Labo d'introduction à l'informatique

pour les mathématiques

Yann Thorimbert

UNIVERSITÉ DE GENÈVE
CENTRE UNIVERSITAIRE D'INFORMATIQUE

Semaine 6

Génération de nombres pseudoaléatoires

Rappel

- Nous avons vu une façon de simuler des processus aléatoires tel que celui du marcheur ivre.

- Nous avons également vu une manière originale (à défaut d'être efficace) pour approximer la valeur de π *via* une méthode de "Monte-Carlo".

- De façon générale, il peut être utile de générer des nombres aléatoires de façon programmatique : simulations de Monte-Carlo, cryptologie, jeux-vidéos en sont des exemples.

Comment l'ordinateur génère-t-il des nombres "aléatoires" ?

- Certaines dispositifs hardwares sont capables d'utiliser des variables de l'environnement suffisamment imprévisibles (petites variations de température par exemple) pour les transformer en séquences de nombres semblant aléatoires, quand bien même le processus est déterministe. D'autres dispositifs exploitent le résultat de mesures quantiques.

- Dans bien des applications, on peut se contenter de séquences qui se répètent en réalité, mais qui donnent suffisamment bien l'impression d'être aléatoires : ce sont des nombres pseudoaléatoires.

L'exemple du carré médian

Une première méthode célèbre, bien que peu efficace car se répétant rapidement, est la méthode du carré médian :

1. Prendre le nombre n, qui possède k chiffres.
2. Calculer n^2
3. Garder les k décimales "du milieu" du nombre n^2

Exemple : $n = 5731 \Rightarrow n^2 = 32844361 \Rightarrow$ Sortie = 8443

L'exemple du carré médian

Une première méthode célèbre, bien que peu efficace car se répétant rapidement, est la méthode du carré médian :

1. Prendre le nombre *n*, qui possède *k* chiffres.
2. Calculer n^2
3. On obtient la sortie *x* en gardant les *k* décimales "du milieu" du nombre n^2

Exemple : ***n* = 5731** ⇒ n^2 = 32844361⇒ *x* = 8443

Ce nombre de "départ" se nomme la **seed**.

L'exemple du générateur congruentiel linéaire

Principe : calculer le modulo d'une fonction linéaire dont la valeur d'abscisse est la seed.

$x = (a \cdot n + b) \% m$

Exemple avec a = 3, b = 5, n = 2 et m = 8 :

$x = (3 \cdot 2 + 5) \% 8 = 11 \% 8 = 3.$

En pratique, on utilise de grandes valeurs de *a* et *b* et m, et il faut bien les choisir !

Séquence de nombres

En général, on peut utiliser la sortie du générateur comme prochaine seed :

$x_{n+1} = (a \cdot x_n + b) \% m$

Utiliser des générateurs pseudoaléatoires

- La plupart des langages proposent un générateur prédéfini.

- Exemple en Python :

```
import random
mon_nombre = random.random() #float entre 0 et 1
print(mon_nombre)
```

La fonction `rand( )`

Quelques utilisations typiques :

```
import random
random.seed(ma_seed)  #permet de spécifier une seed
# Ci-dessous, de x à z, la distribution de probabilité est uniforme
x = random.rand() #float entre 0 et 1 (non compris)
y = random.uniform(a, b) #float entre a et b (non compris)
z = random.randint(a, b) #int entre a et b (compris)

#Exemple pour une distribution gaussienne de moyenne m et écart-type s
k = random.gauss(m, s) #float entre -inf et +inf
```

La fonction rand( ) et le choix de la seed

- Choisir la seed revient à **fixer l'état** à un moment donné. Tous les nombres qui viennent ensuite suivent une séquence **reproductible**, puisqu'on connaît la valeur de départ !

- Si l'on ne précise rien, la seed est fixée via une valeur non reproductible, typiquement déterminée par l'état de l'horloge de l'ordinateur qui exécute le programme.

