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 dictionnaires. 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.
Le code présenté dans les énoncés se trouve sur Moodle, dans le dossier "Exercices > Code". Le temps mentionné (🕓) est à titre indicatif.

1 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 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: (🕓 *5 minutes*) **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.

Conseil

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 [].

Solution

```
1  mon_tuple = (1,2,3,4,5)
2  print(mon_tuple[3])
```

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**.

Conseil

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

Solution

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

Question 3: (*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 ?

```
mon_tuple = (1,2,3,4,5,6)

if 6 in mon_tuple :
    print("6 est contenu dans le tuple")
else :
    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 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

```
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

```
1  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 d'une 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 :

```
1          for c in ma_liste :
2            print(c)
```

Conseil

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

Solution

```
ma_liste = [1,2,3,4,5]

if 6 in ma_liste :
    print("OK")
else :
    ma_liste.append(6)
```

1.3 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 :

```
elon_musk = {
    "prénom": "Elon",
    "nom": "Musk",
    "age": 48,
    "talents": ["programmation", "entreprenariat", "aéronautique"]
}
```

Question 7: (*5 minutes*) **Manipulation d'un dictionnaire - 1**
Créez un dictionnaire nommé **fr_eng** contenant les éléments suivants :

```
"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”,
            ”oiseau”: ”bir”,
            ”poule”: ”chicken”,
            ”papillon”: ”butterfly”,
            ”souris”: ”mouse”,
            ”ours”: ”bear”,
            ”mouton”: ”sheep”,
            ”cochon”: ”pig”
        }
souris_traduite = fr_eng[”souris”]
taille_fr_eng = len(fr_eng)
print(souris_traduite)
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 :

```
    for x in fr_eng :
        print(x + ” : ” + fr_eng[x])
```

Conseil

Comme avec les listes, vous pouvez accéder aux éléments d'un dictionnaire 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.

Solution

```
fr_eng = {
            ”chat”: ”cat”,
            ”chien”: ”dog”,
            ”oiseau”: ”bir”,
            ”poule”: ”chicken”,
            ”papillon”: ”butterfly”,
            ”souris”: ”mouse”,
            ”ours”: ”bear”,
            ”mouton”: ”sheep”,
            ”cochon”: ”pig”
}
fr_eng[”oiseau”] = ”bird”
fr_eng[”cheval”] = ”horse”
```

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

```
elon_musk = {
        ”prénom”: ”Elon”,
        ”nom”: ”Musk”,
        ”age”: 48,
        ”talents”: [”programmation”, ”entreprenariat”, ”aéronautique”]
}

print(elon_musk[”talents”][2])
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 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);
```

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 :

```
List liste = List.of(1,2,3,4);
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 :

```
import java.util.List;
import java.util.LinkedList;
```

Ajoutez ceci à la fin de votre code :

```
for(int i=0;i<ma_liste_m.size();i++){
        System.out.println(ma_liste_m.get(i));
 }
```

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;
import java.util.LinkedList;

public class Main {

   public static void main(String[] args) {
      List ma_liste = List.of(1,2,3,4);
      System.out.println(ma_liste.get(1));
      System.out.println(ma_liste.size());
      LinkedList ma_liste_m = new LinkedList(ma_liste);
      ma_liste_m.addFirst(0);
      ma_liste_m.addLast(6);
      for(int i=0;i<ma_liste_m.size();i++){
      System.out.println(ma_liste_m.get(i));
      }
   }
}
```

Question 11: (*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 :

```
import java.util.HashMap;
import java.util.Map;
```

Ajoutez ceci à la fin de votre code :

```
for (String keys : mon_dictionnaire.keySet()){
    System.out.println(keys + " : "+ mon_dictionnaire.get(keys));
}
```

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;
import java.util.Map;

public class Main {

    public static void main(String[] args) {
        HashMap<String, Integer> mon_dictionnaire = new HashMap<String, Integer>(Map.of("étudiants",
          14000, "enseignants", 2300, "collaborateurs", 0));
        System.out.println(mon_dictionnaire.get("étudiants"));
        int taille_dictionnaire = mon_dictionnaire.size();
        System.out.println(taille_dictionnaire);
        mon_dictionnaire.put("collaborateurs", 950);
        mon_dictionnaire.put("pays", 86);
        for (String keys : mon_dictionnaire.keySet()) {
            System.out.println(keys + " : " + mon_dictionnaire.get(keys));
        }
    }
}
```

Question 12: (*15 minutes*) **Lecture de code**
Qu'affichera le code suivant ?

