Exercices Python

1 - Les bases de la programmation :

1.1 Ce qu'est une variable :

Définition : Une variable correspond à un emplacement dans la mémoire de l'ordinateur auquel on donne :

- **un nom : qui doit commencer par une lettre et ne contenir que des lettres non accentuées, des caractères _ et des chiffres,**
- **une valeur : qui est modifiable,**
- **un type : entier, réel, etc..., que l'on peut obtenir en Python avec la fonction type.**

1.2 Affichage :

Pour afficher la valeur d'une variable, une valeur ou du texte on utilise la fonction print
A l'intérieur des parenthèses, on place notre variable que l'on souhaite afficher.

```
print(expression à afficher)
```

1.3 Saisie :

Pour rendre un programme interactif, on peut demander à l'utilisateur de saisir du texte ou une valeur grâce à la fonction input.

```
a = input("message")
```

1.4 Les Opérateurs :

Python permet d'effectuer les opération classiques sur les nombres

Addition	**+**
Soustraction	**-**
Multiplication	*****
Division	**/**
Division entière	**//**
Puissance	******
modulo	**%**

Exercice 1 : Affiche âge

Faire un programme qui affiche ton âge :

- **tu rentre ta date de naissance et le programme doit afficher ton âge**

Exercice 2 : Epicerie

Une épicerie vend de la farine à 2 euros le kilo, et des tablettes de chocolat à 2,5 euros. Écrire un algorithme qui :

- **demande de saisir un nombre de kilos de farine,**
- **demande de saisir un nombre de tablettes de chocolat,**
- **affiche le prix à payer au total pour la farine et le chocolat.**

Exercice 3 : Bob et ses billes

Bob possède un certain nombre de billes. Il veut regrouper ses billes dans des sacs, en mettant 17 billes par sac. Écrire un algorithme qui :

- **demande de saisir un nombre de billes,**
- **affiche le nombre de sacs qui seront pleins,**
- **affiche le nombre de billes restantes.**

Exercice 4 : Bob part en voyage aux États-Unis.

- **Écrire en Python un algorithme qui demande de saisir une somme en euros, qui convertit cette somme en dollars (1 euro vaut environ 1,17 dollars) et qui affiche le résultat.**
- **Bob veut ramener des chapeaux de cowboy de son voyage. Il trouve des chapeaux à 3,99 dollars pièce. Compléter l'algorithme précédent en affichant combien de chapeaux pourront être achetés au maximum avec la somme saisie à la question a et combien il restera alors de dollars.**

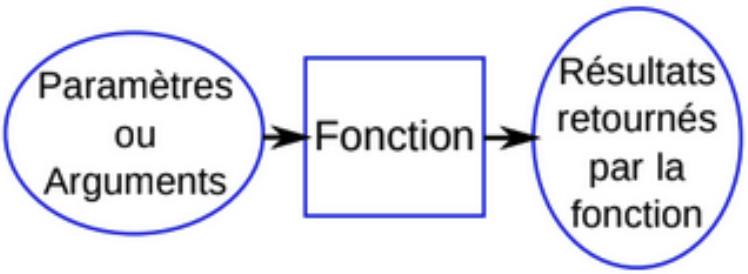
Exercice 5: distributeur

- **Écrire un programme qui demande de saisir une somme en centimes, puis qui donne le nombre de pièces minimum de 2 centimes et 1 centime nécessaires pour payer cette somme.**

Par exemple, si l'utilisateur saisit 17 centimes, l'ordinateur affichera 8 pièces de 2 centimes et 1 pièce de 1 centime.

1.5.1 - Fonctions et Procédures :

Nous avons déjà manipulé des fonctions et procédures comme print et input.
Une fonction est un morceau de code pouvant recevoir en entrée une ou plusieurs valeurs à partir desquelles elle retourne une ou plusieurs valeurs en sortie.
Les valeurs que la fonction reçoit en entrée s'appellent les paramètres ou les arguments de la fonction.
Les valeurs retournées par la fonction sont les résultats de celle-ci.



