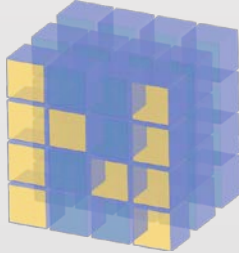




# Introduction à la Data Science en Python

**matplotlib**

 **NumPy**

 **scikit  
learn**

**Pandas**



 **jupyter**



Introduction  
Data Science  
Jupyter notebook  
Module 1  
Module 2  
Module 3  
Module 4

# Season 1

# Python for Data Science



**Introduction**

Data Science

Jupyter notebook

Module 1

Module 2

Module 3

Module 4

# Episod 1

## Pilot



## Introduction

Data Science

Jupyter notebook

Module 1

Module 2

Module 3

Module 4

# Python

Qu'est ce que Python ?

1. Facile à apprendre et très populaire
  - Quel que soit le paramétrage du classement, Python reste le langage de programmation le plus populaire, devant C++ et C pour le mobile, et Java et C# pour le web (Spectrum)
2. Complet
  - Un véritable couteau suisse. Il ne s'arrête pas aux statistiques mais il permet la manipulation et le nettoyage de données, le calcul haute performance (HPC) etc...
3. Possède des librairies de data science très performantes
  - Ecosystème SciPy
  - Scikit learn



## Introduction

Data Science

Jupyter notebook

Module 1

Module 2

Module 3

Module 4



Pandas



matplotlib





## Introduction

Data Science

Jupyter notebook

Module 1

Module 2

Module 3

Module 4

# 4 Modules :

1. Prérequis en Python
2. La boîte à outils Pandas
3. Requêtes et manipulations avancées avec Pandas
4. Analyses statistiques de base avec Numpy et Scipy,  
Mini-projet



