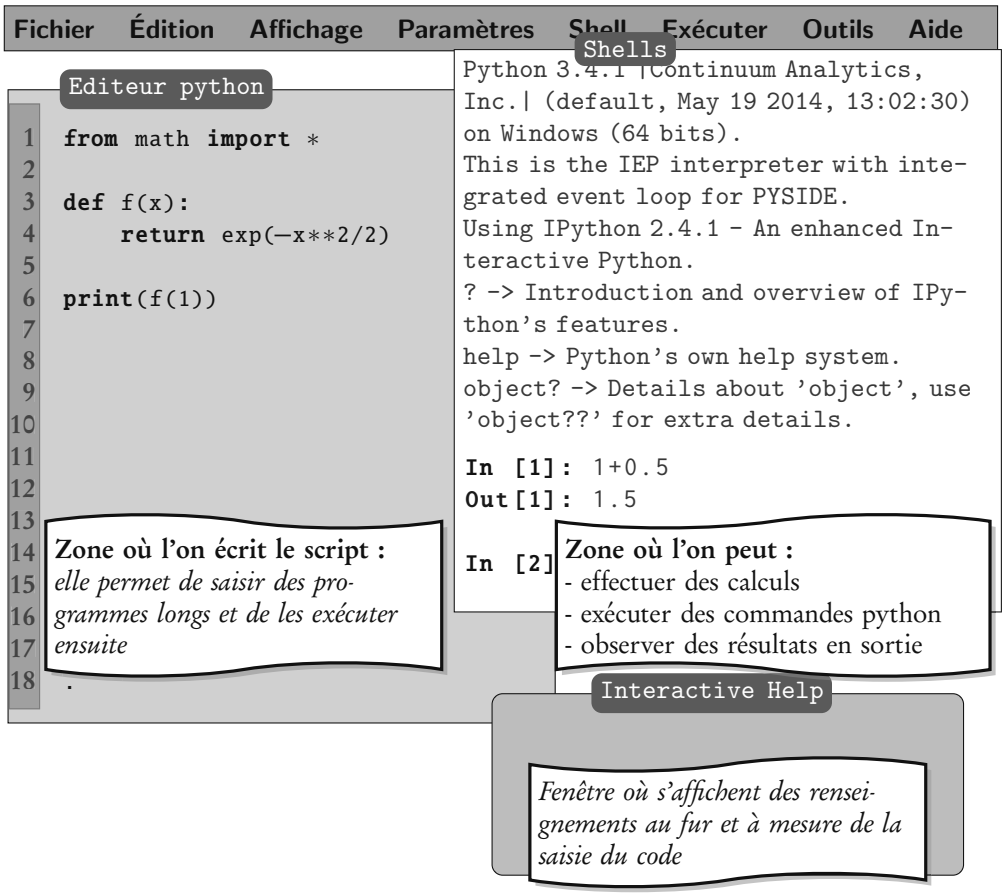
A Présentation et Installation

1 Téléchargement et installation

Nous avons choisi d'utiliser le logiciel PYZO pour programmer en PYTHON. On recommande de le télécharger à l'adresse suivante : `http ://www.pyzo.org/downloads.html`. PYZO est disponible pour WINDOWS, LINUX et OSX. Il convient de choisir la dernière version 64 bits.

2 Quelques principes sur Pyzo

PYZO présente trois panneaux :



Pour ouvrir : un fichier nommé `mon_programme.py`, il faut :

- d'abord ouvrir PYZO.
- puis cliquer sur `Fichier` puis `Ouvrir`. Reste ensuite à choisir `mon_programme.py`.

Pour exécuter : un algorithme saisi dans l'éditeur, on clique sur `Exécuter`, puis dans le menu déroulant `Exécuter le fichier`.

Pour stoper : un algorithme en cours d'exécution, on peut cliquer sur le symbole  *(situé dans le bandeau du Shell).*

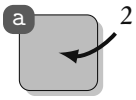
B Instructions élémentaires

1 Instruction d'affectation

Pour garder en mémoire le nombre 2, on affecte le nombre 2 à une variable `a` en écrivant l'instruction : `a=2`

Remarques :

- `a` est une variable.
- On peut voir `a` comme une boîte vide dans laquelle on stocke le nombre 2 :



- On dit aussi que `a` prend la valeur 2.