1.5.2 - Appel d'une procédure :

```
nom_procédure(paramètres)
```

1.6 - Importation de Bibliothèque

Pour charger une bibliothèque, il faut d'abord l'importer grâce au mot import.
Pour utiliser des nombres aléatoires on va importer la bibliothèque random

```
from random import *
# utilisation :
print(randint(0,10))
```

Grâce à l'importation de random, on peut utiliser les fonctions qui proviennent de cette bibliothèque, si on veut savoir toutes les fonctions qui existe dans cette bibliothèque on peut faire la commande suivante :

```
help("random")
```

Exercice 6 (Pour se tester)

1) Qu'est ce qu'une fonction ?
2) Comment définit-on une fonction ?
3) Comment fait-on appel à une fonction ?
4) Quelle est la différence entre procédure et fonction ?
5) Comment fait-on appel à une procédure ?
6) Qu'est une variable locale ? Qu'est ce qui la caractérise ?
7) Comment importe-t-on un module ?

Exercice 7 :

Écrire une fonction produit prenant en paramètres d'entrée deux nombres entiers et retournant leur produit.
exemple : le produit(6, 7) devra retourner 42.

Exercice 8 :

Écrire une fonction echange prenant en paramètres d'entrée deux nombres entiers et retournant ces entiers dans l'ordre inverse.
exemple : echange(12, 98) devra retourner 98, 12.

Exercice 9 :

Écrire une fonction deux_des qui ne prend pas de paramètre et retourne deux entiers aléatoires entre 1 et 6.
exemple : la fonction pourra retourner 4, 2.

Exercice 10 : Conversion duree

Écrire en pseudo-code et en Python une procédure conv_duree prenant en paramètre une durée en secondes puis affichant cette même durée en heures, minutes et secondes.
exemple : conv_duree(7422) affichera "2 heures, 3 minutes et 42 secondes".

2 Les structures conditionnelles :

2.1 La structure :

```
if condition 1 :
    instructions 1
elif condition 2 :
    instructions 2
elif condition 3 :
    instructions 3
# etc...
else :
    autres instructions
```

```
# Algo : conditions et dé
from random import *

de = randint(1,6)
print("de = ", de)
if de <= 2 :
    print("on a 1 ou 2")
elif de < 6 :
    print("on a 3, 4 ou 5")
else :
     print("on a 6")
```

Egal	**==**
Différent	**!=**
Plus grand	**>**
Plus petit	**<**
Plus grand ou égal	**>=**
Plus petit ou égal	**<=**

Exercice 11 : Mot de passe

Le mot de passe d'un ordinateur est "James-Bob". Écrire un algorithme qui demande à l'utilisateur de saisir un mot de passe, et qui affiche un bip d'alarme si le mot de passe saisi est incorrect.

Exercice 12 : Le cinéma

Une place dans un cinéma coûte 5 euros pour les moins de 10 ans, 6 euros pour les moins de 18 ans et 8 euros pour les autres. Écrire une procédure prix qui prend en paramètre un âge et affiche le prix correspondant.

Exercice 13 : Plus grand

Écrire une fonction maximum prenant en paramètres 2 nombres entiers et retournant le plus grand des deux.

Exercice 14: Calcul mental

On veut réaliser un programme de calcul mental. Écrire un algorithme qui :

- **tire aléatoirement deux entiers entre 1 et 10,**
- **demande de saisir le produit de ces deux entiers,**
- **affichage "Gagné !" si le produit saisi est correct, et affiche "Perdu !" ainsi que le vrai résultat si le produit saisi est incorrect.**

Exercice 15: Vacances

Bob ne va à la plage que si l'une des deux conditions suivantes au moins est satisfaite :

- **la température de l'air est supérieure à 25 degrés,**
- **La température de l'eau est supérieure à 20 degrés.**

Écrire un algorithme demandant la température de l'eau et de l'air, et affichant un message si Bob peut aller à la plage.