Introduction

**Data Science**

Jupyter notebook

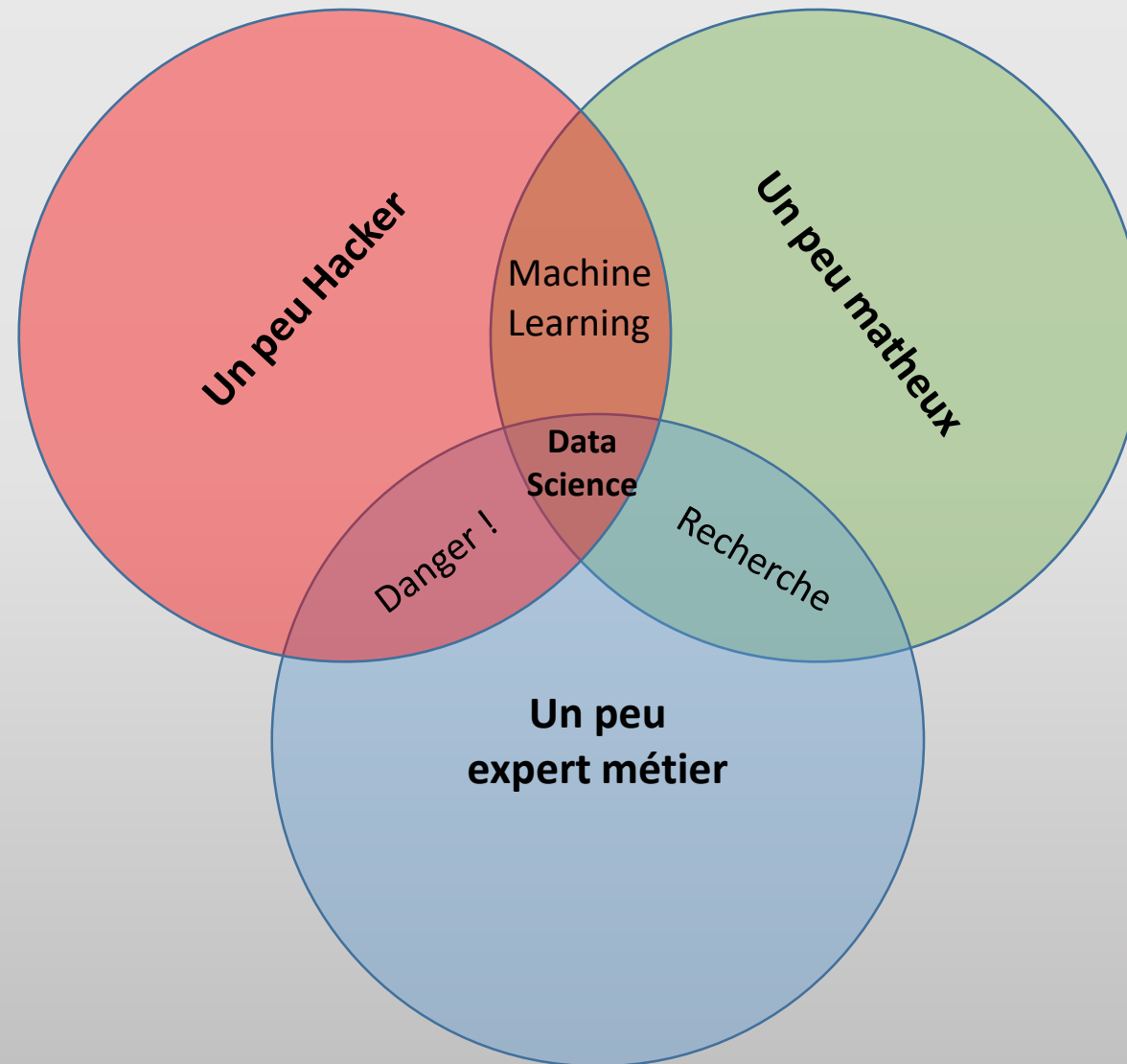
Module 1

Module 2

Module 3

Module 4

# Data Scientist





Introduction  
Data Science

Jupyter notebook

Module 1

Module 2

Module 3

Module 4

## Un peu de lecture



*50 ans de data science*

<https://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf>

- Exploration et préparation des données
- Visualisation et transformations
- Calculs
- Modélisation
- Visualisation et présentation
- Data science



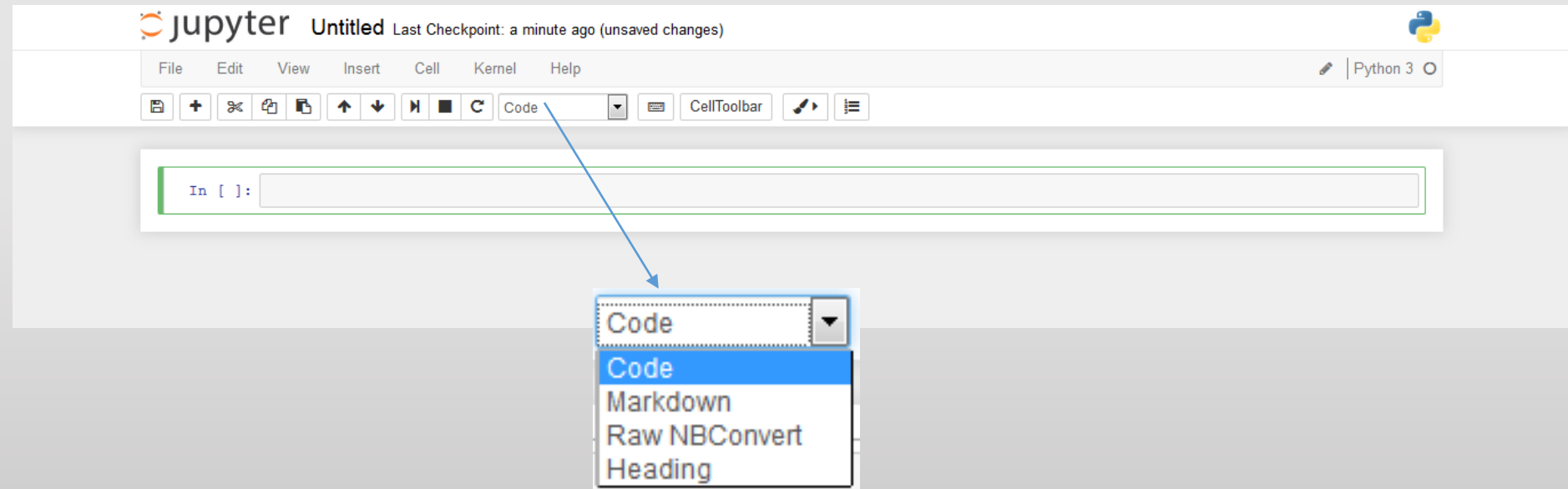


Introduction  
Data Science  
**Jupyter notebook**  
Module 1  
Module 2  
Module 3  
Module 4



# Jupyter notebooks

écrivez, exécutez, documentez et publiez votre code Python





Introduction

Data Science

**Jupyter notebook**

Module 1

Module 2

Module 3

Module 4



## Jupyter notebooks

Les balises Markdown permettent de définir des mises en forme : `<h1>`, `<h2>` ...

