TP 3 : Types de données en **Scilab**
Utilisation des commandes `disp` et `input` dans les programmes

Pré-requis : avant d'entamer ce TP, il faut avoir lu / compris / effectué les manipulations présentes dans les cours « Chapitre 3 : Premiers programmes en **Scilab** »
et « Chapitre 4 : Types de données en **Scilab** ».

- Dans le dossier `Info_1a`, créez le dossier `TP_3`.

I. Types de données

I.1. Le type `constant`

- Évaluer `typeof(3.4)` puis `typeof(%pi)` et `typeof(5+2*3/4)`. Noter le résultat obtenu.

- Que contient le type `constant` de **Scilab** ?

- Évaluer `a = 4.3` puis `typeof(a)`.

- Apporter une précision sur ce que regroupe le type `constant` de **Scilab**.

I.2. Le type `boolean`

- Évaluer `%t | %f`, puis `%t & %f` et enfin `~((%t | %f) & (%t & %t))`.
  Rappeler l'utilité des commandes `|`, `&` et ~.

- Évaluer `typeof(%t)`, `typeof(%t & %f)`, `typeof(%t | %f)`. Noter le résultat obtenu.

- Que contient le type `boolean` de **Scilab** ?

- Selon vous, quel est le type associé aux expressions `3 < 2` puis `3 == 2` ? Le vérifier.

- Évaluer `a = %t & %f` puis `typeof(a)`.

- Apporter une précision sur ce que regroupe le type `boolean` de **Scilab**.

I.3. Le type `string`

- Évaluer `"log"`, puis `'log'`, puis `"3"` et `'3'` et enfin `typeof("Comment allez-vous ?")`. Qu'obtient-on ?

- Évaluer `typeof("log")`, puis `typeof('log')`, puis `typeof("3")` et `typeof('3')` et enfin `typeof("Comment allez-vous ?")`. Qu'obtient-on ?

- Conclure sur ce que regroupe le type `string` de **Scilab**.

- Évaluer `a = "Bonjour"` puis `typeof(a)`.

- Apporter une précision sur ce que regroupe le type `string` de **Scilab**.

- Évaluer `"je vais" + " bien " + "et vous ?"`. Noter le résultat obtenu.
  (on prendra soin de respecter les espaces)

- Décrire précisément l'utilité de l'opérateur `+` dans ce cas.

- Évaluer `typeof(3)` puis `u=string(3)` et `typeof(u)`. Noter le résultat obtenu.

- Décrire précisément l'utilité de l'opérateur `string`.

I.4. Les erreurs de type

- Évaluer `age = input("Quel est votre âge ? ")` en indiquant une réponse au clavier.
- Évaluer `"j''ai actuellement " + age + " ans"`. Noter le message d'erreur obtenu.

- Quel devrait être le type de `age` pour que le message d'erreur précédent ne soit pas levé ?

- Corriger l'appel précédent.

- Évaluer `log("36")`. Noter le message d'erreur obtenu.

- En conclure sur l'une des utilités des types de données en informatique.

I.5. Écriture de programmes avec dialogue utilisateur

- On souhaite écrire un programme qui permet d'afficher la formule de la somme des n premiers entiers où la valeur de n est entrée par l'utilisateur. Écrire un programme qui :
  - × stocke dans une variable `n` un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `calc` la somme des `n` premiers entiers,
  - × stocke dans une variable `chaine` la phrase :
    "`La somme des` n `premiers entiers vaut` n $\star$ `(`n`+1) / 2, à savoir` *calc*"
    où n et *calc* doivent être remplacées par leur valeur,
  - × affiche la prhrase précédente.

  On devra faire appel à la commande `string` lors de la création de la variable `calc`.

  On appellera `sommeEntiers.sce` le fichier contenant ce programme.
- Recopier ci-dessous le programme précédent.

- Dans un nouvel onglet **SciNotes**, écrire un programme `sommeEntiersGen.sce` qui généralise le programme précédent pour une somme $\sum_{k=m}^{n} k$.