- Dans un code scientifique destiné à être reproduit, on fixe donc la seed. C'est d'ailleurs comme cela que vos codes seront testés dans Moodle.

La fonction rand( ) et le choix de la seed

Exemple où toutes les valeurs obtenues sont les mêmes à chaque exécution du code :

```
random.seed(14) # On fixe la seed à 14
x = random.random()
y = random.random()
z = x^2 + y^2
print(z)
```

"Hasard" reproductible :

Choisir un élément au hasard dans une liste

En appliquant ce que l'on vient de voir, effectuer un choix au hasard revient à tirer un indice au hasard. Exemple avec une liste de strings :

```
noms = ["Alice", "Bob", "Charlie"]
indice = random.randint(0, len(noms)-1)
print(noms[indice])
```

Choisir un élément au hasard dans un tableau

Fonction prédéfinie du module random :

```
noms = ["Alice", "Bob", "Charlie"]
print(random.choice(noms))
```

La fonction `sample`

- Et si l'on veut tirer plusieurs prénoms au hasard au sein d'un tableau, sans qu'aucun prénom ne se répète ?

- Avec les ingrédients que l'on possède, on pourrait coder une telle fonctionnalité de plusieurs façons.

- La fonction `sample`, néanmoins, est faite pour ce genre de cas.
  Exemple :

```
a = list(range(10)) #a contient les entiers de 0 à 9
b = random.sample(a, 4) #4 éléments parmi ceux de a
```

La fonction `shuffle`

- Permet d'effectuer un sample sur la taille de la liste entière.
- Exemple d'opération dite **in place** : elle modifie directement l'objet !

```
import random
a = ["Alice", "Bob", "Charlie", "Daniel", "Elena", "Fanny"]
random.shuffle(a)
print(a)
```

- C'est ainsi que vous êtes tirés au sort chaque semaine.

La question de la distribution

- Les fonctions que l'on a vues ici permettent de générer des nombres distribués de façon homogène entre les bornes (sauf `random.gauss`)

- Créer un algorithme pour générer des nombres suivant une distribution prédéfinie peut être un problème difficile, si on veut l'effectuer efficacement. Ce n'est pas le sujet de ce cours.

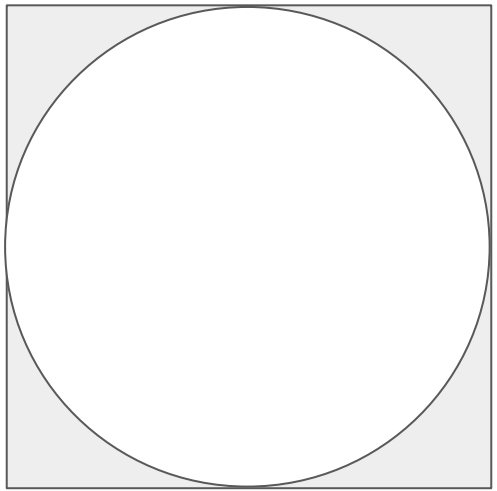
Exemple : méthode de Monte-Carlo

- Il est possible d'estimer la valeur de π en criblant au hasard une cible circulaire avec des "flèches".

- NB : la méthode que l'on va voir n'est **PAS une méthode efficace** pour approximer π, mais elle constitue un exemple simple d'utilisation de nombres pseudo-aléatoires couplés avec des boucles.

