Algorithmique et Programmation 1

Lundi 23 septembre 2024

L1 Mathématiques - L1 Informatique
Semestre 1

Chapitre 4 : Fonctions

Pas facile à lire !

```
i = 0
while i < n:
    j = 0
    while j < n:
        print(caractere, end = '')
        j += 1
    print('\n', end = '')
    i += 1
```

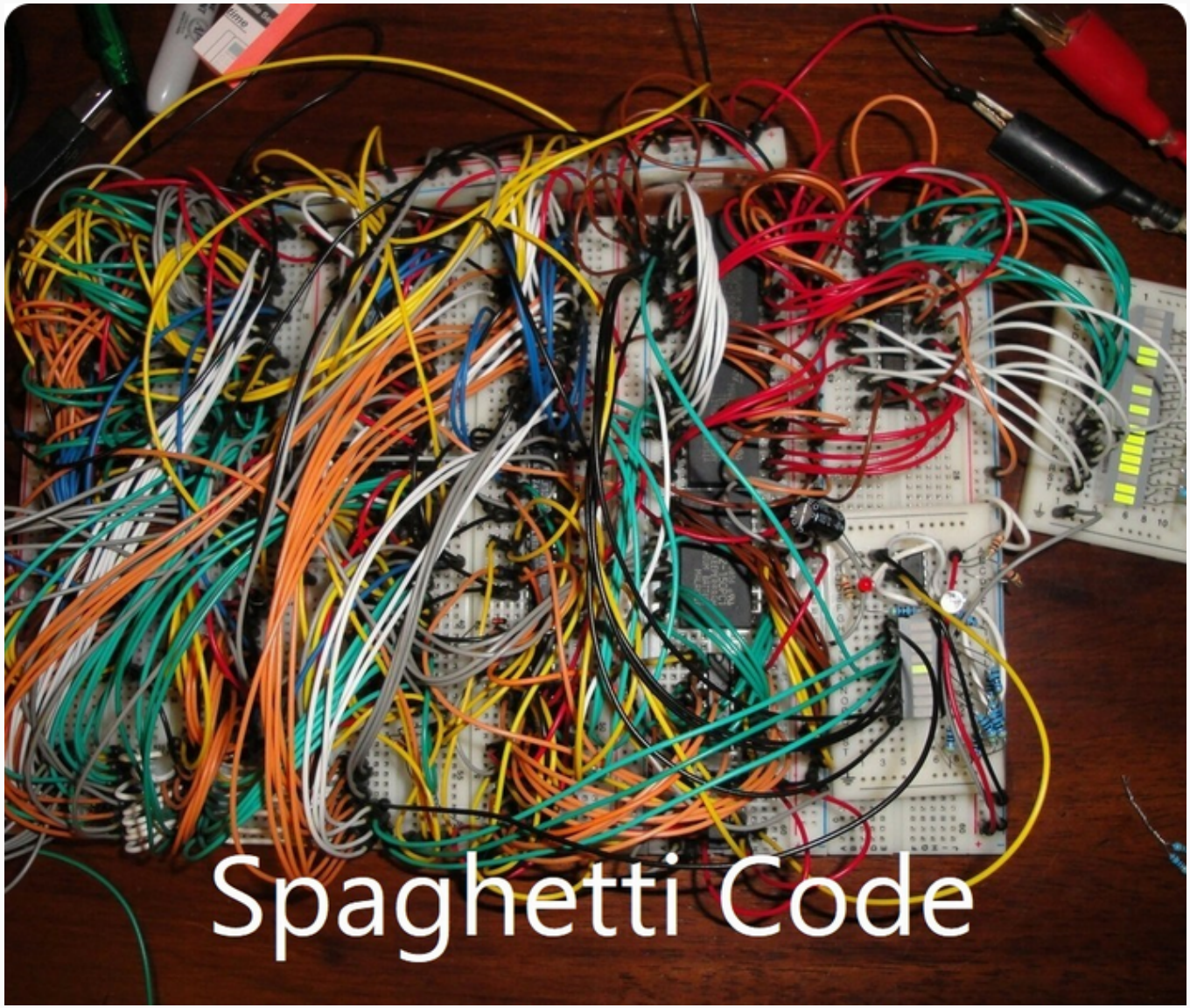
Une fonction : qu'est-ce que c'est ?

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)

➔ Ça ressemble beaucoup aux fonctions vues en mathématiques !

Une fonction : à quoi ça sert ?



Une fonction : à quoi ça sert ?

Lisibilité

- Isoler les parties du programme selon ce qu'elles font
- Éviter trop d'imbrications entre `if` et `while`

Une fonction : à quoi ça sert ?

Lisibilité

- Isoler les parties du programme selon ce qu'elles font
- Éviter trop d'imbrications entre `if` et `while`

Modularité et robustesse

- Réutiliser le code plusieurs fois
- Faciliter la correction des bugs

Une fonction : à quoi ça sert ?

