Algorithmes et Pensée Computationnelle

Structure de données, Itération et récursivité

Le but de cette séance est de mettre en pratique les connaissances apprises en cours. Elle portera essentiellement sur les structures de données qui seront utilisées lors des prochaines séances de cours/Travaux Pratiques. Les structures de données abordées lors de cette séance sont les tuples, les listes et les maps. Lors de cette séance, nous aborderons aussi les notions d'itération et de récursivité.

Au terme de cette séance, l'étudiant sera capable de distinguer une structure de données immuable et non immuable, écrire un programme de façon simplifiée en utilisant les notions d'itération et de récursivité.

1 Les structures de données

Dans la première partie de cette section, les exercices devront être traités en Python. Dans la seconde partie, les exercices seront traités en Java.

1.1 Les tuples

Pour rappel, les tuples sont des listes d'éléments immuables, ce qui signifie que ces listes ne peuvent pas être modifiées. L'une des particularités des tuples est la possibilité d'y inclure des données de types différents. Les tuples sont utiles pour stocker des données que l'on va réutiliser plus tard.

En Python, pour créer un tuple, il suffit de définir une variable et de lui assigner des valeurs entre parenthèses et séparer les valeurs entre elles par des virgules :

>_ Exemple 1 mon_tuple = (1,"une chaîne de caractères",3,4)

Question 1: (Manipulation d'un tuple

Créez un tuple nommé mon_tuple contenant les chiffres 1,2,3,4 et 5. Affichez le 4ème élément de ce tuple.



Pour accéder à un élément d'un tuple, ou d'une liste, vous pouvez utiliser l'indexation. Comme pour accéder aux caractères des chaînes de caractères, utilisez [].

Question 2: (5 minutes) Manipulation d'un tuple - 2

Créez un tuple nommé mon_tuple contenant les chiffres 1,2,3,4 et 5. Obtenez le nombre d'éléments contenus dans votre tuple et stockez le résultat dans une nouvelle variable nommée taille_tuple. Pour finir, affichez le contenu de taille_tuple.



Pour calculer la taille d'un tuple, ou d'une liste, vous pouvez utiliser la fonction len().

- 1 **mon_tuple** = (1,2,3,4,5)
- 2 taille_tuple = len(mon_tuple)
- 3 print(taille_tuple)

Question 3: (O *5 minutes*) **Manipulation d'un tuple - 3**

Vous pouvez contrôler si un élément est contenu dans un tuple, ou une liste, en utilisant l'opérateur in. Si l'élément est dans la liste, la valeur booléenne True sera retournée. S'il n'y est pas, la valeur booléenne False sera retournée.

Qu'affichera le code suivant?

```
1 mon_tuple = (1,2,3,4,5,6)
2
3 if 6 in mon_tuple :
4 print("6 est contenu dans le tuple")
5 else :
6 print("6 n'est pas contenu dans le tuple")
```

Choisissez parmi les éléments suivants :

- 6 est contenu dans le tuple
- True
- 6 n'est pas contenu dans le Tuple
- False

Conseil

Contrôlez si l'élément est contenu dans le tuple et regardez quel branche sera prise.

>_ Solution

"6 est contenu dans le tuple"

6 est un élément de notre tuple, la condition sera donc remplie.

1.2 Les listes

La différence entre une liste et un tuple est que l'on peut modifier les éléments d'une liste. Par exemple, il est possible de remplacer un élément d'une liste par un autre tandis qu'il est impossible de le faire avec un tuple.

Pour créer une liste, il suffit de définir une variable avec des valeurs entre crochets :

>_ Exemple 1 ma_liste = [1,2,3,4,5]

Question 4: (*5 minutes*) **Manipulation d'une liste - 1**

Créez une liste nommée ma_liste contenant les nombres 1,2,3,4 et 5. Stockez le deuxième élément de la liste dans une variable nommée element_2, puis stockez la taille de la liste dans une variable nommée taille_liste. Affichez ces deux variables.

• (

Conseil

Comme pour les tuples, vous pouvez utiliser [] pour accéder à un élément, et la fonction len() pour obtenir le nombre d'éléments contenus dans la liste.

