

# Algorithmes et Pensée Computationnelle

## Fonctions, mémoire et exceptions

Le but de cette séance est d'approfondir vos connaissances en programmation. Au terme de cette séance, l'étudiant sera capable de :

- définir une fonction et l'utiliser dans un programme,
- utiliser des bibliothèques contenant des fonctions prédéfinies,
- connaître quelle est la portée d'une variable,
- comprendre comment fonctionne une pile d'exécution (call stack)

### Attention

Les exceptions n'ayant pu être présentées en cours, les exercices portant sur cette notion seront inclus dans les feuilles d'exercices de la semaine prochaine.

## 1 Variables et Fonctions

### Question 1: (🕒 5 minutes) Les fonctions (fonctions basiques) (Java ou Python)

Définissez une fonction nommée `ping()` qui, lorsqu'elle est appelée, affiche "pong".

### Conseil

Référez vous aux diapositives du cours pour la création et l'appel des fonctions.

### >\_ Solution

#### Python :

```
1 def ping() :
2     print("pong")
3
4 ping()
```

#### Java :

```
1 public class Main {
2     static void Ping(){
3         System.out.println("Pong");
4     }
5     public static void main(String[] args) {
6         Ping();
7     }
8 }
```

### Question 2: (🕒 5 minutes) Les Fonctions (Fonction multiplication) (Java ou Python)

Définissez une fonction nommée `multiplieur()` qui prend deux arguments *multiple1* et *multiple2*, les multiplie et retourne le résultat. Stockez le résultat de `multiplieur(2, 3)` dans une variable *resultat* et affichez la.

### Conseil

- Référez vous aux diapositives du cours pour la création et l'appel des fonctions.
- Pour retourner une valeur au lieu de l'afficher, utilisez le mot-clé `return` (pour Python et Java).

## >\_ Solution

### Python :

```
1 def multiplicateur(multiple1, multiple2):
2     return multiple1 * multiple2
3
4
5 if __name__ == "__main__":
6     resultat = multiplicateur(2, 3)
```

### Java :

```
1 public class Main {
2     public static int multiplicateur(int multiple1, int multiple2){
3         return multiple1*multiple2;
4     }
5
6     public static void main(String[] args) {
7         int resultat = multiplicateur(1, 2);
8     }
9 }
```

### Question 3: (🕒 5 minutes) Les Fonctions (Fonctions Aire et Périmètre) (Java ou Python)

Définissez deux fonctions nommées `aire()` et `perimetre()` qui prennent un argument (`rayon`) et renvoient respectivement l'aire et le périmètre d'un cercle. Stockez les résultats dans des variables `aire` et `perimetre` et affichez le contenu de ces variables.

### 💡 Conseil

- Référez vous aux diapositives du cours pour la création et l'appel des fonctions.
- Pour retourner une valeur au lieu de l'imprimer, utilisez le mot clé `return` (pour Python et Java).
- Pour rappel, le périmètre d'un cercle s'obtient en utilisant la formule  $P = 2 * \pi * r$  et l'aire s'obtient en utilisant la formule  $A = r^2 * \pi$ .

## >\_ Solution

### Python :

```
1 import math
2
3 def aire(rayon):
4     return (rayon**2)*math.pi
5
6 def perimetre(rayon):
7     return 2*math.pi*rayon
8
9 if __name__ == '__main__':
10     rayon = 10
11     aire = aire(rayon)
12     perimetre = perimetre(rayon)
13     print("L'aire d'un cercle de rayon {} est égale à
14           {}".format(rayon, aire))
15     print("Le périmètre d'un cercle de rayon {} est égal à
16           {}".format(rayon, perimetre))
```

### Java :

```
1 public class Main {
2
3     static double aire(int rayon){
4         return Math.pow(rayon, 2)*Math.PI;
5     }
6
7     static double perimetre(int rayon){
8         return 2*Math.PI*rayon;
9     }
10
11     public static void main(String[] args) {
12         int rayon = 5;
13         double aire = aire(rayon);
14         double perimetre = perimetre(rayon);
15         System.out.println("L'aire d'un cercle de rayon "+rayon+" est é
16                             gale à "+aire);
17         System.out.println("Le périmètre d'un cercle de rayon
18                             "+rayon+" est égal à "+perimetre);
19     }
20 }
```

### Question 4: (🕒 5 minutes) Portée des variables (Python)

Qu'affiche le programme suivant ?

```
1 x = 2
2
3 def fonction():
4     x = 3
5
6     def fonction2():
7         global x
8         print("x=" + str(x))
9
10    fonction2()
11    print("x=" + str(x))
12
13 print(fonction())
```

#### Conseil

- Le mot-clé `global` permet d'accéder aux variables globales (définies à l'extérieur de la fonction).
- Soyez attentif au type des éléments retournés par chacune des fonctions.

#### >\_ Solution

x=2  
x=3  
None

#### Question 5: (🕒 10 minutes) Portée des variables (Java)

Qu'affiche le programme suivant ?

```
1 public class Main {  
2     static int a = 0;  
3     public static void f() {  
4         a += 2;  
5         System.out.println("a = " + a);  
6     }  
7  
8     public static void g() {  
9         int b = 3;  
10        System.out.println("a = " + a);  
11        System.out.println("b = " + b);  
12    }  
13  
14    public static void main(String[] args) {  
15        f();  
16        g();  
17        //System.out.println("b = " + b);  
18    }  
19 }
```

Que se passerait-il si on décommente la ligne 17 (enlever les //)?

#### Conseil

Tenir compte de l'endroit où chaque variable est définie et quelles opérations sont réalisées sur celle-ci.

#### >\_ Solution

On obtiendra une erreur de compilation du programme car la variable `b` est créée à l'intérieur de la méthode `g()` et n'existe donc qu'au sein de celle-ci. La variable `b` n'étant pas définie en dehors de la méthode `g()`, celle-ci ne peut pas faire l'objet d'un appel en dehors de la fonction `g()`.