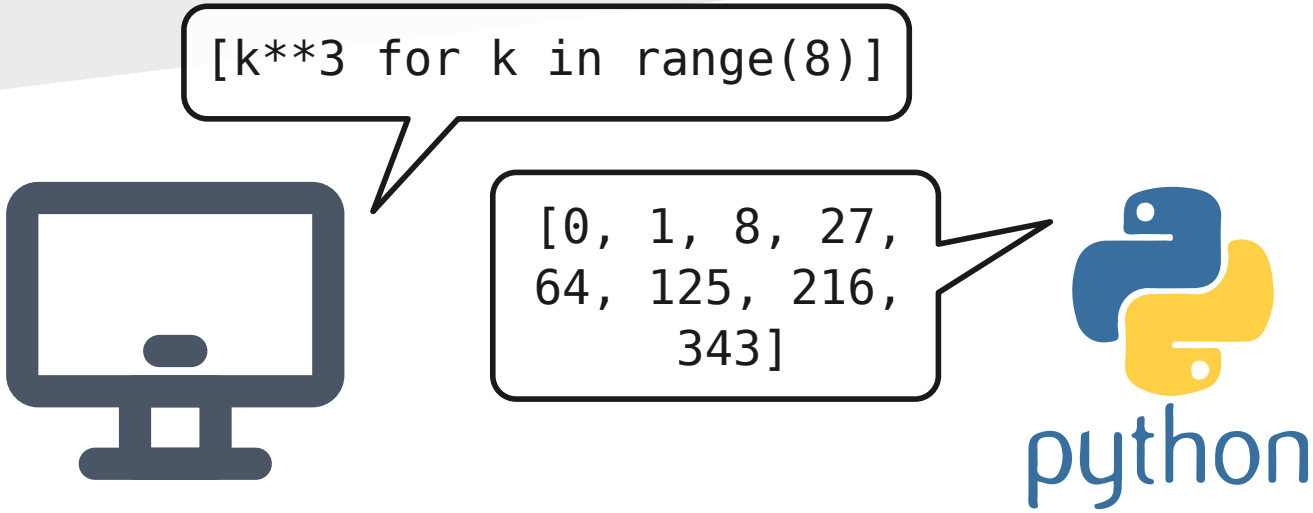
Python pour le supérieur

1) Bases en mode interactif



C’est parti !

- Un aperçu de Python
- Calculatrice
  - Fonctions de bases
  - Limites
- Variables
  - Entiers et flottants
  - Listes
  - Tableaux
- Deux modules scientifiques
  - Numpy
  - Sympy
- Ressources, solutions, etc.

Python

- Python /pi.tɔ̃/, langage de programmation créé en 1991 et régulièrement mis à jour.
- Très utilisé pour :
  - 🤖 l'intelligence artificielle
  - 📊 les *data sciences* et l'analyse de données
  - ⚙️ l'automatisation de tâches, l'administration de systèmes, la cybersécurité
  - 🌐 le développement web
  - 🔬 la recherche scientifique et l'ingénierie
  - 🎲 les loisirs, l'éducation et la création numérique.



Python : comment l'utiliser ?

En ligne :

- Chercher « python en ligne » / « python en ligne avec numpy »

Installation de l' « environnement Python »

- Présent par défaut sur MacOS et Linux.
- Pour Windows : quitter Microsoft pour Linux… 😊 ou installer Python en vous connectant sur `python.org`

Installation d'un logiciel

- Sert à taper le code et/ou à dialoguer.