>_ Solution

- $ma_liste = [1,2,3,4,5]$
- 2 element_2 = ma_liste[1]
- 3 taille_ma_liste = len(ma_liste)
- 4 print(element_2)
- 5 **print**(taille_ma_liste)

Question 5: (*5 minutes*) **Manipulation d'une liste - 2**

Qu'affichera le code suivant?

- 1 $ma_liste = [1,2,2,4,5]$
- 2 $ma_liste[2] = 3$
- 3 ma_liste.append(6)
- 4 ma_liste.insert(0,0)
- 5 print(ma_liste)

Choisissez parmi les éléments suivants :

- **—** 1, 2, 2, 4, 5, 6
- -0, 1, 2, 3, 4, 5
- -0, 1, 2, 3, 4, 5, 6
- -0, 1, 2, 2, 4, 5, 6



Conseil

En Python, vous pouvez accéder à chaque élément de la liste en utilisant [] et modifier sa valeur. Pour ajouter un élément à la fin de la liste, utilisez la fonction append() et pour choisir l'endroit ou vous désirez insérer votre nouvel élément, utilisez la fonction insert().

>_ Solution

[0, 1, 2, 3, 4, 5, 6]

Le script change le deuxième 2 pour un 3, ajoute un 6 à la fin de la liste, et un 0 au début de cette dernière.

Question 6: (5 minutes) Manipulation d'une liste - 3

Créez une liste nommée ma_liste contenant les nombres 1,2,3,4 et 5. Vérifiez si le chiffre 6 est dans votre liste. S'il y est, affichez "OK", s'il n'y est pas, rajoutez le à votre liste.

Utilisez le code ci-dessous pour contrôler que tout a bien été ajouté :

- for c in ma_liste :
 print(c)
 - •

Conseil

Comme pour les tuples, vous pouvez utiliser l'opérateur in pour contrôler si un élément est présent dans votre liste.

>_ Solution 1 ma_liste = [1,2,3,4,5] 2 if 6 in ma_liste : 4 print("OK") 5 else : 6 ma_liste.append(6)

1.3 Les dictionnaires

Les dictionnaires sont des listes associatives, c'est-à-dire des listes qui lient une valeur à une autre. Dans un dictionnaire en papier, les mots sont liés à leur définition.

Dans un dictionnaire Python, les éléments sont liés par une relation dite clé, valeur. La clé étant le moyen de "retrouver" notre valeur dans notre dictionnaire. Par exemple, dans un dictionnaire en papier, nous pouvons retrouver une définition en cherchant le mot qui lui correspond, alternativement, nous pouvons retrouver le contenu d'une page d'un livre en utilisant le numéro de celle-ci, dans ce cas le numéro est la clé et le texte de la page, la valeur.

Pour créer un dictionnaire, il suffit de lister les couples clé, valeurs entre 2 accolades :

Question 7: (5 minutes) Manipulation d'un dictionnaire - 1

Créez un dictionnaire nommé fr_eng contenant les éléments suivants :

```
1 "chat": "cat", "chien": "dog", "oiseau": "bir", "poule": "chicken", "papillon": "butterfly", "souris": "mouse", "ours": "bear", "mouton": "sheep", "cochon": "pig"
```

Ce dictionnaire contient des mots en français (clés) associés à leur traduction en anglais (valeurs).

Accédez à la traduction du mot "souris", et stockez le résultat dans une variable nommée souris_traduite, puis calculez le nombre d'éléments contenus dans le dictionnaire et stockez le résultat dans une variable nommée taille_fr_eng. Affichez le contenu des deux variables que vous venez de créer.

Conseil

Comme pour les listes, vous pouvez accéder aux éléments de dictionnaires en utilisant des crochets []. Seulement cette fois-ci, au lieu d'y mettre l'index, mettez-y la clé associée à l'élément auquel vous voulez accéder. Comme pour les listes, la fonction len() vous aidera à obtenir le nombre d'éléments dans le dictionnaire.

```
>_ Solution
     fr_eng = {
                "chat": "cat",
"chien": "dog",
2
4
                "oiseau": "bir",
5
                "poule": "chicken",
 6
                "papillon": "butterfly",
7
                "souris": "mouse",
8
                "ours": "bear",
9
                "mouton": "sheep",
                "cochon": "pig"
10
11
12
     souris_traduite = fr_eng["souris"]
13
     taille\_fr\_eng = \underline{len}(fr\_eng)
     \frac{print}{(souris\_traduite)}
15
     print(taille_fr_eng)
```

Question 8: (5 minutes) Manipulation d'un dictionnaire - 2

Gardez le même dictionnaire qu'auparavant. La traduction du mot "oiseau" est mal orthographiée, modifiez la valeur associée à "oiseau" pour qu'elle devienne "bird". Ajoutez un nouvel élément au dictionnaire en associant le mot "horse" au mot "cheval".