**# Titre principal**



Introduction  
Data Science  
**Jupyter notebook**  
Module 1  
Module 2  
Module 3  
Module 4



## Jupyter notebooks

# : <h1>, ## : <h2> ...

**Titre principal**

**## Titre secondaire**



Introduction  
Data Science  
**Jupyter notebook**  
Module 1  
Module 2  
Module 3  
Module 4



## Jupyter notebooks

Les cellules de code permettent d'écrire un script python

**Titre principal**

**Titre secondaire**

```
In [ ]: a = 0
        while a <= 10 :
            print (a)
            a = a + 1
```



Introduction  
Data Science  
**Jupyter notebook**  
Module 1  
Module 2  
Module 3  
Module 4



# Jupyter notebooks

MAJ+ENTER

## Titre principal

### Titre secondaire

```
In [1]: a = 0  
while a <= 10 :  
    print (a)  
    a = a + 1
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

```
In [ ]:
```



Introduction

Data Science

**Jupyter notebook**

Module 1

Module 2

Module 3

Module 4



## Jupyter notebooks

[Un petit guide GitHub des balises Markdown](#)

<https://openclassrooms.com/fr/courses/1304236-redigez-en-markdown>

<https://www.markdowntutorial.com/>



Introduction  
Data Science  
Jupyter notebook  
**Module 1**  
Module 2  
Module 3  
Module 4

# Episod 2

## Module 1



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

12

22.314

type("James Bond 007")

str





Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

type 12	int
type 22.314	float
type "James Bond 007"	str



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

<code>type(12)</code>	<code>int</code>
<code>type(22.314)</code>	<code>float</code>
<code>type("James Bond 007")</code>	<code>str</code>



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

Transtypage

`float(2)` → 2.0



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

Transtypage

`float(2) → 2.0`

`int(1.1) → 1`

`int('1') → 1`



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

Transtypage

`float(2)` → 2.0

`int(1.1)` → 1

`int('1')` → 1

`int('A')` → error



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Types

## Transtypage

`float(2)` → 2.0

`int(1.1)` → 1

`int('1')` → 1

`int('A')` → error

`str(2)` → "2"

`str(1.1)` → '4.5'



Introduction  
Data Science  
Jupyter notebook  
Fonctions  
Types  
Chaines  
CSV  
Dates  
Objet  
Lambda  
Numpy

# Types

## Booléens

True
False



Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

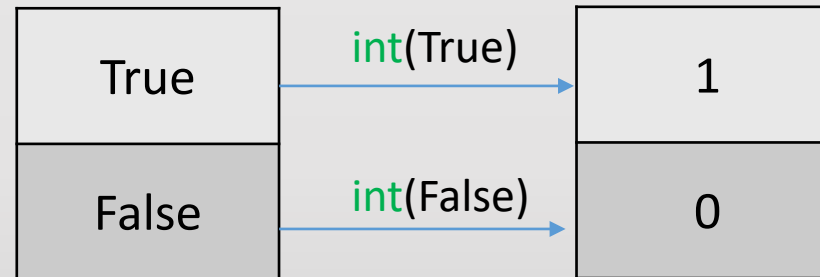
Module 2

Module 3

Module 4

# Types

## Booléens







Introduction  
Data Science  
Jupyter notebook

## Module 1

- **Types**
- variables
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

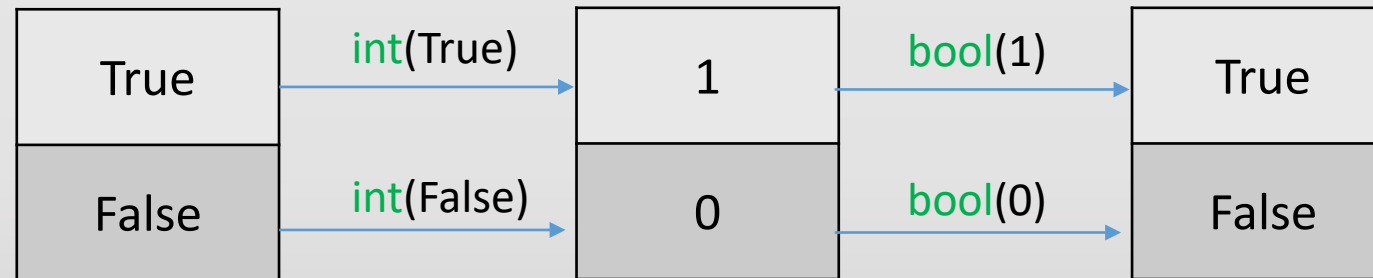
Module 2

Module 3

Module 4

# Types

## Booléens





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- **variables**
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Variables et expressions

symbole	exemples
+	$6+4 \rightarrow 10$
-	$6-4 \rightarrow 2$
*	$6*4 \rightarrow 24$ $1.2 * 1 \rightarrow 1.2$
**	$12**2 \rightarrow 144$
/	$6/4 \rightarrow 1 (*)$ $6./4 \rightarrow 1.5$
//	$6//4 \rightarrow 1$
%	$6\%4 \rightarrow 2$



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- **variables**
- Chaines
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Variables et expressions

Les variables Python ne sont pas typées.

Il convient de respecter quelques règles de nommage :

- Le nom des classes en CamelCase
  - Les (pseudo) constantes en UPPER\_CASE
  - Le reste en snake\_case
  - Explicit is better than implicit
- 
- Voir Zen of Python



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

**"James Bond" ou 'James Bond'**



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

"James Bond" ou 'James Bond'
'007'



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

"James Bond" ou 'James Bond'
'007'
"&#@\[^*"



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

