2 Les variables

Notions

- Une **variable** est un emplacement mémoire qui permet de stocker des objets de **différents types**.
- Le **nom d'une variable** ne peut contenir que des caractères alphanumériques (A-Z, a-z, 0-9) et le tiret du bas (_). Il ne peut pas commencer par un chiffre ni contenir d'espaces. Certains mots réservés ne sont pas autorisés (*in, and, if, for, else, while,* etc.)

Outils

- `nom = valeur` : on déclare une variable en lui affectant une valeur. « = » est un signe d'affectation. Il est différent d'une égalité au sens mathématique.
- `Type de variable` : *str* : chaîne de caractères, *int* : nombre entier, *bool* : Booléen *(True* ou *False)*, *float :* nombre à virgule, *list :* liste.
- `type()` : indique le type de la variable.
- `str(), int() et float()` : convertissent le type des données en *str*, *int* et *float*.

Exercices

1. Analyser le script et sa réponse puis donner la valeur et le type des variables a, b et c.

`print()` : permet d'afficher des données.

```
1 a = bonjour
2 b = 15
3 c = True
4 print(a,type(a))
5 print(b,type(b))
6 print(c,type(c))
```

Réponse :

```
bonjour <class 'str'>
15 <class 'int'>
True <class 'bool'>
```

Nom	Valeur	Type
a		..
b	....	... ...
c	....	..

2. Les variables ci-dessous sont-elles valides ? Si non, proposer un nom valide.

Nom de la variable	Valide ?		Nom valide
#prénom1	☐ Oui	☐ Non	....
ageJoueur	☐ Oui	☐ Non	.....
and	☐ Oui	☐ Non	....

3. Qu'affiche le script suivant ? Le script affiche : bac***.....

```
1   a,b,c,d = "a", "b", "c", "d"
2   a,b = b,a
3   d = "***"
4   print(a,b,c,d)
```

Travaux pratiques 40 mn

A. Tester et comprendre

1. Indiquer le résultat affiché par les scripts suivants.

```
1 d = 1
2 d = d + 2
3 e = str(d) + "3")
4 print(e)
```

Le script affiche :

☐ 6.
☐ 123,0.
☐ 33.

```
1 x = int(input("x:"))
2 plusDeux = x + 2
3 print(plusDeux)
```

`input()` : permet d'entrer des données.

On entre 10, le script affiche :

☐ 12.
☐ erreur.
☐ 102.

2. Tester les scripts pour vérifier vos réponses.

B. Faire connaissance avec l'ordinateur

```
1   # prenom = input("Entrez votre prénom:")
2   # age = input("Entrez votre âge:")
3   # age = int(input("Entrez votre âge:"))
4   # plusUn = age + 1
5   # print("Bonjour" + prenom + ", vous avez"
    + age + "ans")
6   # print("Bonjour" + prenom + ", vous avez"
    + str(age) + "ans")
7   # print("Vous en aurez" + str(plusUn) +
    "dans un an !")
```

Une ligne commentée, c'est-à-dire débutant par un #, n'est pas exécutée.

- Recopier le code précédent puis commenter ou décommenter les lignes afin de tester les scripts indiqués. Relier chaque script à la réponse constatée et à la solution lui correspondant.

Script	Réponse constatée	Solution
Lignes 1, 3 et 5	Erreur L5 : concaténation *str* et *int* !	int(input()) (1, 3, 4, 7)
Lignes 1, 2, 4 et 7	Script complet et correct.	str() (1, 3, 6) ou (1, 2 ,5)
Lignes 1, 3, 4, 6 et 7	Erreur L4 : concaténation *str* et *int* !	Rien à faire.

3 Les conditions (if, elif, else)

Notions

- En Python, on utilise des **structures de contrôle** comme les **conditions** et les boucles.
- Un test est effectué en utilisant une condition (ou **prédicat**). La condition est évaluée et renvoie vrai (**True**) ou faux (**False**).
- Les **structures conditionnelles** permettent d'exécuter des **instructions** en fonction des conditions.

Outils

- `if` : signifie « si ». Il évalue si le résultat du test est vrai (et retourne `True`) ou faux (et retourne `False`).
- `else` : signifie « sinon ».
- `elif` : signifie « sinon si ».
- Les conditions peuvent utiliser des opérateurs de comparaisons (`==, !=, <, >, <=, >=`) et des opérateurs logiques (`not, and, or`).
- L'indentation (espaces) : permet de définir un bloc d'instructions ainsi que les imbrications entre les blocs.

