python

```
print("Hello, world!")
```

FORMATION PYTHON : Les algorithmes de recherche

Ce chapitre a pour but d'initier aux algorithmes de recherche en Python.

https://sites.google.com/site/saiedanis87

said_anis
@hotmail
.com

Introduction

- Les algorithmes de recherche sont très utiles en informatique. Leur rôle est de :
  - Vérifier l'existence d'un ou de plusieurs éléments, satisfaisant une condition donnée, dans un tableau ;
  - Chercher le nombre d'occurrences d'un élément ;
  - Chercher la ou les positions d'un élément donné dans un tableau; etc.
- Il existe essentiellement deux stratégies de recherche :
  - Recherche *séquentielle* ou encore linéaire
  - Recherche *dichotomique*.

Plan

1. RECHERCHE SÉQUENTIELLE
2. RECHERCHE PAR DICHOTOMIQUE
3. EXERCICES (TD)

1. RECHERCHE SEQUENTIELLE

said_anis
@hotmail
.com

1. Recherche séquentielle

- La recherche séquentielle d'un élément dans un tableau consiste à parcourir le tableau séquentiellement jusqu'à trouver l'élément recherché ou atteindre la fin du tableau.
- La recherche peut se faire sur un tableau trié ou non, la différence réside dans le parcours du tableau et dans la condition d'arrêt de la recherche.

1. Recherche séquentielle

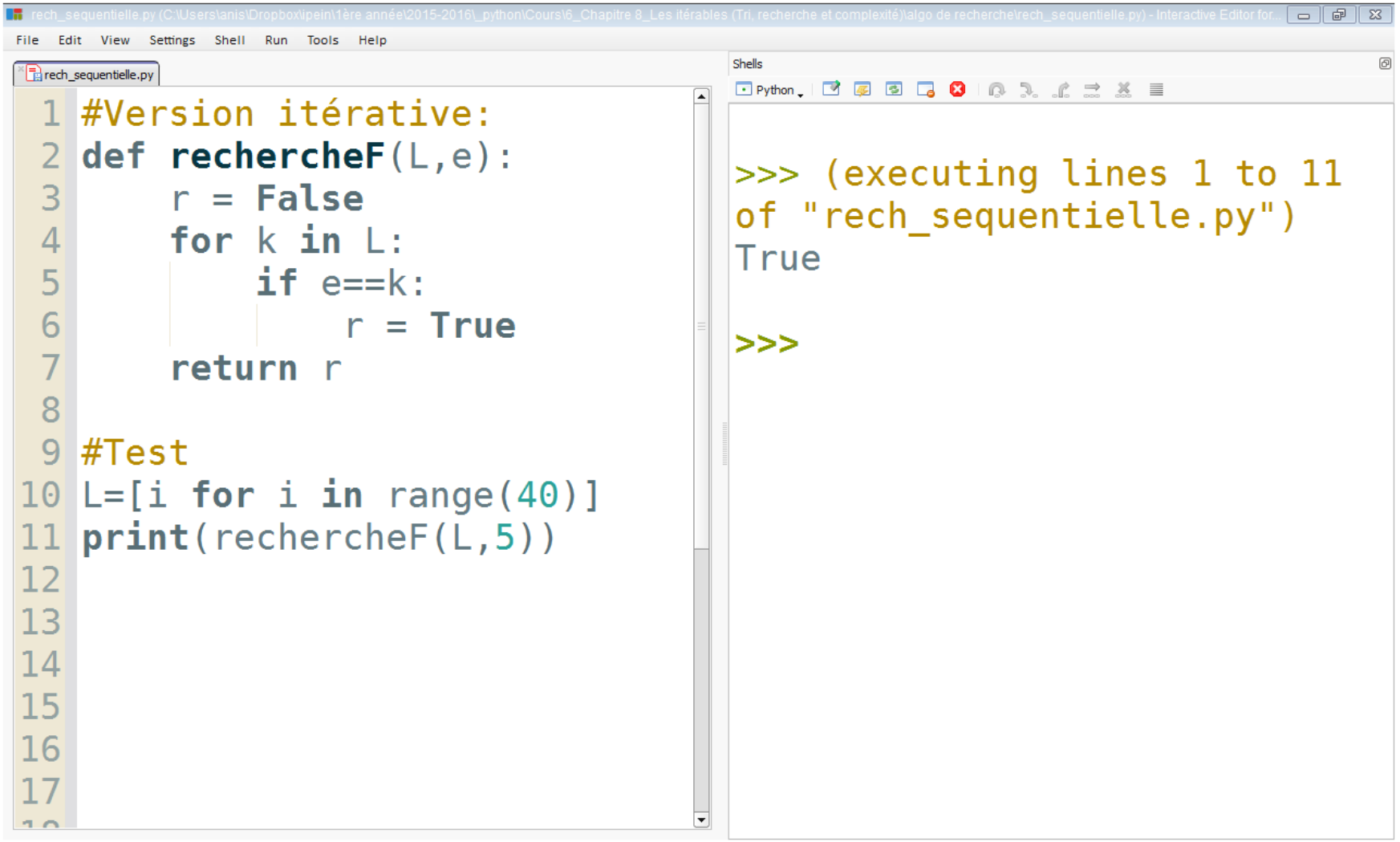
- On considère une liste L ou un tableau T et un objet e.

