



Notions essentielles (niveau 1)

1) Les différents types de données.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

125 -102 1253202210100121012101220120

1) Les différents types de données.

- ▶ Entiers
- ▶ **flottants**
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

15.6

.5

2.3e10

1e-5

1) Les différents types de données.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

True

False

1) Les différents types de données.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

"" "Python" 'Monty' "troupe d'humoristes"

1) Les différents types de données.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

[] [2, 3, 5] ["Pierre", "Paul", "Jacques"] [1, 'pi', [1, 2]]

1) Les différents types de données.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

2) Les variables.

- ▶ Noms
- ▶ Initialisation
- ▶ Affectation

A

Nom_1

Variable_2

List_aux

2) Les variables.

- ▶ Noms
- ▶ Initialisation
- ▶ Affectation

```
A = 12      Nom_1 = 'Paul'      Variable_2  = 3.14
```

```
List_aux   = ['Paul', 12 ]
```

2) Les variables.

- ▶ Noms
- ▶ Initialisation
- ▶ Affectation

```
List_aux = [A]          Nom_1 = Nom_1 + 'e'
```

```
Variable_2 = A * Variable_2      A = A**2
```

2) Les variables.

- ▶ Noms
- ▶ Initialisation
- ▶ Affectation

3) Entrées/sorties.

- ▶ `print`

- ▶ `input`

```
print("Hello World")
```

```
print(A)
```

```
print('Resultat', A + 6)
```

3) Entrées/sorties.

▶ `print`

▶ `input`

```
c = input('Entrez un texte :  ')
```

```
x = float(input('Entrez un nombre :  '))
```

```
liste = list(map(float, input("Entrez des nombres séparé par une virgule : ").split(",")))
```

3) Entrées/sorties.

► print

► input

```
print('Entrer les coefficients de  $aX^2+bX+c$ ')  
a = float(input('Entrer a : '))  
b = float(input('Entrer b : '))  
c = float(input('Entrer c : '))  
print('le discriminant vaut : ',  $b^2-4*a*c$ )
```

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

+

-

*

//

%

**

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

+ - * / **

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

`or` `and` `not`

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

+ *

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

+

*

4) Opérations.

- ▶ Entiers
- ▶ flottants
- ▶ booléens
- ▶ Chaines de caractères
- ▶ Listes

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

$$2 + 3 * 6$$

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

$$2 + 3 * 6 = 20$$

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

$$a / b + c$$

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

(*) 2 <= a < 8

4) Opérations.

Priorités des opérations :

	Opérateur(s)	Description	Ordre
1	()	Parenthèses	De l'int. vers l'ext.
2	**	Exponentiation	Droite à gauche
3	+x, -x	plus, moins	
4	*, /, //, %	Multiplication, division, modulo	Gauche à droite
5	+, -	Addition, soustraction	Gauche à droite
6	==, !=, <, <=, >, >=, in	Comparaisons	(*)
7	not	Négation logique	
8	and	ET logique	Gauche à droite
9	or	OU logique	Gauche à droite

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

`abs()`

`divmod()`

`bin()`

`type()`

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

`abs()`

`round()`

`int()`

`type()`

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

`len(t)`

`list(t)`

`t.lower()`

`t.upper()`

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

`len(L)` `sorted(L)`

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

`L.append()`

`L.sort()`

`L.remove()`

5) Fonctions sur les variables.

- ▶ Sur les entiers.
- ▶ Sur les flottants.
- ▶ Sur les chaines de caractères.
- ▶ Sur les listes.

6) Structures de contrôle.

- ▶ Branchement conditionnel.
- ▶ Boucle conditionnelle.
- ▶ Boucle itérative.

```
if <condition> :  
    bloc d'instructions 1  
else :  
    bloc d'instructions 2
```

6) Structures de contrôle.

- ▶ Branchement conditionnel.
- ▶ Boucle conditionnelle.
- ▶ Boucle itérative.

```
while <condition> :  
    bloc d'instructions répété
```

6) Structures de contrôle.

- ▶ Branchement conditionnel.
- ▶ Boucle conditionnelle.
- ▶ Boucle itérative.

```
for k in <iterable> :  
    bloc d'instructions répété
```

6) Structures de contrôle.

- ▶ Branchement conditionnel.
- ▶ Boucle conditionnelle.
- ▶ Boucle itérative.

7) Définition de fonctions.

- ▶ Définition.
- ▶ Paramètres, arguments
- ▶ variables locales/globales
- ▶ fonctions natives

```
def nom( <paramètres> ):  
    instructions  
    return <objet renvoyé>
```

7) Définition de fonctions.