Utilisez ce code pour avoir un aperçu des changements :

```
1 for x in fr_eng:
2 print(x + ": " + fr_eng[x])
```

Conseil

Comme pour les listes, vous pouvez accéder aux éléments du dictionnaire en utilisant les crochets []. Seulement cette fois-ci, au lieu d'y mettre l'index, mettez-y la clé associée à l'élément auquel vous voulez accéder.

```
>_ Solution
    fr_eng = {
    "chat": "cat",
    ""dog"
3
               "chien": "dog",
4
               "oiseau": "bir",
5
               "poule": "chicken",
6
               "papillon": "butterfly",
               "souris": "mouse",
8
               "ours": "bear",
9
               "mouton": "sheep",
10
              "cochon": "pig"
11
     fr_eng["oiseau"] = "bird"
12
     fr_eng["cheval"] = "horse"
13
```

Question 9: (**1** *5 minutes*) **Manipulation d'un dictionnaire - 3** Qu'affiche le programme suivant?

- 8 print(elon_musk["talents"][2])
- 9 print("aéronautique" in elon_musk["talents"])

Choisissez parmi les possibilités suivantes :

- programmation

True

— Musk

False

— 48

False

- aéronautique

True

Conseil

L'élément associé à la valeur "talents" du dictionnaire elon_musk est une liste.

>_ Solution

aéronautique

True

En effet, via elon_musk["talents"], on accède à la liste ["programmation", "entreprenariat", "aéronautique"]. Il est donc possible de manipuler cette liste comme une liste classique!

1.4 Les Structures de données (Java)

Dans cette partie, les exercices devront être effectués en Java.

Les syntaxes des principales structures de données abordées dans cette partie sont les suivantes :

Déclaration d'un tuple en Java :

```
Object[] mon_tuple = \{1,2,3,4\};
```

Déclaration d'une liste en Java :

List $ma_liste = List.of(1,2,3,4);$

1

Déclaration d'un dictionnaire en Java :

HashMap mon_dictionnaire = new HashMap(Map.of("un", 1, "deux", 2));

>_ Informations utiles

- En Java, les listes sont immuables, comme les tuples en Python. Au cas où vous devriez modifier une liste, on vous recommande d'utiliser une liste liée (LinkedList). Une liste liée se déclare comme suit :
- 1 List liste = List.of(1,2,3,4);
- 2 LinkedList ma_liste = new LinkedList(liste);
- En Java, les listes ne peuvent contenir que des éléments homogènes c'est-à-dire de même type.

Question 10: (10 minutes) Manipulation des listes en Java

Créez une liste nommée ma liste contenant les nombres 1,2,3,4 et 5. Affichez le deuxième élément de la liste ainsi que la taille de la liste.

Créez une liste ma_liste_m liée à la liste ma_liste. Ajoutez le chiffre 6 à la fin de la liste, et le chiffre 0 au début de cette dernière.

Ajoutez ceci au début de votre code :

```
1     import java.util.List;
2     import java.util.LinkedList;
Ajoutez ceci à la fin de votre code :
1     for(int i=0;i<ma_liste_m.size();i++){
2          System.out.println(ma_liste_m.get(i));
3     }</pre>
```

Conseil

Pour obtenir un élément d'une liste, il faut utiliser la fonction **get(index)**. Pour obtenir la taille de la liste, utilisez la fonction **size()**.

Pour ajouter un élément au début d'une LinkedList, vous devez utiliser addFirst(value) et pour l'ajouter à la fin, vous devez utiliser addLast(value).