```
import java.util.List;
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        List<Integer> ma_liste = List.of(3,2,6,4,1,5);
        HashMap<Integer, String> mon_dictionnaire = new HashMap<Integer, String>(Map.of(1,"un",2,"deux",3,"trois",
          4,"quatre",5,"cinq",6,"six"));
        System.out.println(ma_liste.get(3));
        System.out.println(ma_liste.get(2));
        System.out.println(mon_dictionnaire.get(ma_liste.get(4)));
        System.out.println(mon_dictionnaire.get(ma_liste.get(0)));
    }
}
```

Choisissez parmi les possibilités suivantes :

— 3
2
quatre
zero

— 4
6
un
trois

— 3
2
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 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]
for c in ma_liste :
  print(c)

```

Choisissez parmi les possibilités suivantes :
— 1 2 3 4 5
— 5 4 3 2 1
— 3 2 1 5 4
— 4 3 5 1 2

Conseil

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

Solution

1 2 3 4 5

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 :
— 5 4 3 2 1
— 1 2 3 4 5
— 4 3 5 1 2
— 2 3 4 5 6

Conseil

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

Solution

2 3 4 5 6

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

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

```

Choisissez parmi les possibilités suivantes :

— 0 1 2 3 4
— 1 2 3 4 5
— 4 3 2 1 5
— 2 3 4 0 1

Conseil

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

Solution

0 1 2 3 4

Question 16: (*5 minutes*) **Boucle while en Java**

Qu'affichera le code suivant ?

```
1       int i = 0;
2       while(i<5){
3         i++;
4         System.out.print(i + " ");
5       }
6
```

Choisissez parmi les possibilités suivantes :

— 0 1 2 3 4
— 1 2 3 4 5
— 4 3 2 1 0
— 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

1 2 3 4 5

Question 17: (*5 minutes*) **Boucle for et fonction range() - 1**

Qu'affichera le code suivant ?

```
1       ma_liste = ["J'aime","Python"]
2       for x in range(len(ma_liste)):
3           print(x)
4           print(ma_liste[x])
5       #0
6       #J'aime
7       #1
8       #Python
9
```

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

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

Question 19: (*10 minutes*) **Création d'une liste à partir d'un tuple**
Transformer le tuple (1,4,5,8) en une liste à l'aide d'une boucle `for`.

Conseil

- Déclarer une liste vide à l'extérieur de la boucle.
- La boucle permet d'itérer sur le tuple.
- Rajouter un à un les éléments du tuple dans la liste.

Solution

```
liste_finale = []
tuple_initial = (1, 4, 5, 8)
for valeur in tuple_initial:        # A chaque iteration valeur devient une copie d'un element du tuple. La boucle se
    termine au dernier élément de tuple_initial.
    liste_finale.append(valeur)  # On ajoute valeur à la liste créée en dehors de la boucle.
```

Question 20: (*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
while bool_test:                                    # Tant que bool_test est True, la boucle continue
    test_value = int(input("Veuillez entrer un entier"))  # Pour que le test effectué à la ligne suivante ne soit pas toujours juste
                                                    #  ( 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
    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.")
print("Bravo !")
```

3 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 21: (*5 minutes*) **Lecture de code - 1**
Qu'affiche le programme suivant ?

```
def fonction_p(x) :
    print(x)
    if x == 0 :
        pass
    else :
        fonction_p(x-1)

fonction_p(5)
```

Choisissez parmi les possibilités suivantes :

- 4 3 2 1 0
- 5 4 3 2 1 0
- 0 1 2 3 4
- 1 2 3 4 5 0

Solution

5 4 3 2 1 0

Question 22: (*5 minutes*) **Lecture de code - 2**
Qu'affiche le programme suivant ?

```
def fonction_p(x) :
    if x == 0 :
        pass
    else :
        fonction_p(x-1)
        print(x)

fonction_p(5)
```

Choisissez parmi les possibilités suivantes :

- 4 3 2 1 0
- 5 4 3 2 1
- 0 1 2 3 4

— 1 2 3 4 5

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 23: (*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 si n = 0

Solution

Python :

```python
def factoriel(x) :
    if x == 0:
        return 1
    else :
        return x * factoriel(x-1)
```

Java :

```java
public static int factoriel(int x){
    if (x == 0){
       return 1;
    }
    else{
       return x*factoriel(x-1);
    }
  }
  public static void main(String[] args) {
     System.out.println(factoriel(5));
  }
```

Question 24: (*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,...]

Conseil

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 :

```
1  public static int fibonacci_i(int n){
2      if (n==0 || n==1){
3          return n;
4      }
5      else{
6          int old_fib = 1;
7          int new_fib = 1;
8          int tmp;
9          for (int i=2; i<n; i++){
10             tmp = new_fib;
11             new_fib = new_fib+old_fib;
12             old_fib = tmp;
13         }
14         return new_fib;
15     }
16 }
```

Via récursivité :

```
1  public static int fibonacci_r(int n){
2      if (n==0 || n==1){
3          return n;
4      }
5      else{
6          return fibonacci_r(n-1) + fibonacci_r(n-2);
7      }
8  }
```