Mode interactif / mode script

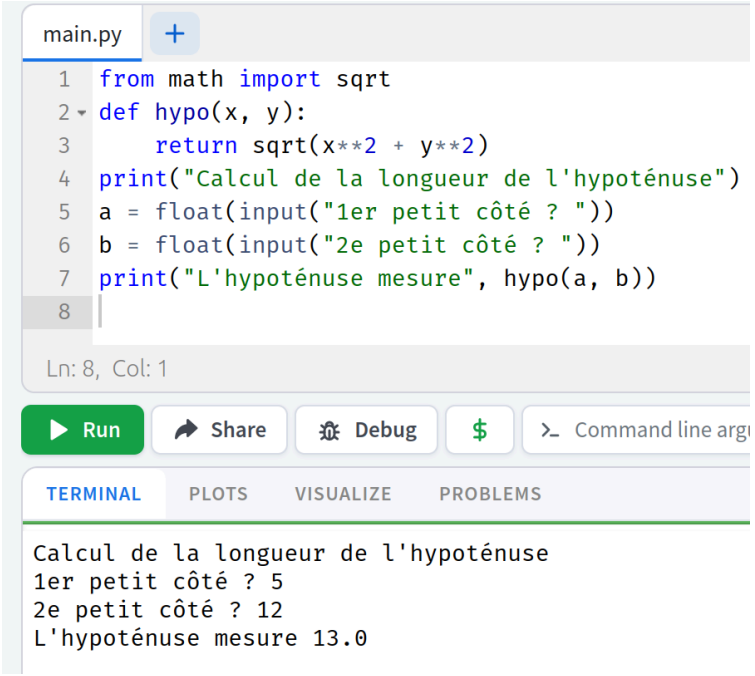
Mode interactif

- Une ou quelques lignes
- La commande est exécutée automatiquement
- Terminal de commande, Jupyter notebook, extensions…

```
Python 3.12.3 (main, Jun 19 2026, 12:46
Type "help", "copyright", "credits" or
>>> print("Salut tout le monde")
Salut tout le monde
>>> from math import *
>>> sqrt(5)
2.23606797749979
>>> 5/7.
0.7142857142857143
>>> round(5/7,1)
0.7
>>>
```

Mode script

- Programmes de toute taille
- Le script doit être lancé
- EduPython, Pyzo, VSCodium, …



Calculatrice : fonctions de base

```
>>> 2 + 3 * 5      # 17
>>> 16 / 3         # 5.333333333333333
>>> 16 // 3        # quotient = 5 (division euclidienne)
>>> 16 % 3         # reste = 1
>>> 10**4 - 1      # 9999
>>> 2**(1/2)       # 1.4142135623730951
```

```
>>> abs(-5.3)          # valeur absolue = 5.3
>>> round(19.725)      # arrondi = 20
>>> round(19.725, 2)   # arrondi à 10^-2 = 19.73
>>> int(19.725)        # troncature = 19  (transforme en entier)
```

```
>>> 1j**2      # i² = -1
    (-1+0j)
```

Calculatrice : le module `math`

```
>>> log(10)
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    NameError: name 'log' is not defined
```

```
>>> from math import *
```

```
>>> log(100)         # 4.605170185988092
>>> log(100, 10)     # 2.0
>>> sin(pi/2)        # 1.0
>>> acos(-1)         # 3.141592653589793
>>> sqrt(2)          # 1.4142135623730951
>>> e**4             # 54.59815003314423
>>> exp(400)         # 5.2214696897641 44e+173
>>> floor(-4.5)      # -5
>>> ceil(-4.5)       # -4
```

Calculatrice : les limitations

Les nombres précédents sont de type « flottants » (*float*).

```
>>> 0.1 + 0.1 + 0.1 - 0.3
    5.551115123125783e-17
```

En binaire, 0,1 s'écrit comme une fraction périodique infinie :

$$0{,}1_{10} = 0{,}0\overline{0011}_2 = 0{,}0001100110011001100110011\ldots_2$$

- `0.1 ≈ 0.1000000000000000055511151231257 8...`
- `0.3 ≈ 0.2999999999999999888977697537484 3...`

```
>>> from decimal import Decimal
>>> Decimal('0.1') + Decimal('0.1') + Decimal('0.1') -
    Decimal('0.3')
    Decimal('0.0')
```

Les variables numériques

```
>>> a = 6
>>> a = a + 3   # a = 9
>>> a += 3      # a = 12
>>> a = a // 3  # a = 4
>>> a = a / 2   # a = 2.0
```

```
>>> a = 6
>>> a + 3       # affiche 9 sans affecter a
```

```
>>> b, c = 10, 8
>>> b*c, b/c, abs(c-b)  # (80, 1.25, 2)
```

```
>>> b, c = 10, 8
>>> b, c = c, b+c       # (8, 18)
```

Les booléens (*boolean*)

```
>>> a = 5       # affectation
>>> b = -3
>>> a == 5      # test d'égalité
    True
>>> b != -2     # test de différence (non égalité)
    True
>>> a >= 10     # supérieur ou égal
    False
```

```
>>> not a>0            # non
    False
>>> a>0 and b<0        # et
    True
>>> a>0 and b>0
    False
>>> a>0 or b>0         # ou
    True
>>> (a>0)^(b<0)        # ou exclusif (xor)
    False
```

Les chaines de caractères (*string*)

```
>>> s = "Bonjour"     # ou s = 'Bonjour'
>>> s[0]      # 'B'
>>> s[-1]     # 'r'
>>> s[2:5]    # 'njo'
>>> s * 2     # 'BonjourBonjour'
```

```
>>> s = s + " tout le monde"
>>> s
    'Bonjour tout le monde'
>>> s.replace("on", "eau")
    'Beaujour tout le meaude'
>>> s.upper()
    'BONJOUR TOUT LE MONDE'
>>> s.split(" ")
    ['Bonjour', 'tout', 'le', 'monde']
```