N'oubliez pas d'écrire votre programme à l'intérieur d'une fonction principale main.

```
public static void main(String[] args) {
    // Ecrire votre code ici
}
```

>_ Solution

```
import java.util.List;
 2
     import java.util.LinkedList;
 3
     public class Main {
 5
       public static void main(String[] args) {
 7
          List ma_liste = List.of(1,2,3,4);
 8
          System.out.println(ma_liste.get(1));
 9
          System.out.println(ma_liste.size());
10
          LinkedList ma_liste_m = new LinkedList(ma_liste);
11
          ma_liste_m.addFirst(0);
12
          ma_liste_m.addLast(6);
13
          for(int i=0:i < ma_liste_m.size():i++){</pre>
14
          System.out.println(ma_liste_m.get(i));
15
16
       }
17
     }
```

Question 11: (**1** *15 minutes*) **Manipulation d'un dictionnaire en Java**

Créez un dictionnaire mon dictionnaire contenant les éléments suivants :

```
("étudiants", 14000, "enseignants", 2300, "collaborateurs", 0)
```

Affichez le nombre d'étudiants. Obtenez la taille du dictionnaire et stockez le résultat dans une variable taille_dictionnaire. Affichez le contenu de taille_dictionnaire. Modifiez la valeur de collaborateurs, remplacez 0 par

950. Rajoutez un nouvel élément pays dans votre dictionnaire avec pour valeur 86.

Ajoutez ceci au début de votre code :

Conseil

Comme pour les listes, accédez aux valeurs via la fonction get(key) et à la taille via la fonction size().

Pour insérer / corriger un élément, utilisez la fonction put(key,value).

```
>_ Solution
    import java.util.HashMap;
2
    import java.util.Map;
3
4
    public class Main {
5
6
       public static void main(String[] args) {
         HashMap mon_dictionnaire = new
          HashMap(Map.of("étudiants",14000,"enseignants",2300,"collaborateurs",0));
 8
         System.out.println(mon_dictionnaire.get("étudiants"));
9
         int taille_dictionnaire = mon_dictionnaire.size();
10
         System.out.println(taille\_dictionnaire);
         mon_dictionnaire.put("collaborateurs",950);
11
         mon_dictionnaire.put("pays",86);
12
13
         for (Object keys : mon_dictionnaire.keySet()){
14
         System.out.println(keys + ":"+ mon_dictionnaire.get(keys));
15
         }
16
       }
    }
17
```

Question 12: (15 minutes) Lecture de code

Qu'affichera le code suivant?

```
import java.util.List;
     import java.util.HashMap;
 3
     import java.util.Map;
     public class Main {
 5
 6
       public static void main(String[] args) {
 7
          List ma_liste = List.of(3,2,6,4,1,5);
 8
         HashMap mon_dictionnaire = new HashMap(Map.of(1,"un",2,"deux",3,"trois",4,"quatre",5,"cinq",6,"six"));
 9
         System.out.println(ma_liste.get(3));
10
         System.out.println(ma_liste.get(2));
          System.out.println(mon\_dictionnaire.get(ma\_liste.get(4)));\\
11
12
          System.out.println(mon_dictionnaire.get(ma_liste.get(0)));
13
14
     Choisissez parmi les possibilités suivantes :
          — 3
             2
             quatre
             zero
```

— 4 6 un

— 3

2

trois

un trois

_ 4

6

quatre

zero

Conseil

Commencez par voir quel est l'élément de la liste qui sera retourné. Ensuite, regardez dans le dictionnaire quelle est la valeur associée à cet élément qui vient de nous être retourné.

>_ Solution

4

6

un

trois

2 L'itération

Quelques rappels de concepts théoriques :

- L'itération désigne l'action de répéter un processus (généralement à l'aide d'une boucle) jusqu'à ce qu'une condition particulière soit remplie.
- Il y a deux types de boucles qui sont majoritairement utilisées :

Boucle for: Une boucle **for** permet d'itérer sur un ensemble. Cet ensemble peut être une liste, un dictionnaire, une collection, ... La syntaxe d'une boucle en Python est la suivante **for nom de variable** in suivi du nom de l'élément sur lequel vous voulez itérer. La variable prendra la valeur de chaque élément dans la liste, un par un.

La syntaxe en java est la suivante : for (type nom_de_variable; condition de fin de boucle; incrémentation à chaque itération) suivi d'une accolade.

Boucle while: Les boucles while sont des boucles qui s'exécutent de façon continue jusqu'à ce qu'une condition soit remplie. La syntaxe est la suivante : En Python, on écrit d'abord while, suivi de la condition à atteindre.

En Java, il n'y a pas de différence majeure à part le fait qu'il faille mettre entre parenthèses la condition et les deux points sont remplacés par des accolades.

