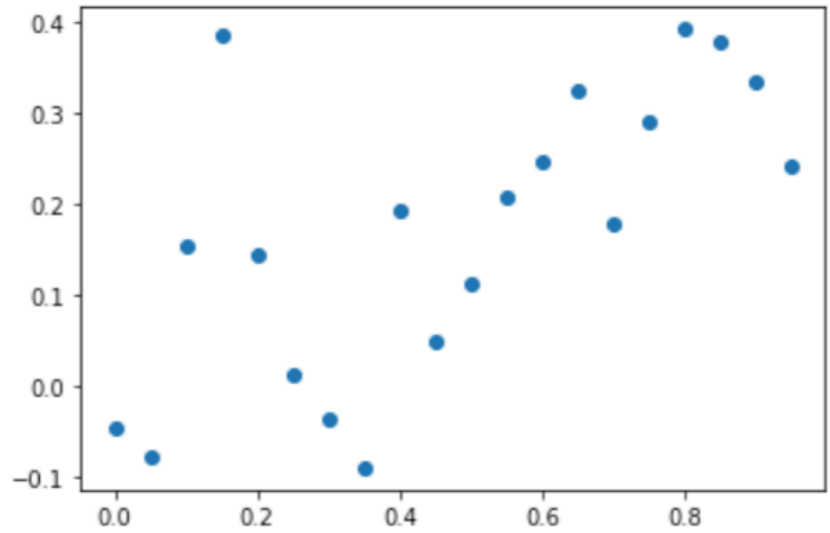
BASES DE LA PROGRAMMATION (2)

Sujet 1 – Devine la corrélation !

Adrien Guille - R2.03 - Université Lumière Lyon 2

```
DEVINE LA CORRÉLATION !
Vies      1
Pièces    0
```




```
Coefficient estimé    0.7
Coefficient réel      0.65
Différence            0.05

Série en cours        0
Erreur moyenne        0.19

+5 pièces
+1 vie
Appuyer sur Entrer pour continuer
```

Exercice 1 – Générer les données

1. Se connecter à Google Colab et créer un nouveau notebook.
2. Importer les fonctions utiles et consulter leur documentation en ligne :

```
from numpy import arange # génère une séquence de valeurs régulièrement espacées
from numpy.random import uniform # réalise un tirage aléatoire selon la loi uniforme
from numpy.random import normal # réalise un tirage aléatoire selon une loi normale
```

3. Définir la fonction `generate_data` qui reçoit en argument un nombre de points `n` et retourne une série bivariée (X, Y) de taille `n` générée selon la procédure suivante :
   - Les valeurs de X sont régulièrement espacées entre 0 et 1.
   - Les valeurs de Y sont définies comme une fonction de X, avec $y_i = \text{pente} \times x_i + \epsilon_i$:
     - La constante pente est choisie uniformément entre 0 et 0,75 ;
     - Le terme d'erreur propre au i-ème individu, ϵ_i, est tiré selon la loi normale centrée en 0 et d'écart-type σ ;
     - La constante σ est choisie uniformément entre 0,01 et 0,2.
4. Tester la fonction.

Exercice 2 – Jouer un tour

1. Importer les fonctions utiles :

```
from scipy.stats import pearsonr # mesure le coefficient de corrélation linéaire et la p-valeur
from matplotlib.pyplot import scatter # configure un graphique en nuage de points
from matplotlib.pyplot import show # affiche les graphiques
```

2. Définir la fonction `play_turn` qui :
   - Appelle la fonction définie précédemment pour générer entre 20 et 30 individus ;
   - Mesure le coefficient de corrélation linéaire, aussi appelé coefficient de corrélation de Pearson, entre X et Y ;
   - Affiche le nuage de points ;
   - Lit l'estimation du coefficient, saisi au clavier par le joueur ;
   - Retourne le coefficient mesuré et celui estimé par le joueur.
3. Tester la fonction.

Exercice 3 – Jouer une partie

1. Définir la fonction `play_game` qui permet d'enchaîner les tours, en gérant les vies et les pièces et retourne le nombre de pièces accumulées lorsque le joueur perd. Les nombres de vies et de pièces obéissent aux règles suivantes :
   - Si la différence absolue entre les deux coefficients est inférieure ou égale à 0,05, le joueur gagne une vie (sans pouvoir dépasser 3 vies) et gagne 5 pièces ;
   - Si la différence absolue est comprise entre 0,05 et 0,1, le joueur gagne une pièce ;

- Si la différence absolue est supérieure à 0,1, le joueur perd une vie.
- Si le joueur enchaîne 5 différences inférieures ou égales à 0,1, il reçoit 5 pièces bonus.

2. Tester la fonction.

Exercice 4 – Jouer au jeu

1. Écrire le programme principal qui permet au joueur de joueur autant parties qu'il le souhaite.
2. Modifier le programme principal pour suivre le nombre de pièces gagnées et alerter le joueur quand il atteint un nouveau record.

Exercice 5 - Analyser le jeu

1. Importer les fonctions utiles :

```
1 from matplotlib.pyplot import hist, legend, title
```

1. Modifier la fonction `play_game` de sorte à ce qu'elle retourne aussi la liste des coefficients réels et estimés au cours du jeu.
2. Modifier le programme principal pour mémoriser tous ces coefficients au fil des parties.
3. Visualiser les distributions des coefficients réels et estimés à l'aide de deux histogrammes.