3 Boucles

3.1 Entraînement

- Ecrire des algorithmes qui affichent les résultats suivants :
  - `1 2 3 4 5`
  - `3 2 1 0 -1 -2`
  - `0 5 10 15 … 100`
  - `10 5 0 -5 -10`
- Refaire cet exercice avec les différents types de boucle.

3.2 Validation des données

- Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.
- Ecrire un algorithme qui demande à l'utilisateur une chaine de caractères, et qui redemande tant que l'utilisateur donne une chaîne vide (= 0 caractères).

3.3 Calculatrice un peu plus évoluée

- Reprendre l'algorithme de l'exercice 2.8 – Calculatrice simple. Faire en sorte que le menu soit reproposé à la fin de chaque calcul, et propose en plus un 5ème choix pour terminer le programme.

3.4 Table de multiplication

- Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

```
Table de 7 :
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
...
7 x 10 = 70
```

3.5 Table de multiplication en double boucle

- Ecrire l'algorithme correspondant à l'affichage suivant :

```
1     2     3     4     5     6     7     8     9     10
2     4     6     8     10    12    14    16    18    20
3     6     9     12    15    18    21    24    27    30
4     8     12    16    20    24    28    32    36    40
5     10    15    20    25    30    35    40    45    50
6     12    18    24    30    36    42    48    54    60
7     14    21    28    35    42    49    56    63    70
8     16    24    32    40    48    56    64    72    80
9     18    27    36    45    54    63    72    81    90
10    20    30    40    50    60    70    80    90    100
```

3.6 Jeu de recherche d'un nombre

- Ecrire un jeu de recherche d'un nombre entre 1 et 100. L'ordinateur choisit un nombre entre 1 et 100 avec une fonction de type **`nombreATrouver ← random(1, 100)`**. Ensuite, le jeu vous propose de saisir un nombre, et indique si ce nombre est plus petit ou plus grand que le nombre à trouver. Vous avez 10 chances. Si au bout de 10 essais vous n'avez pas trouvé, vous avez perdu.
- Ecrire le script python qui correspond à cet algorithme. La génération d'un nombre aléatoire entre 1 et 100 peut se faire avec les commandes **`import random`** puis **`nombreATrouver=random.randint(1,100)`**.

3.7 Force brute

- Un code de carte bancaire est constitué de 4 chiffres. Ecrire un algorithme qui permet de deviner un code de carte bancaire en « force brute », c'est-à-dire en testant toutes les solutions jusqu'à ce qu'il trouve. L'algorithme devra afficher également le nombre de tentatives effectuées. En imaginant que chaque test prenne 1 seconde, quelle serait la durée moyenne pour deviner ce code s'il avait 4 chiffres ? 6 chiffres ? 8 chiffres ?