- ▶ Définition.
- ▶ Paramètres, arguments
- ▶ variables locales/globales
- ▶ fonctions natives

```
def f(x, y):  
    z = x**2 + y**2  
    return z
```

```
>>> f(1, 2)
```

7) Définition de fonctions.

- ▶ Définition.
- ▶ Paramètres, arguments
- ▶ variables locales/globales
- ▶ fonctions natives

```
def f(x, y):  
    z = x**2 + y**2  
    return z  
z = 3  
print(f(1, 2))
```

7) Définition de fonctions.

- ▶ Définition.
- ▶ Paramètres, arguments
- ▶ variables locales/globales
- ▶ fonctions natives

Conversion	<code>int()</code> , <code>float()</code> , <code>str()</code> , <code>bool()</code> , <code>list()</code> , <code>set()</code>
Mathématiques	<code>abs()</code> , <code>pow()</code> , <code>round()</code> , <code>divmod()</code> , <code>max()</code> , <code>min()</code> , <code>sum()</code>
Itérables	<code>len()</code> , <code>range()</code> , <code>reversed()</code> , <code>sorted()</code>
Types et objets	<code>type()</code> , <code>id()</code>
Entrée/sortie	<code>print()</code> , <code>input()</code> , <code>open()</code>
Fonctions utiles	<code>any()</code> , <code>all()</code>

7) Définition de fonctions.

- ▶ Définition.
- ▶ Paramètres, arguments
- ▶ variables locales/globales
- ▶ fonctions natives

8) Modules.

- ▶ Importation.
- ▶ math
- ▶ numpy
- ▶ matplotlib.pyplot
- ▶ random

```
import math          import numpy as np
```

```
from math import cos,sin    from math import *
```

8) Modules.

- ▶ Importation.
- ▶ `math`
- ▶ `numpy`
- ▶ `matplotlib.pyplot`
- ▶ `random`

```
import math as m          m.sin()      m.exp()

m.log()      m.tan()      m.acos()      m.floor()

m.sqrt()      m.comb()      m.factorial()
```

8) Modules.

- ▶ Importation.
- ▶ `math`
- ▶ `numpy`
- ▶ `matplotlib.pyplot`
- ▶ `random`

```
import numpy as np      np.array()      np.zeros()

np.ones()      np.eye()      np.exp()      np.sin()

np.shape()      np.mean()      np.std()      np.polyfit()
```

8) Modules.

- ▶ Importation.
- ▶ math
- ▶ numpy
- ▶ matplotlib.pyplot
- ▶ random

```
import matplotlib.pyplot as plt          plt.plot()

plt.close('all')      plt.show()      plt.legend()

plt.fill()      plt.figure()      plt.errorbar()
```

8) Modules.

- ▶ Importation.
- ▶ math
- ▶ numpy
- ▶ matplotlib.pyplot
- ▶ random

```
import random as rd      rd.random()      rd.randint()

rd.choice()      rd.shuffle()      rd.sample()

rd.randrange()      rd.seed()      rd.gauss()
```

8) Modules.

- ▶ Importation.
- ▶ math
- ▶ numpy
- ▶ matplotlib.pyplot
- ▶ random

9) Exemples.

```
import math

def resoudre(a, b, c):
    delta = b**2 - 4*a*c
    if delta < 0:
        return 'Pas de racine reelle'
    if delta == 0:
        x = -b / (2*a)
        return 'Racine double : x = ' + str(x)
    x1 = (-b - math.sqrt(delta)) / (2*a)
    x2 = (-b + math.sqrt(delta)) / (2*a)
    return 'Deux racines : x1 = ' + str(x1) + ', x2 = ' + str(x2)

a = float(input('Entrer a : '))
b = float(input('Entrer b : '))
c = float(input('Entrer c : '))

print(resoudre(a, b, c))
```

9) Exemples.

```
import matplotlib.pyplot as plt

def f(x):
    if x < 0:
        return -x**2
    else:
        return x**2

a, b = -2, 2
N = 100
x = []
y = []
for k in range(N+1):
    t = a + k*(b-a)/N
    x.append(t)
    y.append(f(t))

plt.figure("Représentation graphique")
plt.plot(x, y, 'r')
plt.show()
```

9) Exemples.

```
import random as rd

def simul_X():
    """ simule la réalisation de X"""
    R = []
    for k in range(10):
        R.append(rd.randint(1,6))
    return sum(R)

N = 1000
s = 0
for k in range(N):
    s += simul_X()    # ce qui remplace s = s + simul_X()

print("La valeur moyenne de X est proche de : ", s/N)
```