Exercices

1) Calculer l'arrondi de $\varphi = \dfrac{1+\sqrt{5}}{2}$ à 10^{-5} près.
2) Calculer l'arrondi de $e - \left(1 + \dfrac{1}{100}\right)^{100}$ à 10^{-6} près.
3) Donner trois instructions d'une ligne chacune qui vérifie que :
   a) un entier `n` est pair.
   b) `a` et `b`, supposés non nuls, sont de même signe.
   c) trois flottants `p`, `q` et `r` sont deux à deux distincts.
   d) une chaine de caractères `s` est tout en majuscules.
4) Un **palindrome** est un mot qui se lit dans les deux sens, comme « kayak » ou « radar ». Écrire une instruction d'une ligne qui permet de vérifier qu'un mot de 5 lettres est un palindrome.

Les listes (*list*)

```
>>> L = [5, -2, 9]
>>> L * 2
    [5, -2, 9, 5, -2, 9]
>>> L + 14
    Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
    TypeError: can only concatenate list (not "int") to list
>>> L + [14]
    [5, -2, 9, 14]
>>> L
    [5, -2, 9]
>>> L.append(14)    # ou L += [14]
>>> L
    [5, -2, 9, 14]
>>> L[1]
    -2
>>> L[0:2]    # de l'indice 0 (compris) à 2 (non compris)
    [5, -2]
```

Les listes

```
>>> L
    [5, -2, 9, 14]
>>> len(L)
    4
>>> del(L[1])
>>> -2 in L
    False
>>> sum(L), min(L), max(L)
    (28, 5, 14)
```

```
>>> M = [7, -1, 4, 6, 4, -4]
>>> M.count(4)
    2
>>> M.sort()           # [-4, -1, 4, 4, 6, 7]
>>> M.remove(6)        # [-4, -1, 4, 4, 7]
>>> M.insert(3, 9)     # [-4, -1, 4, 9, 4, 7]
>>> M.reverse()        # [7, 4, 9, 4, -1, -4]
```

Les listes

```
>>> C = [x**2 for x in range(11)]
>>> C
    [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> C = [x+2 for x in C]
>>> C
    [2, 3, 6, 11, 18, 27, 38, 51, 66, 83, 102]
>>> C = [10*x for x in C]
>>> C
    [20, 30, 60, 110, 180, 270, 380, 510, 660, 830, 1020]
```

```
>>> I = ["farine", "oeuf", "lait", "sucre"]
>>> J = I
>>> I[3] = "cassonade"      # ou I[-1] = "cassonade"
>>> J
    ['farine', 'oeuf', 'lait', 'cassonade']
```

Pour copier une liste, sans la pointer : `J = I.copy()`

Les vecteurs avec numpy

```
>>> import numpy as np
```

```
>>> V = np.array([5, -3, 4])
>>> V + 10      # [15,  7, 14]
>>> V * 2       # [10, -6,  8]
>>> V**2        # [25,  9, 16]
>>> np.sqrt(V)  # [2.23606798, nan, 2. ]
>>> np.exp(V)
    # array([1.48413159e+02, 4.97870684e-02, 5.45981500e+01])
>>> V @ V       # 50
```

```
>>> np.zeros(5)     # [0., 0., 0., 0., 0.]
>>> np.ones(6)      # [1., 1., 1., 1., 1., 1.]
>>> np.full(4, 13)  # [13, 13, 13, 13]
>>> np.arange(1, 16, 3)   # [ 1,  4,  7, 10, 13]
>>> np.linspace(0, 1, 5)      # [0., 0.25, 0.5 , 0.75, 1.]
```

Indicateurs statistiques

```
>>> L = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 100])
>>> L.size          # 10
>>> L.mean()        # 14.5  (moyenne)
>>> np.median(L)    # 5.5   (médiane)
>>> L.std()         # 28.605069480775605   (écart-type)
>>> L.min()         # 1
>>> np.percentile(L, 25)    # 3.25   (1er quartile)
>>> np.percentile(L, 75)    # 7.75   (3e quartile)
>>> L.max()         # 100
>>> sum(L)          # 145
```

Remarque : La fonction `describe` du module `pandas` regroupe tous les indicateurs en une seule fonction.

