Sujet BAC 16: Récursivité



1. Cours

Vous pouvez télécharger une copie au format pdf du diaporama de synthèse de cours présenté en classe .

Diaporama de cours



Attention

Ce diaporama ne vous donne que quelques points de repères lors de vos révisions. Il devrait être complété par la relecture attentive de vos **propres** notes de cours et par une révision approfondie des exercices.

2. France Septembre 2022 : Session de remplacement.

Société immobiliére

Thèmes abordés

- Notions de classes
- Itération
- Récursivité

Une petite société immobilière ne voulant pas investir dans une base de données nécessitant une mise en place longue et fastidieuse a cré un fichier .csv pour stocker ses annonces. La consultation et la mise en place de ce fichier ne seront pas étudiées ici. Pour limiter notre étude, nous considérerons que les données sont stockées temporairement dans une liste v dont voici la structure simplifiée :

#pieces class Piece: def __init__(self,a,b): self.nom = a #nom self.sup = b # superficie de la piece

```
def sup(self):
      return self.sup
#villas
class villa:
   def init (self,a,b,c,d,e):
      self.nom = a #nom de la villa
      self.sejour = b #caracteristiques sejour
      self.ch1 = c #caracteristiques de la 1ere chambre
      self.ch2 = d #caracteristiques de la chambre 2
      self.eqCuis = e # equipement de la cuisine 'eq' ou 'non eq'
   def nom(self):
       return self.nom
   def surface(self):
      return ... ...
   def equip(self):
      return self.eqCuis
#Programme principal
v=[]
v.append(Villa("Les quatre vents", Piece("séjour",40),Piece("Ch1",10),Piece("ch2",20),"eq")))
v.append(Villa("Les goélands", Piece("séjour",50),Piece("Ch1",15),Piece("ch2",15),"eq")))
v.append(Villa("Rêve d'été" , Piece("séjour", \textcolor{red}{\textbf{30}}), Piece("Ch1", \textcolor{red}{\textbf{15}}), Piece("ch2", \textcolor{red}{\textbf{20}}), "non eq")))
v.append(Villa("Les oliviers", Piece("s\'ejour", \textcolor{red}{\bf 30}), Piece("Ch1", \textcolor{red}{\bf 10}), Piece("ch2", \textcolor{red}{\bf 20}), "eq")))
v.append(Villa("Bellevue" \ , \ Piece("s\'ejour", \textcolor{red}{\textbf{30}}), Piece("Ch1", \textcolor{red}{\textbf{10}}), Piece("ch2", \textcolor{red}{\textbf{20}}), "non \ eq")))
```

Bug

- Le code proposé définit pour un objet de la classe Villa un attribut nom ainsi qu'une méthode nom, ce qui n'est pas possible. La méthode est un *getter* qu'on pourrait renommer par exemple en *get_nom*.
- L'attribut eqCuis ne prend que deux valeurs "eq" ou "noneq", il serait donc plus judicieux d'en faire un booléen.

La structure de données retenue pour l'exercice est définie par la classe Villa.

2.1. Partie A: analyse du code et complétion

Question 1

- 1.a Combien d'éléments contient la liste v?
- **1.b** Que retourne l'instruction v[1] .nom() ?

Pour accéder à l'information de la surface habitable du logement, le developpeur souhaite ajouter la méthode surface() qui renvoie cette surface.

1.c Compléter sur votre copie la méthode surface()



L'agent immobilier veut pouvoir consulter les villas qui disposent d'une cuisine équipée. Pour cela vous devez écrire la portion du programme qui affichera la liste des villas équipées. Elle devra parcourir séquentiellement la liste et afficher à l'écran le nom de chaque villa équipée.

Question 2

Rédigez sur votre copie la portion de programme réalissant cette sélection.

2.2. Partie B: Récursivité

L'agent immobilier veut répondre à une demande de client : "Quelle est votre villa la plus grande ?". Pour cela, il souhaite disposer d'une fonction max_surface() qui va extraire de la liste v, la villa désirée. Nous avons décidé d'écrire cette fonction de façon récursive.

Question 3

Recopiez parmi les propositions suivantes celle qui caractèrise un appel récursif.

- appel d'une fonction par elle-même
- appel dont l'exécution est un processus itératif.
- appel d'une fonction comportant une boucle.

L'agorithme suivant a été choisi :

Il faut partir d'une liste de villas :

- si cette liste contient un seul élèment, c'est le résultat;

- si la liste en continet plusieurs, il faut analyser les deux premiers élèments, élimier la villa de plus petite surface, et recommencer avec la liste tronquée.

