

Python : le tout début

leçon de choses

Stéphane Gonnord

stephane@gonnord.org

www.mp933.fr

Lycée du parc - Lyon

Mardi 22 avril 2014 - Luminy

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Langages interprétés ou compilés

- ▶ Les interprétés :
 - ▶ Lisp
 - ▶ Caml (souvent)
 - ▶ *Maple*
 - ▶ Python

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

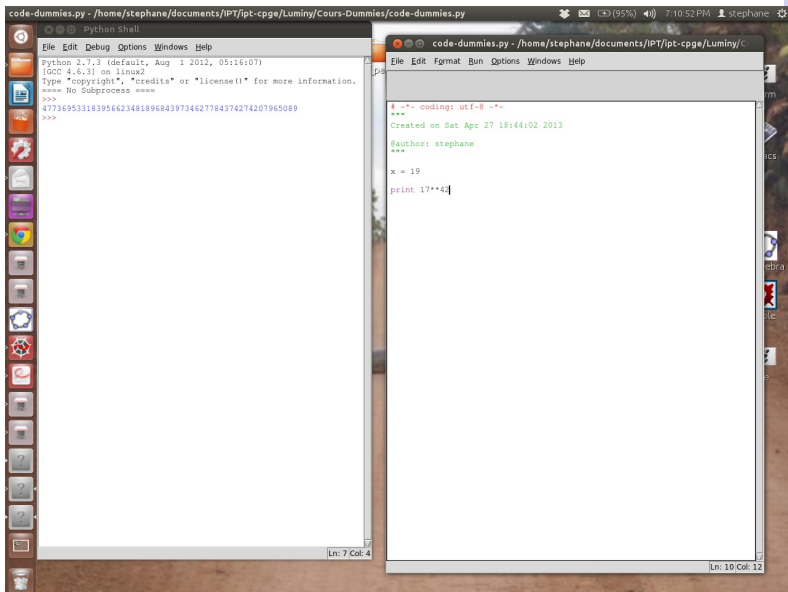
Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Langages interprétés ou compilés

- ▶ Les interprétés :
 - ▶ Lisp
 - ▶ Caml (souvent)
 - ▶ *Maple*
 - ▶ Python
- ▶ Les compilés :
 - ▶ Pascal
 - ▶ C
 - ▶ Caml (si on le souhaite)
 - ▶ *Java* (virtual machine)



Idle

- ▶ Avantages :
 - ▶ très léger ;
 - ▶ existe pour Python 2 **et Python 3** ;
 - ▶ prise en main instantannée

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Idle

- ▶ Avantages :
 - ▶ très léger ;
 - ▶ existe pour Python 2 **et Python 3** ;
 - ▶ prise en main instantannée
- ▶ Inconvénients :
 - ▶ interface trop légère voire déficiente (pour ouvrir un nouveau fichier par exemple) ;
 - ▶ ALT P/N pour visiter l'historique des commandes ; merci bien !
 - ▶ **j'aime pas !**

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

- Entiers
- Les flottants
- Les listes
- Les tuples
- Les chaînes de caractère
- Les ensembles

Les structures de contrôle

- Les boucles
- Les tests

Des programmes

- Factorielle
- Exponentiation
- Fibonacci
- Maximum d'une liste
- Somme des éléments d'une liste

The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script with the following content:

```
1 # -*- coding: utf-8 -*-
2 ...
3 Created on Sat Apr 27 18:44:02 2013
4
5 @author: stephane
6 ...
7
8 x = 19
9
10 print 17**42
11
```

The console window shows the output of the script:

```
4773695331839566234818968439734627784374274207965089
>>>
```

The variable explorer window shows the variable 'x' with the value 19:

Name	Type	Size	Value
x	int	1	19

The bottom status bar shows the following information:

Permissions: RW | End-of-lines: LF | Encoding: UTF-8 | Line: 11 | Column: 1

▶ Avantages :

- ▶ bien designé (à mon goût), sexy ;
- ▶ historique des commandes par UP/DOWN ;
- ▶ aides à la volée sur les fonctions utilisées ;
- ▶ méthodes d'objets proposées à la volée ;
- ▶ **suivi des variables très** pratique ;
- ▶ débogueur bien fichu (il paraît !).

