Atelier d'informatique

Épisode I : Introduction

31 janvier 2017

« *Un programme informatique fait ce que vous lui avez dit de faire, pas ce que vous voulez qu'il fasse.* » — Troisième loi de Greer

1. L'informatique, c'est quoi ?
2. Espace de travail
3. Calculs élémentaires
4. Variables
5. Chaînes de caractères
   - Généralités
   - Opérations sur les chaînes
6. Premiers programmes
7. Structures conditionnelles
   - Booléens
   - Booléens

Informatique : késako ?

Le mot *informatique* est la contraction des mots *information* et *automatique* : il s'agit donc de la science du traitement automatique de l'information.

Informatique : késako ?

Le mot *informatique* est la contraction des mots *information* et *automatique* : il s'agit donc de la science du traitement automatique de l'information.
Un *ordinateur* est la concrétisation de cette notion, une machine qui traite automatiquement des informations données en entrée, selon un *programme informatique* qui dicte comment procéder.

Comment Pythonner

- Démarrez sur votre bureau le programme **Pyzo**.

Comment Pythonner

- Démarrez sur votre bureau le programme **Pyzo**.
- **Pyzo** est un environnement de développement pour le langage de programmation Python. Il inclut une **console** (ou *shell*, ou *interpréteur*), où sont entrées les instructions à exécuter immédiatement, et une zone où écrire des scripts.

Comment Pythonner

- Démarrez sur votre bureau le programme **Pyzo**.
- **Pyzo** est un environnement de développement pour le langage de programmation Python. Il inclut une **console** (ou *shell*, ou *interpréteur*), où sont entrées les instructions à exécuter immédiatement, et une zone où écrire des scripts.

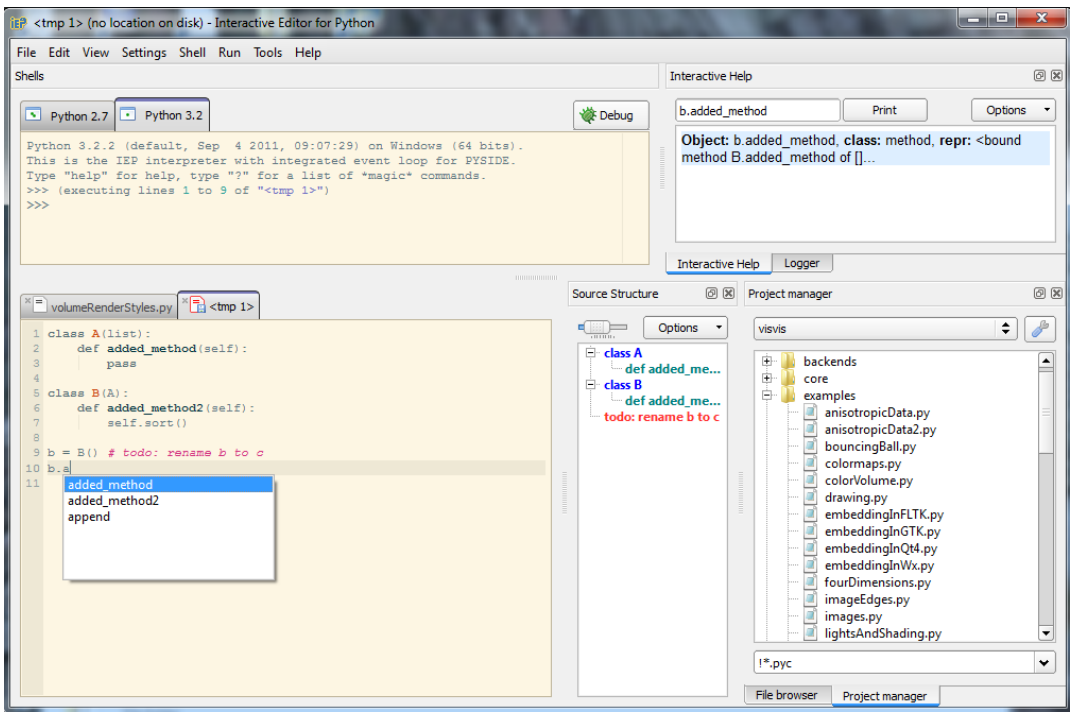

FIG.: Pyzo.

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer 2+2 dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer `2+2` dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Soustraction Entrer `4-3`. Vérifier que l'on trouve `1`.

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer `2+2` dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Soustraction Entrer `4-3`. Vérifier que l'on trouve `1`.

Produit Taper `2*3`. Vérifier que l'on trouve `6`.

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer `2+2` dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Soustraction Entrer `4-3`. Vérifier que l'on trouve `1`.