Exercice 16 : Roulette simplifié

Un sac contient 10 balles numérotées de 1 à 10 et telles que : les balles 1 et 2 sont vertes, les balles 3, 4 et 5 sont bleues et les autres balles sont rouges. Bob vous propose un jeu : vous devez parier sur une couleur, puis, il tire une balle au hasard, et si cette balle est de la couleur que vous avez choisie, vous avez gagné.

1) Écrire d'une fonction tirage qui :

- **ne prend pas de paramètre,**
- **simule le tirage d'une balle et affiche son numéro,**
- **retourne une chaîne de caractères donnant le numéro de la balle.**

2) Écrire un algorithme qui :

- **demande de saisir une couleur,**
- **réalise un tirage de balle grâce à la fonction tirage,**
- **affichage la couleur de la balle tirée,**
- **affichage un message indiquant si l'on a gagné ou non.**

3 - Les boucles :

définition : Une boucle (ou structure itérative), est une structure permettant de répéter plusieurs fois l'exécution d'un bloc d'instructions. Il en existe deux en Python : la boucle "while" et la boucle "for".

3.2 Boucle while :

La boucle while (Tant que) permet de répéter un bloc d'instructions tant qu'une condition
est vérifiée. Lorsque la condition n'est plus vérifiée, on sort de la boucle. Une telle structure se formalise de la façon suivante :

```
# Algo : affichage de 1 à 10


i = 1
while i <= 10 :
    print(i)
    i = i + 1

print("Fin de la boucle")
```

```
while condition :
    instructions
```

exemple d'un lancer de dé avec la boucle while :

```
# Algorithme : lancer de dé
from random import *

de = 0
compteur = 0
while de != 6 :
    de = randint(1,6)
    print("score : ", de)
    compteur = compteur + 1

print("nb de lancers : ",
   compteur)
```

- **Attention**, si la boucle Tant que est mal utilisée, elle peut donner lieu à une exécution infinie. Pour que l'exécution de la boucle Tant que se termine, il faut qu'à un moment la condition devienne fausse. En particulier, les instructions dans la boucle doivent modifier la variable utilisée dans le test.
- Une fois encore, il ne faut pas oublier le " :", ni d'indenter le bloc d'instructions.
- La boucle Tant que est une boucle non déterministe : on ne sait pas forcément à l'avance combien d'itérations seront effectuées. Il arrive qu'elle ne soit pas exécutée du tout lorsque la condition est fausse dès le départ.

3.3 Boucle for :

La boucle for (Pour) utilise une variable, que l'on appelle compteur, qui sert à repérer le nombre d'itérations déjà effectuées. Le compteur part d'une valeur de départ et augmente à chaque itération d'un nombre xe, appelé pas, jusqu'à atteindre une valeur d'arrivée. Pour chaque valeur du compteur, le bloc d'instructions est exécuté une fois. La boucle Pour se formalise de la façon suivante :

```
for compteur in range(depart, arrivee, pas) :
    instructions
```

```
# Algorithme : affichage des nombres pairs de 0 à 100
for compteur in range(0,101,2) :
    print(compteur)
```

- La boucle "Pour" n'est utilisée que lorsque le nombre d'itérations est connu dès le départ.
- Attention, en python, la valeur de départ est toujours incluse, mais la valeur d'arrivée est toujours exclue. Ainsi, si l'on écrit for compteur in range(a,b), le compteur ira de a à b-1. En pseudo-code, il faut préciser (systématiquement) si b est exclu ou inclus.
- Dans la boucle "Pour" la valeur du compteur est modiée automatiquement.

Exercice 17 (Pour se tester)

1) À quoi sert une boucle ?
2) Quels sont les deux types de boucle en Python ?
3) Quelle est la différence entre les deux types de boucle en Python ?
4) Comment faire chacun des deux types de boucle ?