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

- ▶ **Avantages :**
 - ▶ bien designé (à mon goût), sexy ;
 - ▶ historique des commandes par UP/DOWN ;
 - ▶ aides à la volée sur les fonctions utilisées ;
 - ▶ méthodes d'objets proposées à la volée ;
 - ▶ **suivi des variables très** pratique ;
 - ▶ débogueur bien fichu (il paraît !).
- ▶ **Inconvénients :**
 - ▶ **pas encore porté sous Python 3**, mais...

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

- ▶ **Avantages :**
 - ▶ bien designé (à mon goût), sexy ;
 - ▶ historique des commandes par UP/DOWN ;
 - ▶ aides à la volée sur les fonctions utilisées ;
 - ▶ méthodes d'objets proposées à la volée ;
 - ▶ **suivi des variables très** pratique ;
 - ▶ débogueur bien fichu (il paraît !).
- ▶ **Inconvénients :**
 - ▶ **pas encore porté sous Python 3**, mais... Ben si, c'est porté, maintenant !
 - ▶ pénible (lent) au démarrage ;
 - ▶ sous-fenêtres au comportement parfois mystérieux !

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Bon, tout ça c'est bien gentil, mais...

je tape dans quelle fenêtre ?

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Bon, tout ça c'est bien gentil, mais...

je tape dans quelle fenêtre ?

- ▶ Dans l'éditeur :
 - ▶ les fonctions/programmes ;
 - ▶ toutes les opérations que je veux pouvoir sauver ;
 - ▶ copy/paste de résultats (interpréteur) commentés.

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Bon, tout ça c'est bien gentil, mais...

je tape dans quelle fenêtre ?

- ▶ Dans l'éditeur :
 - ▶ les fonctions/programmes ;
 - ▶ toutes les opérations que je veux pouvoir sauver ;
 - ▶ copy/paste de résultats (interpréteur) commentés.
- ▶ Dans l'interpréteur :
 - ▶ des petits calculs n'ayant pas vocation à être sauvés ;
 - ▶ les tests d'un programme écrit par ailleurs ;
 - ▶ une suite d'instructions pour comprendre différents comportements.

Exemple :

The screenshot shows a Python IDE with a code editor on the left and a console window on the right. The code editor contains a Python script with a factorial function and a call to it. The console window shows the output of the script, including the factorial of 10 and 40, and the value of a variable x.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Apr 27 18:44:02 2013
4
5 @author: stephane
6 """
7
8 10*30
9
10 x = 19
11
12 print 17**42
13
14 def facto(n):
15     if n == 0: return 1
16     return n*facto(n-1)
17
18 #>>> facto(40)
19 #815915283247897734345611269596115894272000000000L
20
```

```
4773695331839566234818968439734627784374274207965089
>>> facto(10)
3628800
>>> x
19
>>> facto(40)
815915283247897734345611269596115894272000000000L
>>> |
```

Name	Type	Size	
x	int	1	19

Entiers (1/3)

```
>>> 125 * 2 + 5 % 3
```

```
252
```

```
>>> 5**3
```

```
125
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Entiers (1/3)

```
>>> 125 * 2 + 5 % 3
```

```
252
```

```
>>> 5**3
```

```
125
```

```
>>> 5^3
```

```
6
```

```
>>> 42^13
```

```
39
```

Alors, vous en pensez quoi ?