```
Agent = "James Bond 007"
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

`len(Agent)` → 14



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0] → 'J'



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0] → 'J'

Agent[6] → 'B'



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0] → 'J'	Agent[6] → 'B'	Agent[13] → '7'
----------------	----------------	-----------------



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2  
Module 3  
Module 4

# Chaines de caractères

## Indexation

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0] → 'J'	Agent[6] → 'B'	Agent[13] → '7'
Agent[-14] → 'J'	Agent[-8] → 'B'	Agent[-1] → '7'



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Slicing

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0:3] → "Jam"



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Slicing

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[0:3] → "Jam"

Agent[6:10] → "Bond"



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Slicing

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[::2] → "JmsBn 0"





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Slicing

Agent = "James Bond 007"													
J	a	m	e	s		B	o	n	d		0	0	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Agent[::2] → "JmsBn 0"

Agent[1:8:2] → "ae o"



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Concaténation

```
Agent = "James Bond"
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Concaténation

```
Agent = "James Bond"  
replique = "My name is Bond, " + Agent
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Concaténation

```
Agent = "James Bond"  
replique = "My name is Bond, " + Agent  
  
replique → "My name is Bond, James Bond"
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Concaténation

```
Agent = "James Bond"  
replique = "My name is Bond, " + Agent
```

```
replique → "My name is Bond, James Bond"
```

```
3*"James " → " James James James "
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Caractère d'échappement

Les antislashs permettent d'échapper caractères mais aussi de créer des mises en forme

- `\n`
- `\r`
- `\\`
- `Print(r".....")`



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- **Chaines**
- Conteneurs
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Chaines de caractères

## Méthodes spécifiques

upper()

lower()

<https://docs.python.org/3/library/stdtypes.html#string-methods>

replace()

find()



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Conteneurs standard

Les chaines

Les tuples

Les listes

Les tableaux associatifs

Les ensembles





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Un tuple est une collection ordonnée et non modifiable d'éléments éventuellement hétérogènes.



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Un tuple est une collection ordonnée et non modifiable d'éléments éventuellement hétérogènes.

Les éléments sont encadrés par des parenthèses et séparés par des virgules.

Notes = (10, 9, 6, 5, 10, 8, 9, 6, 4.5)



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chainés
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Un tuple est une collection ordonnée et non modifiable d'éléments éventuellement hétérogènes.

Les éléments sont encadrés par des parenthèses et séparés par des virgules.