- La fonction range(n) permet de créer une liste de nombres de 0 à la valeur passée en argument moins 1 (n-1). Lorsqu'elle est combinée à une boucle for, on peut itérer sur une liste de nombres de 0 à n-1.
- Si vous avez fait une erreur dans votre code, il se peut que la boucle while ne remplisse jamais la condition lui permettant de sortir. On parle dans ce cas d'une boucle infinie. Ceci peut entraîner un

crash de votre programme, voire même de votre ordinateur. Il faut toujours s'assurer qu'il y' ait une condition valide dans une boucle while.

Question 13: (5 minutes) **Boucle for en Python** Qu'affichera le code suivant?

```
ma_liste = [1,2,3,4,5]
2
            for c in ma_liste:
3
              print(c)
4
```

Choisissez parmi les possibilités suivantes :

- 12345
- 54321
- 3 2 1 5 4
- <u>43512</u>

Conseil

Faites les étapes de la boucle une après l'autre et voyez ce que vous obtenez (avec chaque valeur).

>_ Solution

12345

Question 14: (5 *minutes*) **Boucle for en Java** Qu'affichera le code suivant?

```
List ma_liste = List.of((2,3,4,5,6));
for(int i=0;i < ma\_liste.size();i++){}
       System.out.print(ma_liste.get(i));
}
```

Choisissez parmi les possibilités suivantes :

- **—** 54321
- 12345
- 43512
- 23456

2

3

4

Conseil

Faites les étapes de la boucle une après l'autre et voyez ce que vous obtenez (avec chaque valeur).

>_ Solution

23456

Question 15: (*5 minutes*) **Boucle while en Python** Qu'affichera le code suivant?

```
i = 0
              while i<5:
3
                  print(i)
4
                  i += 1
```

Choisissez parmi les possibilités suivantes :

- 01234
- 12345
- 43215
- 23401

•

Conseil

Faites les étapes de la boucle une après l'autre et voyez ce que vous obtenez (avec chaque valeur).

01234

Question 16: (5 minutes) **Boucle while en Java** Qu'affichera le code suivant?

Choisissez parmi les possibilités suivantes :

- 01234
- 1 2 3 4 5
- 43210
- _ 5 4 3 2 1

Conseil

Faites les étapes de la boucle une après l'autre et voyez ce que vous obtenez (avec chaque valeur).

>_ Solution

12345

Question 17: (5 minutes) Boucle for et fonction range() - 1 Qu'affichera le code suivant?

Choisissez parmi les possibilités suivantes :

- 0 J'aime 1 Python
- J'aime 1 Python 2
- J'aime 0 Python 1
- 1 J'aime 2 Python

Conseil

Faites les étapes de la boucle une après l'autre et voyez ce que vous obtenez (avec chaque valeur).

>_ Solution

0 J'aime 1 Python

Dans les exercices suivants, les programmes doivent être écrits en Python.

Question 18: (5 minutes) **Boucle for et fonction range**() - 2

En utilisant la fonction range() et une boucle for, calculez la somme des entiers de 0 à 20 et affichez le résultat.

Conseil

- Lorsque range(n) est combinée à une boucle for, on peut itérer sur une liste de nombres de 0 à n-1.
- Déclarer une variable en dehors de la boucle. Au terme de l'itération, cette variable contiendra la somme des entiers de 0 à 20.

>_ Solution

```
1  somme = 0
2  for i in range(21):
3     somme += i # A chaque iteration on rajoute i à somme.
4  print(somme)
```

Question 19: (10 minutes) **Boucle while et input**()

A l'aide d'une boucle while, demandez à l'utilisateur de rentrer une valeur. Tant que cette valeur ne correspond pas à 10, le programme redemande à l'utilisateur une nouvelle valeur.

Conseil

- Définir à l'extérieur de la boucle une variable de type booléen utilisée pour le test dans while.
- Utilisez la fonction input() pour récupérer la saisie de l'utilisateur.
- La fonction input() retourne un str (chaîne de caractères), pensez à changer son type.