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Entiers (2/3)

► En Python 2 :

```
>>> 17**42
```

```
4773695331839566234818968439734627784374274207965089
```

```
>>> print 17**42
```

```
4773695331839566234818968439734627784374274207965089
```

```
>>> print(17**42)
```

```
4773695331839566234818968439734627784374274207965089
```

Entiers (2/3)

► En Python 2 :

```
>>> 17**42
4773695331839566234818968439734627784374274207965089L
>>> print 17**42
4773695331839566234818968439734627784374274207965089
>>> print(17**42)
4773695331839566234818968439734627784374274207965089
```

► En Python 3 :

```
>>> 17**42
4773695331839566234818968439734627784374274207965089
>>> print 17**42
SyntaxError: invalid syntax
>>> print(17**42)
4773695331839566234818968439734627784374274207965089
```

Entiers (3/3)

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>

Entiers (3/3)

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>
<pre>>>> 7. / 3 2.3333333333333335</pre>	<pre>>>> 7. / 3 2.3333333333333335</pre>

Entiers (3/3)

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>
<pre>>>> 7. / 3 2.3333333333333335</pre>	<pre>>>> 7. / 3 2.3333333333333335</pre>
<pre>>>> 7 // 3 2</pre>	<pre>>>> 7 // 3 2</pre>

Entiers (3/3)

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>
<pre>>>> 7. / 3 2.3333333333333335</pre>	<pre>>>> 7. / 3 2.3333333333333335</pre>
<pre>>>> 7 // 3 2</pre>	<pre>>>> 7 // 3 2</pre>
<pre>>>> 7. // 3 2.0</pre>	<pre>>>> 7. // 3 2.0</pre>

Entiers (3/3)

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>
<pre>>>> 7. / 3 2.3333333333333335</pre>	<pre>>>> 7. / 3 2.3333333333333335</pre>
<pre>>>> 7 // 3 2</pre>	<pre>>>> 7 // 3 2</pre>
<pre>>>> 7. // 3 2.0</pre>	<pre>>>> 7. // 3 2.0</pre>

Vous trouvez ça pénible ?

Entiers (3/3)

Python 2	Python 3
<pre>>>> 7 / 3 2</pre>	<pre>>>> 7 / 3 2.3333333333333335</pre>
<pre>>>> 7. / 3 2.3333333333333335</pre>	<pre>>>> 7. / 3 2.3333333333333335</pre>
<pre>>>> 7 // 3 2</pre>	<pre>>>> 7 // 3 2</pre>
<pre>>>> 7. // 3 2.0</pre>	<pre>>>> 7. // 3 2.0</pre>

Vous trouvez ça pénible ?

Ben oui...

Flottants (1/3)

```
>>> 1. / 7  
0.14285714285714285
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Flottants (1/3)

```
>>> 1. / 7  
0.14285714285714285
```

Au fait, est-ce que Python connaît π ?

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Flottants (1/3)

```
>>> 1. / 7  
0.14285714285714285
```

Au fait, est-ce que Python connaît π ?

```
>>> pi  
Traceback (most recent call last):  
  File "<pyshell#33>", line 1, in <module>  
    pi  
NameError: name 'pi' is not defined  
>>> import math  
>>> math.pi  
3.141592653589793  
>>> math.cos(math.pi)  
-1.0
```

Flottants (2/3)

```
>>> 1. / 3**6  
0.0013717421124828531  
>>> (1. / 3**6) * 3**6  
0.9999999999999999
```

Alors que...

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Flottants (2/3)

```
>>> 1. / 3**6
0.0013717421124828531
>>> (1. / 3**6) * 3**6
0.9999999999999999
```

Alors que...

```
>>> for i in range(1000):
        if (1./2**i) * 2**i <> 1. : print(i)
>>>
```

Flottants (2/3)

```
>>> 1. / 3**6
0.0013717421124828531
>>> (1. / 3**6) * 3**6
0.9999999999999999
```

Alors que...

```
>>> for i in range(1000):
        if (1./2**i) * 2**i <> 1. : print(i)
>>>
```

Attention aux tests d'égalité de flottants...

Flottants (2/3)

Un peu de maple...

```
> for i to 40 do if 2**i*(1./2**i) <> 1.  
then print(i) fi od;
```

34

38

>

Alors que...

Flottants (2/3)

Un peu de maple...

```
> for i to 40 do if 2**i*(1./2**i) <> 1.  
                then print(i) fi od;
```

34

38

>

Alors que...

