Code de partage avec Capytale : 3c1a-1309855 **Corrigé**

Préparation

Nous utiliserons toujours le même jeu de données

```
import pandas
films=pandas.read_csv('films.csv', delimiter=';' ,encoding='utf8')
```

Calculer des indicateurs et suivre leur évolution

1 Calculer des indicateurs

Même si Python possède généralement des fonctions qui répondent aux questions, un objectif du jour est de voir comment manipuler dans le détails les données (« une par une »).

1. Sélectionner les films de l'année 1991 (on pourra créer un extrait).

   ```
   extrait=films[films['release_year']==1991]['revenue']
   ```

   La commande `films[films['release_ year']==1991]` permet de ne garder du jeu de données initiales, que les lignes de l'année 2004. L'ajout de `['revenue']` permet de ne garder que la colonne des revenus, qui contient les valeurs que l'on souhaite analyser. Enfin la définition d'une nouvelle table de données `extrait` vise à faciliter les commandes ultérieures.

2. Puis calculer la moyenne des revenus au cours de cette année et sans utiliser les fonctions Python prédéfinies.

   Pour parcourir les valeurs, on pourra utiliser la commande suivante

   ```
   for i in extrait.index :
     extrait[i]
   ```

   On pourra vérifier avec les fonctions prédéfinies dans Python.

   L'inconvénient de la table `extrait` créée ci-dessus est que les données sont indexées par les numéros de lignes de la table initiale (donc l'enchainement peut être irrégulier, i.e. pas de 1 en 1), mais la commande `for i in extrait.index` permet de pallier cette difficulté en parcourant une table sans connaitre, a priori, ses indices de lignes.

   ```
   somme=0
   population=0
   for i in extrait.index :
       somme=somme+extrait[i]
       population=population+1
   moyenne=somme/population
   print("la moyenne vaut", moyenne)

   print(extrait.mean()) # pour vérifier
   ```

   Le jeu de données contient des lignes où les données ne sont sans doute pas totalement renseignées car plusieurs cellules « revenue » contiennent 0
   On peut ne pas les prendre en compte, en ajoutant `if extrait[i]>0 :` avant `population=population+`
   (attention dans ce cas-là, on ne trouve plus la même valeur qu'avec la commande `extrait.mean()`

3. De même calculer l'écart-type des revenus des films en 1991.

   De manière analogue, on calcule la moyenne des carrés puis on utilise la formule de Kœnig-Huygens pour calculer la variance et enfin l'écart-type (en utilisant la moyenne calculée précédemment. On peut comparer avec `extrait.std()`

```
import numpy as np
somme2=0
for i in extrait.index :
    somme2=somme2+extrait[i]**2
moyenne_carres=somme2/population
variance =moyenne_carres - moyenne**2
ecart_type=np.sqrt(variance)

print(extrait.std()) # pour vérifier
```

   La vérification n'est pas concluante, car on trouve une valeur différente dans les deux cas. En fait Python utilise une définition différente de la nôtre pour l'écart-type. Il calcule la somme des carrés des écarts à la moyenne, puis divise cette somme par $n-1$ (et non n comme nous) où n est l'effectif total. D'où la différence (d'autant que la formule de Kœnig-Huygens n'est alors pas valable).

2 Etudier l'évolution des indicateurs

A l'aide des indicateurs et commandes précédents, étudier l'évolution des revenus de 1991 à 2010. On pourra également utiliser les outils graphiques vus au T.P. 16.

Le programme est le même qu'au T.P. précédent, ici pour étudier l'évolution des revenus moyens :

```
revenue=[]
for i in range(1991,2010):
    revenue.append(films[films['release_year']==i]['revenue'].mean())

x=[i for i in range(1991,2010)]
import matplotlib.pyplot as plt
plt.bar(x,revenue)
plt.show()
```

Et pour l'évolution des écarts-types, on change simplement le `.mean()` en `.std()` :

```
revenue=[]
for i in range(1991,2010):
    revenue.append(films[films['release_year']==i]['revenue'].std())

x=[i for i in range(1991,2010)]
import matplotlib.pyplot as plt
plt.bar(x,revenue)
plt.show()
```