Algorithmique et Programmation 1

Lundi 2 décembre 2024

L1 Mathématiques - L1 Informatique
Semestre 1

Retour sur les fonctions

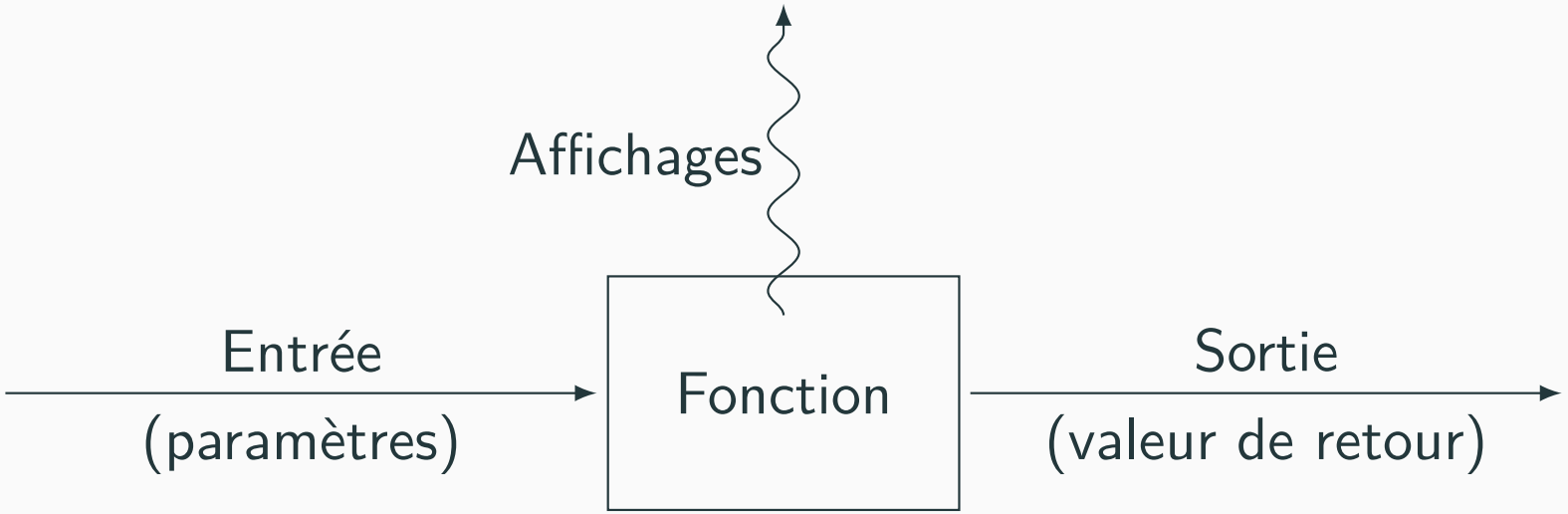
Fonction

En informatique, une fonction est :

- Un morceau de programme
- Portant en général *un nom*
- Prenant un ou plusieurs *paramètres* (ou zéro)
- Renvoyant un résultat (la plupart du temps)

Fonctions

Fonctions


Affichages
Entrée
(paramètres)
Fonction
Sortie
(valeur de retour)

Définition et appel de fonction

Définir une fonction

La syntaxe pour définir une fonction est :

```
def nom_fonction(parametre_1, ..., parametre_n):
    # corps de la fonction
    # utilisant parametre_1, ..., parametre_n
    ...
    # peut renvoyer un résultat :
    return resultat
```

Définition et appel de fonction

Définir une fonction

La syntaxe pour définir une fonction est :

```
def nom_fonction(parametre_1, ..., parametre_n):
    # corps de la fonction
    # utilisant parametre_1, ..., parametre_n
    ...
    # peut renvoyer un résultat :
    return resultat
```

Appeler une fonction

On peut ensuite *appeler* la fonction `nom_fonction` das le code :

```
resultat = nom_fonction(expression_1, ..., expression_n)
```

Exemples

Fonction sans valeur de retour

Dessiner un carré fait du caractère `caractere` :