```
> for i to 1000 do if 10**i*(1./10**i) <> 1.  
                 then print(i) fi od;
```

>

Conclusion ?

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (1/2)

```
>>> l = [5, 3, 42.42, 1, 'tagada']
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (1/2)

```
>>> l = [5, 3, 42.42, 1, 'tagada']
```

```
>>> l[1]
```

```
3
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (1/2)

```
>>> l = [5, 3, 42.42, 1, 'tagada']
```

```
>>> l[1]
```

```
3
```

```
>>> len(l)
```

```
5
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (1/2)

```
>>> l = [5, 3, 42.42, 1, 'tagada']
```

```
>>> l[1]
```

```
3
```

```
>>> len(l)
```

```
5
```

```
>>> l[5]
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#44>", line 1, in <module>
```

```
    l[5]
```

```
IndexError: list index out of range
```

Listes (1/2)

```
>>> l = [5, 3, 42.42, 1, 'tagada']
```

```
>>> l[1]
```

```
3
```

```
>>> len(l)
```

```
5
```

```
>>> l[5]
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#44>", line 1, in <module>
```

```
    l[5]
```

```
IndexError: list index out of range
```

```
>>> l[-1]
```

```
'tagada'
```

Listes (2/2)

Rappel : `l = [5, 3, 42.42, 1, 'tagada']`

```
>>> l[1:4]
[3, 42.42, 1]
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (2/2)

Rappel : `l = [5, 3, 42.42, 1, 'tagada']`

```
>>> l[1:4]
[3, 42.42, 1]
```

```
>>> l[2:-2]
```

Vous avez suivi ?

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Listes (2/2)

Rappel : `l = [5, 3, 42.42, 1, 'tagada']`

```
>>> l[1:4]
[3, 42.42, 1]
```

```
>>> l[2:-2]
```

Vous avez suivi ?

```
[42.42]
```

Listes (2/2)

Rappel : `l = [5, 3, 42.42, 1, 'tagada']`

```
>>> l[1:4]
[3, 42.42, 1]
```

```
>>> l[2:-2]
```

Vous avez suivi ?

```
[42.42]
```

```
>>> l[2] = 1515
```

```
>>> l
[5, 3, 1515, 1, 'tagada']
```

Les listes sont **mutables**

Tuples

```
>>> t = 3, 'plof', 5.3, 4
```

```
>>> t[2]
```

```
5.3
```

```
>>> t[2] = 3
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#62>", line 1, in <module>
```

```
    t[2] = 3
```

```
TypeError: 'tuple' object does not support  
                                item assignment
```

Les tuples sont **non-mutables**

Tuples

```
>>> t = 3, 'plof', 5.3, 4
```

```
>>> t[2]
```

```
5.3
```

```
>>> t[2] = 3
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#62>", line 1, in <module>
```

```
    t[2] = 3
```

```
TypeError: 'tuple' object does not support  
item assignment
```

Les tuples sont **non-mutables**

```
>>> t = 1, 15
```

```
>>> a, b = t
```

```
>>> b
```

```
15
```

Chaînes

```
>>> c = 'plof'
```

```
>>> d = "plof"
```

```
>>> c == d
```

```
True
```

```
>>> len(c)
```

```
4
```

```
>>> c[2]
```

```
'o'
```

```
>>> c[1:3]
```

```
'lo'
```

```
>>> c[1] = 'a'
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#77>", line 1, in <module>
```

```
    c[1] = 'a'
```

```
TypeError: 'str' object does not support item assignment
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Ensembles

```
>>> s = {3, 4.5, 3, 'toto'}
>>> len(s)
3
>>> s
set([4.5, 3, 'toto'])
>>> 3 in s, 'truc' in s
(True, False)
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Ensembles

```
>>> s = {3, 4.5, 3, 'toto'}
>>> len(s)
3
>>> s
set([4.5, 3, 'toto'])
>>> 3 in s, 'truc' in s
(True, False)
```

Mais aussi : les ensembles sont **itérables** :

Ensembles