>_ Solution

```
bool_test = True
                                              # Tant que bool_test est True, la boucle continue
    while bool test:
2
         test_value = int(input("'Veuillez entrer un entier")) # Pour que le test effectué à la ligne suivante ne soit pas
3
          toujours juste
4
                                            # (comparaison entre un string et un entier), il faut changer le type de
          l'input.
         bool\_test = not(test\_value == 10)
                                                        # On veut sortir de la boucle si la test_value est 10. Pour sortir il
          faut que bool_test soit
6
         if bool_test == True:
                                                   # False. D'où l'utilisation de not qui transforme true en false et vice
          verse.
              print("Ce n'est pas le bon entier.")
7
    print("Bravo !")
```

Question 20: (**1** *10 minutes*) **Boucle while et des étudiants**

Considérons une liste d'étudiant-e-s contenant le nom de l'étudiant-e suivi d'une valeur booléenne indiquant si l'étudiant-e est présent-e en classe ou non.

```
1 = ["Schmitt", True, "Irma", False, "Khalif", True, "Yasser", False, "Wang", True]
```

À partir de cette liste et du type de boucle de votre choix, créez un dictionnaire ayant pour clés les noms et pour valeur un booléen indiquant la présence au cours.

Conseil

- i % 2 : renvoie le reste de i par la division euclidienne de 2. (i % 2 == 0) Indique si i est pair.
- Si vous utilisez la boucle for : Utiliser les indices pour accéder aux éléments de la liste et éviter d'itérer manuellement sur les éléments de la liste.
- Si vous utilisez la boucle while : La condition de sortie devrait être en rapport avec la longueur de la liste.

Solution avec la boucle for:

```
l = ["Schmitt", True, "Irma", False, "Khalif", True, "Yasser", False, "Wang", True]
2
    dict_final = {}
3
                                   # i va de 0 à 9
    for i in range(len(l)):
4
         if i \% 2 == 0:
                                   # Test si i est pair. Si oui, on crée une nouvelle entrée dans le dictionnaire
              dict_final[l[i]] = l[i + 1]
    Solution avec la boucle while:
    I = ["Schmitt", True, "Irma", False, "Khalif", True, "Yasser", False, "Wang", True]
    dict_final = \{\}
2
3
    i = 0
                             # Variable qui permettra d'itérer et de gérer la boucle while.
4
    while i < len(l):
                                  # La boucle s'arrêtera donc au bout de 10 itérations. La valeur maximale de i sera 9.
5
         if i \% 2 == 0:
6
           dict_final[l[i]] = l[i + 1]
                                # On incrémente i
         i +=1
```

Pour ce problème, la boucle **for** est plus simple à mettre en place et évite le danger d'une boucle infinie. La boucle **while** est utilisée pour des problèmes plus précis qui ne peuvent pas être résolus par une boucle **for**. La boucle **for** offre aussi l'avantage d'être facile à lire. Pour des problèmes de nécessitant pas un très grand nombre d'itérations, utilisez de préférence une boucle **for**. Par contre, si vous l'utilisez pour trouver un élément dans un grand ensemble (par exemple, une liste contenant un grand nombre d'enregistrements), utilisez une boucle **while** car elle offre l'avantage de sortir de la boucle une fois que la condition initiale est fausse sans avoir à parcourir l'ensemble des éléments (par exemple, votre boucle s'arrêtera lorsque vous aurez trouvé l'élément que vous recherchez dans la liste volumineuse).

Ouestion 21: (5 minutes) **Différence entre Itération et Récursivité**

Quel est la différence dans le traitement de problèmes entre l'itération et la récursivité?

Conseil

- L'itération traite les problèmes de manière "contrôlée". En effet, lors d'un appel itératif, la boucle est répétée tant que la condition est vérifiée.
- La récursivité résout les problèmes en résolvant des sous-problèmes de ceux-ci.

>_ Solution

- L'itération va résoudre de manière plus "intuitive" les problèmes, c'est-à-dire répéter certaines actions qui permettent de résoudre un problème souvent au prix de l'efficience.
- La récursivité permet de diviser un problème en sous-problèmes et ainsi de suite jusqu'au moment où le sous-problème est soluble et permet de résoudre tous les problèmes jusqu'à celui original. La récursivité cache une faille, un appel à elle même infini. De manière semblable aux boucles while, la récursivité peut causer le crash du programme voir de l'ordinateur. Lorsque vous cherchez à faire une récursivité il faut donc systématiquement poser avant tout la condition finale pour l'arrêt.

