Introduction à l'algorithmique et la programmation

Algorithmique et structuration de données

Léna Gaubert

lena.gaubert@proton.me

L3 TIM/TAL @ INALCO

Semestre 5

Introduction

présentations, organisation et syllabus

About me

Data Scientist @ Arlequin AI
Master pluriTAL (mention R&D) @ Sorbonne Nouvelle
Mentorat @ Wild Code School & Centrale Supélec
Contact: lena.gaubert@proton.me

Organisation

- Séances de 3h : 1h de théorie et 2h de pratique
- Supports de cours (TP et diapos) en ligne sur mon site (https://kittog.github.io/teachings)
- 12 semaines de cours
- 2 partiels (mi et fin semestre) sur papier (& more?)

Apprentissage du langage Python

- connaître et manipuler différents types de données (entiers, chaînes, listes...)
- résoudre des problèmes algorithmiques
- déboguer un code Python
- produire du code lisible, clair, et *propre*

Quelques pré-requis

- vous avez accès à un ordinateur (Linux/Windows/MacOs)
- vous êtes familiarisé avec l'utilisation du terminal
- *vous voulez apprendre à programmer en Python*

Syllabus

Ce cours couvrira les sujets suivants :

- **Types de données** (*data types*) : entier, float, chaînes de caractères, listes, dictionnaires...
- **Opérations** & **instructions conditionnelles** (`if`, `else`)
- **Boucles** (`while`, `for`)
- **Fonctions**

Pourquoi (*encore*) apprendre à coder ?

What about vibe-coding?

Pourquoi Python, finalement ?

- **multiplateforme** (Linux/MacOs/Windows/Android...)
- langage de **haut niveau**, et **interprété** (pas besoin d'être compilé pour être exécuté)
- langage très **polyvalent**, utilisé dans de nombreux domaines (TAL, physique, biologie, finances, data science...)

Mise en place

installation Python, éditeur de texte...

Installation Python & mise en place

Vérifier son installation Python (depuis le terminal)

- Dans le `shell Unix` (MacOs, Linux) ou `powershell` (Windows) :

```
python3 --version
```

> Si vous n'avez pas Python sur votre machine : How to install Python: a guide (Real Python).

- Pour lancer l'interpréteur Python depuis le terminal : `python3`

Créer un répertoire pour le cours

Toujours dans le terminal (shell/Powershell), depuis le répertoire HOME (pour y retourner : `cd ~`) : créer le répertoire `intro-python/scripts`.

```
mkdir intro-python/scripts
```

Éditeur

Vous pouvez utiliser l'éditeur de votre choix !

- Visual Studio Code
- PyCharm
- Sublime Text
- Pulsar (successeur d'Atom, archivé depuis fin 2022)
- Vim (à vos risques et périls ; pour une alternative moderne voir Helix)

Structure d'un script

Un script écrit en langage Python suit généralement la structure suivante :

1. **Imports de modules**
2. **Fonctions**
3. **Instructions** : Commandes exécutées par l'interpréteur Python (opérations, appel fonctions...)
4. **Commentaires** : Expliquent votre code en langage naturel. Précédé du caractère `#`, un commentaire est ignoré par l'interpréte Python jusqu'à la fin de la ligne.

Premiers scripts

Hello world, déclaration de variables...

Premier script : Hello world!

Dans l'éditeur de votre choix, ouvrez un fichier `hello_world.py` et écrivez les instructions suivantes.

```
# print message
print("Hello, world!")
```

Note : vous pouvez aussi écrire ces lignes de code directement dans l'interpréteur Python !

Pour exécuter le script, écrire la commande suivante dans votre terminal :

```
python3 hello_world.py
```

Hello Alan!

Petite variation du script `hello_world.py`

```
# declare variables
first_name = input("first name:")
last_name = input("last name:")
# create message
message = "Hello " + first_name + "!"
# print message
print(message)
```

Hello Alan!