Notes = (10, 9, 6, 5, 10, 8, 9, 6, 4.5)

Film = ("Dr No", 1962, "Sean Connery")



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chainés
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Film = ("Dr No", 1962, "Sean Connery")

Les éléments sont accessibles via leur index :

Film[0] → *"Dr No"*



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Film = ("Dr No", 1962, "Sean Connery")

Les éléments sont accessibles via leur index :

Film[0] → "Dr No"

.

.

Film[-1] → "Sean Connery"



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chainés
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Opérations :

- Addition :

Film = Film + (1000000 , 5.8)

Film → ("Dr No", 1962, "Sean Connery", 5.8)



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

Opérations :

- Addition :  
Film = Film + (1000000 , 5.8)  
Film → ("Dr No", 1962, "Sean Connery", 5.8)
- Slicing :  
Film[1:3] → (1962, "Sean Connery")



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

## Opérations :

- Addition :  
Film = Film + (1000000 , 5.8)  
Film → ("Dr No", 1962, "Sean Connery", 5.8)
- Slicing :  
Film[1:3] → (1962, "Sean Connery")  
Film[1:] → (1962, "Sean Connery", 5.8)





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

## Opérations :

- Addition :  
Film = Film + (1000000 , 5.8)  
Film → ("Dr No", 1962, "Sean Connery", 5.8)
- Slicing :  
Film[1:3] → (1962, "Sean Connery")  
Film[1:] → (1962, "Sean Connery", 5.8)
- Longueur  
Len(Film) → 4



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les tuples

## Opérations :

- Addition :  
Film = Film + (1000000 , 5.8)  
Film → ("Dr No", 1962, "Sean Connery", 5.8)
- Slicing :  
Film[1:3] → (1962, "Sean Connery")  
Film[1:] → (1962, "Sean Connery", 5.8)
- Longueur  
Len(Film) → 4
- Tri  
notes1 = sorted((10, 9, 6, 5, 10, 8, 9, 6, 4.5))  
Film1 → [4.5, 5, 6, 6, 8, 9, 9, 10, 10]



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les listes

Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes

Elles sont encadrées par des crochets :

Exemple :

```
Film = ["Dr No", 1962, "Sean Connery"]
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les listes

Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes

Elles sont encadrées par des crochets :

Exemple :

```
Film = ["Dr No", 1962, "Sean Connery"]
```

```
Films_c64 = [("Dr No", "From Russia with love", "Goldfinger"), [1962, 1963, 1964]]
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chainés
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les listes

L'accès se fait alors par double indexation (Nesting)

```
Films_c64 = [("Dr No", "From Russia with love", "Goldfinger"), [1962, 1963, 1964]]
```

```
Films_c64[0][2] → "GoldFinger"
```



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les listes

## Opérations :

- **Addition :**  
Film = Film + [1000000 , 5.8] **ou** Film.extend([1000000 , 5.8])  
Film → ["Dr No", 1962, "Sean Connery", 5.8]



Introduction  
Data Science  
Jupyter notebook  
**Module 1**  
• Types  
• variables  
• Chaines  
• **Conteneurs**  
• Structures  
• Fonctions  
• POO  
• Fichiers  
• Numpy  
Module 2  
Module 3  
Module 4

# Les listes

## Opérations :

- Addition :  
Film = Film + [1000000 , 5.8] ou Film.extend([1000000 , 5.8])  
Film → ["Dr No", 1962, "Sean Connery", 1000000, 5.8]
- Ajout d'un élément :  
Film.append([1000000 , 5.8])  
Film → ["Dr No", 1962, "Sean Connery", [1000000, 5.8]]

[4.5, 5, 6, 6, 8, 9, 9, 10, 10]



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les listes

Opérations :

Film → ["Dr No", 1962, "Sean Connery", 1000000, 5.8]





- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les listes

Opérations :

Film → ["Dr No", 1962, "Sean Connery", 1000000, 5.8]

- **Modification :**

Film[4] = 6.5

Film → ["Dr No", 1962, "Sean Connery", 1000000, 6.5]



Introduction  
Data Science  
Jupyter notebook  
**Module 1**  
• Types  
• variables  
• Chaines  
• **Conteneurs**  
• Structures  
• Fonctions  
• POO  
• Fichiers  
• Numpy  
Module 2  
Module 3  
Module 4

# Les listes

Opérations :

Film → ["Dr No", 1962, "Sean Connery", 1000000, 5.8]