2 Opérations de base

Opérations	Symboles	Exemples
Addition	+	`In [1]: 3+5` `Out[1]: 8`
Soustraction	-	`In [2]: 13-4` `Out[2]: 9`
Multiplication	*	`In [3]: 7*5` `Out[3]: 35`
Puissance	**	`In [4]: 3**4` `Out[4]: 81`
Division	/	`In [5]: 7/2` `Out[5]: 3.5`
Reste de la division euclidienne	%	`In [6]: 7%2` `Out[6]: 1`
Quotient de la division euclidienne	//	`In [7]: 7//2` `Out[7]: 3`

Opérateurs relationnels	$=$	$\neq$	$>$	$<$	$\leqslant$	$\geqslant$
Avec PYTHON	==	!=	>	<	<=	>=

3 Entrée - sortie

1 Instruction d'entrée avec la commande input()

Objectif

On souhaite saisir en mémoire un nombre dans une case mémoire a.

Façons possibles :

Langage naturel	PYTHON
Saisir en mémoire l'**entier** a	`a=int(input())`
Afficher `'Donner la valeur de a :'` ET saisir en mémoire l'**entier** a	`a=int(input('Donner la valeur de a :'))`
Saisir en mémoire le **réel** a	`a=float(input())`
Afficher `'Donner la valeur de a :'` ET saisir en mémoire le **réel** a	`a=float(input('Donner la valeur de a :'))`

Remarque : la saisie de la valeur a se fera dans l'ÉDITEUR PYTHON ou la SHELL de PYTHON.

2 Instruction de sortie avec la commande print()

Langage naturel	PYTHON
`Affichage d'une valeur numérique :`	
Afficher « ***a*** »	`print(a)`
`Affichage d'un texte :`	
Afficher « `la valeur de a est :` »	`print('la valeur de a est :')`
`Affichage d'un texte avec apostrophe :`	
Afficher « `l'entier` »	`print("l'entier")`
`Affichage d'un texte et d'une variable numérique a :`	
Afficher « `la valeur de a est :a` »	`print('la valeur de a est :',a)`

4 Instruction conditionnelle

Langage naturel	Code Python

Si condition **Alors**
  instructions
Sinon
  autres instructions
FinSi
	`if condition :`
	`  instructions`
	`else:`
	`  autres intructions`

Ce bloc est facultatif

Exemple : *test de parité*

Langage naturel	Code Python

Entrée : Saisir l'entier a
Si le reste de la division de a par 2 est nul **Alors**
  Afficher « a est pair »
Sinon
  Afficher « a est impair »
FinSi
	`a=int(input("saisir l'entier a : "))`
	`if a%2==0:`
	`  print('a est pair')`
	`else:`
	`  print('a est impair')`

5 Instructions itératives

1 Boucle For

Langage naturel	Code Python

Pour chaque valeur de k parcourant l'ensemble des entiers $[\![m,n]\!]$ **Faire**
  instructions
FinPour
Pour k **allant de** m **à** n **Faire**
  instructions
FinPour
	`for k in range(m,n+1):`
	`  instructions`

A noter :

– $[\![m,n]\!]$ désigne l'ensemble des entiers naturels compris entre m et n.

– `range(m,n+1)` contient les mêmes éléments que $[\![m;n]\!]$. C'est un ensemble que l'on peut parcourir, on dit qu'il est **itérable**.

Exemples :

Code Python	Correspond à l'ensemble
`range(10)`	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
`range(1,11)`	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
`range(1,10,2)`	1, 3, 5, 7, 9

Remarque : on peut utiliser à la place de k, les lettres i, j, n, m, p, q… Par commodité, ce sont les lettres souvent utilisées pour désigner une variable entière.

Exemple : *somme des* 100 *premiers carrés d'entiers* $\sum_{k=1}^{100} k^2$

Langage naturel	Code Python

Début
  S = 0
  Pour k **allant de** 1 **à** 100 **Faire**
  S = S + k^2
  FinPour
Fin
Sortie : Afficher S
	`S=0`
	`for k in range(1,101):`
	`  S=S+k**2`
	`print(S)`
	En sortie
	338350

2 Parcourir une liste de valeurs par « compréhension »

Définitions

- Une **liste** est un ensemble d'objets séparés par des virgules et mis entre [ ].
- On dit que L est une **liste vide** si `L=[]`.
- **Parcourir** une liste L par « **compréhension** » pour « **agir** » sur tous ses éléments signifie :

  Pour chaque élément t contenu dans la liste L **Faire**
  | « Agir » sur t
  FinPour

- Une liste est donc aussi **itérable**.

Exemple :

```
L=[1,4,9,12,21]
for t in L:
    print(t**2)
```

En sortie

```
1
16
81
144
441
```

3 Créer une liste de valeurs par « compréhension »

Principe

Supposons que T est une liste de nombres et f une fonction.

