Chap. 1 Introduction à la programmation Python Thème 1

Interaction textuelle

Sommaire du chapitre 1

La façon la plus simple d'interagir avec un programme est de lire et d'afficher du texte dans la console. On parle d'interaction « en mode console » ou « en ligne de commande ».

I Afficher du texte

Nous avons déjà vu la fonction `print` pour afficher des valeurs sous forme textuelle. Cette fonction admet autant de paramètres que l'on veut, de n'importe quel type (chaîne de caractères, nombre, ...). Elle affiche la *concaténation* de ces paramètres en les « collant » l'un après l'autre, séparés par des espaces :

```
1 a = 3
2 b = 5.4
3 print("La somme de", a, "et", b, "est égale à", a+b)
```

A l'écran s'affichera « La somme de 3 et 5.4 est égale à 8.4 ».

Remarques

`print(a)` est différent de `print("a")`.
`print(a+b)` est différent de `print("a"+"b")`

Python permet d'utiliser des chaînes de caractères particulières, appelées **chaînes de formatage**, qui simplifient l'affichage de chaînes complexes faisant intervenir des valeurs de variables et d'expressions. L'affichage ci-dessus peut être produit par :

```
1 print(f"La somme de {a} et {b} est égale à {a+b}")
```

Une chaîne de formatage commence par la lettre `f` (pour « format ») *avant* le guillement ouvrant (simple ou double). La chaîne peut contenir des expressions Python (ici `a`, `b` et `a+b`)

entre accolades. Chaque expression est alors remplacée par sa valeur.

Remarques

Une expression entre accolades peut être terminée par : suivi d'un nombre pour indiquer le nombre de caractères à utiliser pour afficher l'expression : `f'{a:10}'` affiche la valeur de `a` sur 10 caractères.

II Lire du texte

Pour « lire » du texte en Python, on utilise la fonction `input()` qui met en pause le programme et attend que l'utilisateur saisisse du texte et termine la saisie en appuyant sur la touche `Entrée`.

Remarques

ATTENTION !
La valeur retournée par `input()` est toujours une chaîne de caractères, même si ce que l'utilisateur a tapé ressemble à un nombre ! Pour effectuer des opérations arithmétiques, il faut d'abord **convertir** ces chaînes de caractères en nombre entier (`int()`) ou en nombre à virgule flottante (`float()`). Par exemple, un programme qui ajoute deux nombres entrés par l'utilisateur s'écrit ainsi :

```
1 # Attendre que l'utilisateur entre les valeurs de a et b
2 a = input("Valeur de a ? ")
3 b = input("Valeur de b ? ")
4 # Convertir a et b en flottants, et calculer la somme
5 print("Résultat = ", float(a) + float(b))
```

III Gérer les erreurs dans les entrées : l'instruction try

Que se passe-t-il si l'utilisateur tape comme valeur de `a` une chaîne de caractères qui ne peut pas être interprétée comme un nombre ? Voici le résultat de l'exécution du programme :

```
Valeur de a ? toto
Valeur de b ? tutu
Traceback (most recent call last):
  File "c:/Users/Sophie/OneDrive - ac-reunion.fr/02 - NSI Verger/NSI Premiere/01_Intro
duction a la programmation avec Python/toto.py", line 5, in <module>
    print("Résultat ", float(a) + float(b))
ValueError: could not convert string to float: 'toto'
```

La conversion en nombre flottant, ligne 5, provoque une **erreur d'exécution** de type `ValueError`. Pour éviter que le programme ne s'arrête à cause de cette erreur, il faut l'**intercepter** grâce à l'instruction Python `try`. Cette instruction permet de « prévenir » Python qu'une erreur peut se produire lors de l'exécution du bloc d'instructions qui suit `try`, et de définir ce qu'il faut faire si cette erreur survient (plutôt qu'interrompre le programme par défaut) dans le bloc d'instructions qui suit `except` :

```
1 try :
2     a = float(input("Valeur de a ? "))
3     print("Résultat = ", a)
4 except ValueError :
5     print("Oups, a n'est pas un nombre.")
```

Si une erreur de type `ValueError` se produit pendant l'exécution des instructions comprises dans le bloc `try` (lignes 2-3), le code du bloc `except` (ligne 5) est exécuté au lieu de provoquer l'arrêt du programme :

```
Valeur de a ? toto
Oups, a n'est pas un nombre.
```

Pour rendre le programme plus convivial et redemander à l'utilisateur la saisie si celle-ci est incorrecte, il faut programmer une boucle :

```
1 ok = False
2 while not ok :           # Tant que la saisie n'est pas correcte
3     try :
4         a = float(input("Valeur de a ? "))
5         ok = True        # Saisie correcte
6     except ValueError :  # Saisie incorrecte
7         print("Erreur de saisie ! Entrez une valeur flottante :")
```

```
Valeur de a ? toto
Erreur de saisie ! Entrez une valeur flottante :
Valeur de a ? 123.456
```

Remarques

- D'autres erreurs communes sont `ZeroDivisionError` pour une division par zéro (`x/0`), ou `TypeError` pour une erreur de type (`'2'+2`).
- Si une erreur autre que `ValueError` se produit, par exemple une erreur `ZeroDivisionError`, le bloc `except :` est ignoré et l'erreur fera « planter » le programme.