Produit Taper `2*3`. Vérifier que l'on trouve `6`.

Division Taper `12/3`. Vérifier que l'on trouve `4`. Taper `4/5`. Vérifier que l'on trouve `0.8`.

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer `2+2` dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Soustraction Entrer `4-3`. Vérifier que l'on trouve `1`.

Produit Taper `2*3`. Vérifier que l'on trouve `6`.

Division Taper `12/3`. Vérifier que l'on trouve `4`. Taper `4/5`. Vérifier que l'on trouve `0.8`.

...de CM1 Taper `13//4`, et `13%4`. Compléter la division posée suivante :

$$\begin{array}{r|l} 13 & 4 \\ \hline \ldots & \ldots \end{array}$$

Que remarquez-vous ?

Premiers pas

Exercice 1 Calculs élémentaires

On va commencer par se familiariser avec les opérations arithmétiques de base que peut faire Python.

Addition Entrer `2+2` dans la console, appuyer sur `Entrée` pour exécuter votre saisie. Vérifier que ça fait `4`.

Soustraction Entrer `4-3`. Vérifier que l'on trouve `1`.

Produit Taper `2*3`. Vérifier que l'on trouve `6`.

Division Taper `12/3`. Vérifier que l'on trouve `4`. Taper `4/5`. Vérifier que l'on trouve `0.8`.

...de CM1 Taper `13//4`, et `13%4`. Compléter la division posée suivante :

$$\begin{array}{r|l} 13 & 4 \\ \hline \ldots & \ldots \end{array}$$

Que remarquez-vous ?

Puissances Taper `2**3`. Quel est le résultat ? Et celui de `3**2` ?

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

- Créer une variable x qui a la valeur 3. Est-ce qu'on peut écrire `3=x` ?

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

- Créer une variable x qui a la valeur 3. Est-ce qu'on peut écrire `3=x` ?
- Créer une variable y qui a la valeur $7x$. Modifier la valeur de x. Cela modifie-t-il la valeur de y ?

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

- Créer une variable x qui a la valeur 3. Est-ce qu'on peut écrire `3=x` ?
- Créer une variable y qui a la valeur $7x$. Modifier la valeur de x. Cela modifie-t-il la valeur de y ?
- Rajouter 1 à x en la réaffectant, vérifier en évaluant. Même question avec 0.5.

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

- Créer une variable x qui a la valeur 3. Est-ce qu'on peut écrire `3=x` ?
- Créer une variable y qui a la valeur $7x$. Modifier la valeur de x. Cela modifie-t-il la valeur de y ?
- Rajouter 1 à x en la réaffectant, vérifier en évaluant. Même question avec 0.5.
- Que font `x+=1`, `x-=1`, `x*=2` ou encore `x/=2` ?

Mais Python est bien plus puissant qu'une simple *Casio collège*...

Exercice 2 Variables

Une *variable* est la donnée d'un emplacement mémoire où une valeur est stockée, et d'un nom.
Le fait d'affecter une valeur à une variable s'appelle une « *affectation* ».
En Python, on fait ça avec une expression de la forme `x=...`
On peut également réaffecter une variable en recyclant simplement son nom.
Pour afficher la valeur d'une variable, on peut demander à Python de l'évaluer en tapant son nom.

- Créer une variable x qui a la valeur 3. Est-ce qu'on peut écrire `3=x` ?
- Créer une variable y qui a la valeur $7x$. Modifier la valeur de x. Cela modifie-t-il la valeur de y ?
- Rajouter 1 à x en la réaffectant, vérifier en évaluant. Même question avec 0.5.
- Que font `x+=1`, `x-=1`, `x*=2` ou encore `x/=2` ?
- Entrer l'instruction `x,y=y,x`. Quelles sont alors les valeurs de x et de y ?

Remarque Dans un langage de programmation un peu plus *kasher*, on déclare et on affecte séparément une variable. Mais Python permet de déclarer une variable directement par affectation.

Remarque Dans un langage de programmation un peu plus *kasher*, on déclare et on affecte séparément une variable. Mais Python permet de déclarer une variable directement par affectation.

À propos des types

La fonction `type`, appliquée à une variable x via l'instruction `type(x)`, renvoie le type de la variable x. Essayez sur plusieurs valeurs : `type(0)`, `type(0.5)`, `type(type)`, et `type(0.)`.

Remarque Dans un langage de programmation un peu plus *kasher*, on déclare et on affecte séparément une variable. Mais Python permet de déclarer une variable directement par affectation.

