Fonctions

Algorithmique et structuration de données

Léna Gaubert
lena.gaubert@proton.me

L3 TIM/TAL @ INALCO

Informations générales

Reste du semestre, examen final...

Où en sommes-nous ?

Jusqu'ici

- *6 cours*, *7 TP* (dont un de révisions)
- **Différents types de données** : chaînes, listes, entiers, décimaux, booléen
- **Boucles** : for & while
- **Structures conditionnelles** : if-else-elif

Il nous reste :

- **fonctions**
- **dictionnaires**

Examen final

Examen final : lundi 8/12 @ 16h30
Durée : 1h30-2h (+ tiers-temps)
Programme : tous (sauf les dictionnaires)
Aucun document autorisé !

Examen final

Pour bien réviser :

- Reprendre les TP (sans correction d'abord !)
- Revoir les diapositives des cours pour le *vocabulaire*.
- Ne pas hésiter à poser des questions ! (à ses camarades, ou à moi !)

Conventions d'écriture

Guidelines

- Utiliser des **caractères ASCII** (encodage)
- **Variables** : noms simples, mais *explicites*.
- **Langue** : de préférence, opter pour l'anglais. (garder la même langue pour tout le programme !)
- **Pas d'espace** entre variable (type séquence) et crochet d'indentation !
- **Pas d'espace** non plus entre fonction et parenthèses !

Pour aller plus loin : voir PEP 8 – Style Guide for Python Code

Fonctions

Concepts théoriques, motivations...

Qu'est-ce qu'une fonction ?

Une **fonction** est un *bloc de code réutilisable* qui effectue une tâche *spécifique*.Elle prend des **entrées** (arguments), effectue des **opérations**, et produit une **sortie** (résultat).

Motivations: à quoi sert une fonction dans Python?

- **Organiser du code** : découper un programme complexe en petites parties
- **Éviter la répétition** : écrire une fois la fonction, et l'appeler plusieurs fois
- **Faciliter la maintenance** : on corrige le code à un seul endroit donné
- **Améliorer la lisibilité** : donner un nom explicite à une opération

Théorie et motivations

Vous connaissez déjà plusieurs fonctions de Python :

- `len()` : renvoie la longueur d'une séquence (chaîne, liste...)
- `range()` : génère un intervalle
- `print()` : affiche un message
- `input()` : demande à l'utilisateur un message

Fonctions

Syntaxe Python et exemples

Syntaxe

On définit une fonction selon la syntaxe suivante :

```
def fonction(arg):
    return result
```

Syntaxe

On définit une fonction selon la syntaxe suivante :

```
def fonction(arg):
    return result
```

- `def` pour indiquer qu'on *définit* une fonction
- `return` pour indiquer ce que la fonction *renvoie*
- la fonction prend le nom que vous souhaitez
- autant d'arguments que nécessaire

Une fonction peut prendre...

- Aucun argument : `def hello(): return "Bonjour"`
- Un ou plusieurs arguments : `def add(a, b): return a + b`
- Arguments optionnels (avec valeur par défaut) :

```
def greet(name, greeting="Bonjour"):
    return f"{greeting}, {name}!"
```

L'ordre des arguments est important ! Les arguments *optionnels* sont passés après les arguments *positionnels*.

Opérations simples

```
def addition(a, b):
    return a + b

r = addition(a, b)
```

Attention à bien conserver les bons noms d'arguments (*variables locales*) dans le programme de la fonction !

Renvoyer plusieurs résultats

```python
def square_cube(n):
    return n**2, n**3

r = square_cube(3)
# type(r) ?
s, c = square_cube(3)
# type(s), type(c) ?
```

On peut renvoyer le type de données qu'on souhaite !

Fonction sans return

Une fonction peut ne rien renvoyer (elle effectue juste une action) :

```
def display_message(name):
    print(f"Bienvenue, {name}!")

display_message("Alice")
```

Our all-time favourite: odd or even?

```
def odd_or_even(n):
    is_even = False
    if n % 2 == 0:
        is_even = True
    return is_even

result1 = odd_or_even(2)
result2 = odd_or_even(5)
```

Time to practice! („> ˘ <„) .´.´