Fonction sans valeur de retour

Dessiner un carré fait du caractère `caractere` :

```
def dessine_carre(n, caractere):
    i = 0
    while i < n:
        j = 0
        while j < n:
            print(caractere, end = '')
            j += 1
        print('\n', end = '')  # ou simplement print()
        i += 1
```

Fonction sans valeur de retour

Dessiner un *rectangle* fait du caractère `caractere` :

Fonction sans valeur de retour

Dessiner un *rectangle* fait du caractère `caractere` :

```
def dessine_rectangle(m, n, caractere):
    for i in range(m):
        for j in range(n):
            print(caractere, end = '')
        print()
```

Composition de fonctions

On peut appeler une fonction dans une fonction ! (et ainsi de suite)

Composition de fonctions

On peut appeler une fonction dans une fonction ! (et ainsi de suite)

Dessiner un rectangle : variante

```
def dessine_ligne(n, caractere):
    for j in range(n):
        print(caractere, end = '')
    print()

def dessine_rectangle(m, n, caractere):
    for i in range(m):
        dessine_ligne(n, caractere)
```

Composition de fonctions

On peut appeler une fonction dans une fonction ! (et ainsi de suite)

Dessiner un rectangle : variante

```
def dessine_ligne(n, caractere):
    for j in range(n):
        print(caractere, end = '')
    print()

def dessine_rectangle(m, n, caractere):
    for i in range(m):
        dessine_ligne(n, caractere)
```

Question

➔ Qu'est-ce qui empêche une fonction de s'appeler *elle-même* ?

Dessiner un rectangle : deuxième variante

```
def dessine_ligne(n, caractere):
    for j in range(n):
        print(caractere, end = '')
    print()

def dessine_rectangle(m, n, caractere):
    if m > 0:
        dessine_ligne(n, caractere)
        dessine_rectangle(m-1, n, caractere)
```

La récursivité

Récursivité

Fonction récursive

➔ Une fonction qui s'appelle elle-même.

Exemple : factorielle

$$0! = 1$$

$$n! = n \times (n-1)!$$

Récursivité

Fonction récursive

➔ Une fonction qui s'appelle elle-même.

Exemple : factorielle

$$0! = 1$$

$$n! = n \times (n-1)!$$

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

Exemple : dessiner une spirale

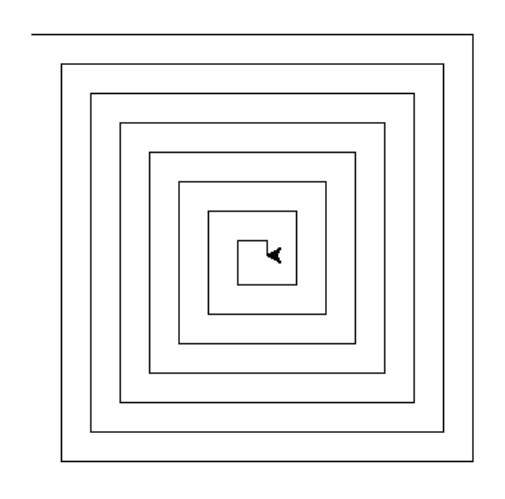


Figure 1 : Une spirale

Pour dessiner une spirale de côté $c > 0$:

1. On trace un trait de longueur c
2. On tourne le stylo de 90 degrés
3. On trace une spirale de côté $c - d$, où d est l'écart entre un côté de la spirale et le suivant.