- Modification :  
Film[4] = 6.5  
Film → ["Dr No", 1962, "Sean Connery", 1000000, 6.5]
- Suppression:  
del(Film[3])  
Film → ["Dr No", 1962, "Sean Connery", 5.8]



# Les listes

## Opérations :

- Conversion de chaine :  
`"Sean Connery".split()` → `["Sean", "Connery"]`

Remarque : la méthode `split()` découpe les chaines en détectant les caractères "espace" mais il est possible d'en specifier un autre :

`"1963,1964,1965".split(",")` → `["1963", "1964", "1965"]`



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les listes

Opérations :

- Copie et clone

```
Film = ["Dr No", 1962, "Sean Connery", 1000000, 5.8]
```

```
Film1 = Film
```

Les deux sont dépendants : `Film[4] = 6.5`

`Film1[4] → 6.5` et réciproquement



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les listes

## Opérations :

- Copie et clone  
Film = ["Dr No", 1962, "Sean Connery", 1000000, 5.8]  
Film1 = Film  
  
Les deux sont dépendants : Film[4] = 6.5  
Film1[4] → 6.5 et réciproquement  
  
Film1=Film[:]  
Les deux sont indépendants.



Introduction

Data Science

Jupyter notebook

**Module 1**

- Types
- variables
- Chainés
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les listes

Pour plus d'infos :

```
Film = ["Dr No", 1962, "Sean Connery", 1000000, 5.8]
```

```
help(Film)
```



Introduction

Data Science

Jupyter notebook

**Module 1**

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les dictionnaires

- Les dictionnaires sont modifiables, mais non ordonnés : les couples enregistrés n'occupent pas un ordre immuable, leur emplacement est géré par un algorithme spécifique (algorithme de hash). Le caractère non ordonné des dictionnaires est le prix à payer pour leur rapidité !



# Les dictionnaires

- Les dictionnaires sont modifiables, mais non ordonnés : les couples enregistrés n'occupent pas un ordre immuable, leur emplacement est géré par un algorithme spécifique (algorithme de hash). Le caractère non ordonné des dictionnaires est le prix à payer pour leur rapidité !
- Une clé pourra être alphabétique, numérique... en fait tout type hachable (donc liste et dictionnaire exclus).
- Les valeurs pourront être de tout type sans exclusion.
- Collection de couples **clé** : **valeur** entourée d'accolades.

Film = {"Titre" : "Dr No", "annee" : 1962, "acteur": "Sean Connery"}





Introduction  
Data Science  
Jupyter notebook  
**Module 1**  
• Types  
• variables  
• Chaines  
• **Conteneurs**  
• Structures  
• Fonctions  
• POO  
• Fichiers  
• Numpy  
Module 2  
Module 3  
Module 4

# Les dictionnaires

- Les dictionnaires sont modifiables, mais non ordonnés : les couples enregistrés n'occupent pas un ordre immuable, leur emplacement est géré par un algorithme spécifique (algorithme de hash). Le caractère non ordonné des dictionnaires est le prix à payer pour leur rapidité !
- Une clé pourra être alphabétique, numérique... en fait tout type hachable (donc liste et dictionnaire exclus).



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les dictionnaires

- Les dictionnaires sont modifiables, mais non ordonnés : les couples enregistrés n'occupent pas un ordre immuable, leur emplacement est géré par un algorithme spécifique (algorithme de hash). Le caractère non ordonné des dictionnaires est le prix à payer pour leur rapidité !
- Une clé pourra être alphabétique, numérique... en fait tout type hachable (donc liste et dictionnaire exclus).
- Les valeurs pourront être de tout type sans exclusion.



Introduction  
Data Science  
Jupyter notebook  
**Module 1**

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy
- Module 2
- Module 3
- Module 4

# Les dictionnaires

- Les dictionnaires sont modifiables, mais non ordonnés : les couples enregistrés n'occupent pas un ordre immuable, leur emplacement est géré par un algorithme spécifique (algorithme de hash). Le caractère non ordonné des dictionnaires est le prix à payer pour leur rapidité !
- Une clé pourra être alphabétique, numérique... en fait tout type hachable (donc liste et dictionnaire exclus).
- Les valeurs pourront être de tout type sans exclusion.
- Collection de couples **clé** : **valeur** entourée d'accolades.

