Si vous devez enregistrer votre travail

Sur votre espace réseau personnel de l'ENT, dans le dossier NSI \ PROGRAMATION, créer à l'intérieur un sous dossier Fiche2.

I. Variable Booléenne

Vrai ou faux ?

Si quelqu'un vous dit que « 4 est égal à 5», vous lui répondez quoi ? **«C'est faux».**
Si maintenant la même personne vous dit que «7 est égal à 7», vous lui répondrez bien évidemment que **«c'est vrai».**

En Python, ces deux «affirmations» («4 est égal à 5» et «7 est égal à 7») s'écriront «4 == 5» et «7 == 7» (**notez bien le double signe égal**).

1. Tester ce programme à l'aide de Pyzo.
   print(4 == 5) Regarder ce qu'affiche la console

2. Quel est le résultat attendu après l'exécution de ce programme dans la console ?

   7 == 7 .

Le « **double égal** » (**==**) est **l'opérateur d'égalité**.
L'opérateur d'égalité est soit **vrai (True)** soit **faux (False).**
L'utilisation de l'opérateur d'égalité va prendre tout son sens avec des variables.

3. Soit le programme suivant :

   ```
   a = 4
   b = 7
   a == b
   a = 7
   a == b
   ```

   Quels sont les résultats attendus pendant l'exécution de ce programme dans la console ?

   Vérifiez votre hypothèse en saisissant ce programme dans la console Pyzo.

ATTENTION :
Il ne faut pas confondre **l'opérateur d'égalité (==)** et **l'opérateur d'affectation (=)** utilisé pour attribuer une valeur aux variables.
La **confusion** entre ces 2 opérateurs est une **erreur classique** qui est parfois très difficile à détecter !
Il est possible d'utiliser aussi **l'opérateur « différent de » : !=**

4. Soit le programme suivant :

   ```
   a = 4
   b = 7
   a != b
   a = 7
   a != b
   ```

   Quels sont les résultats attendus pendant l'exécution de ce programme dans la console ?
   .

   Vérifiez votre hypothèse en saisissant ce programme dans la console Pyzo

Notez aussi l'existence des opérateurs :

- « strictement inférieur à » : **<**
- « strictement supérieur à » : **>**
- « inférieur ou égal à » : **<=**
- « supérieur ou égal à » : **>=**

A chaque fois **ces opérateurs sont True (vrai) ou False (faux).**

5. Soit le programme suivant :

```
a = 4
b = 4
a < b
a = 7
a < b
```

Quels sont les résultats attendus pendant l'exécution de ce programme dans la console ?

Vérifiez votre hypothèse en saisissant ce programme dans la console Pyzo.

Pour terminer, notez qu'une variable qui ne peut contenir que True ou False est de type booléen.

II. Les conditions

Nous allons maintenant étudier une structure fondamentale en programmation

le « si ..... alors.....sinon.....».

L'idée de base est la suivante :

```
si expression :
    suite_instruction1
sinon:
    suite_instruction2
```

Si « **condition » est vraie** (**True**) alors « **suite_instruction1 » est exécuté** et « suite_**instruction2** » est ignoré.

Sinon (sous-entendu **que « expression » est Fausse**) « **suite_instruction2** » est **exécuté** et « suite_instruction1 » est ignoré.

Notez l'utilisation d'un élément nouveau : le **décalage de «suite_instruction1»** et de **« suite_instruction2».**
Ce décalage est appelé indentation, il permet de rendre le code plus lisible.

Dans le cas de l'**utilisation** d'un **si/alors**, **l'indentation est obligatoire en Python**.

1. **Exo1**

Soit le programme suivant :

```
a = 4
b = 7
if a < b :
      print ("Je suis toto.")
      print ("Je n'aime pas titi.")
else :
      print ("Je suis titi.")
      print ("Je n'aime pas toto.")
      print ("En revanche, j'aime le Python.")
```

Quel est le résultat attendu après l'exécution de ce programme ?
Vérifiez votre hypothèse à l'aide de Pyzo.

2. **Exo2**

Écrire un programme qui demande l'âge de l'utilisateur.
Si l'utilisateur a 18 ans ou plus, le programme devra afficher « Bonjour, vous êtes majeur.».
Si l'utilisateur a moins de 18 ans, le programme devra afficher « Bonjour, tu es mineur.»

3. **Exo3**

Soit le programme suivant :

```
def annonce(num, prov, dest):
   if dest != "0":
         msg = f"le train n° {num} en provenance de {prov} et à
         destination de {dest}, entre en gare."
   else:
         msg = f"le train n° {num} en provenance de {prov} entre en gare.
         Ce train est terminus Triffouillis-les-Oies."
   return msg
```

Quel est le résultat attendu après l'exécution de ce programme si vous saisissez dans la console " annonce("4557", "Paris", "Marseille") " ?

Et si vous saisissez dans la console " annonce("5768", "Bonneville", "0") " ?