Problématique:

On souhaite tout d'abord de tester l'appartenance de e à L.

- Des primitives de Python réalisent ce travail (e in L), mais notre objectif est de comprendre comment on les réalise.

1. Recherche séquentielle : V.I.



1. Recherche séquentielle

Complexité :

- Un appel à la procédure rechercheF(L,e) provoque n itérations,

  avec n = len(L), dans tous les cas ce qui fait n comparaisons

  if (e==L[k])

- O(n) : Complexité linéaire

1. Recherche séquentielle

Exercice :

- Écrire une fonction qui renvoie les deux plus grands éléments d'un tableau d'entiers. On supposera que le tableau est de longueur au moins 2 ; en revanche, on veillera à ne le parcourir qu'une seule fois.

1. Recherche séquentielle

Exercice :

1. Écrire une fonction qui renvoie l'indice de la première occurrence de l'élément maximal d'un tableau d'entiers.
2. Évaluer la complexité de cette fonction.

1. RECHERCHE DICHOTOMIQUE

said_anis
@hotmail
.com

2. Recherche dichotomique

- L'algorithme que nous proposons ici suppose que la liste dans laquelle on recherche un élément est **triée**.
- Quand la liste (tableau) est triée, la recherche d'un élément dans une liste peut être réalisée de manière plus efficace en appliquant le principe "*diviser pour régner*".
- L'idée est que l'on compare l'élément cherché au terme du milieu de la liste, ce qui conduit, tant qu'on ne le trouve pas, à le rechercher soit dans la première partie soit dans la seconde partie de la liste.

2. Recherche dichotomique : principe

Exemple : Chercher la valeur 9 dans le tableau [1, 3, 5, 6, 9, 12, 14].

La recherche s'effectue ainsi :

On cherche dans *a*[0:7] : 1 3 5 6 9 12 14 avec : len(a)=7

On compare *x* à *a*[3] : 1 3 5 6 9 12 14 9 >6 => continuer à droite

On cherche x dans *a*[4:7] : 1 3 5 6 9 12 14

On compare *x* à *a*[5] : 1 3 5 6 9 12 14 12>9 => continuer à gauche

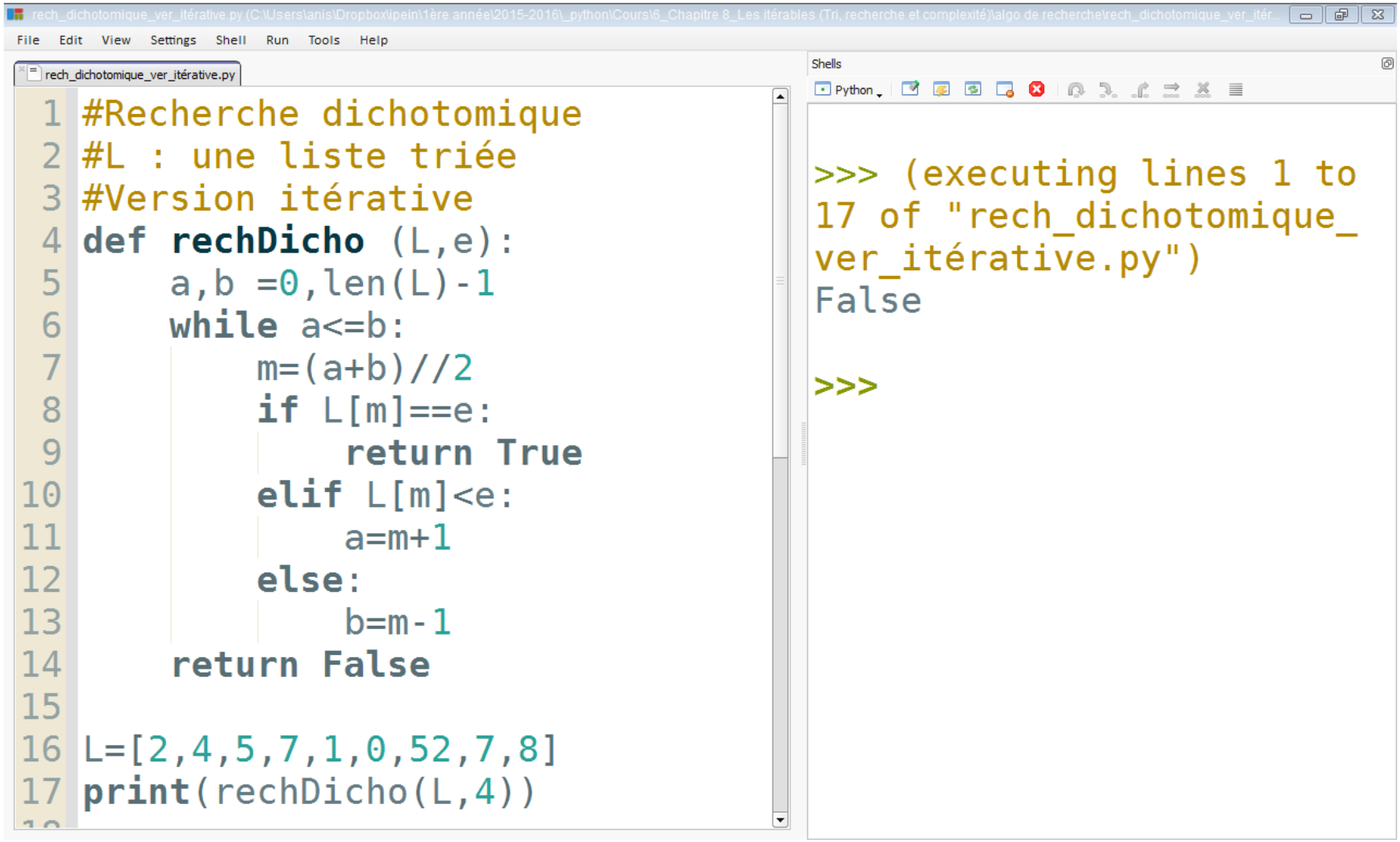
On cherche dans *a*[4:4] : 1 3 5 6 9 12 14

