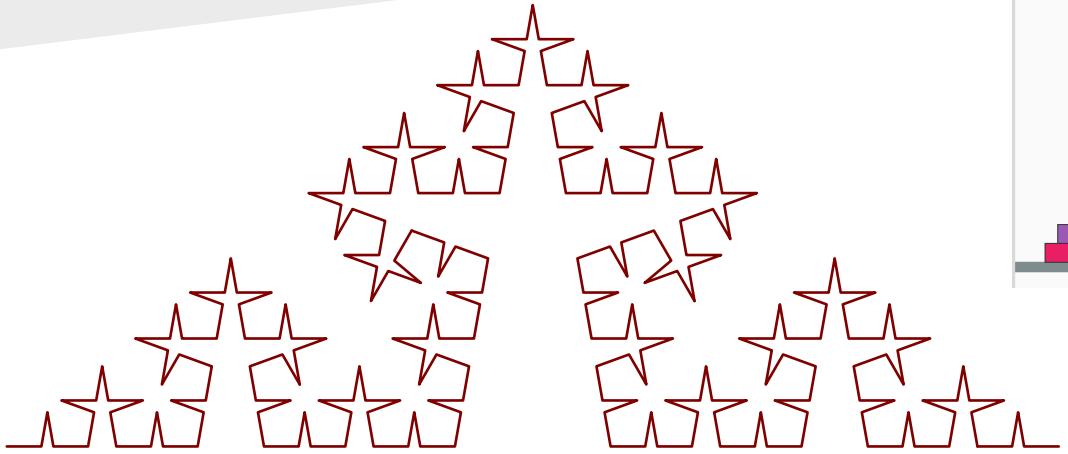
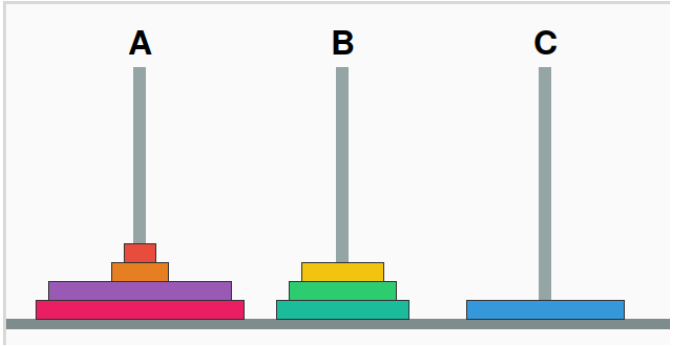
Python pour le supérieur