Teaser : l'arbre d'Auto-Warhol

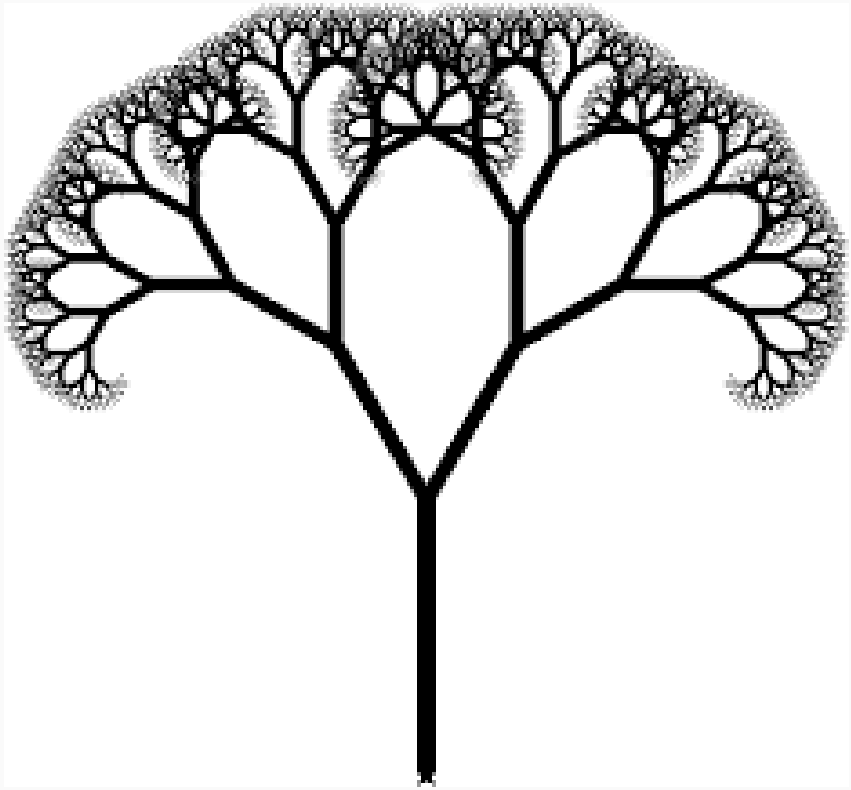


Figure 2 : Un arbre

Cas de base

Fonction récursive

➔ Une fonction qui s'appelle elle-même.

```
def fact(n):
    return n * fact(n-1)
```

Cas de base

Fonction récursive

➔ Une fonction qui s'appelle elle-même.

```
def fact(n):
    return n * fact(n-1)
```

Attention !

Pour qu'elle termine, une fonction récursive doit avoir un ou plusieurs *cas de base*.
(comme une boucle `while` doit avoir une condition d'arrêt)

Appels récursifs

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

Appels récursifs

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

Attention !

Pour qu'elle termine, une fonction récursive doit s'auto-appeler sur des arguments *plus simples*.

(comme une boucle `while` qui dépend de `n` doit modifier `n`, sinon elle ne termine pas)

