Consignes

— On considère qu'on est au début d'une session de travail en Python.

— Le parcours de liste par *for e in L* ou *for i in range(len(L))* est autorisé. La méthode *append()* est autorisée, les autres commandes Python relatives aux listes sont interdites, sauf précision.

— Le module **random** (mot anglais pour "aléatoire") contient une fonction *randint(a,b)* que l'aide nous présente ainsi : *Return random integer in range [a, b], including both end points.*

Génération de liste

1. Définir en compréhension une liste L1 de cent entiers naturels, chacun étant pris au hasard, strictement inférieur à cent.
2. Écrire un script pour remplir progressivement une liste vide, afin d'obtenir une liste L2 de même structure.
3. Construire la liste L12 des nombres communs à L1 et L2 ; le test *if élément in liste* est autorisé.

Recherche d'éléments

On veut écrire une fonction 'cherche' prenant comme argument une liste L et une valeur x, et qui cherche la première occurrence de x dans L, sans utiliser de méthodes de listes intégrées. Si la valeur x est trouvée, la fonction renvoie son indice ; sinon, elle renvoie *None*.

4. Programmer cette fonction en utilisant une variable locale pour le résultat, et une boucle 'for...' parcourant toute la liste.
5. Ajouter une instruction interrompant la recherche si la valeur cherchée est trouvée.
6. Programmer le comportement précédent avec une boucle 'while...'.

Dictionnaire

7. Créer le dictionnaire dont les clés sont les entiers naturels strictement inférieurs à 100, avec comme valeurs les résultats de la recherche précédente pour L1.
8. À partir de ce dictionnaire, créer la liste des nombres qui ne sont pas dans L1.
9. Calculer le pourcentage d'entiers absents de L1.

Approche statistique

10. Écrire un script qui détermine la moyenne du pourcentage précédent, calculée sur N listes générées successivement.
    Une étude probabiliste montre que lorsqu'on tire au hasard une série de p entiers parmi n, le nombre moyen de valeurs distinctes est : $n\,(1-(1-1/n)^p)$. Retrouve-t-on cela ?

$$-=\mathcal{FIN}=-$$

Correction

```
from random import randint # y penser !
L1 = [randint(0,99) for k in range(100)]
L2 = []
for k in range(100):
    L2.append(randint(0,99)) # ou L2 = L2 + [randint(0,99)]

L12 = [e for e in L1 if e in L2]
print(L12, '\n', len(L12))
```

```
cherche(x,L):
    resultat = None
    for i in range(len(L)):
        if x == L[i]:
            resultat = i
    return resultat # 'resultat' n'a pas été modifiée si x n'a pas été trouvée

def cherche2(x,L):
    resultat = None
    for i in range(len(L)):
        if x == L[i]:
            resultat = i ; break # boucle interrompue car résultat obtenu
    return resultat

def cherche3(x,L): # plus "propre" mais il faut gérer le compteur et la condition...
    resultat = None
    i = 0
    while i < len(L) and not(resultat):
        if x == L[i]:
            resultat = i
        i += 1
    return resultat
```

```
dico1 = {k : cherche2(k,L1) for k in range(100)}
absents = [k for k in dico1 if dico1[k] == None]

pca = len(absents) # car = len(absents)/len(L1)*100
print(absents,'\n\n',pca)
```

```
c = 0 ; N = 100000
# mieux que de mettre la valeur numérique de N partout où elle est nécessaire.
for i in range(N):
    L3 = [randint(0,99) for k in range(100)]
    dico3 = {k : cherche2(k,L3) for k in range(100)}
    absents3 = [k for k in dico3 if dico3[k] == None]
    c += len(absents3)
print('Avec N =', N,', le pourcentage moyen est',c/N)

# on trouve 36.6% ; c'est ce qui est prévu puisque (1-1/100)^100 = 0.366032...
```

$$-= \mathcal{FIN} =-$$