Pascal Lafourcade et Malika More

15 ÉNIGMES LUDIQUES POUR S'INITIER À LA PROGRAMMATION PYTHON

DUNOD

Découvrez aussi :

- P. Lafourcade, M. More, *25 énigmes ludiques pour s'initier à la cryptographie*, Dunod, 2021
- J.-G. Dumas, P. Lafourcade, A. Tichit et S. Varrette, *Les blockchains en 50 questions – Comprendre le fonctionnement et les enjeux de cette technologie*, Dunod, 2019
- J.-G. Dumas, P. Lafourcade, P. Redon, *Architectures de sécurité pour internet – Protocoles, standards et déploiement*, Dunod, 2020
- J. Guillod, *Programmation Python par la pratique – Problèmes et exercices corrigés*, Dunod, 2021
- B. Cordeau, L. Pointal, *Python 3 – Apprendre à programmer dans l'écosystème Python*, 2e édition, Dunod, 2020

Direction artistique : Studio Dunod
Graphisme de couverture : Nicolas Wiel

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-083430-3

À tous ceux qui sont prêts à forger des codes pour gagner de l'argent.

Sommaire

Avant-propos

Ces 15 énigmes sont des challenges pour vous faire découvrir des concepts importants de l'informatique en vous amusant. Elles nécessitent plus ou moins de logique, de réflexion et d'astuce. Cependant, elles sont toutes accessibles à l'aide de concepts abordés au lycée.

Pour chaque énigme, nous avons conçu trois niveaux progressifs d'indices, qui se trouvent dans un chapitre séparé au milieu du livre. Ainsi, si après avoir commencé à réfléchir, vous êtes bloqué, vous trouverez avec les indices une aide graduée pour vous donner un coup de pouce et vous mettre sur la piste de la solution.

Les niveaux de difficulté des énigmes sont indiqués par des étoiles. Le niveau facile est représenté par ☆. Les énigmes de ce niveau sont accessibles à tous, moyennant parfois un peu de persévérance. Pour les résoudre il faudra écrire des programmes Python 3 assez simples.

Le niveau intermédiaire est représenté par ☆☆. Dans ces énigmes, la réflexion ou les calculs sont plus complexes, et il arrive que la solution repose sur une astuce un peu moins évidente que dans le premier niveau. Les programmes à écrire pour ces énigmes sont clairement un peu plus complexes.

Le niveau ☆☆☆ est le niveau difficile. Il comporte deux énigmes qui nécessitent beaucoup de réflexion ou qui demandent des connaissances en mathématiques et en informatique un peu plus avancées. Il faudra faire preuve de ténacité pour concevoir ces programmes afin de trouver la solution.

Cet ouvrage s'adresse principalement à des lycéens scientifiques. L'objectif est de proposer des énigmes dont la solution peut être obtenue à l'aide d'un programme Python 3. Il fallait bien choisir un langage pour présenter les solutions, mais il est bien entendu possible de programmer dans votre langage de programmation favori pour résoudre ces énigmes. En début d'ouvrage, nous fournissons en quelques pages la description des principales commandes en Python 3 utiles pour la résolution des énigmes.

Les thèmes des énigmes sont l'occasion de découvrir de nombreux concepts importants en informatique. La plus grande partie de cet ouvrage est constituée des solutions détaillées de toutes les énigmes. Chaque solution contient non seulement le programme Python 3 amenant à la résolution de l'énigme,

mais aussi des explications détaillées sur la conception de l'algorithme correspondant. En guise de clin d'œil, chaque solution est accompagnée d'une citation scientifique ou littéraire en rapport avec l'énigme ou sa solution.

L'aspect ludique de cet ouvrage motivera certains lecteurs pour coder les solutions et les poussera à faire preuve de créativité pour écrire des programmes résolvant les énigmes.

Ce livre s'adresse en premier à un public qui aime programmer et vise à proposer des problèmes accessibles pour les programmeurs, afin qu'ils découvrent et apprennent des concepts importants de l'informatique. Il s'adresse aussi, indirectement, à tous les enseignants de Lycée en mathématiques, puisqu'il leur est prescrit de faire programmer leurs élèves avec Python 3. Ces énigmes sont une banque d'exercices corrigés au même titre qu'un manuel de cours.