On compare *x*=9 avec *a*[4]=9 :

1 3 5 6 9 12 14 9=9

La valeur 9 a été trouvée en seulement trois.

2. Recherche dichotomique : V.I.



2. Recherche dichotomique : V.R.


```
#Recherche dichotomique dans une liste triée
def rechDicho (L,e):
    a,b=0,len(L)
    m=(a+b)//2
    if(a==b==0):
        return False
    elif L[m]==e:
        return True
    elif L[m]>e:
        return rechDicho(L[:m],e)
    else:
        return rechDicho(L[m+1:],e)

#Test : La liste (tableau) doit être trié
L=[2,4,5,6,7,8,9]
print(rechDicho(L,59))
```

2. Recherche dichotomique : principe

Complexité :

- La complexité de cette fonction est au pire en $O(\log_2 n)$ où n est la longueur du tableau initial car à chaque itération la longueur du tableau est divisée en deux.
- La complexité de la recherche dichotomique est au pire en $O(\log n)$ où n est la longueur du tableau, alors que celle de la recherche séquentielle est $O(n)$.
- Il ne faut cependant pas oublier qu'elles ne s'appliquent pas dans les mêmes conditions : une recherche dichotomique est exclue si les données ne sont pas triées.

2. Recherche dichotomique : Exercice

Recherche d'un mot (une séquence) dans un texte

- Un problème classique en informatique consiste à rechercher, non pas une seule valeur, mais une *séquence* de valeurs dans un tableau. Cela revient à chercher une occurrence d'un tableau dans un autre ou, pour les chaînes de caractères, une occurrence d'un mot dans un texte.
- On souhaite donc écrire une fonction **recherche_mot** qui, étant donnés deux tableaux *m* et *t*, détermine la position de la première occurrence de *m* dans *t*, si elle existe, et qui renvoie **None** sinon.

Exemple : pour les tableaux *m*=[1,2,3] et *t*=[2,1,4,1,2,6,1,2,3,7],

La fonction **recherche_mot** renvoie 6 : [2, 1, 4, 1, 2, 6, ***1, 2, 3***, 7]

2. Recherche dichotomique : Exercice

Solution :

```
def recherche_mot(m, t):
    """renvoie, si elle existe, la position de la première
    occurrence du mot m dans le texte t et renvoie None sinon"""
    for i in range(len(t)+1 - len(m)):#pour pas avoir de débordement
    #si m apparaît à la position i, comparer les caractères
    # de m et t un à un,sinon on passe au i suivant
        j = 0
        while j < len(m) and m[j] == t[i + j]:
            j += 1
        if j == len(m):  # mot m existe à la position i
            return i
    return None  # pas d'occurrence de m dans t
```

2. Recherche dichotomique : Exercice

1. Modifier la fonction **recherche_mot** pour qu'elle retourne *toutes* les occurrences de *m* dans *t*. La complexité est-elle différente après cette modification ?
2. Ecrire une fonction qui vérifie qu'une chaîne de caractères est un palindrome, c'est-à-dire qu'elle est identique qu'on la lise de gauche à droite ou de droite à gauche.
3. Adapter cette fonction pour qu'elle ne tienne pas compte des espaces ni des signes de ponctuation.
4. Écrire une fonction qui vérifie qu'une chaîne de caractères est composée uniquement de lettres de l'alphabet, d'espaces et des symboles de ponctuation usuels.
5. Évaluer sa complexité.

2. Recherche dichotomique : Exercice

Solution :