TD 2 : Fonctions, bibliothèque standard

Exercice 2.1 : Fonctions

(a) Réaliser une fonction `envers` à un argument supposé d'être du type `str`. Elle retournera son argument à l'envers : `envers("Monty Python")` produira `"nohtyP ytnoM"`. (Se servir de la fonction `len` qui retourne la longueur d'une chaîne de caractères.)

(b) Réaliser une fonction `Gamma` à deux arguments `z` et `N`, où `N` est un nombre naturel (avec une valeur par défaut de 100) et `z` est un nombre complexe. Elle retournera une approximation à la fonction Gamma $\Gamma(z)$ calculée par la formule d'Euler :

$$\Gamma(z) = \frac{1}{z} \prod_{k=1}^{\infty} \frac{(1+1/k)^z}{1+z/k}$$

avec les N premiers termes du produit.

(c) Réaliser un programme qui lit un nombre romain du clavier et affiche sa valeur. Il incluera une fonction `valeur_numerique` qui convertit un chiffre romain en nombre. *Rappel :* un nombre romain se compose des chiffres M (1000), D (500), C (100), L (50), X (10), V (5) et I (1). Sa valeur est la somme des valeurs des chiffres, lu de gauche à droite, sauf si un chiffre de valeur inférieure précède un chiffre de valeur supérieure ; dans ce cas, ce premier est compté avec un signe négatif. Exemples : LXIV $= 50 + 10 + (-1) + 5 = 64$, MMXXIV $= 1000 + 1000 + 10 + 10 + (-1) + 5 = 2024$, MCMXLIX $= (1000 + (-100) + 1000 + (-10) + 50 + (-1) + 10 = 1949$.

Exercice 2.2 : Récursivité

Réaliser une fonction `pgcd` à deux arguments, supposés d'être des nombres naturels `x` et `y`, qui calcule leur PGCD (plus grand commun diviseur) récursivement par l'algorithme d'Euclide. Tester cette fonction dans un programme.

Algorithme d'Euclide : Soient x et y des entiers avec $x \geq y \geq 0$. Si r est le reste de la division entière de x par y, alors le PGCD de x et y est égal au PGCD de y et r. On répète avec $x \leftarrow y$ et $y \leftarrow r$ jusqu'à ce qu'on tombe sur 0. (Par définition, $\mathrm{PGCD}(x, 0) = x$.)

Exercice 2.3 : Fonctions d'ordre supérieur ($\star$)

(a) Réaliser une fonction `deriv` à trois arguments `f`, `x`, `epsilon`, où `x` et `epsilon` sont des float et `f` représente une application dérivable $f : \mathbb{R} \to \mathbb{R}$. Les valeurs par défaut de `x` et `epsilon` seront 0 et 10^{-5}. La valeur de retour sera la dérivée de f au point `x` avec une précision numérique d'au moins $\mathcal{O}($`epsilon`$)$.

(b) Réaliser une fonction `d` à deux arguments `f` et `epsilon` comme ci-dessus. Elle renverra une fonction réelle qui est la dérivée numérique f' de f avec une précision numérique d'au moins $\mathcal{O}($`epsilon`$)$.

(c) Tester ces fonctions dans un programme. En particulier, il faut que `d(f)(x)` = `deriv(f, x)` pour toute fonction `f` et tout `x`. Vous pouvez vous servir des fonctions définies dans la bibliothèque `math` (par exemple, vérifiez que $\exp' = \exp$ et que $\sin' = \cos$ pour plusieurs valeurs des arguments).

Rappel : La dérivée d'une fonction dérivable $f : \mathbb{R} \to \mathbb{R}$ au point x est la limite

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \qquad \text{ou bien} \qquad f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}\,.$$

La deuxième expression est à préférer car elle donne une meilleure approximation numérique pour un h donné.

Exercice 2.4 : Le module `math`

Réaliser une fonction `gauss` à trois arguments `x`, `sigma`, `mu` supposés d'être du type float. Elle retournera la valeur de la fonction de Gauss au point x,

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} .$$

Exercice 2.5 : Le module `statistics` ($\star$)

On donne une liste de nombres `L = [12.5, 2, 7.5, 8, 13.5, 8, 10, 6.5, 6, 16]`. Au moyen de la documentation en ligne (voir `https://docs.python.org`) se renseigner sur les fonctionnalités du module `statistics` de la bibliothèque standard. Trouver des fonctions appropriées de ce module pour calculer la moyenne, la médiane et l'écart-type de `L` (en anglais : "mean", "median", "standard deviation").

Exercice 2.6 : Blackjack (devoir à rendre)

Réaliser un programme qui permet à l'utilisateur de jouer le jeu aux cartes de blackjack. Utilisez des listes, des structures conditionnelles, des boucles et des fonctions.

Règles (légèrement simplifiées) :

- Le jeu oppose le joueur (l'utilisateur) contre la banque (l'ordinateur).
- Le joueur gagne le jeu si la valeur de ses cartes dépasse celle de la banque sans dépasser 21 points. Il perd si ses cartes valent soit ≥ 22 points soit moins que celles de la banque.
- Il y a 13 types de cartes, caracterisés soit par un nombre entier entre 2 et 10, soit par une des lettres V, D, R ou A (valet, dame, roi ou as). Les valeurs des cartes sont les valeurs numériques pour les cartes numérotés et 10 pour les V, D et R. Les cartes A valent soit 1 soit 11.
- Toute partie commence avec deux cartes visibles données au joueur par hasard. Puis, deux cartes sont données à la banque dont une est visible au joueur et l'autre est masquée. Il est possible de recevoir plusieurs cartes du même type. La probabilité de recevoir une carte d'un certain type est toujours 1/13.
- Ensuite, s'il le souhaite, le joueur peut demander d'avoir une troisième, puis une quatrième et une ciquième carte etc. Dès que la valeur de sa main dépasse 21 points, il perd le jeu immédiatement.
- Quand le joueur est satisfait, la banque peut recevoir des cartes supplémentaires :
  - Tant que ses cartes valent ≤ 16 points, elle demandera des cartes en plus.
  - Lorsque sa main vaut ≥ 17 points, elle s'arrête.
- Si enfin les cartes de la banque valent plus que 21 points, le joueur gagne le jeu. Sinon, et si les cartes du joueur valent plus que celles de la banque, il le gagne également. Si les cartes de la banque valent plus que celles du joueur sans dépasser 21 points, le joueur perd le jeu. Si les deux mains ont la même valeur, la partie est nulle.
- La valeur des A (1 ou 11 points) est toujours en faveur du détenteur : si avec un A de valeur 11 une donne vaudrait au maximum 21 points, sa valeur est 11, sinon 1.

Indication : Servez-vous de la fonction `randrange` du module `random`. La commande `random.randrange(r)` (avec `r` un entier) retourne un entier (quasi-)aléatoire entre 0 et `r`-1.

Remarque : Ce devoir n'est pas noté. Son seul intérêt est de vous donner l'opportunité de pratiquer Python et d'avoir un retour sur votre travail ; il y a donc zéro intérêt de rendre une solution préfabriquée trouvée sur internet ou de se faire aider par une IA.