TP n°2

Entrées/sorties et documentation de programmes

1 Saisie d'une expression au clavier

Pour demander à l'utilisateur de saisir une expression au clavier, on peut utiliser la fonction `input`. Par exemple, l'exécution de :

```
input()
```

donne la main à l'utilisateur pour qu'il saisisse une expresion au clavier. Pour indiquer que la saisie est achevée, on utilise le retour chariot (Return ou Entrée).

☞ Exécuter la ligne de commande

```
input()
```

dans la console, puis saisir

```
1996
```

au clavier.

On observe que la chaîne de caractères `'1996'` est affichée après la saisie.

☞ Déterminer le type de l'expression `'1996'`.

Dans l'exemple précédent, bien que nous ayons saisi le nombre 1996, la fonction `input` n'a pas retourné une expression de type `int`, mais une chaîne de caractères qui correspond au type .....

D'une manière générale, la fonction `input` renvoie la **chaîne de caractères** saisie au clavier (sans le retour chariot terminal).

La fonction `input` donne, en outre, la possibilité d'afficher un message, avant de demander la saisie d'une expression au clavier. Pour cela, il suffit d'insérer une chaîne de caractères entre les parenthèses lors de l'appel de la fonction `input`.

☞ Exécuter la ligne de commande suivante dans la console

```
input('Saisir un nombre entier')
```

dans la console, puis saisir

```
1996
```

au clavier. Exécuter à nouveau cette ligne de commande, mais saisir cette fois-ci

au clavier.

Dans le deuxième cas, aucun message d'erreur n'apparaît à l'écran, bien que nous ayons saisi une expression non entière (puisque de type `float`).

Le message (éventuellement) affiché avant la demande de saisie a uniquement pour but d'informer l'utilisateur. **Il n'a aucune fonction de contrôle**.

2 Affichage d'un message, d'une expression, d'une valeur de variable dans la console

Pour demander à la machine d'afficher un message dans la console, on peut utiliser la fonction `print`. Par exemple, l'exécution de

```
print('PTSI is fun ;-)')
```

affiche le message

```
PTSI is fun ;-)
```

dans la console. On peut aussi afficher des valeurs d'expressions grâce à `print`. Par exemple, l'éxécution de

```
print(2**3-4*3+50)
```

affiche la valeur de l'expression `2**3-4*3+50` dans la console (ici l'expression `46` de type `int`). On peut encore afficher des valeurs de variables grâce à `print`. Par exemple, l'éxécution de

```
1. x=26
2. print(x)
```

affiche la valeur de la variable `x` dans la console (ici l'expression `26` de type `int`). Enfin, on peut mixer les différents types d'affichages (message, valeurs d'expressions et valeurs de variables) dans un même appel de la fonction `print`. Par exemple, l'éxécution de

```
1. x=26
2. triple=3*x
3. print('Le double de',x,'est',2*26,'et le triple de',x,'est',triple)
```

affiche

```
Le double de 26 est 52 et le triple de 26 est 78
```

dans la console.

3 Documentation de programmes

Il est fondamental de documenter les programmes que l'on écrit, en ajoutant au code brut des commentaires qui :

- permettent de comprendre le but du programme ;
- expliquent le rôle des variables, des fonctions ;
- éclairent sur sa construction, sa structure ;
- pointent des insuffisances ou des améliorations possibles.

Cet effort est utile pour rendre intelligible du code. Personne ne se plonge dans des dizaines de lignes de programme non commentées !

Pour indiquer un commentaire dans un programme, on peut placer le caractère

#

avant le commentaire proprement dit. Lors de l'exécution du programme, tout ce qui se trouve après un caractère #, et sur la même ligne que ce caractère, est ignoré. Voici un exemple de programme documenté.

```
#    But : calcul de la somme des entiers de 1 à n

#    Amélioration possible : se passer de la boucle et utiliser le fait que la somme vaut n*(n+1)/2

1.  n=10   # initialisation de la variable n (donnée) à 10
2.  k=1    # initialisation de l'indice de parcours de boucle/de somme k à 1
3.  s=0    # initialisation de la variable s qui sert à calculer la somme de proche en proche à 0
4.
5.  while(k<=n) :       # on exécute le bloc d'instructions indenté suivant, tant que k<=n
6.      s=s+k       # on ajoute la valeur de k (valeur qui évolue) à la somme s déjà calculée
7.      k=k+1       # on incrémente k de 1 pour accéder à l'entier suivant
8.
9.  print('La valeur de la somme des entiers de 1 à',n,'est',s)       # affichage du résultat
```

La pertinence des commentaires est un élément important d'appréciation d'un programme.

4 Exercices

Consigne : Pour chaque exercice, écrire un fichier `.py` et l'envoyer à l'adresse :

`ipt.ptsi@gmail.com`

en mettant comme objet du courriel

TP 2 - Exercice **x** - **Nom 1** - **Nom 2**

et en usant de votre sagacité pour remplacer les termes en gras par ce qu'il convient.

Exercice 1 : Documenter le programme suivant.

```
1.  x=input('Saisir un nombre réel:')
2.  x=float(x)
3.
4.  if (x>0):
5.      print('Le nombre',x,'est positif.')
6.  elif (x<0):
7.      print('Le nombre',x,'est négatif.')
8.  else:
9.      print('Le nombre',x,'est nul.')
```

Exercice 2 : Écrire un programme qui demande à l'utilisateur de saisir un nombre entier au clavier et qui affiche le cube de l'entier saisi à la console.

Exercice 3 : Écrire un programme qui demande à l'utilisateur de saisir la partie réelle a, puis la partie imaginaire b d'un nombre complexe (supposé non nul) et qui affiche les deux solutions de l'équation $z^2 = a + ib$ d'inconnue $z \in \mathbb{C}$.
Indication : on pourra utiliser la fonction racine carrée, d'argument un réel positif ou nul, nommée `math.sqrt` *après avoir chargé la bibliothèque* `math` *grâce à la commande* `import math` *(que l'on place usuellement en tête de fichier).*