```
>>> s = {3, 4.5, 3, 'toto'}
>>> len(s)
3
>>> s
set([4.5, 3, 'toto'])
>>> 3 in s, 'truc' in s
(True, False)
```

Mais aussi : les ensembles sont **itérables** :

```
>>> for x in s: print(x)
4.5
3
toto
```


Boucles (1/2)

```
>>> for i in range(3):  
    print(i)
```

0

1

2

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Boucles (1/2)

```
>>> for i in range(3):  
    print(i)
```

0

1

2

```
>>> for i in range(3,5):  
    print(i)
```

3

4

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Boucles (1/2)

```
>>> for i in range(3):  
    print(i)
```

0

1

2

```
>>> for i in range(3,5):  
    print(i)
```

3

4

```
>>> for i in range(10,25,5):  
    print(i)
```

10

15

20

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Boucles (2/2)

```
>>> for i in range(2,5):  
        for j in range(2,5):  
            print(i,j)
```

(2, 2)

(2, 3)

(2, 4)

(3, 2)

(3, 3)

(3, 4)

(4, 2)

(4, 3)

(4, 4)

L'environnement de travail

- Interprétation vs compilation
- Spyder vs Idle
- Dans quelle fenêtre travailler ?

Les types de base

- Entiers
- Les flottants
- Les listes
- Les tuples
- Les chaînes de caractère
- Les ensembles

Les structures de contrôle

- Les boucles**
- Les tests

Des programmes

- Factorielle
- Exponentiation
- Fibonacci
- Maximum d'une liste
- Somme des éléments d'une liste

Tests (1/2)

```
>>> if 2013%2 == 1:  
        print('Test 1 réussi')  
Test 1 réussi
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Tests (1/2)

```
>>> if 2013%2 == 1:  
    print('Test 1 réussi')
```

```
Test 1 réussi
```

```
>>> if ((2013%2 == 1) and (2013 > 10**4))\  
    or (2013 < 0):  
    print('Test 2 réussi')
```

```
>>>
```

Tests (1/2)

```
>>> if 2013%2 == 1:
        print('Test 1 réussi')
```

```
Test 1 réussi
```

```
>>> if ((2013%2 == 1) and (2013 > 10**4))\
        or (2013 < 0):
        print('Test 2 réussi')
```

```
>>>
```

```
>>> if ((2013%2 == 1) and (2013 > 10**4))\
        or (2013 < 0):
        print('Test 2 réussi')
    else:
        print('Test 2 loupé')
```

```
Test 2 loupé
```

Tests (2/2)

```
>>> for b in range(2,7):  
    for k in range(15):  
        if b**k*(1./b**k) <> 1. :  
            print(b,k)
```

(3, 6)

(3, 9)

(3, 10)

(5, 11)

(6, 6)

(6, 9)

(6, 10)

La fonction factorielle (1/2)

► Version **récursive** :

```
def factol(n):  
    if n == 0 : return 1  
    return n * factol(n-1)  
  
>>> factol(5)  
120
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

La fonction factorielle (1/2)

► **Version récursive :**

```
def factol(n):  
    if n == 0 : return 1  
    return n * factol(n-1)  
>>> factol(5)  
120
```

► **Version itérative :**

```
def facto2(n):  
    res=1  
    for k in range(2,n+1):  
        res = res*k  
    return res  
>>> facto2(5)  
120
```

La fonction factorielle (2/2) - reprenons !

```
def facto3(n):  
    res=1  
    for k in range(2,n+1):  
        res = res*k  
    return res  
  
>>> facto3(5)  
2
```

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

La fonction factorielle (2/2) - reprenons !

```
def facto3(n):  
    res=1  
    for k in range(2,n+1):  
        res = res*k  
    return res  
  
>>> facto3(5)  
2
```

WTF ?

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

La fonction factorielle (2/2) - reprenons !

```
def facto3(n):  
    res=1  
    for k in range(2,n+1):  
        res = res*k  
    return res
```

```
>>> facto3(5)  
2
```

WTF ?