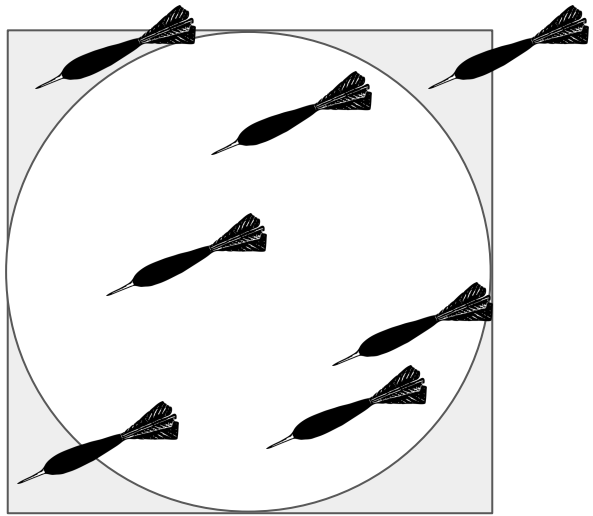
Estimation de π avec un tirage au sort

Il est possible d'estimer la valeur de π en criblant au hasard une cible circulaire avec des "flèches".



Estimation de π avec un tirage au sort

Il est possible d’estimer la valeur de π en criblant au hasard avec des "flèches" une cible circulaire inscrite dans un carré.



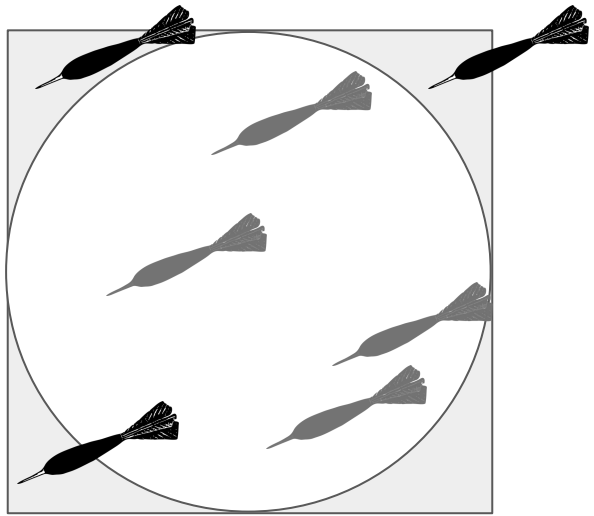
Estimation de π avec un tirage au sort

Il est possible d'estimer la valeur de π en criblant au hasard avec des "flèches" une cible circulaire inscrite dans un carré.

Exemple :

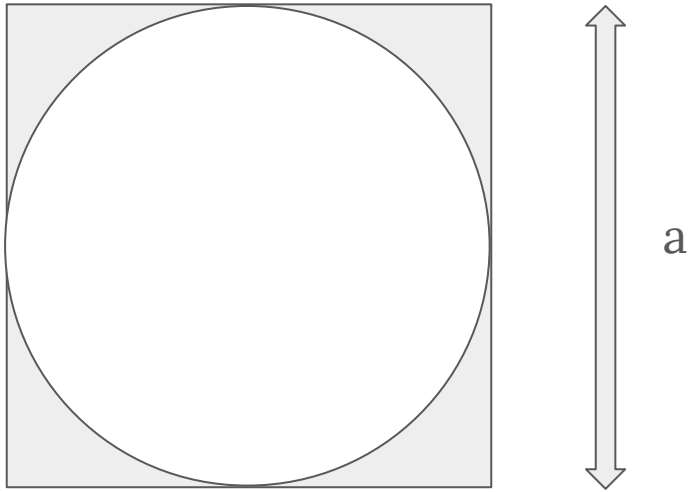
$n_{cible} = 4$

$n_{tot} = 7$



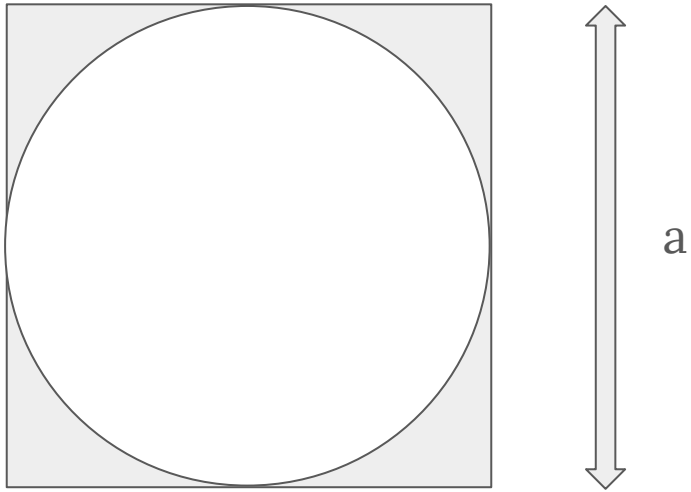
Estimation de π avec un tirage au sort

Il est possible d’estimer la valeur de pi en criblant au hasard une cible circulaire avec des "flèches".