A la fin du processus, une seule villa est renvoyée.

Pour écrire cette fonction, dans un premier temps, nous alloons donc distinguer deux cas :

- celui àù la liste des villas ne contient qu'un villa :



il faut renvoyer la villa

- celui où la liste en contient au moins deux :

il faut supprimer v[1] de la liste

Texte

si la surface de la villa v[0] est inférieure à celle de la villa v[1] il faut supprimer v[0] de la liste sinopn

Question 4

Ecrivez sur votre copie le code de cette fonction en Python.

& Script Python

def max_surface(v):

Reponse

- 1. a. La liste v contient 5 éléments (elle était initialement vide et on y ajouté 5 éléments à l'aide de 5 append)
 - b. v[1] est le deuxième élément de la liste v c'est à dire un objet de la classe Villa sa méthode self.nom renvoie l'attribut nom c'est à dire "Les goélands".
 - c. Le calcul de la surface s'effectue naturellement en sommant la surface de chacune des pièces de la villa :

```
& Script Python

def surface(self):
    return self.sejour.sup() + self.ch1.sup() + self.ch2.sup()
```

2. On parcours la liste v des villas et on affiche les noms de celles ayant une cuisine équipée (la méthode equip renvoie "eq" lorsque la cuisine est équipée).

```
Script Python

for villa in v:
    if villa.equip()=="eq":
        print(villa.nom())
```

- 3. Un appel récursif est caractérisé par appel d'une fonction par elle-même
- def max_surface(v):
 if len(v)==1:
 return v.nom()
 if v[0].surface() > v[1].surface():
 v.pop(1)
 else:

return max surface(v)

🐍 Script Python

v.pop(0)

Remarque

4.

Le type list étant mutable, la fonction ci-dessous vide en même temps la liste v qui contiendra à la fin un unique élément (celui ayant la surface maximale)

3. Sujet B 2023

Fonctions récursives

Thèmes abordés

- Récursivité
- \bullet Cet exercice est consacré à l'analyse et à l'écriture de programmes récursifs.

1.a) Expliquer en quelques mots ce qu'est une fonction récursive.

Réponse

Une fonction récursive est une fonction qui possède un appel à elle-même dans son code source.

1.b) On considère la fonction Python suivante :

```
def compte_rebours(n):
    """ n est un entier positif ou nul """
    if n >= 0:
        print(n)
        compte_rebours(n - 1)
```

L'appel compte_rebours(3) affiche successivement les nombres 3, 2, 1 et 0. Expliquer pourquoi le programme s'arrête après l'affichage du nombre 0.

Réponse

Une fois l'affichage de 0 effectué, il y a un appel récursif compte rebours(0 - 1).

Lors de cet appel récursif, n vaut -1, on ne rentre pas donc dans la structure conditionnelle.

La pile d'appel récursif se vide sans qu'il ait d'autres instructions effectuées.

Ainsi le programme s'arrête après avoir affiché 0 et vidé la pile d'appels récursifs.

- **2.** En mathématiques, la factorielle d'un entier naturel (n) est le produit des nombres entiers strictement positifs inférieurs ou égaux à (n). Par convention, la factorielle de (0) est (1). Par exemple :
 - la factorielle de \(1\) est \(1\)
 - la factorielle de (2) est $(2 \times 1 = 2)$
 - la factorielle de (3) est $(3 \times 2 \times 1 = 6)$
 - la factorielle de (4) est $(4 \times 3 \times 2 \times 1 = 24)$

Recopier et compléter sur votre copie le programme donné ci-dessous afin que la fonction récursive fact renvoie la factorielle de l'entier passé en paramètre de cette fonction.

Exemple: fact(4) renvoie 24.

```
def fact(n):
    """ Renvoie le produit des entiers strictement positifs
    et inférieurs ou égaux à n.
    if n == 0:
```

```
if n == 0:
    return ... # À compléter
else:
    return ... # À compléter
```

_

Réponse

```
& Script Python
```

```
def fact(n):
    """ Renvoie le produit des entiers strictement positifs
    et inférieurs ou égaux à n.
    """

if n == 0:
    return 1
else:
    return n * fact(n - 1)
```

3. La fonction somme_entiers_rec ci-dessous permet de calculer la somme des entiers, de 0 à l'entier naturel n passé en paramètre.