2) Bases en mode script



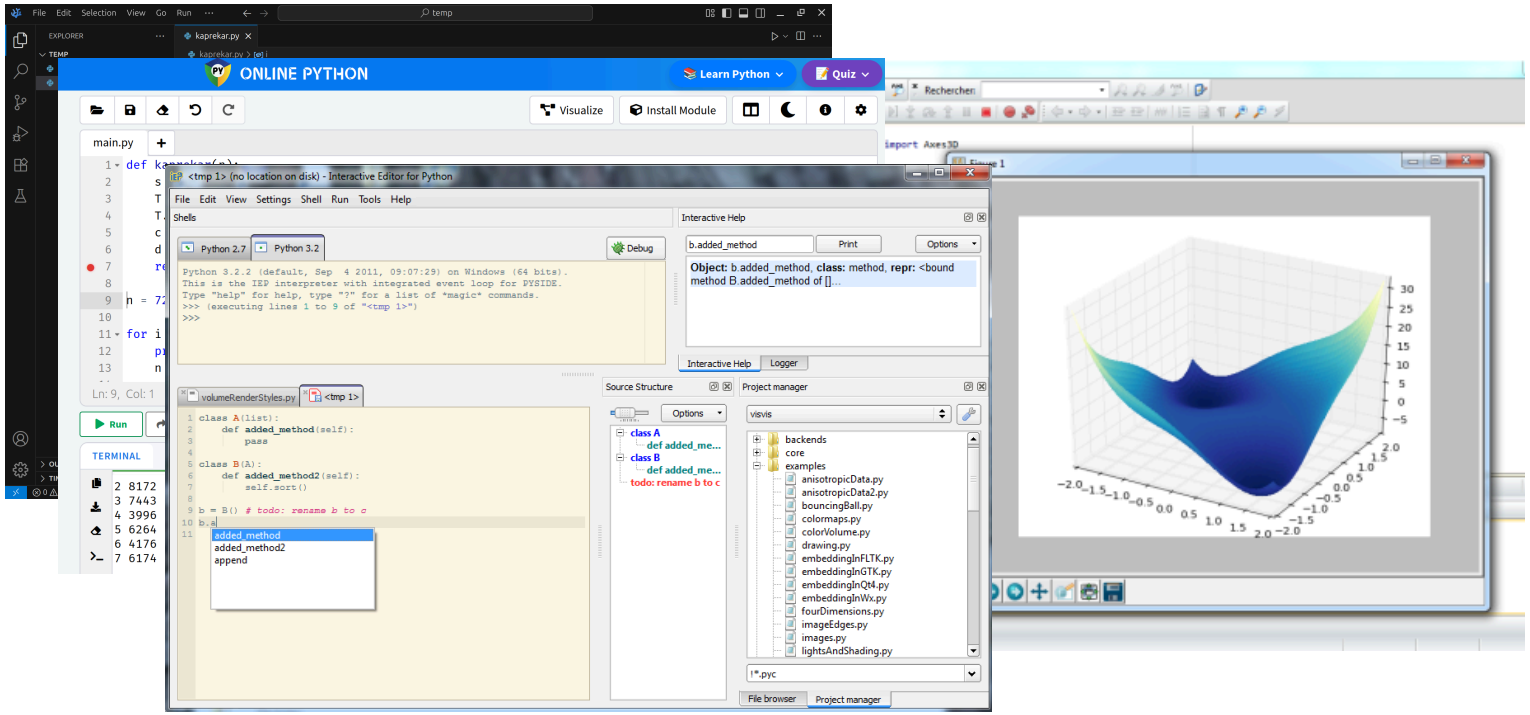


Mode script

- Définir et utiliser une fonction
- Instructions conditionnelles `if`, `while`, `for`
- Premières applications
  - Convertisseur binaire-décimal
  - Algorithme de Kaprekar
- Récursivité
  - Principe et fonctions standards
  - Points de vigilance

Mode script

- Utilisé de préférence pour écrire des algorithmes supérieurs à plusieurs lignes.
- IDLE : EduPython, Pyzo, VSCodium, … « Python en ligne »



Définition de fonctions

Exemple 1 : Pythagore

```
1  from math import *
2
3  def hypo(x, y):
4      return sqrt(x**2 + y**2)
5
6  print("Calcul de la longueur de l'hypoténuse")
7  a = float(input("1er petit côté ? "))
8  b = float(input("2e petit côté ? "))
9
10 print("L'hypoténuse mesure", hypo(a, b))
```

Exécution du code

```
Calcul de la longueur de l'hypoténuse
1er petit côté ? 5
2e petit côté ? 12
L'hypoténuse mesure 13.0
```

Instructions conditionnelles

Exemple 2 : Nature d'un triangle

```
1  a, b, c = 5, 7, 7
2
3  if a==b==c :
4      nature = "équilatéral"
5  elif a==b or b==c or a==c:
6      nature = "isocèle"
7  else:
8      nature = "scalène"
9
10 print(f"Le triangle est {nature}.")
```

Exécution du code

```
Le triangle est isocèle.
```

Boucle `while`

Exemple 3 : Partie entière de $\sqrt{x}$, $x \geq 0$

```
1  x = 150
2
3  k = 0
4
5  while k**2 < x:
6      k += 1
7
8  print(f"{k} est le plus petit entier dont le carré est
   inférieur à {x}")
```