Les matrices avec numpy

```
>>> A = np.array([[5, 1, 2], [-4, 13, 0], [10, -3, 3]])
>>> I = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
>>> A*I
    array([[ 5,  0,  0],
           [ 0, 13,  0],
           [ 0,  0,  3]])
```

$$A = \begin{pmatrix} 5 & 1 & 2 \\ -4 & 13 & 0 \\ 10 & -3 & 3 \end{pmatrix}$$

```
>>> A@I                    # ou np.dot(A, I)
    array([[ 5,  1,  2],
           [-4, 13,  0],
           [10, -3,  3]])
```

Attention :

- `A**2` est égal à `A*A` (donc généralement différent de A^2)
- A^2 s'obtient avec `A@A` ou `np.linalg.matrix_power(A, 2)`

Calcul matriciel avec numpy

Fonction	Usage
`np.transpose(A)`	Transposée d'une matrice
`np.linalg.inv(A)`	Inverse d'une matrice carrée
`np.linalg.det(A)`	Déterminant d'une matrice carrée
`np.linalg.solve(A, B)`	Résolution d'un système d'équations linéaires $AX = B$
`np.linalg.eig(A)`	Calcul des valeurs propres et vecteurs propres d'une matrice carrée
`np.linalg.norm(x)`	Calcul de la norme d'un vecteur ou d'une matrice
`np.linalg.qr(A)`	Décomposition QR d'une matrice
`np.linalg.cholesky(A)`	Décomposition de Cholesky pour matrices symétriques définies positives

Exercices

1) Calculer la valeur approchée de $\displaystyle\sum_{k=0}^{10\,000} \frac{1}{k+1} - \ln(10\,000)$

2) Soit $A = \begin{pmatrix} 3 & 1 & 2 \\ 6 & 4 & 5 \\ 9 & 7 & 8 \end{pmatrix}$. Trouver une matrice de permutation `P` telle que `A@P = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

3) Compléter `L = [... for k in range(12)]` pour obtenir la liste `[1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0]`

4) On définit trois variables `a, b, n`.
   Compléter `M = [... for k in range(n)]` pour que la liste `M` ait les mêmes valeurs que `np.linspace(a, b, n)`
   Tester avec `a, b, n = 0, 100, 26`

Le module sympy

Étude de la fonction $x \mapsto x^3 - 2x + 1$

```
>>> from sympy import *
>>> x = symbols('x')
>>> f = x**3 - 2*x + 1
>>> factor(f)
    (x - 1)*(x**2 + x - 1)
>>> solve(f)
    [1, -1/2 + sqrt(5)/2, -sqrt(5)/2 - 1/2]
>>> integrate(f)
    x**4/4 - x**2 + x
>>> integrate(f, (x, 0, 5))
    545/4
>>> diff(f)
    3*x**2 - 2
```

Remarque. Si l'expression a deux indéterminées, on précise celle utilisée (exemple : `diff(g, y)`)

Le module sympy

Décomposition en éléments simples :

$$\frac{3x-2}{(x+1)(x+2)} = \frac{8}{x+2} - \frac{5}{x+1}$$

```
>>> apart( (3*x-2) / ((x+1)*(x+2)) )
    8/(x + 2) - 5/(x + 1)
```

Regroupement sous le même dénominateur :

$$\frac{7}{x+2} - \frac{2}{x+7} = \frac{5(x+9)}{(x+2)(x+7)}$$

```
>>> together( 7/(x+2) - 2/(x+7) )
    5*(x + 9)/((x + 2)*(x + 7))
```

Conclusion

Quelques conseils :

- Apprendre les syntaxes de base par cœur.
- Connaitre les fonctionnalités plus que les noms des fonctions.
- Savoir où retrouver les syntaxes moins courantes.

Ressources :

- Alain Busser, *Python pour les (futurs) professeurs*, Ellipses, 2025.
- Julien Guillod, *Programmation Python par la pratique* (2e édition), Dunod, 2024. `https://python.guillod.org/fr/`

Solutions

1) `phi = (1+sqrt(5))/2` si le module `math` a été importé.
   `phi = 1.618033988749895`

2) `round(e-(1+1/100)**100, 6) = 0.013468`

3) a) `n%2 == 0`
   b) `a*b > 0`
   c) `p!=q and q!=r and r!=p`
   d) `s.upper() == s`

4) `s[0]==s[-1] and s[1]==s[-2]`

Solutions

1) `sum([1/(k+1) for k in range(10000)])-log(10000)`
   `0.577265640681983`

2) `K = np.array([[0, 0, 1], [1, 0, 0], [0, 1, 0]])`

3) `L = [int(cos(k*pi/2)) for k in range(12)]`

4) `M = [a+k*(b-a)/(n-1) for k in range(n)]`

```
[0.0, 4.0, 8.0, 12.0, 16.0, 20.0, 24.0, 28.0, 32.0, 36.0,
40.0, 44.0, 48.0, 52.0, 56.0, 60.0, 64.0, 68.0, 72.0,
76.0, 80.0, 84.0, 88.0, 92.0, 96.0, 100.0]
```