Par exemple:

- Pour n = 0, la fonction renvoie la valeur 0.
- Pour n = 1, la fonction renvoie la valeur 0 + 1 = 1.
- ...
- Pour n = 4, la fonction renvoie la valeur 0 + 1 + 2 + 3 + 4 = 10.

```
1
    def somme entiers rec(n):
2
       """ Renvoie, de manière récursive,
      la somme des entiers de 0 à l'entier naturel n.
3
4
5
      if n == 0:
6
         return 0
7
      else:
         print(n) # pour vérification
8
9
         return n + somme entiers(n - 1)
```

L'instruction print(n) de la ligne 7 dans le code précédent a été insérée afin de mettre en évidence le mécanisme en œuvre au niveau des appels récursifs.

3.a) Écrire ce qui sera affiché dans la console après l'exécution de la ligne suivante :

```
Console Python
>>> res = somme_entiers_rec(3)
```

Réponse Console Python >>> res = somme_entiers_rec(3) 3 2 1 >>> L'appel somme_entiers_rec(3) affiche 3 puis appelle somme_entiers_rec(2) L'appel somme_entiers_rec(2) affiche 2 puis appelle somme_entiers_rec(1) L'appel somme_entiers_rec(1) affiche 3 puis appelle somme_entiers_rec(0) L'appel somme_entiers_rec(0) n'affiche rien.

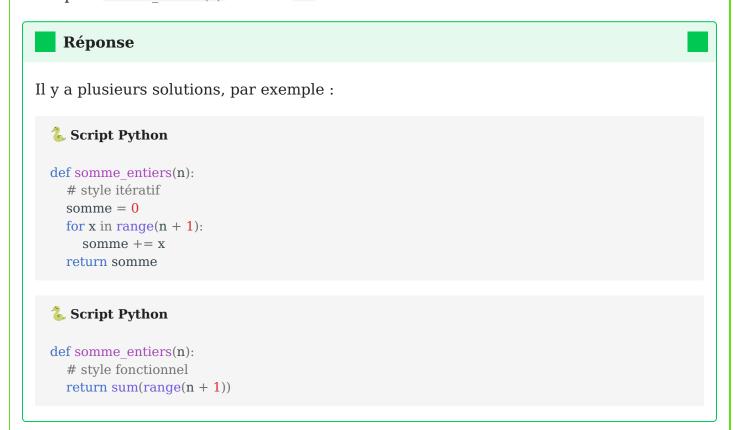
3.b) Quelle valeur sera alors affectée à la variable res?



La valeur 6 est affectée à la variable res, la somme \(3+2+1+0\).

4. Écrire en Python une fonction somme_entiers non récursive : cette fonction devra prendre en argument un entier naturel n et renvoyer la somme des entiers de 0 à n compris. Elle devra donc renvoyer le même résultat que la fonction somme_entiers_rec définie à la question 3.

Exemple: somme_entiers(4) renvoie 10.



4. Centres Etrangers J2 2022

Produit de chaines de caractères par un entier

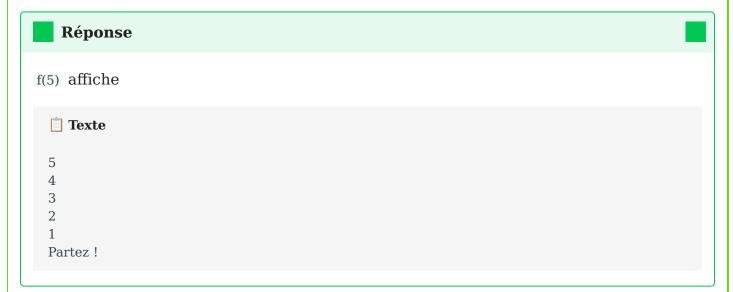
Thèmes abordés

- Programmation
- Récursivité

1. Voici une fonction codée en Python :

```
def f(n):
    if n == 0:
        print("Partez !")
    else:
        print(n)
        f(n-1)
```

1.a. Qu'affiche la commande f(5) ?



1.b. Pourquoi dit-on de cette fonction qu'elle est récursive?



Le code source de f contient un appel à elle-même, c'est donc une fonction récursive.

2. On rappelle qu'en python l'opérateur + a le comportement suivant sur les chaines de caractères :

```
Console Python
>>> S = 'a' + 'bc'
>>> S
'abc'
```

Et le comportement suivant sur les listes :

```
& Console Python

>>> L = ['a'] + ['b', 'c']

>>> L

['a', 'b', 'c']
```

On a besoin pour les questions suivantes de pouvoir ajouter une chaine de caractères s en préfixe à chaque chaine de caractères de la liste chaines .