Exemple : l'arbre d'Auto-Warhol

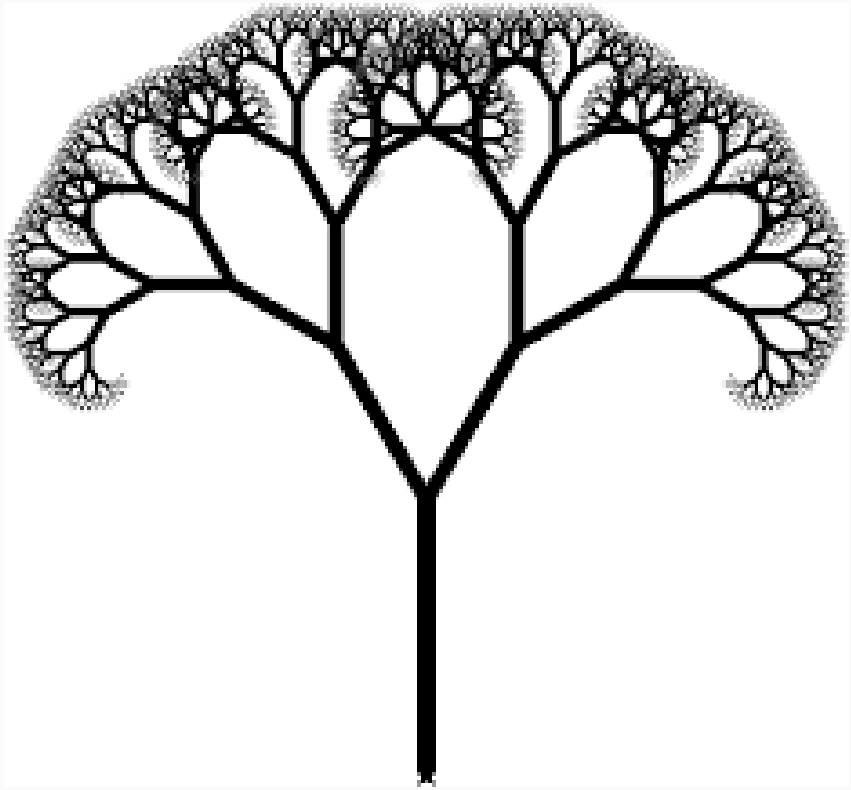


Figure 3 : Un arbre

Exemple : l'arbre d'Auto-Warhol

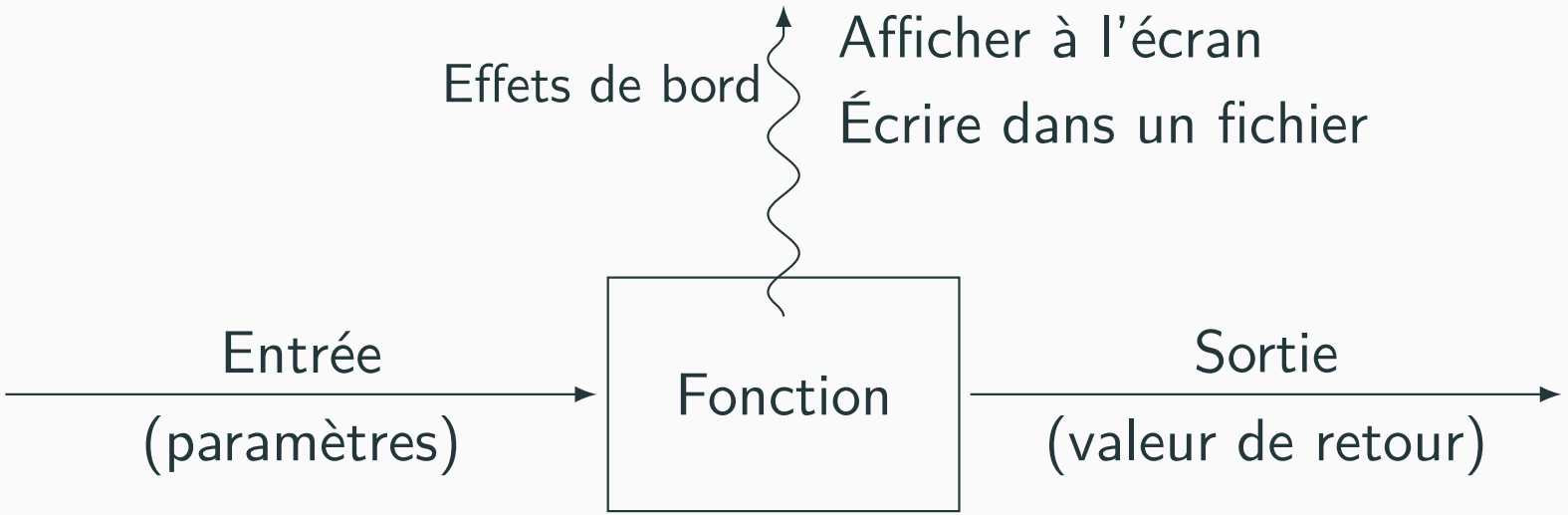
Pour dessiner un arbre dont le tronc mesure L (si $L > 1$) :

1. On trace un trait de longueur L
2. On tourne le stylo de 30 degrés vers la gauche
3. On trace un arbre de tronc $\frac{2}{3}L$
4. On tourne le stylo de 30 degrés vers la droite
   pour le remettre en position
5. On tourne le stylo de 30 degrés vers la droite
6. On trace un arbre de tronc $\frac{2}{3}L$
7. On tourne le stylo de 30 degrés vers la gauche
   pour le remettre en position
8. On ramène le stylo à sa place en traçant un trait de longueur L à l'envers