Film = {"Titre" : "Dr No", "annee" : 1962, "acteur": "Sean Connery"}



Introduction

Data Science

Jupyter notebook

**Module 1**

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les dictionnaires

- Les clés sont **non modifiables** et **uniques**



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les dictionnaires

- Les clés sont **non modifiables** et **uniques**
- L'accès à un élément se fait par clé ou par index :

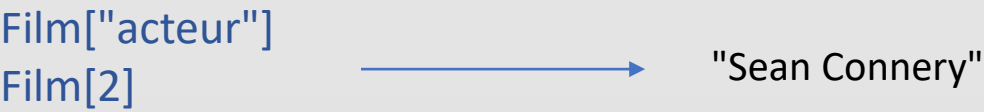
Film["acteur"]  
Film[2]                      →                      "Sean Connery"



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les dictionnaires

- Les clés sont **non modifiables** et **uniques**
- L'accès à un élément se fait par clé ou par index :



- Ajouter un élément  
Film["note"] = 6.5



Introduction  
Data Science  
Jupyter notebook

**Module 1**

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2  
Module 3  
Module 4

# Les dictionnaires

- Les clés sont **non modifiables** et **uniques**
- L'accès à un élément se fait par clé ou par index :

Film["acteur"]  
Film[2]                      →                      "Sean Connery"

- Ajouter un élément  
Film["note"] = 6.5
- Liste des clés ou des valeurs :  
Film.keys()  
Film.values()



- Introduction
- Data Science
- Jupyter notebook
- Module 1**
  - Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2
- Module 3
- Module 4

# Les sets

- un set est la transposition informatique de la notion d'ensemble mathématique.
- En Python, il existe deux types d'ensembles, les modifiables : set et les non modifiables : frozenset (équivalent des listes et tuples)





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les sets

- un set est la transposition informatique de la notion d'ensemble mathématique.
- En Python, il existe deux types d'ensembles, les modifiables : set et les non modifiables : frozenset (équivalent des listes et tuples)
- Ils ne contiennent pas de doublons :  
Notes = {10, 9, 6, 5, 10, 8, 9, 6, 4.5}  
Notes → {4.5, 5, 6, 8, 9, 10 }



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les sets

Opérations :

`X = set('spam')`

$X \rightarrow \{'s', 'p', 'm', 'a'\}$

`Y = set('pass')`

$Y \rightarrow \{'s', 'p', 'a'\}$



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les sets

## Opérations :

`X = set('spam')`

$X \rightarrow \{'s', 'p', 'm', 'a'\}$

`Y = set('pass')`

$Y \rightarrow \{'s', 'p', 'a'\}$

`'p' in X`

$\rightarrow$

True

`'m' in Y`

$\rightarrow$

False



Introduction  
Data Science  
Jupyter notebook  
**Module 1**

- Types
  - variables
  - Chaines
  - **Conteneurs**
  - Structures
  - Fonctions
  - POO
  - Fichiers
  - Numpy
- Module 2  
Module 3  
Module 4

# Les sets

## Opérations :

`X = set('spam')`                       $X \rightarrow \{'s', 'p', 'm', 'a'\}$

`Y = set('pass')`                       $Y \rightarrow \{'s', 'p', 'a'\}$

`'p' in X`                       $\rightarrow$                       True

`'m' in Y`                       $\rightarrow$                       False

`X - Y`                      # dans X mais pas dans Y                       $\rightarrow$                       `\{'m'\}`

`X ^ Y`                      # soit dans X soit dans Y                       $\rightarrow$                       `\{'m'\}`

`X | Y`                      # union                       $\rightarrow$                       `\{'s', 'p', 'm', 'a'\}`

`X & Y`                      # intersection                       $\rightarrow$                       `\{'s', 'p', 'a'\}`



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- **Conteneurs**
- Structures
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les sets

## Les méthodes :

`add(...)`

| Add an element to a set.

|

| This has no effect if the element is already present.

|

`clear(...)`

| Remove all elements from this set.

|

`copy(...)`

| Return a shallow copy of a set.

|



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

Il existe deux grandes structures de contrôles :

