.

NOM	.....	PRENOM	.....

Exercices

Testons l'aléatoire de `numpy.random`

Et si vous commenciez par lire la section 9 ? Tester les exemples serait aussi une bonne idée. Pour les pandas et la statistique pas d'urgence (en revanche)

Après cela, ouvrir un éditeur de script Python et importer les bibliothèques adéquates.

1. Générer simul : $U \hookrightarrow \mathcal{U}[0;1[$ tel que décrit dans le mnémo Python :

```
>>>
...
```

2. Créer 10 vecteurs lignes $V_1 \dots V_{10}$ composés chacun de 100 valeurs de type simul : $U \hookrightarrow \mathcal{U}[0;1[$:

```
V=np.zeros([10,100])
for .......................
    V[k,]=..................
.............................
```

3. Mais quel est donc le format de l'objet `V` ?

```
...
```

4. *Et maintenant avec les pandas :* Après lecture du mnémo, faites-donc l'exercice proposé :

```
import pandas as pd
L=.......................
.........................
M=L.mean()
s=L.std()
print("moyenne=",M)
print("standard sigma=",s)
```

5. Déterminer les valeurs $\mu_1 \dots \mu_{10}$ des moyennes de chaque séries de valeurs contenues dans les vecteurs $V_1 \dots V_{10}$:

```
Mu=np.zeros([10])
for ............................:
    Mu[k]=.......................
    print(.......................)
```

6. Faire de même avec les écart-types standardisés $\sigma_1 \dots \sigma_{10}$ (on veut le code !) :

```
...
...
...
...
```

7. Organiser la synthèse des résultats sous forme d'une matrice à deux lignes (l'une pour les moyennes, l'autre pour les écart-types) :

```
...
...
```

8. Et fournissez la matrice de sortie (ne dépassez pas 3 chiffres après la virgule...) :

```
...
...
...
```

Pour créer Dédé

La commande Python `rd.randint(a, b + 1)` renvoie un nombre entier aléatoire compris entre a et b (de préférence entier sauf si aimez les bugs). Donc si vous avez bien compris, quelle commande simule un lancé de dé "usuel" ?

```
D6 = rd.randint(................)
```

On notera dans cette section DN le résultat d'un dé à N faces équilibrées. Le but de cette partie est de construire une simulation de DN sans avoir recourt à `rd.randint(1, N + 1)` (ce qui reste un attendu des programmes).

1. Testez le programme suivant cinq fois (à vous de compléter par l'importation des bibliothèques nécessaires !) :

```
...
...
...
U = rd.random()
X = 6*U
X = floor(X)
```

Que vous a renvoyé ce programme ?

```
mes résultats :
```

et surtout, que fait ce programme ?

```
...
...
```

2. Proposez un script qui prend N en entrées et qui réalise une simulation d'un lancer de DN :

```
def de(N):
        ...
        ...
        ...
```

Testez ensuite votre programme pour vérifier (non, pas la peine de fournir vos tests !)

3. Simulez 100 résultats de lancés de D20 puis conjecturez la moyenne attendue pour cette expérience aléatoire d'après la liste des résultats obtenus :

```
...
...
```

4. Utiliser Python pour obtenir la moyenne et l'écart-type standardisé résultant des 100 simulations effectuées :

```
...
...
...
...
...
...
...
```

En répétant, en répétant, en répétant ...

Cette (sous)-section propose de réaliser des simulations de loi binômiale (voir cours associé)
La commande Python `rd.binomial(n, p)` renvoie un nombre entier aléatoire compris entre 0 et n (un entier positif). Donc si vous avez bien compris, quelle commande simule une variable aléatoire de loi $\mathcal{B}(12; \frac{1}{4})$?

```
BnpTest = ..............................
```

Et on va essayer de faire sans la commande (étonnant non ?)

1. Interpréter le code suivant :

```
p = float(input("fournir un paramètre de [0;1]"))       #un tel p sera fourni
U = rd.random()
if U<= p :
        X=1
else :
        X=0
print(X)
```

En particulier, donner $\mathbb{P}[X = 1]$ et décrire la loi de X :

```
...
...
...
Loi de X :
```

2. Réécrivez-nous ça en version *fonction* avec p en entrée :

```
def Ber(p):
        ............................
        ............................
        ............................
        ............................
```

3. En vous aidant de la question précédante, créer une fonction `binomiale(n, p)` qui simule n variables aléatoires indépendantes $X_1 \dots X_n$ de même loi que X qui précère et renvoie le nombre K d'entre elles qui réalisent la valeur 1.

```
def binom(n,p):
    K=0
        ......................
        ......................
        ......................
```

On fera le lien avec la théorie sur la loi binomiale (par la pensée)