```
def facto2(n):  
    res=1  
    for k in range(2,n+1):  
        res = res*k  
    return res
```

```
>>> facto2(5)  
120
```

Calculer x^n (1/3)

► Une façon astucieuse

```
def pow1(x,n):  
    return x**n  
  
>>> pow1(2,10)  
1024
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer x^n (1/3)

- ▶ Une façon astucieuse

```
def pow1(x,n):  
    return x**n  
  
>>> pow1(2,10)  
1024
```

- ▶ Plus laborieux - **récurif**

```
def pow2(x,n):  
    if n==0 : return 1  
    return x * pow2(x,n-1)  
  
>>> pow2(2,10)  
1024
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer x^n (2/3) - itératif

```
def pow3(x,n):  
    res=1  
    for k in range(n):  
        res = res*x  
    return res  
  
>>> pow3(2,10)  
2
```

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer x^n (2/3) - itératif

```
def pow3(x,n):  
    res=1  
    for k in range(n):  
        res = res*x  
    return res  
  
>>> pow3(2,10)  
2
```

HAHAHAHAHAHA!

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer x^n (2/3) - itératif

```
def pow3(x,n):  
    res=1  
    for k in range(n):  
        res = res*x  
    return res
```

```
>>> pow3(2,10)  
2
```

HAHAHAHAHAHA!

```
def pow4(x,n):  
    res=1  
    for k in range(n):  
        res = res*x  
    return res
```

```
>>> pow4(2,10)  
1024
```

Calculer x^n (3/3) - exponentiation rapide

```
def pow5(x, n):  
    if n == 0: return 1  
    if n %2 ==0: return pow5(x, n//2)**2  
    return x*pow5(x, (n-1)//2)**2  
  
>>> pow5(2,10)  
1024
```

Python : le tout début

Stéphane Gonnord

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer x^n (3/3) - exponentiation rapide

```
def pow5(x, n):  
    if n == 0: return 1  
    if n %2 ==0: return pow5(x, n//2)**2  
    return x*pow(x, (n-1)//2)**2  
  
>>> pow5(2,10)  
1024
```

```
def pow6(x,n):  
    if n == 0: return 1  
    res, p, N= 1,x, n  
    while N>1:  
        if N%2 == 1: res = res*p  
        p = p*p  
        N = N//2  
    return res*p  
  
>>> pow6(2,10)  
1024
```

Calculer f_n (1/3)

```
def fibol(n):  
    if n <= 1: return n  
    return fibol(n-1)+fibol(n-2)  
  
>>> fibol(10)  
55
```

Trop fastoche...

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer f_n (1/3)

```
def fibol(n):  
    if n <= 1: return n  
    return fibol(n-1)+fibol(n-2)  
  
>>> fibol(10)  
55
```

Trop fastoche...

```
import time  
def chrono(n):  
    t0 = time.time()  
    _ = fibol(n)  
    return time.time()-t0  
  
>>> [chrono(n) for n in range(28,34)]  
[0.30255794525146484, 0.48483800888061523, 0.784882068634  
1.261225938796997, 2.0136120319366455, 3.2120370864868164
```

Calculer f_n (2/3)

```
def fibo2(n):  
    if n <= 1 : return n  
    fk, fkp1 = 0, 1  
    for k in range(1,n):  
        fk, fkp1 = fkp1, fk+fkp1  
    return fkp1  
  
>>> fibo2(10)  
55
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer f_n (2/3)

```
def fibo2(n):  
    if n <= 1 : return n  
    fk, fkp1 = 0, 1  
    for k in range(1,n):  
        fk, fkp1 = fkp1, fk+fkp1  
    return fkp1
```

```
>>> fibo2(10)  
55
```

```
def fibo3(n):  
    if n <=1 : return n  
    val = [0,1]+[0]*(n-1)  
    for k in range(2,n+1):  
        val[k] = val[k-1]+val[k-2]  
    return val[-1] # héhé...
```

```
>>> fibo3(10)  
55
```

L'environnement de travail

Interprétation vs compilation

Spyder vs Idle

Dans quelle fenêtre travailler ?

