

Approfondissement

Algorithmique et programmation en Python

Pour aller plus loin : chiffres d'un entier, vol de Syracuse, méthode de Monte-Carlo, dichotomie, listes

5 exercices · niveau ●●● · durée conseillée : 2 h 30 · sans ordinateur

Exercice 1 Les chiffres d'un entier



- 1.1** Soit n un entier naturel. Justifier que $n \% 10$ est le chiffre des unités de n et que $n // 10$ supprime ce chiffre.
- 1.2** Recopier et compléter le tableau d'exécution de `somme_chiffres(2026)` : à chaque tour, donner s et n à la fin du tour. Que renvoie l'appel ?

Tour	1	2	3	4
s				
n				

```

1 def somme_chiffres(n):
2     s = 0
3     while n > 0:
4         s = s + n % 10
5         n = n // 10
6     return s

```

Programme 1 — Somme des chiffres

- 1.3** Que renvoie `somme_chiffres(0)` ? Combien de tours sont effectués ? Le résultat est-il satisfaisant ?
- 1.4** Écrire une fonction `nb_chiffres(n)` renvoyant le nombre de chiffres d'un entier $n \geq 1$.
- 1.5** Rappeler le critère de divisibilité par 9, puis écrire `multiple_de_9(n)` à l'aide de `somme_chiffres`. La tester sur 3618 et 3619.

Exercice 2 Le vol de Syracuse



On part d'un entier $n \geq 1$: si n est pair on le divise par 2, sinon on le remplace par $3n + 1$; on s'arrête lorsque l'on obtient 1.

- 2.1** Écrire la suite complète des valeurs prises par n lorsque l'on part de 6. Que renvoie alors `vol(6)` ?
- 2.2** Que renvoient `vol(1)` et `vol(8)` ? Justifier.
- 2.3** Modifier le programme pour qu'il renvoie également la plus grande valeur atteinte au cours du vol. Que vaut-elle pour $n = 6$?
- 2.4** On sait vérifier que le programme s'arrête pour tous les entiers testés jusqu'ici, mais personne n'a démontré qu'il s'arrête pour *tout* entier n . Expliquer pourquoi le raisonnement « la suite finit toujours par diminuer » n'est pas une démonstration.

```

1 def vol(n):
2     d = 0
3     while n != 1:
4         if n % 2 == 0:
5             n = n // 2
6         else:
7             n = 3 * n + 1
8             d = d + 1
9     return d

```

Programme 1 — Durée du vol

Exercice 3 Estimer π par la méthode de Monte-Carlo



On tire au hasard un point M de coordonnées $(x; y)$ dans le carré de sommets $O(0; 0)$, $A(1; 0)$, $B(1; 1)$ et $C(0; 1)$: les deux coordonnées sont obtenues par `random()`, qui renvoie un flottant de $[0; 1]$. On admet que la probabilité que M appartienne à une partie du carré est égale à l'aire de cette partie (hors programme — pour aller plus loin).

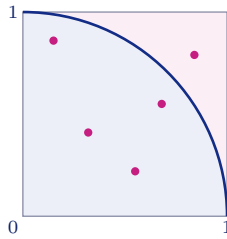


Figure 1 — Cinq points tirés au hasard

```

1 from random import random
2
3 def monte_carlo(n):
4     c = 0
5     for i in range(n):
6         x = random()
7         y = random()
8         if x * x + y * y <= 1:
9             c = c + 1
10    return 4 * c / n

```

Programme 1 — Estimation de π par tirages aléatoires

- 3.1** Que traduit géométriquement la condition $x^2 + y^2 \leq 1$? Combien des cinq points de la figure 1 la vérifient ?
- 3.2** Calculer l'aire du carré, puis celle du quart de disque. En déduire la probabilité que M appartienne au quart de disque.
- 3.3** Que représente $\frac{c}{n}$ lorsque n est grand ? Justifier alors le facteur 4 de la ligne 10.
- 3.4** Trois exécutions de `monte_carlo(10000)` ont donné 3,1416 ; 3,1284 ; 3,1508. Commenter, puis dire comment améliorer la précision et à quel prix.

Exercice 4 Deviner un nombre par dichotomie



Un ami choisit un entier cible entre 1 et 100. Le programme ci-contre le devine : à chaque étape il propose le milieu m de l'intervalle des valeurs encore possibles, puis ne conserve que la moitié utile.

- 4.1** Recopier et compléter le tableau d'exécution pour cible = 73 : à chaque tour, donner m , puis a et b à la fin du tour.

Tour	1	2	3	4	5	6	7
m							
a							
b							

```

1 def recherche(cible):
2     a = 1
3     b = 100
4     n = 0
5     while a < b:
6         m = (a + b) // 2
7         n = n + 1
8         if m < cible:
9             a = m + 1
10        else:
11            b = m
12    return a, n

```

Programme 1 — Dichotomie

- 4.2** Que renvoie `recherche(73)` ?
- 4.3** Justifier que, s'il reste N entiers possibles au début d'un tour, il en reste au plus $\frac{N+1}{2}$ à la fin de ce tour ; en déduire le nombre maximal de tours entre 1 et 100.
- 4.4** Même question entre 1 et 1000 (on remplace 100 par 1000 à la ligne 3), puis comparer avec une recherche essayant 1, puis 2, puis 3...

Exercice 5 Une liste pour compter (hors programme — pour aller plus loin)



```

1 from random import randint
2
3 # liste de 13 cases numerotees de 0 a 12 :
4 # effectifs[k] compte les sorties de k.
5 effectifs = [0] * 13
6
7 def simulation(n):
8     for i in range(n):
9         s = randint(1, 6) + randint(1, 6)
10        effectifs[s] = effectifs[s] + 1
11    return effectifs

```

Programme 1 — Somme de deux dés

- 5.1** Pourquoi 13 cases ? Quelles cases resteront égales à 0, quel que soit le nombre de lancers ?
- 5.2** En dénombrant les 36 couples possibles, compléter le tableau des probabilités $P(S = k)$, en trente-sixièmes.
- 5.3** Une exécution de `simulation(3600)` a donné, pour k de 2 à 12 : 92 ; 214 ; 288 ; 393 ; 522 ; 587 ; 511 ; 389 ; 305 ; 196 ; 103. Vérifier que le total vaut 3600, puis comparer aux effectifs théoriques.
- 5.4** Écrire, **sans utiliser de liste**, une fonction qui renvoie la fréquence des sommes égales à 9 sur n lancers, et donner la probabilité théorique correspondante.

k	2	3	4	5	6	7	8	9	10	11	12
$P(S = k)$											