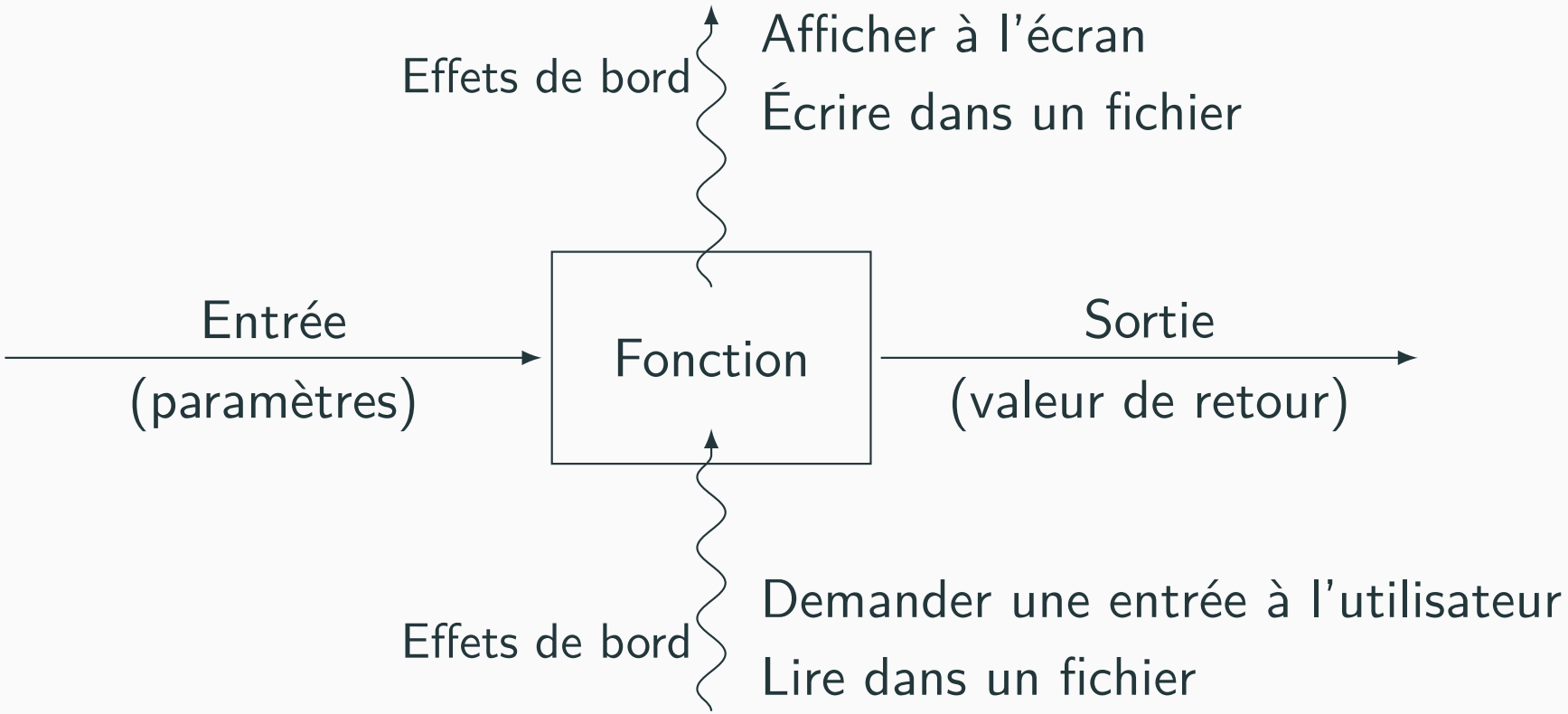
Suppléments sur les fonctions

Fonctions

Fonctions


Effets de bord
Afficher à l'écran
Écrire dans un fichier
Entrée
(paramètres)
Fonction
Sortie
(valeur de retour)

Fonctions


Effets de bord
Afficher à l'écran
Écrire dans un fichier
Entrée
(paramètres)
Fonction
Sortie
(valeur de retour)
Effets de bord
Demander une entrée à l'utilisateur
Lire dans un fichier