Vérifiez votre réponse en testant ce programme avec Pyzo.

4. **Exo4**

Vous êtes gérant d'un magasin et vous désirez écrire un programme Python qui calculera automatiquement le montant de la facture des clients.
Tout client qui achète au moins 5 fois le même article se voit octroyer une remise de 5 % (uniquement sur le montant de l'achat de cet article).
Afin de simplifier le problème, on considère qu'un client n'achète qu'un seul type d'article.
Écrire une fonction qui prend en paramètre le prix unitaire de l'article et le nombre d'articles achetés.
Cette fonction doit renvoyer le montant de la facture.

III. Le "ou" et le "et"

Un **if (Si) peut contenir plusieurs conditions**, nous aurons alors une structure de la forme :

```
si condition1 op_logique condition2 :
        suite_instruction1
sinon:
        suite_instruction2
```

« op_logique » étant un opérateur logique.

Nous allons étudier **2 opérateurs logiques** :

- le **«ou»** (noté en Python **or**) et
- le **«et»** (noté en Python **and**).

Par exemple (condition1 or condition2) est vrai si condition1 est vraie et condition2 est vraie.
Autre exemple (condition1 and condition2) est faux si condition1 est vraie et condition2 est faux.

Les résultats peuvent être regroupés dans ce que l'on appelle une table de vérité :

table de vérité pour le « ou »

condition1	condition2	condition1 or condition2
vrai	vrai	vrai
vrai	faux	vrai
faux	vrai	vrai
faux	faux	faux

table de vérité pour le « et »

condition1	condition2	condition1 and condition2
vrai	vrai	vrai
vrai	faux	faux
faux	vrai	faux
faux	faux	faux

Exo5

Soit le programme suivant :

```
a = 5
b = 10
if a > 5 and b == 10:
      print ("Toto")
else:
      print("Titi")
if a > 5 or b == 10:
      print ("Tata")
else:
      print("Tutu")
```

Quel est le résultat attendu après l'exécution de ce programme ?

Vérifiez votre réponse en testant ce programme.

IV. Notion de boucle

La notion de boucle est **fondamentale** en informatique.
Une boucle permet d'exécuter **plusieurs fois** des instructions qui **ne sont présentes qu'une seule fois** dans le code.

Les boucles **"for"** et **"while"** sont **interchangeables** dans un programme, cependant la boucle "while" est souvent utilisée quand le programmeur ne connait pas à l'avance le nombre de "tours" que devra effectuer la boucle.
La boucle "for" est souvent préférée dans les cas où le programmeur connait à l'avance le nombre de "tours" que devra effectuer la boucle.

1. La boucle while

La structure de la boucle while est la suivante :

```
while expression:
      instruction1
      instruction2
suite programme
```

Tant que l' **expression reste vraie (True)**, les instructions à l'intérieur du bloc (partie indentée) seront exécutées.

Exo6

Soit le programme suivant :

```
i = 0
while i < 10:
      print(f"i vaut : {i}")
      i = i + 1
print("C'est terminé.")
```

Quel est le résultat attendu après l'exécution de ce programme ?
Vérifiez votre réponse en testant le programme à l'aide de Pyzo.

Exo7 Créer "un générateur automatique de punition"

Écrire une fonction qui prendra 2 paramètres : une chaîne de caractère (*la phrase à recopier*) et un nombre entier (*le nombre de fois où la phrase devra être recopiée*).

Par exemple : Si on passe comme paramètres à notre fonction : "Je ne dois pas discuter en classe" et 3,
la fonction devra afficher :

Je ne dois pas discuter en classe
Je ne dois pas discuter en classe
Je ne dois pas discuter en classe

Exo8 Table de multiplication

Écrire une fonction permettant d'afficher une table de multiplication.

Cette fonction devra prendre en paramètre la table désirée (de 1 à 9),
La fonction permet alors d'afficher la table demandée.

Par exemple :

si l'utilisateur demande la table des 3, le programme devra afficher :

1 x 3 = 3
2 x 3 = 6
...
...
10 x 3 = 30

2. *La boucle for*

Il existe un autre type de boucle : **la boucle for**

La structure de la **« *boucle for* »** est la suivante :

```
for i in range(a,b):
        instruction1
        instruction2
suite programme
```

Nous aurons ici une boucle où la variable i prendra toutes **les valeurs entières** comprises entre a et b (**a inclus** et **b exclu**).

Exo9 Soit le programme suivant :

```
for i in range(0,10):
    print(f"i vaut : {i}")
print("C'est terminé.")
```

Quel est le résultat attendu après l'exécution de ce programme ?
Vérifiez votre hypothèse à en utilisant Pyzo.

Cette fonction devra prendre en paramètre la table désirée (de 1 à 9),
La fonction permet alors d'afficher la table demandée.

Par exemple :

1 x 3 = 3
2 x 3 = 6
...
...
10 x 3 = 30