On appellera cette fonction ajouter.

Par exemple, ajouter("a", ["b", "c"]) doit renvoyer ["ab", "ac"].

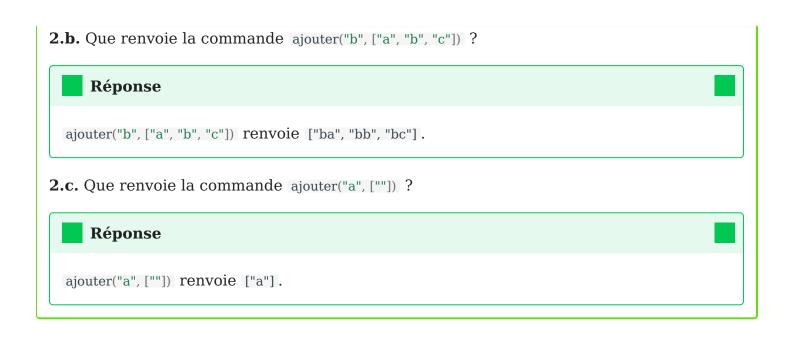
2.a. Recopiez le code suivant et complétez ... sur votre copie :

```
def ajouter(s, chaines):
    resultat = []
    for mot in chaines:
        resultat ...
    return resultat
```

```
Réponses

Caript Python

def ajouter(s, chaines):
    resultat = []
    for mot in chaines:
        resultat.append(s + mot)
    return resultat
```



3. On s'intéresse ici à la fonction suivante écrite en Python où s est une chaine de caractères et n un entier naturel.

def produit(s, n): if n == 0: return [""] else: resultat = [] for c in s: resultat = resultat + ajouter(c, produit(s, n - 1)) return resultat

3.a. Que renvoie la commande produit("ab", 0) ? Le résultat est-il une liste vide ?

Réponse

produit("ab", 0) utilise le paramètre n égal à 0 , donc elle renvoie [""]. (L'affichage en console sera [''])

[""] n'est pas une liste vide, c'est une liste qui contient **un** élément : la chaine de caractères vide.

3.b. Que renvoie la commande produit("ab", 1) ?

Réponse

produit("ab", 1) fait une boucle for avec deux tours:

- Premier tour, avec c = 'a', resultat devient ["a"].
- Second tour, avec c = b', resultat devient ["a"] + ["b"].

['a', 'b'] est renvoyé.

3.c. Que renvoie la commande produit("ab", 2) ?

Réponse

produit("ab", 2) fait une boucle for avec deux tours :

- Premier tour, avec c = 'a', resultat devient ["aa", "ab"].
- Second tour, avec c = b', resultat devient ["aa", "ab"] + ["ba", "bb"].

['aa', 'ab', 'ba', 'bb'] est renvoyé.

5. Polynésie J1 2022

Construction récursive de chaine de caractères

Thèmes abordés

- Programmation
- Récursivité

On rappelle qu'une chaine de caractères peut être représentée en Python par un texte entre guillemets et que :

- la fonction len renvoie la longueur de la chaine de caractères passée en paramètre ;
- si une variable ch désigne une chaine de caractères, alors ch[0] renvoie son premier caractère, ch[1] le deuxième, etc;
- l'opérateur + permet de concaténer deux chaines de caractères.

Exemples

& Console Python

```
>>> texte = "bricot"

>>> len(texte)

6

>>> texte[0]

"b"

>>> texte[1]

"r"

>>> "a" + texte

"abricot"
```

On s'intéresse dans cet exercice à la construction de chaines de caractères suivant certaines règles de construction.

Règle A

Une chaine est construite suivant la règle A dans les deux cas suivants :

- soit elle est égale à "a" ;
- soit elle est de la forme "a" + chaine + "a", où chaine est une chaine de caractères construite suivant la règle A.

Règle B

Une chaine est construite suivant la règle B dans les deux cas suivants :

- soit elle est de la forme "b" + chaine + "b", où chaine est une chaine de caractères construite suivant la règle A;
- soit elle est de la forme "b" + chaine + "b", où chaine est une chaine de caractères construite suivant la règle B.

Fonction choice du module random

On a reproduit ci-dessous l'aide de la fonction choice du module random.