Conditions

Itérations



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

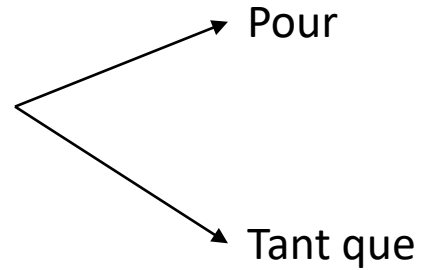
Il existe deux grandes structures de contrôles :

Conditions

Itérations

Pour

Tant que





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

Les conditions :

```
if condition.s :  
    instruction.s
```





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

Les conditions :

```
if condition.s :  
    instruction.s  
else :  
    instruction.s
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chainés
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

Les conditions :

```
if condition.s :  
    instruction.s  
elif condition.s :  
    instruction.s  
else :  
    instruction.s
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

Les opérateurs :

==	Égal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inferieur
<=	Inferieur ou égal à



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Compteur d'itération basé sur un conteneur :

**for** i **in** conteneur :



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Compteur d'itération basé sur un conteneur :

**for** i **in** conteneur :

Le conteneur peut être une liste, un tuple un set ou un dict (!)



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Compteur d'itération basé sur un conteneur :

**for** i **in** conteneur :

Le conteneur peut être une liste, un tuple un set ou un dict (!)

Le plus souvent on utilise un objet de type range :

range(n) : 0, 1, ....., n-1



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Compteur d'itération basé sur un conteneur :

**for** i **in** conteneur :

Le conteneur peut être une liste, un tuple un set ou un dict (!)

Le plus souvent on utilise un objet de type range :

range(n) : 0, 1, ....., n-1

range(n1, n2) : n1, n1+1, ....., n2-1



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Compteur d'itération basé sur un conteneur :

**for** i **in** conteneur :

Le conteneur peut être une liste, un tuple un set ou un dict (!)

Le plus souvent on utilise un objet de type range :

range(n) : 0, 1, ....., n-1

range(n1, n2) : n1, n1+1, ....., n2-1

range(n1,n2,r) : n1, n1+r, ....., <n2-1





Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle for :

Récupération des index et des valeurs :  
Fonction "enumerate"

```
for i,x in enumerate(['A','B','C']):  
    print(i,x)
```

enumerate contient les tuples suivants :

```
(0, 'A')  
(1, 'B')  
(2, 'C')
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- **Structures**
- Fonctions
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les structures de contrôle

## Les itérations :

### Boucle while :

Le nombre d'itérations dépend d'une condition:

**while** condition :

Exemple :

**#initialisation**

**i = 0**

**while** i < 10 :

**print(i)**

**i += 2** **#incrementation**



Introduction  
Data Science  
Jupyter notebook

## Module 1

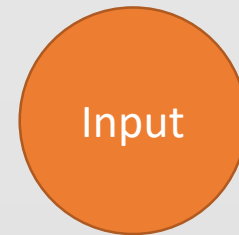
- Types
- variables
- Chainés
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions





Introduction  
Data Science  
Jupyter notebook

## Module 1

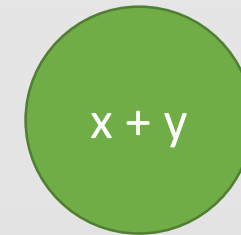
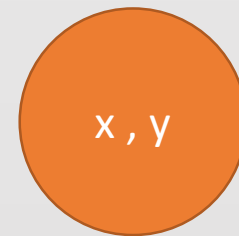
- Types
- variables
- Chaines
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions



```
def add(x, y):  
    return x + y
```

```
add(1, 2)
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions

Des fonctions natives :

`len()`

`sum()`

`sorted()` (! À la différence de la méthode `sort()`)



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions

Documentation d'une fonction

```
def add(x, y):  
    """ Fonction qui additionne deux valeurs """  
    return (x + y)  
  
help(add)
```



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions

Argument avec valeur par défaut :

```
def add(x, y=2) :  
    ''' Fonction qui additionne deux valeurs '''  
    return (x + y)
```

La valeur de y, si elle n'est pas spécifiée, sera égale à 2

add(12) → 14

add(12,5) → 17



Introduction  
Data Science  
Jupyter notebook

## Module 1

- Types
- variables
- Chaines
- Conteneurs
- Structures
- **Fonctions**
- POO
- Fichiers
- Numpy

Module 2

Module 3

Module 4

# Les fonctions

Portée des variables

Variables locales vs variables globales

Mot clé global