TP 06

Fonctions récursives

Savoirs et compétences :

- ❒ *AA.C9 : Choisir un type de données en fonction d'un problème à résoudre*
- ❒ *AA.S12 : Fichiers*

Activité 1 – Introduction et définition

Définitions

Définition Une fonction récursive est une fonction qui s'appelle elle-même.
On appelle récursion l'appel de la fonction à elle-même.

La programmation est un paradigme de programmation au même titre que la programmation itérative. Un programme écrit de manière récursive peut être traduit de manière itérative, même si dans certain cas, cela peut s'avérer délicat.

Méthode
- Une fonction récursive doit posséder une condition d'arrêt (ou cas de base).
- Une fonction récursive doit s'appeler elle-même (récursion).
- L'argument de l'étape de récursion doit évoluer de manière à se ramener à la condition d'arrêt.

Exemples

Suites récurrentes

Soit la suite u_n définie par récurrence pour tout $n \in \mathbb{N}^*$ par $\begin{cases} u_1 = 1 \\ u_{n+1} = \dfrac{u_n+6}{u_n+2} \end{cases}$.

```
def un_it (n : int) -> float :
    if n == 1 :
        return 1
    else :
        u = 1
        for i in range(2,n+1):
            u = (u+6)/(u+2)
        return u
```

```
def un_rec (n : int) -> float :
    if n == 1 :
        return 1
    else :
        return (un_rec(n-1)+6)/(un_rec(n-1)+2)
```

Activité 2 – Programmation de la factorielle n!

On rappelle la définition de la factorielle : $n! = \prod_{i=1}^{n} i$ et $n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{si } n \geq 1 \end{cases}$.

Question 1 *Proposer une fonction* `fact_it(n:int)->int` *renvoyant la factorielle de n par une méthode itérative*

Question 2 *Proposer une fonction* `fact_rec(n:int)->int` *renvoyant la factorielle de n par une méthode récursive*

Activité 3 – Recherche dichotomique

On se donne une liste `L` de nombres de longueur `n`, triée dans l'ordre croissant, et un nombre `x0`.

Pour chercher `x0`, on va couper la liste en deux moitiés et chercher dans la moitié de tableau qui encadre `x0` et ainsi de suite...

On appelle `g` l'indice de l'élément du début de la sous-liste dans laquelle on travaille et `d` l'indice de l'élément de fin.

Au début, `g = 0` et `d = n - 1`.

On utilise la méthode suivante :

- on compare `x0` à « l'élément du milieu » `L[m]` avec `m = (g + d) // 2`;
- si `x0 = L[m]`, on a trouvé `x0`, on peut alors s'arrêter;
- si `x0` < `L[m]`, c'est qu'il faut chercher entre `L[g]` et `L[m-1]`;
- si `x0` > `L[m]`, c'est qu'il faut chercher entre `L[m+1]` et `L[d]`.

On poursuit jusqu'à ce qu'on a trouvé `x0` ou lorsque l'on a épuisé la liste `L`.

On donne l'algorithme de recherche dichotomique construit avec une méthode itérative.

```
def dichotomie(x0,L):
    g=0
    d=len(L)-1
    m=(g+d)//2
    while g<=d:
        if L[m]==x0:
            return True
        elif L[m]<x0:
            g=m+1
        else:
            d=m-1
        m=(g+d)//2
    return False
```

Question 3 *Écrire une fonction* `dichotomie_rec(x0,L)` *qui renvoie* `True` *ou* `False` *selon que* `x0` *figure ou non dans* `L` *par cette méthode. On utilisera une boucle* `while` *que l'on interrompra soit lorsque l'on a trouvé* `x0`*, soit lorsque l'on a fini de parcourir la liste.*

Activité 4 – Recherche de palindrome

On appelle palindrome un mot qui peut se lire indifféremment de gauche à droite ou de droite à gauche. Ainsi par exemple *radar*, *rotor* ou kayak sont trois palindromes.

Question 4 *Écrire une fonction récursive* `palindrome(x:str) -> bool` *prenant en paramètre une chaîne de caractère* `x` *et qui retourne* `True` *si il s'agit d'un palindrome et* `False` *sinon. (Les espaces seront gérés comme de simples caractères). Pour cela on remarquera que :*

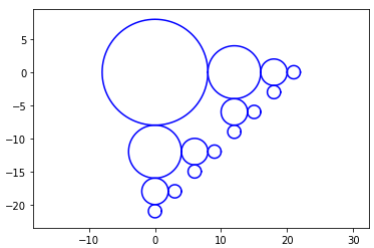
- *tout mot de longueur* ≤ 1 *est un palindrome;*
- *un mot est un palindrome si et seulement si ses premier et dernier caractères sont identiques et son sous-mot allant du* $2^{ème}$ *caractère jusqu'à l'avant dernier est un palindrome.*

Activité 5 – Bubble bobble

On suppose disposer d'une fonction `circle(coords:list, r:float) -> None` qui trace à l'écran un cercle de centre `coords=(x,y)` et de rayon `r`.

```
import matplotlib.pyplot as plt
import numpy as np
def circle(coords:list, r:float) -> None :
    X, Y = [], []
    for t in range(101):
        X.append(coords[0]+r*np.cos(t*np.pi/50))
        Y.append(coords[1]+r*np.sin(t*np.pi/50))
    plt.plot(X, Y, 'b')
```

Question 5 *Définir la fonction* `bubble1(n:int, x:float, y:float, r:float) -> None` *permettant de tracer la figure suivante.*



Question 6 *Définir la fonction* `bubble2(n:int, x:float, y:float, r:float, d:str) -> None` *permettant de tracer la figure suivante.*