À propos des types

La fonction `type`, appliquée à une variable x via l'instruction `type(x)`, renvoie le type de la variable x. Essayez sur plusieurs valeurs : `type(0)`, `type(0.5)`, `type(type)`, et `type(0.)`.

Les types `int` et `float` servent respectivement à représenter les nombres entiers et les nombres réels à virgule dans Python. On peut faire les opérations arithmétiques vues plus haut avec elles, même quand les variables en jeu sont d'un type et de l'autre.

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (`'`) ou doubles (`"`).
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Exercice 3 Chaînes de caractères

- Définir une variable `Bonjour` prenant la valeur 0.

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Exercice 3 Chaînes de caractères

- Définir une variable `Bonjour` prenant la valeur 0.
- Que fait `print(Bonjour)` ? Et les instructions `print("Bonjour")` et `print('Bonjour')` ?

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Exercice 3 Chaînes de caractères

- Définir une variable `Bonjour` prenant la valeur 0.
- Que fait `print(Bonjour)` ? Et les instructions `print("Bonjour")` et `print('Bonjour')` ?
- Comparer `type(Bonjour)`, `type('Bonjour')` et `type("Bonjour")`.

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Exercice 3 Chaînes de caractères

- Définir une variable `Bonjour` prenant la valeur 0.
- Que fait `print(Bonjour)` ? Et les instructions `print("Bonjour")` et `print('Bonjour')` ?
- Comparer `type(Bonjour)`, `type('Bonjour')` et `type("Bonjour")`.
- Évaluer `len("Bonjour")`. À quoi cela correspond-t-il ?

Chaînes de caractères

Généralités

Un nouveau type de variable important : le type `str`, pour *string* ou *chaîne de caractères*, en français. On peut les définir explicitement en incluant son message entre guillemets simples (') ou doubles (").
Il faut faire attention : quand on démarre par une guillemet d'un type, l'apparition d'une autre guillemet du même type clôt la chaîne.

Exercice 3 Chaînes de caractères

- Définir une variable `Bonjour` prenant la valeur 0.
- Que fait `print(Bonjour)` ? Et les instructions `print("Bonjour")` et `print('Bonjour')` ?
- Comparer `type(Bonjour)`, `type('Bonjour')` et `type("Bonjour")`.
- Évaluer `len("Bonjour")`. À quoi cela correspond-t-il ?
- Afficher `J'aime la tartiflette` et `Il dit : "Bonjour !"`. Attention à utiliser des types de guillemets différents !

Chaînes de caractères
Opérations

Exercice 3 (Suite)

- Que se passe-t-il quand on affiche une chaîne contenant `\n` ?

Chaînes de caractères
Opérations

Exercice 3 (Suite)

- Que se passe-t-il quand on affiche une chaîne contenant `\n` ?
- Définir deux chaines de caractères `x` et `y`: que fait `print(x, y)` ? Est-ce que ça marche aussi si `x` est une variable de type `int` ?

Chaînes de caractères
Opérations

Exercice 3 (Suite)

- Que se passe-t-il quand on affiche une chaîne contenant `\n` ?
- Définir deux chaines de caractères `x` et `y`: que fait `print(x, y)` ? Est-ce que ça marche aussi si `x` est une variable de type `int` ?
- Évaluer la valeur de `x+y`, l'afficher via `print`.

Chaînes de caractères
Opérations

Exercice 3 (Suite)

- Que se passe-t-il quand on affiche une chaîne contenant `\n` ?
- Définir deux chaines de caractères `x` et `y`: que fait `print(x, y)` ? Est-ce que ça marche aussi si `x` est une variable de type `int` ?
- Évaluer la valeur de `x+y`, l'afficher via `print`.
- Afficher « `1/100 est petit` » en remplaçant `1/100` par sa valeur. On utilisera `"{} est petit".format()` où `x` est la valeur voulue.

Premiers programmes

On va maintenant basculer sur la zone d'écriture de scripts. Un programme est une succession d'instructions qui sont effectuées lorsqu'il est exécuté.

Premiers programmes

On va maintenant basculer sur la zone d'écriture de scripts. Un programme est une succession d'instructions qui sont effectuées lorsqu'il est exécuté.

Exercice 4

- Écrire un programme qui stocke la chaîne « `Bonjour` » dans une variable `x`, puis l'affiche, puis affiche « `Bonjour Bonjour` ».

Premiers programmes

On va maintenant basculer sur la zone d'écriture de scripts. Un programme est une succession d'instructions qui sont effectuées lorsqu'il est exécuté.