La liste Y des images est la liste de $\left\{ f(t) \text{ telle que } t \in \mathtt{T} \right\}$.

Elle peut être obtenue par « **compréhension** » :

Y=[$f(t)$ pour chaque élément t de la liste T]

Exemple :

```
X=[1,4,9,12,21]
Y=[x**2 for x in X]
print(Y)
```

En sortie

```
[1, 16, 81, 144, 441]
```

4 Boucle While

Langage naturel	Code Python
TantQue condition vraie **Faire** instructions **FinTantQue**	`while condition :` `  instructions`

Exemple : *division euclidienne* $\begin{array}{c|c} a & b \\ \hline r & q \end{array}$

Langage naturel	Code Python

Entrée :
∘ Saisir « le dividende a= : » a
∘ Saisir « le diviseur b= : » b
Début
  $q=0$
  TantQue $a \geqslant b$ **Faire**
  $a=a-b$
  $q=q+1$
  FinTantQue
Fin
Sortie : Afficher « le quotient de la division euclidienne de a par b est q= », q , « et son reste est r= », a
	`a=int(input('Saisir le dividende a : '))`
	`b=int(input('Saisir le diviseur b : '))`
	`q=0`
	`while a>=b:`
	`  a=a-b`
	`  q=q+1`
	`print('Le quotient de la division\`
	`euclidienne est q= ',q,' et`
	`son reste est r= ',a)`
	En sortie
	`Saisir le dividende a : 107`
	`Saisir le diviseur b : 6`
	`Le quotient de la division eucli-`
	`dienne est q= 17 et son reste est`
	`r= 5`

Remarque : par confort de saisie dans l'éditeur PYTHON, pour faire un retour à la ligne dans un texte saisi, il suffit d'insérer un antislash \. *(Voir l'algorithme précédent).*

C Modules utiles

- Les fonctions PYTHON sont organisées par modules.
- La bibliothèque STANDARD PYTHON (PYTHON STANDARD LIBRARY) est la collection de modules donnant accès aux fonctions de base.
- Un module doit être importé avant de pouvoir être utilisé.

1 Module math

Pour importer le module `math`, on saisit :

```
import math
```

Maintenant le module `math` peut être utilisé.
Question : comment savoir ce qu'il contient ?

Pour répondre à cette question, on peut saisir dans le SHELL les instructions suivantes :

1 – On invoque le module `math` :

```
In [8]: import math
```

2 – On peut saisir ensuite soit `dir(math)` soit `math.`.

Première façon :

```
In [9]: dir(math)
Out[9]:
[\'__doc__\', '__name__', '__package__',
'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'hypot',
'isinf', 'isnan', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'modf', 'pi', 'pow',
'radians', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'trunc']
```

Deuxième façon :

```
In [10]: math.
```

A ce moment, une fenêtre apparaît et propose la liste de toutes les fonctions contenues dans le module `math`. *Il ne reste plus qu'à choisir.*

Exemple :

```
1 import math
2 print(math.pi)
3 print(math.exp(2))
```

Ligne n° 1 : permet de charger le module `math`

Ligne n° 2 : utilise la commande `pi` contenue dans le module `math`

Ligne n° 3 : utilise la commande `exp` contenue dans le module `math`

En sortie

```
3.141592653589793
7.38905609893065
```

Remarque : en utilisant `import` $*$, on peut alléger l'écriture précédente :

```
1 from math import *
2 print(pi)
3 print(exp(2))
```

Ligne n° 1 : permet de charger le module `math`

Ligne n° 2 : la commande `math.pi` se réduit à `pi`

Ligne n° 3 : la commande `math.exp()` se réduit à `exp()`

En sortie

```
3.141592653589793
7.38905609893065
```

2 Module numpy

1 Son usage

Le module `numpy` permet de créer des TABLEAUX [1] *(`array` en anglais)* de **nombres** et de les gérer.
Pour l'invoquer, on saisit en préambule :

```
import numpy as np
```

Remarque : `np` est un surnom de `numpy`.

si on veut voir ce qu'il contient, on peut utiliser la commande `dir` [2] :

```
import numpy as np
print(dir(np))
```

Remarques : l'affichage en sortie est trop conséquent, il n'est donc pas retranscrit ici...

1. Un tableau est une liste de nombres « améliorée » dans lequel on peut opérer sur chaque élément qui le compose, voir page 13.
2. `dir` est l'abrégé de DIRECTORY, c'est-à-dire RÉPERTOIRE en anglais. Elle retourne l'information sur n'importe quelle variable en donnant par ordre alphabétique le nom des fonctions qui peuvent être utilisées.