Les énigmes, les indices et les solutions contiennent de nombreux encarts biographiques, historiques, techniques, mathématiques, Python 3, culturels, de solution de codes Python 3 ou encore d'objectifs pédagogiques pour les enseignants en rapport avec les concepts abordés. Ils sont représentés respectivement par ces icônes :







Les fichiers correspondant aux énoncés des énigmes, aux programmes des solutions et des compléments en Python 3 sont disponibles à l'adresse :

https://www.dunod.com/ean/9782100834303

Remerciements : Nous remercions Cédric Lauradoux pour nous avoir montré la voie pour la création de ces énigmes. Nous adressons aussi nos remerciements à Flavien Binet, Guenaëlle De Julis, Emmanuel Delay, Christine Solnon, Thibault Ralet pour leurs contributions à l'élaboration du contenu de ce livre. De plus, nous exprimons également notre gratitude à Jean-Luc Blanc, Matthieu Daniel et Anne Le Duc pour leurs commentaires et suggestions constructives, à la suite de leurs relectures assidues.

Clermont-Ferrand, le 13 septembre 2022.
Malika More et Pascal Lafourcade *.

*Nous serons heureux de répondre à vos questions par email.

1

Python 3

1 Une brève introduction

Dans cet ouvrage, les solutions des énigmes sont données en Python 3 à titre d'illustration, un logiciel libre [#]. Il est bien entendu possible de rédiger les solutions dans le langage de programmation de votre choix, puisqu'un langage de programmation comprenant des instructions pour effectuer des calculs arithmétiques, des affectations de variables, des instructions conditionnelles et des boucles (`while`) permet de programmer n'importe quel algorithme.

Dans ce chapitre introductif, les commandes les plus utilisées dans les solutions des différentes énigmes sont présentées. Dans certains cas, des précisions supplémentaires sont données dans les énoncés ou dans les indices. Pour avoir plus de détails sur Python 3, et pour apprendre à programmer en Python 3 de façon organisée, il existe de nombreux ouvrages [Gui21] et des sites de cours dédiés. Le site de référence, mais pas le plus simple pour débuter, est le manuel officiel accessible en ligne `https://docs.python.org/fr/3/`. Si vous êtes déjà à l'aise en Python 3, vous n'aurez probablement pas besoin de vous référer à ce chapitre, sinon, n'hésitez pas à le parcourir pour chercher des idées lorsque vous manquez d'outils pour programmer la solution d'une énigme.

Tout d'abord, notez qu'en Python 3 le symbole `#` permet de commenter le reste de la ligne, ce qui signifie que tout ce qui est écrit après ce symbole sur la même ligne ne sera pas pris en compte lors de l'exécution du programme. De plus, il est essentiel de savoir que les *tabulations* en Python 3 jouent un rôle crucial et que leur présence ou leur absence aux bons endroits peut modifier le comportement d'un programme et même générer des erreurs. Pour éviter d'avoir de longues lignes en Python 3, il existe l'opérateur `\` également connu sous le nom de saut de ligne explicite. Il permet de diviser une seule longue ligne continue en plusieurs lignes de code plus petites et plus faciles à lire.

```
1 string = "Cette" + " chaine" + " est" + " bien" + " trop" + " longue" \
2  + " pour" + " tenir" + " sur" + " une" + " ligne."
```

[#] `https://www.python.org`

Affectation et opérations mathématiques simples

Le symbole `=` permet d'affecter la valeur d'une expression à une variable. Par exemple, les deux instructions `x=3` suivie de `x=x+1` affectent successivement la valeur `3` à la variable `x` puis la valeur `3+1`, soit `4`, à cette même variable.

Les opérations numériques d'addition, soustraction, multiplication, division, puissance, ainsi que les opérations booléennes négation, et, ou, se notent respectivement `a+b`, `a-b`, `a*b`, `a/b`, `a**b`, `not a`, `a and b`, `a or b`. Concernant la division euclidienne, si $a = b.q + r$ avec $0 \leq r < b$, alors q = `a//b`, et r = `a%b` ou r =`divmod(a,b)`.

Lorsque ces opérations sont possibles, pour convertir la variable `y` en entier il faut écrire `x=int(y)`, et pour convertir la variable `x` en chaîne de caractères, il faut écrire `y=str(x)`.