& Console Python

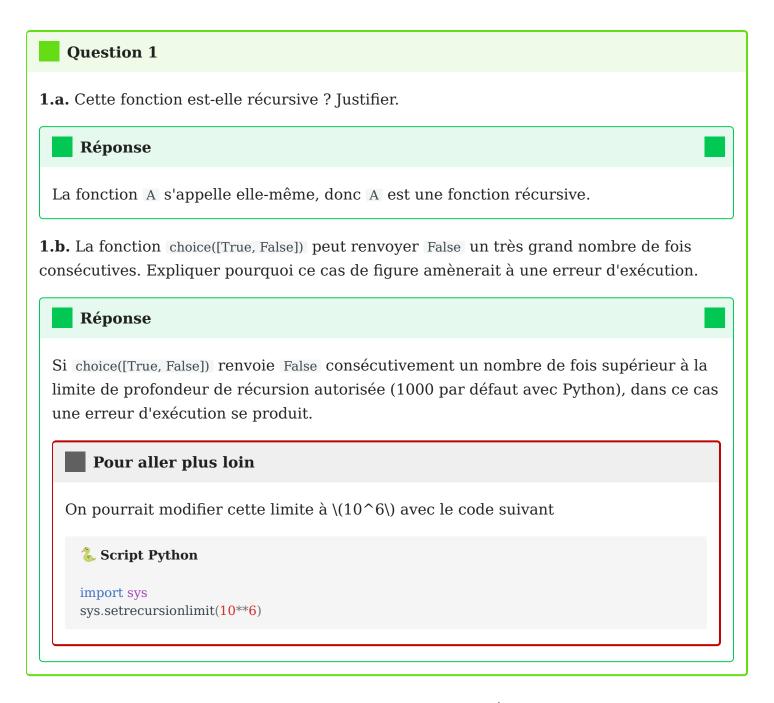
```
>>> from random import choice
>>> help(choice)
Help on method choice in module random:
choice(seq) method of random.Random instance
Choose a random element from a non-empty sequence.
```

choice(seq) renvoie un élément de seq (qui peut être une liste) de façon pseudo-aléatoire.

La fonction A ci-dessous renvoie une chaine de caractères construite suivant la règle A, en choisissant aléatoirement entre les deux cas de figure de cette règle.

& Script Python def A(): if choice([True, False]): return "a" else:

return "a" + A() + "a"



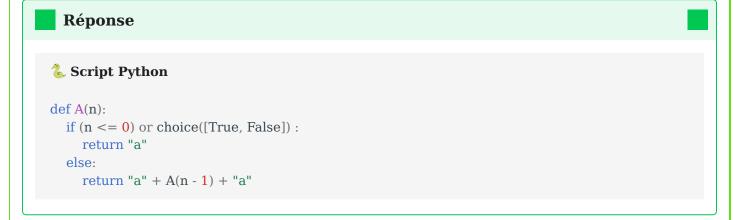
Dans la suite, on considère une deuxième version de la fonction A. À présent, la fonction prend en paramètre un entier n tel que,

- si la valeur de n est négative ou nulle, la fonction renvoie "a" ;
- si la valeur de n est strictement positive, elle renvoie une chaine de caractères construite suivant la règle A avec un n décrémenté de 1, en choisissant aléatoirement entre les deux cas de figure de cette règle.

```
def A(n):
    if ... or choice([True, False]) :
        return "a"
    else:
        return "a" + ... + "a"
```



2.a. Recopier sur la copie et compléter aux emplacements des points de suspension ... le code de cette nouvelle fonction A.



2.b. Justifier le fait qu'un appel de la forme A(n) avec n un nombre entier positif inférieur à 50, termine toujours.

Réponse

Pour (n > 0), l'appel à A(n) provoque ou bien un arrêt de la fonction, ou bien un appel récursif avec le paramètre n-1.

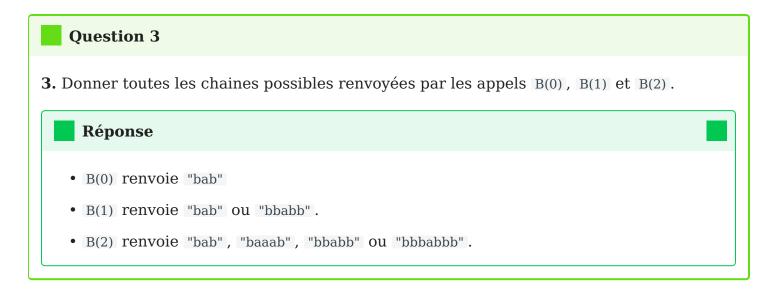
Un appel à A(50) pourrait provoquer dans le pire des cas 50 appels récursifs pour arriver à A(0) qui termine, ou alors terminer avant !