Exécution du code

```
13 est le plus petit entier dont le carré est inférieur à
150.
```

Boucle `while`

Exemple 4 : Jeu passionnant

```
1  import random as rd
2
3  n = rd.randint(1, 100)
4  k = -1
5
6  while(k != n):
7      k = int(input(("Quel est ton choix ? ")))
8      if k<n:
9          print("-> Plus grand...")
10     elif k>n:
11         print("-> Plus petit...")
12
13 print("BRAVO !")
```

Boucle `while`

Exemple 4 : Jeu passionnant

```
1  import random as rd
2
3  n = rd.randint(1, 100)
4
5
6  while True :
7      k = int(input(("Quel est ton choix ? ")))
8      if k<n :
9          print("-> Plus grand...")
10     elif k>n :
11         print("-> Plus petit...")
12     else:
13         break
14
15 print("BRAVO !")
```

Boucle for

Exemple 5 : Diviseurs d'un entier n

```
 1  def diviseurs(n):
 2
 3      D = []
 4
 5      for k in range(1, n+1):
 6          if(n%k==0):
 7              D.append(k)
 8
 9      return D
10
11  print(diviseurs(24))
```

Exécution du code

```
[1, 2, 3, 4, 6, 8, 12, 24]
```

Conversion décimal-binaire

But. Passer d'une écriture décimale `d` à une écriture binaire `b`.

Exemple. $326_{10} = 101000110_2$

Algorithme.

```
Prendre le plus petit m tel que 2^(m+1) dépasse d.

Faire une boucle de m à 0 par pas de -1 :
    Si d ≥ 2^m :
        Le prochain chiffre binaire est 1
        On enlève 2^m à d
    Sinon :
        Le prochain chiffre binaire est 0
        d n'est pas affecté

Afficher b
```

8	$2^8 = 256$	1	326
7	$2^7 = 128$	0	70
6	$2^6 = 64$	1	70
5	$2^5 = 32$	0	6
4	$2^4 = 16$	0	6
3	$2^3 = 8$	0	6
2	$2^2 = 4$	1	6
1	$2^1 = 2$	1	2
0	$2^0 = 1$	0	0

Conversion décimal-binaire

```python
d = 326      # décimal

m = 0        # nb de chiffres binaires
while(2**(m+1) <= d):
    m += 1

b = ""       # chiffres binaires  (chaine de caractères)

for k in range(m, -1, -1):
    if(d >= 2**k):
        b += "1"
        d = d - 2**k
    else:
        b += "0"

print(b)
```

Exécution du code

```
101000110
```

Algorithme de Kaprekar

Soit n un entier naturel à 4 chiffres, dont au moins 2 sont distincts.

$n = \overline{abcd}$ avec $a \neq 0$

On classe a, b, c, d par ordre croissant et on note $\alpha, \beta, \gamma, \delta$ cette liste triée.

On pose $c = \overline{\alpha\beta\gamma\delta}$ et $d = \overline{\delta\gamma\beta\alpha}$

La fonction de Kaprekar associe à n l'entier (positif) $d - c$.

Exemple. Avec $n = 7231$, $c = 1237$, $d = 7321$, $d - c = 6084$

Question. Que se passe-t-il quand on itère cette fonction ?

Algorithme de Kaprekar

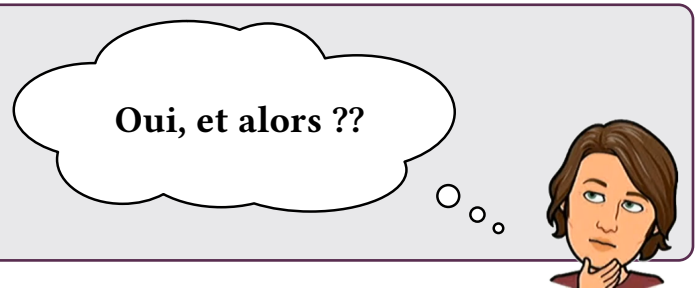
```python
def kaprekar(n):
    s = str(n).zfill(4)          # Chaine de 4 caractères
    T = [s[0], s[1], s[2], s[3]]
    T.sort()
    c = int(T[0]+T[1]+T[2]+T[3])   # concaténation puis
    d = int(T[3]+T[2]+T[1]+T[0])   # passage en integer
    return d-c

n = 7231

for i in range(8):
    print(i, n)
    n = kaprekar(n)
```