Instruction conditionnelle : `if`

Pour exprimer une condition, les mots clés `if`, `else` et `elif` sont utilisés. Le code suivant affecte la valeur `3` à la variable `res` si la valeur de la variable `x` est strictement plus grande que `3`, sinon, si `x` vaut zéro alors `res` reçoit la valeur `0` et sinon `res` prend la valeur `1`.

```
1 if x > 3 :
2     res = 3
3 elif x == 0 :
4     res = 0
5 else :
6     res = 1
```

Il est important de remarquer que `==` est utilisé dans les tests des conditions et que ce symbole est différent de celui utilisé pour symboliser l'affectation. Par ailleurs, le symbole : après la comparaison ainsi qu'une tabulation pour chacun des cas sont obligatoires. Les parties du code correspondant à `elif` et à `else` ne sont utilisées que si l'algorithme les rend nécessaires, dans le cas contraire il est possible de ne pas les écrire.

Les opérateurs de comparaison usuels se notent comme suit : `a<b`, `a<=b`, `a>b`, `a>=b`, `a==b`, et `a !=b` (pour `a` différent de `b`).

Boucles : `while` et `for`

Le programme suivant affiche les entiers de 0 à 9 grâce à une boucle `while`.

```
1 i=0
2 while (i<10) :
3     print(i)        #0 1 2 3 4 5 6 7 8 9
4     i=i+1
```

Le programme suivant fait la même chose avec une boucle `for`. En Python 3, la fonction `range(i)` retourne l'ensemble des valeurs entières entre `0` et `i-1`.

```
1 for i in range(10):
2     print(i)        #0 1 2 3 4 5 6 7 8 9
```

Chaînes de caractères

- La chaîne de caractères vide est notée `""` (sans espace entre la paire de guillemets).
- Pour concaténer deux chaînes de caractères, comme `"Bon"` et `"jour"`, il suffit d'utiliser le symbole d'addition `+`. Ainsi la concaténation de `"Bon"` et de `"jour"` vaut `"Bonjour"`.
- Pour convertir un entier en le caractère dont il est le code ASCII [†], lorsque cela est possible, la fonction `chr` renvoie le symbole correspondant. Par exemple, le code `chr(97)` vaut `'a'` et `chr(122)` vaut `'z'`, mais le code `chr(123456789)` renvoie une erreur, car les valeurs des codes ASCII ne vont pas si loin.
- Il est en général indifférent d'utiliser les guillemets simples `'` ou doubles `"` pour encadrer une chaîne de caractères, comme dans les deux exemples ci-dessus.

Lecture de données dans un fichier

Dans de nombreuses énigmes, un fichier contenant les données de l'énigme est fourni en ligne (lien page VIII), afin d'éviter de recopier l'énoncé. Il est donc important de savoir lire ces données pour pouvoir les manipuler dans un programme en Python 3.

Le programme suivant permet de lire ligne par ligne et un par un les mots du fichier `f.txt`

[†] American Standard Code for Information Interchange

```
fichier = open('f.txt','r')
for line in fichier :
    for word in line.split() :
        print(word)
fichier.close()
```

Le programme suivant, quant à lui, permet de lire caractère par caractère le contenu du fichier `f.txt`

```
fichier=open('f.txt','r')
caractere=fichier.read(1)
while caractere :
    print(caractere)
    caractere=fichier.read(1)
fichier.close()
```

Listes

La liste vide est notée `[]`. Pour ajouter un élément `x` à la fin de la liste `L`, l'instruction `L.append(x)` est utilisée. Pour accéder au premier élément de la liste `L`, il faut écrire `L[0]`, car les indices des listes commencent à `0` en Python 3. Pour avoir le dernier élément de la liste il est possible d'écrire `L[-1]`.

```
L=[]
L.append(4)
L.append(5)
L.append(6)
print(L)              # [4, 5, 6]
print(L[1])           # 5
print(len(L))         # affiche 3, le nombre d'élements de la liste L
print(L[-1])          # affiche le dernier élement de la liste L
print(L[len(L) - 1])  # affiche le dernier élement de la liste L
```

Fonctions et passage de paramètres

Pour définir une fonction en Python 3, il faut utiliser le mot clef `def` suivi du nom de la fonction, par exemple `echange` dans l'exemple ci-dessous, puis des paramètres de la fonction, ici `x` et `y` entre parenthèses.

```
def echange(x,y):
    x=x+y
    y=x-y
    x=x-y
    return (x,y)

a=3
b=4
print(echange(3,b)) # affiche (4,3)
print(a,b)          # affiche 3 4
```

Les variables définies dans la fonction sont des variables dites *locales* et n'ont pas de lien avec les variables définies à l'extérieur de la fonction bien qu'elles puissent porter le même nom. Les variables locales n'existent qu'à l'intérieur de la fonction et des valeurs leur sont affectées lors de l'appel de la fonction. Dans l'exemple, `x` prend la valeur 3 et `y` la valeur du paramètre `y` soit 4.