On donne ci-après le code de la fonction récursive B qui prend en paramètre un entier n et qui renvoie une chaine de caractères construite suivant la règle B.

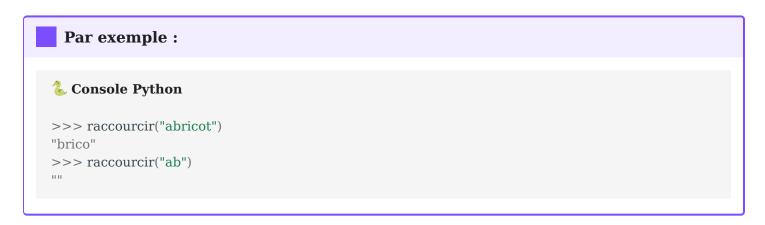
```
def B(n):
    if (n <= 0) or choice([True, False]):
        return "b" + A(n - 1) + "b"
    else:
        return "b" + B(n - 1) + "b"</pre>
```

On admet que:

- les appels A(-1) et A(0) renvoient la chaine "a" ;
- l'appel A(1) renvoie la chaine "a" ou la chaine "aaa" ;
- l'appel A(2) renvoie la chaine "a", la chaine "aaa" ou la chaine "aaaaa".



On suppose maintenant qu'on dispose d'une fonction raccourcir qui prend comme paramètre une chaine de caractères de longueur supérieure ou égale à 2, et renvoie la chaine de caractères obtenue à partir de la chaine initiale en lui ôtant le premier et le dernier caractère.



4.a. Recopier sur la copie et compléter les points de suspension ... du code de la fonction regle_A ci-dessous pour qu'elle renvoie True si la chaine passée en paramètre est construite suivant la règle A, et False sinon.

```
def regle_A(chaine):
    n = len(chaine)
    if n >= 2:
        return (chaine[0] == "a") and (chaine[n - 1] == "a") and regle_A(...)
    else:
        return chaine == ...
```

Réponse Constitution def regle_A(chaine): n = len(chaine): if n >= 2: return (chaine[0] == "a") and (chaine[n - 1] == "a") and regle_A(raccourcir(chaine))) else: return chaine == "a"

4.b. Écrire le code d'une fonction regle_B, prenant en paramètre une chaine de caractères et renvoyant True si la chaine est construite suivant la règle B, et False sinon.

```
Réponse

def regle_B(chaine):
    n = len(chaine)
    if n >= 2:
        return (chaine[0] == "b") and (chaine[n - 1] == "b") and (
            regle_A(raccourcir(chaine)) or regle_B(raccourcir(chaine))
    )
    else:
        return False
```

6. France J2 2021 - Candidat libre

Mélange d'une liste

Thèmes abordés

Programmation
Récursivité

On s'intéresse dans cet exercice à un algorithme de mélange des éléments d'une liste.

1. Pour la suite, il sera utile de disposer d'une fonction echange qui permet d'échanger dans une liste valeurs les éléments d'indice i et j.

Expliquer pourquoi le code Python ci-dessous ne réalise pas cet échange et en proposer une modification.

& Script Python def echange(valeurs, i, j): valeurs[j] = valeurs[i] valeurs[i] = valeurs[j]

Réponse

On perd la valeur initiale de <code>list[j]</code> dès la première instruction.

On peut procéder de deux façons différentes :

• Avec une variable temporaire :

% Script Python

```
def echange(valeurs, i, j):
  temp = valeurs[j]
  valeurs[j] = valeurs[i]
  valeurs[i] = temp
```

• En utilisant l'affectation multiple :

& Script Python

```
\begin{aligned} & def \ echange(valeurs, \ i, \ j); \\ & valeurs[i], \ valeurs[j] = valeurs[j], \ valeurs[i] \end{aligned}
```

2. La documentation du module random de Python fournit les informations ci-dessous concernant la fonction randint :

randint(a, b)

Renvoie un entier aléatoire N tel que $a \le N \le b$. Alias pour randrange(a, b + 1).

Parmi les valeurs ci-dessous, quelles sont celles qui peuvent être renvoyées par l'appel randint(0, 10) ?

- 0
- 1
- 3.5
- 9
- 10
- 11

Réponse

L'appel randint(0, 10) renvoie une valeur entière entre 0 et 10 inclus l'un et l'autres.

Donc 0, 1, 9 et 10 sont des valeurs possibles.