Exécution du code

```
0 : 7231        4 : 3996
1 : 6084        5 : 6264
2 : 8172        6 : 4176
3 : 7443        7 : 6174
```



Algorithme de Kaprekar

0	7231	6121	8352	9802	6429	5353
1	6084	5085	6174	9531	7173	2178
2	8172	7992	6174	8172	6354	7443
3	7443	7173	6174	7443	3087	3996
4	3996	6354	6174	3996	8352	6264
5	6264	3087	6174	6264	6174	4176
6	4176	8352	6174	4176	6174	6174
7	6174	6174	6174	6174	6174	6174



Exercices

1) Tester l'algorithme de Kaprekar pour quelques nombres à 3 chiffres, conjecturer et tester la conjecture sur tous les autres.

2) Programmer un convertisseur binaire vers décimal.

3) Écrire cet algorithme qui teste si un entier $n \geq 5$ est premier :

```
Si n est pair ou multiple de 3 -> retourner "Faux"

Poser k = 5

Tant que k*k ≤ n :
    si n est multiple de k ou de k+2 -> retourner "Faux"
    ajouter 6 à k

 Retourner "Vrai" si la boucle précédente s'est terminée.
```

Récursivité

Soit n un entier positif. On définit le $(n+1)$-uplet $(u_0, u_1, ..., u_n)$ de la manière suivante :

$$u_0 = 1$$
$$u_n = n \times u_{n-1}$$

Que vaut u_n ? $\rightarrow$ $u_n = n!$

```python
def factorielle(n):
    if(n==0):
        return 1
    else:
        return n*factorielle(n-1)

print(factorielle(5))   # 120
```

Récursivité : exponentiation rapide

Soient n et a deux entiers naturels non nuls. On définit la fonction $e(a, n)$ par :

$$e(a,n) = \begin{cases} a & \text{si } n = 1 \\ e(a, \frac{n}{2})^2 & \text{si } n \text{ est pair} \\ a \times e(a, n-1) & \text{si } n \text{ est impair} \end{cases}$$

On a : $\boldsymbol{e(a,n) = a^n}$

En effet, si n est pair, $a^n = \left(a^{\frac{n}{2}}\right)^2$ et s'il est impair, $a^n = a \times a^{n-1}$

La récursivité diminue le nombre d'opérations : $a^9 = a \times \left((a^2)^2\right)^2$

$a^9 = a \times a \times a \times a \times a \times a \times a \times a \times a$

Récursivité : exponentiation rapide

Implémentation de l'exponentiation rapide :

```python
def e(a, n):
    if(n==1):
        return a
    elif(n%2==0):
        return e(a, n/2)**2
    else:
        return a*e(a, n-1)

print(e(3, 10))
```

Exécution du code

```
59049
```

On a bien : $3^{10} = 59\,049$

Récursivité : exponentiation rapide

La fonction `pow` implémente l'exponentiation rapide.

Syntaxe : `pow(a, b)` pour a^b et `pow(a, b, n)` pour $a^b \bmod n$.