En Python 3, les objets dits *non mutables*, comme les entiers et les chaînes de caractères, sont donc passés par valeurs dans les fonctions. Cela signifie que leur valeur ne peut pas être modifiée par la fonction.

En revanche, les objets dits *mutables*, comme les listes, sont passés par référence comme le montre l'exemple ci-dessous. Cela signifie que leur valeur peut être modifiée par la fonction.

```
1 def ajoutListe( liste ,x):
2     liste .append(x)
3
4 L =[1,2,3]
5 print(L)          # [1,  2,  3]
6 ajoutListe(L,4)
7 print(L)          # [1,  2,  3,  4]
```

La modification faite sur la liste passée en paramètre est effective sur la liste définie en dehors de la fonction `ajoutListe`.

Bibliothèques

Dans Python 3, il est possible d'inclure des *bibliothèques*, en général thématiques, composées de fonctions déjà programmées. Pour cela, il faut indiquer le nom de la bibliothèque au début du programme, précédé du mot clé `import`. Par exemple, `import math` permet d'utiliser dans la suite du programme les fonctions mathématiques `math.exp(x)`, `math.sqrt(x)`, `math.cos(x)` ou `math.sin(x)`, qui calculent respectivement l'exponentielle, la racine carrée, le cosinus et le sinus d'un nombre `x`.

D'autres bibliothèques sont accessibles en Python 3 comme par exemple les bibliothèques `random` ou `time`. La bibliothèque `random` permet de générer des nombres aléatoires. Avec la commande `time()` de la bibliothèque `time`, il est possible de connaître l'heure et ainsi de mesurer par exemple la durée d'exécution d'un programme.

```
1 import time
2 import random
3
4 debut = time()                  # donne l'heure courante
5 r = random.randint(10,100)      # tire  au hasard un entier entre 10 et  100
6 fin  = time()
7 duree = fin  - debut
```

Il est aussi possible d'ajouter des bibliothèques qui ne sont pas par défaut dans Python 3. Par exemple, pour installer la bibliothèque `turtle` nécessaire pour l'énigme 11, il est possible d'utiliser cette commande dans un terminal :

```
python3 -m pip install turtle
```

Par commodité, il est possible de renommer dans un programme Python 3 le nom d'une bibliothèque, comme la bibliothèque `numpy` qui est abrégée par `np` dans le programme ci-dessous.

```
1 import numpy as np
2
3 zeros = np.zeros([10,10])      # construit une matrice de taille 10 x 10 remplie de 0
4 uns = np.ones([10,10])         # construit une matrice de taille 10 x 10 remplie de 1
5 t = np.linspace(3, 4, 11)      # construit une liste de 11 valeurs allant de 3 a 4
6 print(t)                       # [3.  3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.]
```

Les principales fonctionnalités des bibliothèques requises pour résoudre les énigmes sont les suivantes : `pil` (énigme 5 page 18), `pycryptodome` (énigme 8 page 27), `turtle` (énigme 11 page 35) et `numpy` (énigme 13 page 42). Elles sont présentées dans des encadrés au fur et à mesure des besoins.