3. Le mélange de Fischer-Yates est un algorithme permettant de permuter aléatoirement les éléments d'une liste. On donne ci-dessous une mise en œuvre récursive de cet algorithme en Python.

```
from random import randint

def melange(valeurs, i):
    print(valeurs)
    if i > 0:
        j = randint(0, i)
        echange(valeurs, i, j)
        melange(valeurs, i - 1)
```

3.a. Expliquer pourquoi la fonction melange se termine toujours.

Réponse

On suppose que i est un entier positif compris entre 0 et l'indice du dernier élément de la liste (len(valeurs) - 1).

Lors des appels récursifs, on décrémente la valeur de [i] et ces appels n'ont lieu que si cette valeur est strictement positive. Donc la fonction s'arrêtera toujours.

3.b. Lors de l'appel de la fonction melange, la valeur du paramètre i doit être égal au plus grand indice possible de la liste valeurs.

Pour une liste de longueur \(n\), quel est le nombre d'appels récursifs de la fonction melange effectués, sans compter l'appel initial ?

Réponse

Pour une liste de longueur (n), on appelle tout d'abord melange(valeurs, n - 1).

Le premier appel récursif est donc melange(valeurs, i - 2). Il est suivi d'appels récursifs correspondants aux différents indices de valeurs jusqu'au dernier appel melange(valeurs, 0).

Donc il y a (n-1) appels récursifs.

3.c. On considère le script ci-dessous :

Script Python

```
valeurs = [x for x in range(5)]
melange(valeurs, 4)
```

On suppose que les valeurs successivement renvoyées par la fonction randint sont 2, 1, 2 et 0.

Les deux premiers affichages produits par l'instruction print(valeurs) de la fonction melange sont :

• Premier affichage: [0, 1, 2, 3, 4],

• Deuxième affichage : [0, 1, 4, 3, 2].

Donner les affichages suivants produits par la fonction melange.

Réponse

On a les étapes suivantes :

Valeur de ind	Valeur de valeurs affichée	Valeur renvoyée par randint
ind = 4	[0, 1, 2, 3, 4]	2
ind = 3	[0, 1, 4, 3, 2]	1
ind = 2	[0, 3, 4, 1, 2]	2
ind = 1	[0, 3, 4, 1, 2]	0
ind = 0	[3, 0, 4, 1, 2]	

3.d. Proposer une version itérative du mélange de Fischer-Yates.

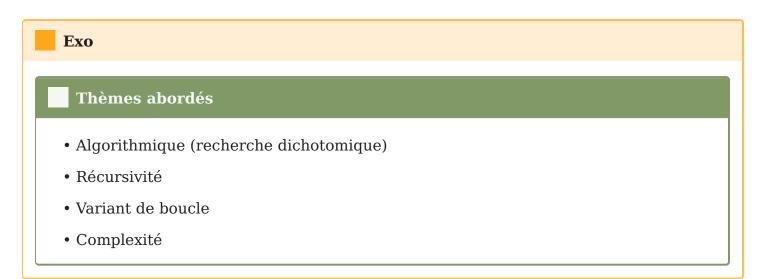
```
Réponse

Script Python

from random import randint

def melange(valeurs):
    indice_dernier = len(valeurs) - 1
    for i in range(indice_dernier, 0, -1):
        j = randint(0, i)
        echange(valeurs, i, j)
```

7. France Septembre 2021 J1

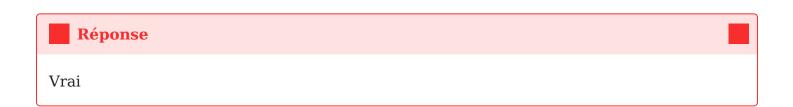


7.1. Partie A: La recherche dichotomique

Question 1

La recherche d'un élément dans un tableau avec une méthode dichotomique ne peut se faire que si le tableau est trié.

- a) Vrai
- b) Faux



Le coût d'un algorithme de recherche dichotomique est :

a) Constant : Complexité O(1)

b) Linéaire : Complexité O(n)

c) Logarithmique : Complexité O(log(n))

Réponse

c) Logarithmique

Question 3

Justifier pourquoi l'entier fin - deb est un **variant de boucle** qui montre la terminaison du programme de recherche dichotomique de l'annexe.