Exemple 6 : Calcul de $2440212^{523238} \bmod 7$

1ère possibilité : `2440212**(523238) % 7`

→ Durée = 1,77 s

2ème possibilité : `pow(2440212, 523238, 7)`

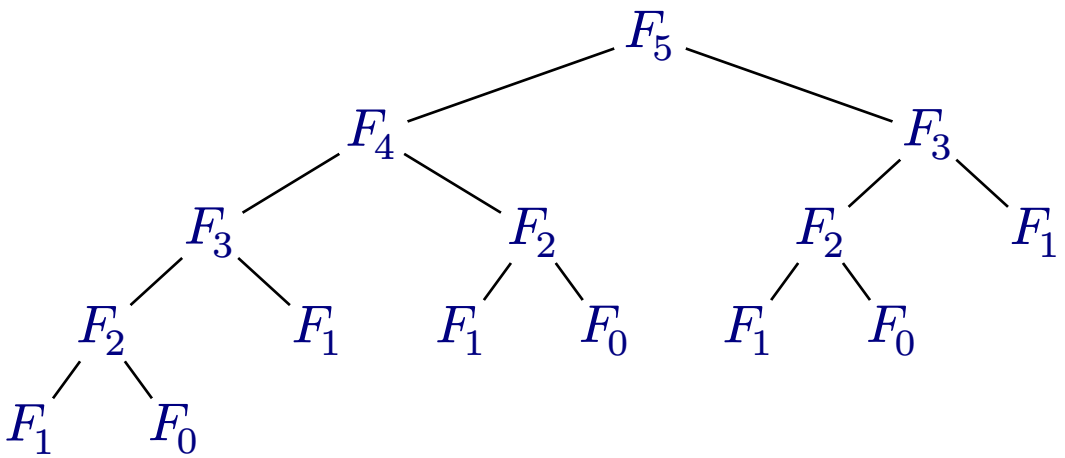
→ Durée = 3 µs

Optimation importante en cryptographie

Récursivité : points de vigilance

- La lecture du code est parfois plus difficile.
- Il faut s'assurer de la finitude de la fonction.
- Il peut y avoir une explosion combinatoires des appels.

Exemple : $F_n = \begin{cases} 0 & \text{si } n=0 \\ 1 & \text{si } n=1 \\ F_{n-1}+F_{n-2} & \text{si } n \geq 2 \end{cases}$ (suite de Fibonacci)



Applications de la récursivité

- Algorithmes de type « diviser pour régner » :
  - Tri de données
  - Parcours d'arbres et de graphes…
- Géométrie fractale.
- Certains problèmes comme la Tour de Hanoï :

```
1 def hanoi(n, source="A", cible="C", intermediaire="B"):
2     if n == 0:
3         return None
4     hanoi(n-1, source, intermediaire, cible)
5     print(f"Déplacer le disque {n} de {source} vers
  {cible}")
6     hanoi(n-1, intermediaire, cible, source)
7
8 hanoi(3)
```

Conclusion

Pour approfondir :

- S'intéresser aux autres **types** en Python : *t*-uplets (*tuple*), dictionnaires (*dict*) et ensembles (*set*) et le lien graphes-matrices.
- Algorithme du tri rapide.
- Les « quatre cycles de Conway » sur `joliesmaths.fr`
- Découvrir le module de tracé `turtle`.
  « Autour de la courbe de Koch » sur `joliesmaths.fr`

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

```python
def kaprekar(n):
    s = str(n).zfill(3)
    T = [s[0], s[1], s[2]]
    T.sort()
    c = int(T[0]+T[1]+T[2])
    d = int(T[2]+T[1]+T[0])
    return d-c

for m in range(100, 1000):
    k = m
    for i in range(7):
        k = kaprekar(k)
    if(k!=495):
        print(m)
```

Exécution du code

```
111     444     777
222     555     888
333     666     999
```

Solutions

```python
b = "101000110"      # binaire
m = len(b)           # nb de chiffres binaires

d = 0                # décimal

for k in range(0, m):
    d = d + int(b[k])*2**(m-k-1)

print(d)
```

```python
def est_premier(n):
    if (n%2==0 or n%3==0):
        return False

    k = 5
    while (k*k <= n):
        if (n%k==0 or n%(k+2)==0):
            return False
        k = k + 6
    return True
```