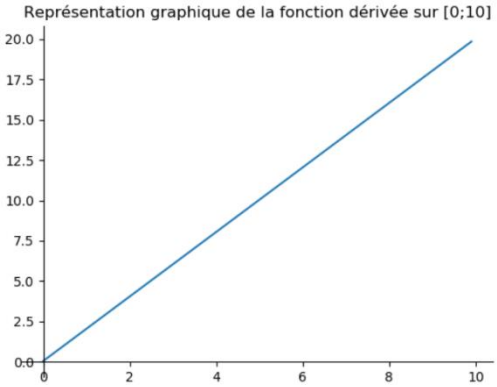
$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Il suffit donc de stocker dans une première liste (notée par exemple `x`) tous les x_i et dans une seconde (`ordonnees`) tous les $\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}$ et on plot tout ça.
Attention, la liste des $\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}$ a un point de moins par rapport à la subdivision x du départ. Donc quand on plot tout ça, il faut s'arranger pour enlever le dernier élément de x et pour cela l'instruction `x[:-1]` s'y prête bien (étant donnée une variable `liste`, l'instruction `liste[-1]` donne le dernier élément tandis que `liste[:-1]` donne tous les éléments à partir du dernier en ne l'incluant pas).

Le but de l'exo (qu'il faudra guider ?) est de faire un fonction `derivee(f,a,b)` qui, étant donnée une fonction `f` passée en paramètre et deux réels `a,b`, trace la fonction dérivée f' sur l'intervalle $[a,b]$.

Exemple de ce que donne :

```
>>> def f(x):
...     return x**2
...
>>> derivee(f,0,10)
```



Voici un code possible (à retrouver dans le fichier Python joint).

```
def derivee(f,a,b):

    x = arange(a,b,0.05) # prendre un pas petit
    ordonnees = [(f(x[i+1]) - f(x[i]))/(x[i+1]-x[i]) for i in range(len(x)-1)]
    plot(x[:-1],ordonnees)
    axesnormaux()
    show()
    title("Représentation graphique de la fonction dérivée sur [{};{}]".format(a,b))
```