Lisibilité

- Isoler les parties du programme selon ce qu'elles font
- Éviter trop d'imbrications entre `if` et `while`

Modularité et robustesse

- Réutiliser le code plusieurs fois
- Faciliter la correction des bugs

Généricité

- Changer la valeur des paramètres

Fonctions prédéfinies

➔ On en a déjà vu !

Exemples

Fonctions prédéfinies

➔ On en a déjà vu !

Exemples

- `print(s)` : afficher la chaîne de caractères `s`
- `input(s)` : afficher la chaîne de caractères `s` et attendre une entrée utilisateur
- `int(x)` : convertit l'objet `x` en entier
- `len(x)` : renvoie la longueur de l'objet `x`

Fonctions prédéfinies

➔ On en a déjà vu !

Exemples

- `print(s)` : afficher la chaîne de caractères `s`
- `input(s)` : afficher la chaîne de caractères `s` et attendre une entrée utilisateur
- `int(x)` : convertit l'objet `x` en entier
- `len(x)` : renvoie la longueur de l'objet `x`

Modules

Bibliothèque de fonctions, de types et d'objets

Exemples

Fonctions prédéfinies

➔ On en a déjà vu !

Exemples

- `print(s)` : afficher la chaîne de caractères `s`
- `input(s)` : afficher la chaîne de caractères `s` et attendre une entrée utilisateur
- `int(x)` : convertit l'objet `x` en entier
- `len(x)` : renvoie la longueur de l'objet `x`

Modules

Bibliothèque de fonctions, de types et d'objets

Exemples

- `randint(a,b)` (module `randint`)

➔ renvoie un entier aléatoire compris entre `a` et `b`

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` dans le code :

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

Exemples

Fonction à paramètres et résultats

Calculer le maximum de deux entiers :

Fonction à paramètres et résultats

Calculer le maximum de deux entiers :

```
def maximum(a,b):
    if a >= b:
        return a
    else:
        return b
```

Fonction à paramètres et résultats

Calculer le maximum de deux entiers :

```
def maximum(a,b):
    if a >= b:
        return a
    else:
        return b
```

Entraînement

- Ouvrir le programme `maximum.py`
- Décrire l'exécution pas à pas du programme (avec état de la mémoire)
- On peut aussi essayer avec Python Tutor
- Dresser un tableau de valeurs de l'exécution du programme

Fonction sans paramètres

Simuler un lancé de dés, et compter combien de coups il faut pour faire un 6.

Fonction sans paramètres

Simuler un lancé de dés, et compter combien de coups il faut pour faire un 6.

```
from random import randint
def lance_de():
    return randint(1,6)

compteur = 1
while lance_de() != 6:
    compteur = compteur + 1
print('Obtenu un 6 en', compteur, 'jets de dé.')
```

Fonction sans paramètres

Simuler un lancé de dés, et compter combien de coups il faut pour faire un 6.

```
from random import randint
def lance_de():
    return randint(1,6)


compteur = 1
while lance_de() != 6:
    compteur = compteur + 1
print('Obtenu un 6 en', compteur, 'jets de dé.')
```

Ce que ça donne sur Python Tutor.

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 = '')
        i += 1
    return
```

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 = '')
        i += 1
    return
```

Ce que ça donne sur Python Tutor.

Erreur fréquente : confusion paramètre / saisie retour / affichage

Paramètre / saisie

```
def maximum(a,b):
    a = int(input()) # NON !
    b = int(input()) # NON !
    if a >= b:
        return a
    else:
        return b
```

Erreur fréquente : confusion paramètre / saisie retour / affichage

Paramètre / saisie

```
def maximum(a,b):
    a = int(input()) # NON !
    b = int(input()) # NON !
    if a >= b:
        return a
    else:
        return b
```

Retour / affichage

```
def maximum(a,b):
    if a >= b:
        print(a) # NON !
    else:
        print(b) # NON !
```

Composition de fonctions

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

Dessiner un carré : variante

```
def dessine_ligne(n, caractere):
    j = 0
    while j < n:
        print(caractere, end = '')
        j += 1
    print('\n', end = '')
    return

def dessine_carre(n, caractere):
    i = 0
    while i < n:
        dessine_ligne(n,caractere)
        i += 1
    return
```

Fonctions et espaces de nom

- les paramètres et variables définies dans le corps d'une fonction sont indépendantes des autres variables du programme
- elles n'existent plus une fois l'exécution de la fonction terminée
- on les appelle des variables *locales*

Par conséquent, on peut renommer les variables d'une fonction.

$\rightarrow$ Démonstration sur Thonny (`maximum.py`)

(Contre-)exemple : intervertir des variables

- Dans `maximum.py`, changer les valeurs de `a` et `b` dans la fonction n'a pas d'effet sur `x` et `y` dans le programme principal !
- Dans `interversion.py`, la variable `temp` n'existe plus après l'exécution de la fonction