TP 7 : Les erreurs - À la chasse aux bugzillas !

Bienvenue dans ce TP où nous allons partir à la recherche des créatures les plus insidieuses du monde Python : les erreurs ! Ces vilaines petites bêtes se cachent partout dans le code et notre mission est de les attraper une par une.

Partie 1 : Présentation des Bugzillas

1. Erreur n°1 : La variable non définie !

Exemple :

```
# Oups, j'ai oublié de définir ma_variable
print(ma_variable)
```

Bugzilla en cause : *NameError* – Python te dira qu'il n'a jamais entendu parler de `ma_variable`. Eh oui, c'est parce qu'elle n'est pas définie !

2. Erreur n°2 : La bourde d'indentation

Exemple :

```
# Qui a bougé cette ligne ?
if True:
print("J'aime Python !")
```

Bugzilla en cause : *IndentationError* – Python aime bien quand c'est bien aligné. Si tu ne fais pas attention, il te le fera savoir ! (Et il sera grognon.)

3. Erreur n°3 : L'oubli de deux petits points (:)

Exemple :

```
# Ah ! Les deux petits points magiques !
if True
    print("Oups, pas de deux points...")
```

Bugzilla en cause : *SyntaxError* – Python a besoin de son « : » pour savoir que tu lui donnes un bloc de code à exécuter.

4. Erreur n°4 : Division par zéro

Exemple :

```
# On n'a pas encore trouvé comment diviser par zéro...
a = 10 / 0
```

Bugzilla en cause : *ZeroDivisionError* – Les maths sont formels, tu ne peux pas diviser par zéro !

5. Erreur n°5 : La confusion entre indice et numéro d'élément

Exemple :

```
# ehhh, mais il n'y a pas d'élément de cet indice !
ma_liste = [1, 2, 3]
print(ma_liste[3])
```

Bugzilla en cause : *IndexError* – Si tu essaies d'accéder à un élément qui n'existe pas, Python te le rappellera (gentiment ou pas) et ici, le dernier indice est 2 et non 3 (Python commence à 0).

6. **Erreur n°6 : Le mélange explosif de int et de str**

Ah, le mélange des types... un grand classique ! Python, lui, aime bien quand tu es bien clair. Soit tu travailles avec des nombres (`int`), soit avec des chaînes de caractères (`str`), mais mélanger les deux sans faire attention, c'est comme mélanger du lait et du citron : ça tourne mal !

Cas n°1 : L'input non converti

```
# Oups, ici on n'a pas converti l'input en nombre...
age = input("Quel âge as-tu ? ")
age += 10
print("Dans 10 ans, tu auras", age, "ans !")
```

Bugzilla en cause : *TypeError* – Le `input` renvoie **toujours** une chaîne de caractères (str), et on ne peut pas additionner une chaîne avec un entier sans conversion.

Solution : Convertir avec int() !

```
age = int(input("Quel âge as-tu ? "))
age += 10
print("Dans 10 ans, tu auras", age, "ans !")
```

Cas n°2 : La concaténation sauvage

```
# Mélanger un int dans une chaîne sans conversion... ça va exploser !
pommes = 3
a = "J'ai " + pommes + " pommes."
```

Bugzilla en cause : *TypeError* – On ne peut pas combiner des `int` directement dans une chaîne avec +. Python est strict sur ce point !

Solution : Convertir le `int` en `str`.

```
a = "J'ai " + str(pommes) + " pommes."
```

7. **Erreur n°7 : Le f-string fail !**

Les f-strings sont cools, mais ils peuvent devenir chaotiques si tu ne gères pas bien tes variables. Fais attention à ne pas laisser des variables sans définition, ou des erreurs de syntaxe se glisser dans les accolades.

Cas n°1 : Variable non définie dans un f-string

```
# On utilise une variable qui n'existe pas encore dans une f-string
nom = "Python"
print(f"Bonjour, {name}, prêt pour coder ?")
```

Bugzilla en cause : NameError – `name` n'a jamais été défini (on voulait sans doute dire `nom`).

Solution : Utiliser la bonne variable ! Remplacez `name` par `nom` dans les accolades…

Cas n°2 : Oublier les accolades dans un f-string

```
# Oups, une accolade manquante fait tout planter !
pi = 3.14159
print(f"La valeur de Pi est environ : pi}")
```

Bugzilla en cause : *SyntaxError* – Quand tu veux inclure une variable, il faut l'entourer des deux accolades `{}`. Sinon Python est perdu !

Solution : Ajouter les accolades autour de la variable.

```
print(f"La valeur de Pi est environ : {pi}")
```

Partie 2 : Chasse aux Bugzillas – Exercices pratiques

Exercice 1 : Les carottes de Python

Dans le code ci-dessous, il y a quelques petits bugs. Corrige-les !

```
# Nos carottes préférées !
carottes = 20
print("Il y a " + carottes + " carottes dans le jardin.")
```

Exercice 2 : Le combat de sumos

Un code simple qui calcule la force d'un sumo... mais il y a une erreur, sauras-tu la trouver ?

```
force_sumo = 10
force_totale = force_sumo + force_sumo *
print(force_totale)
```

Exercice 3 : La boucle infinie

Une petite boucle pour afficher les nombres de 1 à 10. Mais attention, des Bugzillas s'y cachent !

```
n = 1
while n < 10
print(n)
```

Exercice 4 : Sauve les tortues !

Voici un bout de code pour avancer une tortue sur une piste de course, mais il semble qu'une tortue ne bouge pas du tout. Pourquoi ?

```
tortues = [0, 0, 0, 0, 0]
for i in range(5):
    if tortue[i] < 100:
        tortue[i] += 10
print(tortues)
```

Exercice 5 : L'âge du capitaine (ou pas)
Tu as un petit programme pour demander l'âge de quelqu'un, mais attention, il y a des Bugzillas qui traîne... Corrige-le !

```
age = input("Quel âge as-tu ? ")
age += 5
print(f"Dans 5 ans, tu auras age ans.")
```

Exercice 6 : La potion magique de Python
Un petit sorcier veut fabriquer une potion magique et utilise une f-string... mais il a fait une erreur dans les ingrédients. Corrige-la !

```
ingredient = "larmes de dragon"
quantite = 5
print(f"Ajoute {quantite larmes de dragon} à la potion.")
```

Partie 3 : Bonus – Écris ton propre Bugzilla

Tu es prêt ? Crée un petit code où tu cacheras **au moins** 2 Bugzillas différents, et fais tester ton voisin de classe pour voir s'il les trouve. Le premier qui capture les Bugzillas gagne ! 🎉

Voilà, tu es désormais prêt(e) à traquer les Bugzillas les plus coriaces du royaume Python ! Bonne chasse ! 🎯