Les types de base

Entiers

Les flottants

Les listes

Les tuples

Les chaînes de caractère

Les ensembles

Les structures de contrôle

Les boucles

Les tests

Des programmes

Factorielle

Exponentiation

Fibonacci

Maximum d'une liste

Somme des éléments d'une liste

Calculer f_n (3/3)

```
import numpy as np
def fibo4(n):
    return np.linalg.matrix_power(\
        np.array([[0,1],[1,1]]),n)[1,0]
>>> fibo4(10)
55
```

L'environnement de travail

- Interprétation vs compilation
- Spyder vs Idle
- Dans quelle fenêtre travailler ?

Les types de base

- Entiers
- Les flottants
- Les listes
- Les tuples
- Les chaînes de caractère
- Les ensembles

Les structures de contrôle

- Les boucles
- Les tests

Des programmes

- Factorielle
- Exponentiation
- Fibonacci**
- Maximum d'une liste
- Somme des éléments d'une liste

Calculer f_n (3/3)

```
import numpy as np
def fibo4(n):
    return np.linalg.matrix_power(\
        np.array([[0,1],[1,1]]),n)[1,0]
>>> fibo4(10)
55
```

Coût logarithmique ?

Calculer f_n (3/3)

```
import numpy as np
def fibo4(n):
    return np.linalg.matrix_power(\
        np.array([[0, 1], [1, 1]]), n) [1, 0]
>>> fibo4(10)
55
```

Coût logarithmique ? **NON**

Calculer f_n (3/3)

```
import numpy as np
def fibo4(n):
    return np.linalg.matrix_power(\
        np.array([[0,1],[1,1]]),n)[1,0]
>>> fibo4(10)
55
```

Coût logarithmique ? **NON**

```
>>> fibo4(600)
-849170400
```

CARAMBA !

Calculer f_n (3/3)

```
import numpy as np
def fibo4(n):
    return np.linalg.matrix_power(\
        np.array([[0,1],[1,1]]),n)[1,0]
>>> fibo4(10)
55
```

Coût logarithmique ? **NON**

```
>>> fibo4(600)
-849170400
```

CARAMBA !

```
>>> fibo3(600)
110433070572952242346432246767718285942590237357555606380
```

Maximum des éléments d'une liste

► Fastoche :

```
def max1(l):  
    prov = l[0]  
    for i in range(1, len(l)):  
        if l[i] > prov: prov = l[i]  
    return prov  
  
>>> max1([12, 42, 2, 17])  
42
```

Maximum des éléments d'une liste

► Fastoche :

```
def max1(l):  
    prov = l[0]  
    for i in range(1, len(l)):  
        if l[i] > prov: prov = l[i]  
    return prov
```

```
>>> max1([12, 42, 2, 17])  
42
```

► Encore plus facile :

```
>>> max([12, 42, 2, 17])  
42
```

Somme des éléments d'une liste

► Fastoche :

```
def sum1(l):  
    s=0  
    for i in range(len(l)): s = s+l[i]  
    return s  
>>> sum1([12,42,2,17])  
73
```


Somme des éléments d'une liste

► Fastoche :

```
def sum1(l):  
    s=0  
    for i in range(len(l)): s = s+l[i]  
    return s  
>>> sum1([12,42,2,17])  
73
```

► Mais il y a (un peu) plus élégant...

```
def sum2(l):  
    s=0  
    for x in l: s = s+x  
    return s  
>>> sum2([12,42,2,17])  
73
```

Somme des éléments d'une liste

► Fastoche :

```
def sum1(l):  
    s=0  
    for i in range(len(l)): s = s+l[i]  
    return s  
>>> sum1([12,42,2,17])  
73
```

► Mais il y a (un peu) plus élégant...

```
def sum2(l):  
    s=0  
    for x in l: s = s+x  
    return s  
>>> sum2([12,42,2,17])  
73
```

► Et il y a plus simple !

```
>>> sum([12,42,2,17])  
73
```