2

Les énigmes à résoudre

1 À ski ☆

67 104 97 113 117 101 32 103 233 110 233 114 97 116 105 111 110 32 97 32 115 111 110
32 112 104 105 108 111 115 111 112 104 101 44 32 233 99 114 105 118 97 105 110 32 111
117 32 97 114 116 105 115 116 101 32 113 117 105 32 115 97 105 115 105 116 32 101 116
10 105 110 99 97 114 110 101 32 108 39 105 109 97 103 105 110 97 105 114 101 32
100 117 32 109 111 109 101 110 116 46 32 73 108 32 97 114 114 105 118 101 32 113 117
101 32 99 101 115 32 112 104 105 108 111 115 111 112 104 101 115 32 115 111 105 101 110
116 10 114 101 99 111 110 110 117 115 32 100 101 32 108 101 117 114 32 118 105 118 97
110 116 44 32 109 97 105 115 32 108 101 32 112 108 117 115 32 115 111 117 118 101 110
116 32 105 108 32 102 97 117 116 32 97 116 116 101 110 100 114 101 32 113 117 101 32
108 97 10 112 97 116 105 110 101 32 100 117 32 116 101 109 112 115 32 102 97 115 115
101 32 115 111 110 32 101 102 102 101 116 46 32 81 117 101 32 99 101 116 116 101 32 114
101 99 111 110 110 97 105 115 115 97 110 99 101 32 115 111 105 116 10 105 109 109 233
100 105 97 116 101 32 111 117 32 100 105 102 102 233 114 233 101 44 32 117 110 101
32 233 112 111 113 117 101 32 101 115 116 32 109 97 114 113 117 233 101 32 112 97 114
32 99 101 115 32 104 111 109 109 101 115 32 113 117 105 10 101 120 112 114 105 109
101 110 116 32 108 101 117 114 115 32 105 100 233 97 117 120 44 32 100 97 110 115
32 108 101 115 32 109 117 114 109 117 114 101 115 32 100 39 117 110 32 112 111 232 109
101 32 111 117 32 100 97 110 115 32 108 101 10 103 114 111 110 100 101 109 101 110 116
32 100 39 117 110 32 109 111 117 118 101 109 101 110 116 32 112 111 108 105 116 105 113
117 101 46 32 78 111 116 114 101 32 103 233 110 233 114 97 116 105 111 110 32 97 32 117
110 10 112 104 105 108 111 115 111 112 104 101 46 32 67 101 32 110 39 101 115 116 32
110 105 32 117 110 32 97 114 116 105 115 116 101 32 110 105 32 117 110 32 233 99 114
105 118 97 105 110 46 32 67 39 101 115 116 32 117 110 10 105 110 102 111 114 109 97
116 105 99 105 101 110 46 10

Figure 1 – Texte à décoder.

Énigme 1 : *Après avoir compris la technique de codage utilisée, saurez-vous écrire un programme Python 3 permettant de décoder le texte de la figure 1 ?*

Solution page 65.

1 À ski ☆

« Je puis expliquer la base philosophique du logiciel libre en trois mots : liberté, égalité, fraternité. Liberté, parce que les utilisateurs sont libres. Égalité, parce qu'ils disposent tous des mêmes libertés. Fraternité, parce que nous encourageons chacun à coopérer dans la communauté. »
Richard Stallman, « *Liberté, égalité, fraternité* », entretien dans la revue *Programmez !*, numéro 95, mars 2007.

Pour décoder le texte proposé, il faut d'abord se rendre compte que c'est le code ASCII qui a été utilisé pour représenter chaque caractère par un nombre. La correspondance pour les lettres majuscules et minuscules est donnée dans le tableau page 54. Pour décoder, il faut donc prendre successivement chaque nombre et le remplacer par le caractère correspondant. Cela pourrait être fait manuellement, mais serait très fastidieux car le texte est assez long (526 caractères, en comptant les espaces, qui ont 32 pour code ASCII). Voilà une bonne motivation pour écrire un programme qui fera ce travail tout seul !

Pour réaliser le décodage du texte proposé, le programme Python 3 doit successivement ouvrir le fichier `Enigme1.txt`, lire un par un les éléments (chaînes de caractères séparées par des espaces) du fichier, convertir chacun d'eux en entier, puis traduire cet entier en caractère à l'aide du code ASCII, et, au fur et à mesure, concaténer tous ces caractères ensemble dans une seule chaîne de caractères. Dans le programme 2 ci-dessous, c'est la ligne `8` qui effectue l'ensemble des calculs, tandis que les lignes `1` à `7` permettent le traitement du fichier et de son contenu. Les fonctions Python 3 utilisées sont décrites dans l'indice 3 de l'énigme 1 page 58.

Programme 2 – Décodage ASCII.

```
1  ## Ouverture d'un fichier texte en lecture avec un parcours ligne à ligne et mot par mot
2  resultat=""
3  fichier=open('../Enigmes/Enigme1.txt','r')
4  for ligne in fichier :
5      for mot in ligne.split():    ## conversion de chaque mot en entier puis en caractère ASCII
6          resultat = resultat + chr(int(mot))
7  print(resultat)
8  fichier.close()
```