Séance 3: Chaînes, booléens et boucles

Université Paris-Diderot

Exercice 1 (Cadre, ⋆)
Écrire une procédure `frame` *qui prend en paramètre une chaîne de caractères, et affiche cette chaîne de caractères entourée d'un cadre de taille adaptée.*

> ***Contrat:***
> *Par exemple,* `frame("Hello World!")` *doit afficher :*
> ```
> +--------------+
> | Hello World! |
> +--------------+
> ```

On rappelle qu'on peut obtenir la taille (le nombre de caractères) d'une chaine de caractères `s` *grâce à l'expression* `s.length()`. □

Exercice 2 (Palindromes, ⋆)
Un palindrome est une chaîne de caractères qui est identique dans les deux sens. Par exemple, `"rotor"` *ou* `"ressasser"`.

1. *Écrire une fonction* `reverse` *qui inverse le sens d'une chaîne de caractères.*

   > ***Contrat:***
   > *Par exemple,* `reverse("hello")` *doit renvoyer* `"olleh"`.

2. *Utiliser la fonction* `reverse` *pour écrire une fonction* `palindrome` *qui renvoie* `true` *si son argument est un palindrome,* `false` *sinon. On pourra utiliser la fonction* `reverse` *définie juste avant. On rappelle que l'on peut utiliser* `s1.equals(s2)` *pour déterminer si les chaînes de caractères* `s1` *et* `s2` *sont égales.*
3. *Écrire une autre fonction* `palindrome_bis` *qui fait la même chose sans utiliser la fonction* `reverse`. *Votre fonction compare-t-elle chaque paire de lettres une seule fois ? Si non, comment l'améliorer ?*

□

Exercice 3 (Premier, ⋆⋆)
Ecrire une fonction `isPrime` *qui renvoie un booléen indiquant si son argument est un nombre premier.*

> ***Contrat:***
> *Par exemple,* `isPrime(17)` *renvoie* `true`*, alors que* `isPrime(12)` *et* `isPrime(1)` *renvoient* `false`.

□

Bonus

Exercice 4 (Nombres amicaux, ⋆⋆⋆)
Vous avez déjà vu les nombres parfaits dans le Cours-TD 2, les nombres amicaux sont une notion proche.

1. *Écrire une fonction* `sumDiv` *qui renvoie la somme des diviseurs propres d'un entier.*

   > ***Contrat:***
   > *Par exemple,* `sumDiv(6)` *vaut 6,* `sumDiv(1184)` *vaut 1210.*

2. *Deux entiers* `n` *et* `m` *sont dits amicaux si* `sumDiv(n) == m` *et* `sumDiv(m) == n`.
   Vérifier que 1184 et 1210 sont amicaux.

3. *Utiliser cette caractérisation pour trouver un couple de nombres amicaux inférieurs à 500. Indication : On pourra utiliser une boucle imbriquée.*
4. *On veut maintenant trouver un couple de nombres amicaux supérieurs à 10000. Est-ce-que la méthode que vous avez utilisée à la question précédente fonctionne encore ? Si non, trouver une méthode qui vous permettra de déterminer un tel couple. Ecrire une fonction* `firstFriendlyAfter(n)` *qui renvoie le plus petit entier* $\mathtt{m} \geq \mathtt{n}$ *tel qu'il a un pair amical.*

□