3 La récursivité

La récursivité est le fait d'appeler une fonction de façon récursive c'est-à-dire une fonction qui s'appelle elle-même de façon directe ou indirecte.

Question 22: (5 minutes) Lecture de code - 1 Qu'affiche le programme suivant?

```
def fonction_p(x):
     print(x)
2
3
     if x == 0:
       pass
4
5
     else:
6
       fonction_p(x-1)
   fonction\_p(5)
   Choisissez parmi les possibilités suivantes :
       — 43210
       — 543210
       — 0 1 2 3 4
```

543210

— 1 2 3 4 5 0

Question 23: (5 minutes) Lecture de code - 2

Qu'affiche le programme suivant?

```
1     def fonction_p(x):
2         if x == 0:
3         pass
4     else:
5         fonction_p(x-1)
6         print(x)
7
8     fonction_p(5)
```

Choisissez parmi les possibilités suivantes :

- 43210
 54321
- -01234

Conseil

- La principale différence entre ces 2 programmes est l'endroit où est placé le print().
- Reproduire étape par étape le développement sur un papier pourrait aider.

>_ Solution

 $0\,1\,2\,3\,4\,5$

Question 24: (5 minutes) Lecture de code - 3 Optionnel

Qu'affiche le programme suivant?

```
def fct_a(mot):
 2
     if len(mot) == 1 :
 4
           if mot[0] == "a":
 5
               return 1
 6
 7
                return 0
 8
     else:
 9
          if mot[0] == "a":
10
                \textcolor{return}{return} \ 1 + fct\_a(mot[1:])
11
12
                return 0 + fct_a(mot[1:])
13
     print(fct_a("blablabla"))
```

Choisissez parmi les possibilités suivantes :

- __ 1
- _ 2
- **—** 3
- _ 4

A quoi sert la fonction fct_a(mot)?

Conseil

Aidez vous d'un papier et effectuez les étapes une à une afin de bien comprendre comment le code est articulé.

>_ Solution

3

Cette fonction compte le nombre de "a" contenus dans un mot que vous lui passez.

Question 25: (10 minutes) **Fonction factoriel**()

Ecrivez la fonction factoriel() suivant une approche récursive.

Pour rappel, cette fonction prend un entier et retourne le factoriel de ce dernier tel que :

factoriel(1) = 1, factoriel(2) = 1 * 2 = 2, factoriel(3) = 1 * 2 * 3 = 6, factoriel(4) = 1 * 2 * 3 * 4 = 24,...

Conseil

On peut écrire cette fonction comme suit :

$$f(n) = f(n) * f(n-1) si n > 0$$

$$f(n) = 1 \text{ si } n = 0$$

>_ Solution

def factoriel(x):

Python:

1

```
Java:

1    public static int factoriel(int x){
2         if (x == 0){
3            return 1;
4         }
5         else{
6            return x*factoriel(x-1);
7         }
8      }
9     public static void main(String[] args) {
10            System.out.println(factoriel(5));
11     }
```

Question 26: (15 minutes) Fibonacci - Java

Ecrivez deux fonctions permettant de calculer un nombre donné de la suite de Fibonacci. L'une doit utiliser la récursivité, et l'autre l'itération. Pour rappel, chaque élément de la suite de Fibonacci est la somme des deux derniers éléments. L'élément 0 vaut 0, l'élément 1 vaut 1 et l'élément 2 vaut 1.

Début de la suite : [0,1,1,2,3,5,8,13,...]



L'algorithme permettant de faire une suite de Fibonacci a été présenté dans les diapositives du cours. En cas de difficulté, n'hésitez pas à vous y référer.

>_ Solution Via Itération: $public \ static \ \underline{int} \ fibonacci \underline{_i}(\underline{int} \ n) \big\{$ if (n==0 || n==1){ 2 3 4 return n; 5 6 else{ int old_fib = 1; 7 int new_fib = 1; 8 int tmp; for (int i=2; i<n; i++){ 10 tmp = new_fib; new_fib = new_fib+old_fib; 11 12 $old_fib = tmp;$ 13 14 return new_fib; 15 16 } Via récursivité: $public \ static \ \underline{int} \ fibonacci_r(\underline{int} \ n) \big\{$ if (n==0 | | n==1)3 4 return n; 5 else{ 6 7 return fibonacci_r(n-1) + fibonacci_r(n-2); 8 }