Exercice 18 :

Un compte informatique de Bob est protégé par le mot de passe "James-Bob". Écrire en pseudo-code puis en Python un algorithme qui demande de saisir un mot de passe jusqu'à ce que le bon mot de passe soit saisi.

Exercice 19 :

Écrire un algorithme qui :
• simule les tirages successifs d'un dé jusqu'à ce qu'un 2 apparaisse,
• afficher le nombre de lancers effectués pour obtenir le 2.

Exercice 20 :

Dans la mare qui est dans son jardin, Bob a mis un nénuphar. Au début, la surface du nénuphar est de 9 cm2

- **Tous les mois, la surface du nénuphar augmente de 20%.**

a) Écrire une fonction seuil prenant en paramètre une surface de mare, et retournant le
nombre de mois mis par le nénuphar pour dépasser cette surface.
b) Tester la fonction seuil sachant que la surface de la mare est de 6 m2
(c'est-à-dire 60 000cm2)

Exercice 21 :

Bob se lance dans la vente de plants de nénuphars. Il en vend 3 le premier jour. Le nombre de plants vendus double ensuite chaque jour.
a) Écrire un algorithme affichant le nombre de plants vendus chaque jour jusqu'au trentième jour.
b) Modifier l'algorithme précédent pour calculer le nombre total de plants vendus en 30 jours.

Exercice 22 :

a) Écrire une procédure prenant en paramètre un entier, et affichant la liste des diviseurs de cet entier.
b) Modifier la procédure précédente en une fonction retournant le nombre de diviseurs de l'entier choisi comme paramètre.

Exercice 23 : Le jeu du nombre à deviner

Le but du jeu est de deviner un nombre entier entre 0 et 50 choisi aléatoirement par l'ordinateur. Écrire un algorithme qui :

- **tire au hasard un entier entre 0 et 50 qui sera le nombre cible,**
- **répète les étapes suivantes jusqu'à ce que le nombre cible soit deviné : demander de saisir un entier entre 0 et 50 afficher un message disant si l'entier saisi est égal plus petit ou plus grand que le nombre choisi par l'ordinateur**
- **afficher un message de victoire.**

Exercice 24 : Jeu du 1 fatal :

On veut jouer à un jeu à deux joueurs dont les règles sont les suivantes :

- **Le joueur 1 commence.**
- **Lorsque c'est son tour, chaque joueur lance un dé. Il peut relancer son dé autant qu'il veut jusqu'à 9 fois (10 lancers en tout), mais si il tombe sur un 1, il doit s'arrêter. Le joueur gagne alors : 0 points si il est tombé sur un 1, la somme de ses dés sinon.**
- **Le premier joueur à avoir 100 points ou plus gagne.**

1) Écrire une fonction tour qui :

- **ne prend pas de paramètre,**
- **simule un tour de jeu, en respectant les spécifications suivantes :**

À chaque lancer de dé, la valeur du dé doit être affichée,
Si le dé tombe sur 1 ou si 10 lancers ont été effectués, le tour prend fin.
Sinon, après le lancer, le programme doit demander au joueur s' il veut relancer.
Le joueur devra alors saisir 0 pour arrêter et 1 pour relancer.

- **Affiche puis retourne le nombre de points gagnés au cours du tour.**

2) Écrire l'algorithme du jeu. Cet algorithme devra :

- **initialiser les scores des deux joueurs à 0,**
- **répéter jusqu'à ce que l'un des joueurs ait atteint ou dépassé 100 points :**

afficher le score des deux joueurs,
afficher un message indiquant que c'est au joueur 1 de jouer,
faire jouer le joueur 1 grâce à la fonction tour
si le joueur 1 n'a pas atteint 100 points, recommencer les trois étapes précédentes pour le joueur 2

- **Afficher les scores finaux et le gagnant.**