Réponse

À chaque étape, si on net rouve pas l'élément :

- Si liste[m]>elem alors \(fin_{k+1} = m - 1\) et \(debut_{k+1} = debut_k\) or \(m \le fin_k\) donc \(fin_{k+1} < fin_k\) et ainsi \(fin_{k+1} - debut_{k+1} < fin_k - debut_k\) - Si liste[m] < elem alors \(fin_{k+1} = fin_k\) et \(debut_{k+1} = m+1\) or \(m \ge debut_k\) donc \(debut_{k+1} < fin_k\) et ainsi \(fin_{k+1} - debut_{k+1} < fin_ - debut_k\) On a donc toujours \(fin_{k+1} - debut_{k+1} < fin_k - debut_k\), c'est bien un variant de boucle. Il existe donc un \(p\) tel que \(fin_p < debut_p\) et l'algorithme se termine.

7.2. Partie B : La recherche dichotomique itérative

Le programme de recherche dichotomique de l'annexe est utilisé pour effectuer des recherches dans une liste.

Dans l'ensemble de cette partie, on considère la liste :

& Script Python

Lnoms = ["alice", "bob", "etienne", "hector", "lea", "nathan", "paul"]

Expliquer pourquoi en ligne 2, on a «fin = len(liste)-1» plutôt que «fin = 6».

Réponse

Le programme s'adapte à la longueur de la liste

Question 2

En Python, l'opérateur // donne le quotient de la division euclidienne de deux nombres entiers.

Proposer un algorithme pour obtenir ce quotient.

Réponse

Si on cherche le quotient de a par b :

Script Python

$$c = a - b$$

$$q = 0$$
while $c > 0$:
$$q = q + 1$$

$$c = c - b$$

Question 3

Donner la trace complète de l'exécution rechercheDicho("lea", Lnoms) en complétant le tableau ci-dessous sur votre copie :

Variables			Condition	Valeur renvoyée
deb	fin	M	deb <= fin	

Réponse

		Condition	Valeur renvoyée
fin	M	deb <= fin	
6	3	True	
6	5	True	
4	4	True	True
	6	6 3 6 5	fin M deb <= fin 6 3 True 6 5 True

Question 4

Sur votre copie, modifier le code du corps de la fonction rechercheDicho() pour qu'elle renvoie aussi la position (indice) de l'élément cherché ou -1 si l'élément n'est pas trouvé. On pourra indiquer sur la copie le numéro des lignes modifiées, à supprimer ou à insérer s'il y a lieu.

Réponse

À la ligne 6 : return True, m À la ligne 12 : return False,-1

7.3. Partie C: La recherche dichotomique récursive

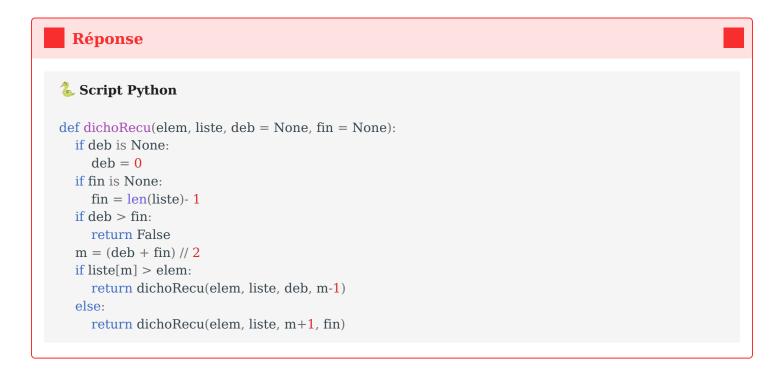
Question 1

Donner la définition d'une fonction récursive en programmation.

Réponse

Une fonction est récurssive si elle s'appelle elle-même.

Écrire en langage naturel ou en python, l'algorithme de recherche dichotomique d'un élément dans une liste, triée de façon croissante, en utilisant une méthode récursive. Il renverra True si l'objet a été trouvé, False sinon.



ANNEXE

On considère la fonction de recherche dichotomique suivante :

```
def rechercheDicho (elem, liste):
 1
 2
        """ Cette fonction indique si un élément se trouve dans un tableau.
 3
        Elle utilise la méthode de recherche dichotomique.
 4
        Elle prend en arguments :
 5
        - elem : élément à rechercher de type string
 6
        - liste : liste d'éléments de type string triée par ordre croissant
 7
        Elle renvoie un booléen correspondant à la présence ou non de l'élément
 8
        deb = 0
 9
        fin = len(liste)-1
10
11
        m = (deb+fin)//2
12
        while deb \le fin :
13
          if liste[m] == elem :
14
             return True
15
          elif liste[m] > elem :
16
             fin = m-1
17
           else:
18
             deb = m+1
19
          m = (deb+fin)//2
20
        return False
```