Exercice 4

- Écrire un programme qui stocke la chaîne « `Bonjour` » dans une variable `x`, puis l'affiche, puis affiche « `Bonjour Bonjour` ».
- Écrire un programme qui demande à l'utilisateur d'entrer un texte, affiche « `Vous avez entré :` » suivi du texte entré. On utilisera la fonction `input`:

  ```
  texte = input()
  ```

  demande, à son exécution, une chaîne de caractère à l'utilisateur, puis la stocke dans la variable `texte`.

Premiers programmes

On va maintenant basculer sur la zone d'écriture de scripts. Un programme est une succession d'instructions qui sont effectuées lorsqu'il est exécuté.

Exercice 4

- Écrire un programme qui stocke la chaîne « `Bonjour` » dans une variable `x`, puis l'affiche, puis affiche « `Bonjour Bonjour` ».
- Écrire un programme qui demande à l'utilisateur d'entrer un texte, affiche « `Vous avez entré :` » suivi du texte entré. On utilisera la fonction `input`:

  ```
  texte = input()
  ```

  demande, à son exécution, une chaîne de caractère à l'utilisateur, puis la stocke dans la variable `texte`.
- Écrire un programme qui demande à l'utilisateur un nombre, le stocke dans une variable `x`, puis affiche « `f(x) =` » suivi de la valeur de $\frac{1}{1+x^2}$. On pourra utiliser `input`, convertir son entrée (initialement de type `str`) en un entier via le constructeur `int`.

Structures conditionnelles

Sémantique

On peut demander à un programme de traiter différemment ses données selon les valeurs des variables introduites. Pour cela, on utilise une structure conditionnelle, qui prend la forme :

```
if (condition):
        < traitement des données >
else:
        < traitement des données >
```

le `else` étant optionnel si on ne désire rien faire si la condition n'est pas vérifiée.

Structures conditionnelles

Sémantique

On peut demander à un programme de traiter différemment ses données selon les valeurs des variables introduites. Pour cela, on utilise une structure conditionnelle, qui prend la forme :

```
if (condition):
        < traitement des données >
else:
        < traitement des données >
```

le `else` étant optionnel si on ne désire rien faire si la condition n'est pas vérifiée.
La condition est une expression de type `bool` pour *booléen*, nommé après le mathématicien et logicien George Boole. Une variable de type `bool` prend deux valeurs : `True` (*Vrai*) et `False` (*Faux*).

Structures conditionnelles

Booléens

Parmi les façons de construire des conditions booléennes, les plus courantes sont celles qui comparent les variables entre elles. Pour cela, on utilise les tests logiques de la table suivante :

Structures conditionnelles

Booléens

Parmi les façons de construire des conditions booléennes, les plus courantes sont celles qui comparent les variables entre elles. Pour cela, on utilise les tests logiques de la table suivante :

Égalité	`x==y`
Différent	`x !=y`
Inférieur ou égal	`x<=y`
Inférieur strictement	`x<y`
Supérieur ou égal	`x>=y`
Supérieur strictement	`x>y`

Structures conditionnelles

Booléens

Parmi les façons de construire des conditions booléennes, les plus courantes sont celles qui comparent les variables entre elles. Pour cela, on utilise les tests logiques de la table suivante :

Égalité	`x==y`
Différent	`x !=y`
Inférieur ou égal	`x<=y`
Inférieur strictement	`x<y`
Supérieur ou égal	`x>=y`
Supérieur strictement	`x>y`

Exemple

Définir une variable `x` égale à 3. Quel est le type de l'expression `x==3` ? L'évaluer. Que donne l'évaluation de `x==2` ? Celle de `x<=4` ? Celle de `x<3` ? Celle de `x !=2` ?

Structures conditionnelles

Opérations sur les booléens

On peut combiner deux booléens `a` et `b` pour faire des opérations logiques en utilisant les opérateurs suivants :

Structures conditionnelles

Opérations sur les booléens

On peut combiner deux booléens `a` et `b` pour faire des opérations logiques en utilisant les opérateurs suivants :

Négation	`not(a)`
Conjonction (« et »)	`a and b`
Disjonction (« ou »)	`a or b`

Structures conditionnelles

Opérations sur les booléens

On peut combiner deux booléens `a` et `b` pour faire des opérations logiques en utilisant les opérateurs suivants :

Négation	`not(a)`
Conjonction (« et »)	`a and b`
Disjonction (« ou »)	`a or b`

Exercice 5

Écrire un programme qui demande deux nombres `x` et `y` à l'utilisateur et affiche s'ils sont tous les deux strictement positifs (c'est-à-dire s'ils sont > 0).

Exercice 6

Écrire un programme qui demande à l'utilisateur deux nombres x et y, et affiche « `Maximum de x et y =  ` » suivi du maximum de x et y (le plus grand des deux).