Python: Les boucles while - Validation

La validation de l'input utilisateur est essentielle pour s'assurer que les données saisies sont correctes et évitent les erreurs de programme. En programmation, une bonne pratique consiste à vérifier les entrées des utilisateurs avant de les utiliser dans le programme. Cela permet d'éviter les erreurs et de garantir que les données sont conformes aux attentes.

5TTR 6TTR Exploration

Validation d'un Input Utilisateur en Python

En Python, il est courant d'utiliser une **boucle** `while` pour demander l'input de l'utilisateur et **vérifier qu'il est valide**. Si l'input n'est pas valide, on redemande à l'utilisateur de saisir une valeur jusqu'à ce qu'une valeur valide soit fournie. Voici un exemple de code qui illustre comment valider qu'un nombre se trouve entre 1 et 5 :

Il existe 2 façons principales pour valider l'input utilisateur à l'aide d'une boucle `while` :

- une boucle infinie dont on sort dès qu'on a validé l'entrée utilisateur
- une boucle qui s'exécute tant que l'utilisateur n'a pas fourni de valeur valide

Vous pouvez choisir la version qui vous convient le mieux même si je trouve que la version boucle infinie est moins "propre" et moins correcte d'un point de vue algorithmique. Pourquoi? Parce dans ce cas la condition qui permet de sortir de la boucle est "cachée" quelque part dans le bloc de code au lieu de se trouver à côté du while.

D'un autre côté, cette version a l'avantage de ne devoir demander qu'une seule fois l'entrée à l'utilisateur (`input`)... alors que la version 'propre' nécessite de le faire 2x. Inspectez les programmes ci-dessous pour bien vous en rendre compte.

Avec une boucle infinie

```python
def demander_texte():

    while True:
        texte = input("Veuillez entrer un texte non vide: ").strip()
        if texte:
            return texte
        else:
            print("L'entrée ne doit pas être vide. Veuillez réessayer.")

# Utilisation de la fonction
```

© 2025 Mr. Lambrechts

```
11 texte_valide = demander_texte()
12 print(f"Vous avez entré le texte: {texte_valide}")
```

> Remaquez l'instruction `return` à la ligne 5: c'est elle qui met fin à l'exécution de la boucle. Elle a pour effet de renvoyer la valeur, et donc de mettre fin à l'exécution de la fonction et, par conséquent, de la boucle.

Voici une explication détaillée du code fourni :

Décomposition du Code

1. **Déclaration de la fonction `demander_texte()`**:
   - La fonction est définie avec le mot-clé `def` suivi du nom de la fonction `demander_texte` et des parenthèses `()`.
2. **Boucle infinie `while True`**:
   - `while True` crée une boucle infinie qui continuera à s'exécuter jusqu'à ce qu'une condition de sortie soit rencontrée. Ici, la boucle se termine lorsqu'un texte valide est saisi et retourné.
3. **Demande d'input à l'utilisateur**:
   - `texte = input("Veuillez entrer un texte non vide: ").strip()`:
     - `input("Veuillez entrer un texte non vide: ")` affiche le message et attend que l'utilisateur saisisse du texte.
     - `.strip()` est utilisé pour enlever les espaces blancs en début et en fin de la chaîne de caractères saisie. Cela garantit que les entrées composées uniquement d'espaces sont traitées comme vides.
4. **Condition pour vérifier si le texte est non vide**:
   - `if texte:` vérifie si la chaîne de caractères `texte` n'est pas vide.
     - En Python, une chaîne de caractères vide `""` est évaluée comme `False`, tandis qu'une chaîne non vide est évaluée comme `True`.
5. **Retour du texte valide**:
   - `return texte` **renvoie le texte valide saisi par l'utilisateur, mettant ainsi fin à l'exécution de la fonction**.

Sans boucle infinie

Voici un code qui produit le même effet, mais sans utiliser de boucle infinie:

```
1 def demander_texte():
2     texte = input("Veuillez entrer un texte non vide: ").strip()
3     while texte == "":
4         print("L'entrée ne doit pas être vide. Veuillez réessayer.")
5         texte = input("Veuillez entrer un texte non vide: ").strip()
6     return texte
7
8 # Utilisation de la fonction
```

© 2025 Mr. Lambrechts

```
texte_valide = demander_texte()
print(f"Vous avez entré le texte: {texte_valide}")
```

Comme indiqué plus haut, tu peux remarquer aux lignes 2 et 5 qu'il faut utiliser 2x l'instruction qui demande l'entrée à l'utilisateur.

Vérifier/valider des nombres

La validation de l'input utilisateur permet d'éviter les erreurs et de garantir que les données sont conformes aux attentes.

En Python, il est donc courant d'utiliser une boucle pour demander l'input de l'utilisateur et vérifier qu'il est valide. Si l'input n'est pas valide, on redemande à l'utilisateur de saisir une valeur jusqu'à ce qu'une valeur valide soit fournie. Voici un exemple de code qui illustre comment valider qu'un nombre se trouve entre 1 et 5 :

```
def demander_nombre():
    nombre = input("Veuillez entrer un nombre entre 1 et 5: ")
    while not (nombre.isdigit() and 1 <= int(nombre) <= 5):
        print("Entrée invalide. Le nombre doit être un entier entre 1 et 5.")
        nombre = input("Veuillez entrer un nombre entre 1 et 5: ")
    return int(nombre)

# Utilisation de la fonction
nombre_valide = demander_nombre()
print(f"Vous avez entré le nombre {nombre_valide}.")
```

Explication du Code

La ligne 3 est importante: on vérifie que la chaîne de caractère ne contient que des chiffres et, si c'est le cas, alors on la convertit en entier.

1. **Initialisation de l'input**: On demande à l'utilisateur de saisir un nombre une première fois.
2. **Condition de la boucle** `while` : La boucle continue tant que l'input n'est pas valide. La validité est vérifiée en s'assurant que :
    - L'input est un chiffre (`nombre.isdigit()` vérifie que tous les caractères de la chaîne sont des chiffres).
    - L'input converti en entier est compris entre 1 et 5.
3. **Message d'erreur et nouvelle saisie**: Si l'input est invalide, un message d'erreur est affiché et l'utilisateur est invité à saisir un nouveau nombre.
4. **Retour du nombre valide**: Lorsque l'input est valide, la boucle se termine et la fonction retourne le nombre converti en entier.

Bonnes Pratiques

1. **Messages d'erreur clairs**: Fournissez des messages d'erreur clairs et instructifs pour aider l'utilisateur à comprendre ce qu'il a fait de mal et comment corriger son erreur.

© 2025 Mr. Lambrechts

2. **Validation stricte**: Assurez-vous de valider toutes les entrées utilisateur pour éviter les erreurs et les comportements inattendus.
3. **Boucles de validation**: Utilisez des boucles pour redemander l'input jusqu'à ce qu'il soit correct, ce qui améliore l'expérience utilisateur en évitant les plantages du programme.

Conclusion

Valider les entrées utilisateur est une pratique fondamentale en programmation. Elle permet d'éviter de nombreux problèmes et d'améliorer la robustesse des applications. En utilisant des boucles et des exceptions, vous pouvez gérer efficacement les entrées incorrectes et guider les utilisateurs vers des saisies valides.

© 2025 Mr. Lambrechts
© 2025 Mr. Lambrechts