Exercices

1. Compléter le tableau, en indiquant pour chaque prédicat, s'il est vrai ou faux.

Conditions	Vrai	Faux
a. "7" != 7	☐	☐
b. 9 < 2	☐	☐
c. not(False)	☐	☐
d. 6 == 4 + 2 and 6 > 8	☐	☐

2. Compléter le script suivant afin qu'il affiche « Admis » lorsque la note entrée est supérieure ou égale à 10.

```
1   note = int(input("Note"))
2   if note ..... :
3       print(.....)
```

3. Que réalise le script suivant ?

```
1   note = int(input("Note :"))
2   if note >= 10 :
3       print("Admis")
4   else :
5       print("Non admis")
```

Affichage obtenu :

.....

.....

.....

Travaux pratiques 30 mn

A. Contrôle de l'âge

- Compléter et tester le script suivant afin d'afficher un message d'accès pour chaque catégorie d'âge.

```
1   age = int(input("Entrez votre âge :"))
2   if .....
3       print("Accès à partir de 18 ans")
4   elif age >= 16 :
5       print("Accès à partir de 16 ans")
6   .....
7       print("Accès à partir de 12 ans")
8   .....
9       print("Accès interdit !")
```

B. Commande d'un volet programmable

- Le script suivant permet de commander un volet programmable. Le volet se ferme la nuit. Le jour, il s'ouvre automatiquement à partir de 9 heures ou bien s'il détecte un mouvement. Compléter et tester le script.

```
1   capteur... = input("Entrer j (si jour), autre lettre (si nuit): ")
2   heure = int(input("Entrer heure : "))
3   mouvement = input("Entrer m (si mouvement), autre lettre sinon : ")
4   if capteur == "j" :
5   .....
6           print("Ouverture volets")
7       else :
8           print("Volets fermés")
9   ..... .....
10      print("Fermeture volets")
```

Défi ! Proposer un script qui demande la note obtenue puis affiche « Assez bien » à partir de 12, « Bien » à partir de 14 et « Très bien » à partir de 16.

4 Les boucles bornées (for... in)

Notions

- Les **boucles** servent à **répéter** plusieurs fois une ou plusieurs opérations.
- Elles permettent également de **parcourir des données** de types chaîne de caractères, listes, etc.
- Les boucles sont dites **bornées** si le nombre d'**itérations** (répétitions) est connu à l'avance. On utilise l'instruction ***for***.

Outils

- Structure d'une boucle *for* :

```
for i in séquence :
    bloc d'instructions
fin boucle for
```

- *i* va prendre les valeurs successives de *séquence*.
- `range(n)` : fonction qui donne une liste de n éléments allant de 0 à n-1.

Exercices

1. Que réalisent les scripts suivants ? Cocher la bonne réponse.

a.

```
for i in range(10):
    print(i)
```

- ☐ Affiche 0 à 10.
- ☐ Affiche 0 à 9.
- ☐ Affiche 1 à 10.

b.

```
for i in range(3):
    for j in range(11):
        print(i*j)
```

- ☐ Affiche les tables de multiplication de 0 à 2.
- ☐ Affiche 3 à 11.
- ☐ Affiche 1 à 10.

2. Compléter le script à l'aide des étiquettes suivantes afin d'afficher les tables de multiplications de 0 à 9.

i,"×",*j*,"=",*i***j* ; range(10) ; for j in ; "Table de",i .

```
for i in .... ..... ..... :
    print(.....)
    ..... range(11):
        print(.....)
```

Les boucles *for* peuvent s'imbriquer grâce à l'indentation.

5 Les boucles non bornées (while)

Notions

- Les boucles sont dites **non bornées** quand le nombre d'itérations n'est pas connu à l'avance. On utilise, dans ce cas, l'instruction ***while*** (« tant que » en français).
- Les boucles non bornées doivent avoir une **condition de sortie** afin d'éviter qu'elles s'exécutent à l'infini.