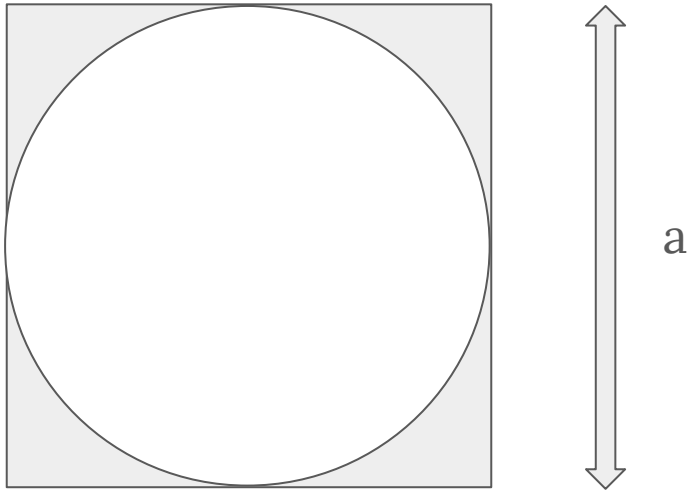
Estimation de π avec un tirage au sort

- $A_s = a^2$
- $A_c := \pi r^2 = \pi a^2/4$



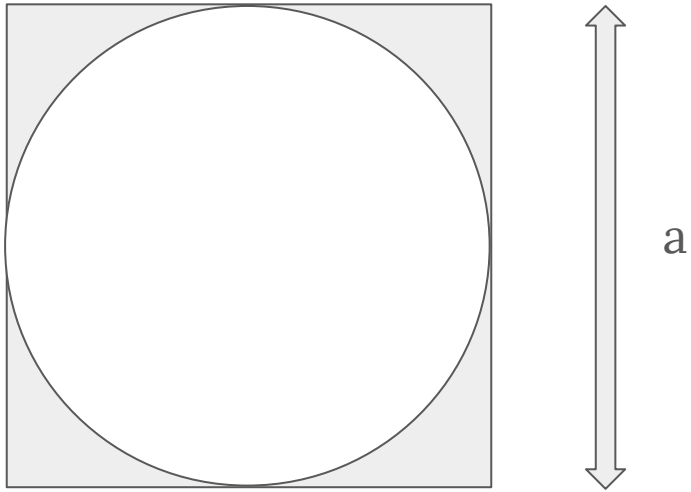
Estimation de π avec un tirage au sort

Si le nombre de flèches est grand et qu'elles sont réellement lancées aléatoirement, alors la densité de flèche est constante et $A_c / A_s \approx n_{cible} / n_{tot}$



Estimation de π avec un tirage au sort

$$n_{cible}/n_{tot} \approx A_c / A_s = (\pi a^2/4) / a^2 = \pi / 4 \Leftrightarrow \boxed{\pi \approx 4\, n_{cible}/n_{tot}}$$



Estimation de π avec un tirage au sort

$$\pi \approx 4\, n_{cible} / n_{tot}$$

Approximation avec une boucle for :

1. Tirer n_{tot} coordonnées au hasard
2. Compter combien sont "dans" le cercle
3. En déduire π

Question supplémentaire :
Comment évolue le nombre de décimales correctes de π_{approx} en fonction du nombre de flèches qu'il faut tirer (en moyenne) pour l'obtenir ?