Quelques explications (ligne par ligne) :

1. `first_name = intput("first name:")` demande à l'utilisateur de saisir son prénom et le stocke dans first_name
2. `last_name = input("last name:")` demande à l'utilisateur de saisir son nom de famille et le stocke dans last_name
3. `message = "Hello " + first_name + "!"` crée une phrase en ajoutant le prénom dans un message de bienvenue

Attribuer une valeur à une variable

- attribuer une valeur à une variable en une seule ligne (langage de *haut niveau*)

```
name = "Alan"
age = 41
```

- attribuer des valeurs à plusieurs variables en une seule ligne

```
# three different students
s1, s2, s3 = "Parry", "Tay", "Ada"
```

- donner la même valeur à plusieurs variables en une seule ligne

```
# three students with the same name
s1 = s2 = s3 = "Eliza"
```

Types de variables

Nom	Python	Description	Exemple
Integer	`int`	Nombre entier	`age=25`
Float	`float`	Nombre décimal	`temp=29.5`
String	`str`	Chaîne de caractères	`name="Eliza"`
Boolean	`bool`	Booléen. `True` = 1 ; `False` = 0	`is_here=True`

Types de variables

Nom	Python	Description	Exemple
List	`list`	Liste *ordonnée*	`l=[12, 14.5]`
Tuple	`tuple`	Liste *immutable*	`a=(2.0, 4.3)`
Dict	`dict`	Dictionnaire	`{"lang": "en"}`
Set	`set`	Ensemble	`{"en", "fr"}`
None	`NoneType`	Aucune valeur	`comment=None`

Catégories de variables

On peut regrouper les différents types de variables dans les catégories suivantes :

- *Text type* : `str`
- *Numeric types* : `int`, `float`
- *Sequence types* : `list`, `tuple`
- *Mapping type* : `dict`
- *Set type* : `set`
- *Boolean type*: `bool`
- *None type* : `None`

Casting : passer d'un type à un autre

```
# décimal -> entier
x = 10.6  # float
int_x = int(x)

# entier -> décimal
y = 10
float_y = float(y)

# nombre -> chaîne
z = 42
str_z = str(z)

# nombre -> booléen
w = 0
bool_w = bool(w)
```

Type d'une variable

Pour connaître le type d'une variable, utiliser la fonction `type()`.

```
age = 25
name = "Alan"
is_student = False

print(type(age))
print(type(name))
print(type(is_student))
```

Fonctions et opérations

`input()`, `print()`, opérations mathématiques...

Fonctions print()

print()

La fonction `print()` permet d'afficher des messages, des variables, des résultats, au fil de votre script. `print()` reçoit les arguments suivants :

- `*objects` : ce qu'on souhaite afficher
- `sep` : définit le séparateur entre plusieurs objets (par défaut, `sep=" "`)
- `end` : définit ce qui est ajouté à la **fin** du message (par défaut : `end = \n`)

> `sep` et `end` sont des arguments *optionnels*.
> On les passe en arguments en *dernier*.
> (Sinon, erreur!!!!)

Fonctions print()

Quelques exemples !

```
x = 4
print("Le résultat est : ", x, end="!")
# affiche : Le résultat est 4 !
```

```
print("a", "b", "c", sep="-", end="!")
# affiche : a-b-c!
```

```
name = "Ada"
print("Hello", name, "!",
      sep="?", end="#")
# affiche : Hello?Ada?!#
```

Fonction input()

input()

La fonction `input()` permet de recevoir des données d'entrée de l'utilisateur, via la console/le terminal.

- Elle prend en argument un message (`prompt`) à afficher à l'utilisateur (par défaut : `prompt=""`)
- La fonction `input()` renvoie *toujours* une chaîne de caractère.

```
x = input(prompt="Enter your name: ")
y = input(prompt="Enter your age: ")
print(type(x))  # <class str>
print(type(y))  # <class str>
```

Fonction input()

Quelques exemples !

Si on demande à l'utilisateur de renseigner un nombre (entier ou flottant/décimal), il se peut qu'on veuille convertir le texte renvoyé par `input()` !

```
age = input(prompt="Enter your age: ")
int_age = int(age)
# vous pouvez aussi écrire :
temp = float(input("What's the temperature?"))
```

Opérations mathématiques

Opérateur	Dénomination	Effet
`+`, `-`	Addition, soustraction	additionne, soustrait
`*`	Multiplication	multiplie
`/`	Division	divise
`//`, `%`	Quotient et reste de la division euclidienne	`11//2=5` `11%2=1`
`**`	Puissance	$2^3 = 2 * 2 * 2$

Opérations mathématiques

Opérations sur les types numériques

```
x = 45
y = x + 2  # 47
z = y - x  # 2
p = z ** 2  # 4
d = x / 4  # 11.25
e = x // 4 # 11
r = x % 4  # 1
```

Opérations sur les chaînes de caractères

```
greeting = " Hello"
greet_w_name = greeting + " Alan!"
print(greet_w_name * 3)
```

La fonction `print()` affichera : `Hello Alan! Hello Alan! Hello Alan!`

Opérations invalides

```
>>> greeting ** 2
>>> greeting + 4
>>> greeting - "a"
```

Time to practice! ヽ(。> ᵕ <)ノ♡