Outils

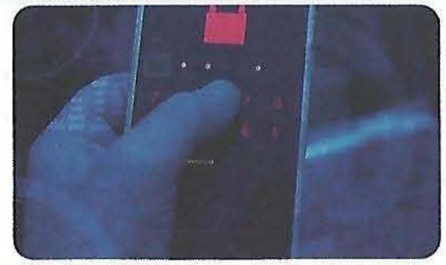
- Structure d'une boucle *while* :

```
i=0
while condition :
    bloc d'instructions
    i+=1
```

- `i=0` : avant la boucle, on **initialise** i à 0.
- `i+=1` : signifie i = i +1. Cette instruction permet d'incrémenter une variable (l'augmenter de 1).

Exercice

- Compléter le script suivant à l'aide des commentaires (#).



```
codePin, pinSaisi = "1234", input("Entrez votre code PIN :")
# initialiser compteur à 0
.....
# tant que pinSaisi ≠ codePin et compteur ≠ 2
.....
    compteur += 1
    print("Réessayez...", "Il vous reste", 3-compteur,
    "essai(s)")
    pinSaisi = input("Entrez à nouveau votre code PIN:")
# si pinSaisi est égale à codePin
.....
    print("Code PIN correct. Bienvenue !")
# sinon
.....
    print("Essais dépassés. Compte bloqué !")
```

6 Définir une fonction

Notions

- Une **fonction** est une partie d'un programme informatique qui accomplit une tâche spécifique.
- **Définir une fonction** c'est expliquer à la machine ce que doit réaliser la fonction.
- Une fonction qui ne retourne pas de résultat s'appelle une **procédure**.

Outils

- `def nom_de_la_fonction(a,b)` : permet de définir une fonction en lui donnant un nom et des paramètres `(a,b)`. Si la fonction ne possède pas de paramètres on écrit : `def nom_de_la_fonction ()`. La ligne se termine par : .
- `return` : retourne le résultat d'une fonction.

Exercices

1. Corriger le script de la fonction `inferieur(a,b)` suivante.

```
1 definition inferieur[a,b]
2       if a<b :
3           renvoi True
4       else :
5           renvoi False
```

```
1 .....
2 .....
3 .....
4 .....
5 .....
```

Tout le code contenu dans une fonction est indenté d'un niveau.

2. Dans le script suivant, surligner en jaune le nom de la fonction et en bleu ses paramètres. Indiquer ce qu'elle réalise et s'il s'agit d'une fonction ou d'une procédure.

```
1   def parite(nombre):
2       if nombre%2==0 :
3           print("paire")
4       else :
5           print("impaire")
```

Action : .....

.....

☐ Fonction ☐ Procédure

3. Écrire une fonction `addition(a,b)`, qui prend en paramètres deux nombres a et b et retourne leur somme.

```
1 .....
2 .....
```


7 Appeler une fonction

Notions

- **L'appel de fonction** est l'opération qui permet d'exécuter le code contenu dans une fonction.
- Une **bibliothèque** est un ensemble de fonctions, de classes et d'autres éléments qui peuvent être utilisés dans un programme.
- Exemple de bibliothèques : math, random, networkx, folium, pillow, tkinter, micro:bit.

Outils

- `afficher_bonjour()` : appel la fonction par son nom.
- `resultat = fonction()` : stock le résultat de la fonction dans une variable `resultat`.
- `import random` : charge la bibliothèque random.
- `random.randint()` : appel de la fonction `randint()` de la bibliothèque random.

Exercice

- En utilisant les fonctions suivantes, écrire les appels de fonctions correspondant aux actions indiquées.

```
1   # Appel de la fonction afficher_bonjour sans paramètre
2   afficher_bonjour() # affiche « Bonjour ! »
3   # Appel de la fonction calculer_carre avec un paramètre
4   carre = calculer_carre(3) # carre contiendra la valeur 9
5   # Appel de la fonction addition avec deux paramètres
6    somme = addition(2, 3) # somme contiendra la valeur 5
```

a. Afficher « Bonjour ! » : .....

b. Calculer le carré de 7 : .....

c. Calculer le carré de -2 : .....

d. Faire l'addition de 1 et 5 : .....

e. Faire l'addition du carré de -3 et de 4 : .....

f. Faire le carré de (4+11) : .....

Il possible d'appeler une fonction dans une fonction.

Défi !

La bibliothèque *Turtle* permet de dessiner à l'écran.
En utilisant cette bibliothèque, dessiner un damier à l'écran.