  La valeur de m doit, elle aussi, être entrée au clavier par l'utilisateur.

- Dans un nouvel onglet **SciNotes**, écrire un programme `sommeCarrésEntiers.sce` qui adapte le programme `sommeEntiers.sce` au cas de la somme $\sum_{k=1}^{n} k^2$.

- Dans un nouvel onglet **SciNotes**, écrire un programme `sommeCarrésEntiers.sce` qui adapte le programme `sommeEntiers.sce` au cas de la somme $\sum_{k=1}^{n} k^3$.

- Enfin, dans un nouvel onglet **SciNotes**, écrire un programme similaire dans le cas d'une somme géométrique $\sum_{k=m}^{n} q^k$.

  Les valeurs de m et n doivent être entrées au clavier par l'utilisateur.

II. Les fonctions `disp` et `input` : premières manipulations

II.1. La fonction `disp`

▶ Évaluer dans la console `"Bonjour à tous"`. Écrire très précisément le résultat obtenu.

▶ Évaluer dans la console `disp("Bonjour à tous")`. Écrire très précisément le résultat obtenu.

▶ Quelle différence y a-t-il entre le résultat de ces deux appels ? À quoi sert la fonction `disp` ?

II.2. La fonction `input`

▶ Évaluer dans la console `help input`.
Recopier ci-dessous la première phrase (en anglais), de la rubrique **Description**.

▶ Évaluer dans la console `input("Prière d''entrer un nombre au clavier : ")`.
Qu'obtient-on ?

▶ Quelle est l'utilité de la fonction `input` ? On pourra traduire la description précédente.

▶ Par quel appel stocke-t-on, dans une variable `n`, un nombre entré au clavier par l'utilisateur ?

- Ouvrir **SciNotes** en cliquant sur l'icône  en haut à gauche.
  Arrimer cette fenêtre à droite de votre environnement **Scilab**.
- À l'intérieur de cette fenêtre, écrire un programme qui :
  - × stocke dans une variable `n1`, un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `n2`, un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `res` la somme de `n1` et `n2`,
  - × affiche le contenu de la variable `res`.

  Sauvegarder et exécuter à l'aide de la touche `F5` (`fn-F5` sous Mac OS).
  On appellera ce fichier `input_disp.sce`. Le recopier ci-dessous.

- Afin de rendre le résultat d'affichage plus agréable, on souhaite afficher la phrase : `La somme des deux nombres entrés vaut ...` où le symbole `...` doit contenir le résultat calculé.
  Évaluer dans la console `disp("La somme des deux nombres entrés vaut res")`.
  Est-ce le résultat attendu ? Expliquer.

- Évaluer `help plus`. Repérer dans la `Description` la phrase parlant des chaînes de caractères.
  La recopier ci-dessous.

- Que signifie le terme concaténation ? *(chercher éventuellement dans Wikipédia)*

- Évaluer dans la console `"La somme des deux nombres entrés vaut " + "res"`.
  Qu'obtient-on ?

- Évaluer dans la console `"La somme des deux nombres entrés vaut " + res`.
  Quel message d'erreur obtient-on ?

- On utilise alors la fonction `string` qui permet de convertir un réel **Scilab** en chaîne de caractères. Évaluer `"La somme des deux nombres entrés vaut " + string(res)`.
  Expliquer le résultat obtenu.

- Modifier le programme précédent afin qu'il réalise l'affichage de cette phrase. Exécuter.

III. Étude de programmes

Dans la suite, on considère les programmes `discrim.sci`, `racP.sci` (du TP précédent) et le programme `racPDisp.sci` défini plus loin.
On rappelle tout d'abord le contenu du programme `racP.sci`.

```
1  // racP(a, b, c) permet de calculer les
2  // racines du polynôme aX ^2+bX+c
3  function [xPlus, xMoins] = racP(a, b, c)
4      delta = discrim(a, b, c)
5      xPlus = (-b + sqrt(delta))/(2*a)
6      xMoins = (-b - sqrt(delta))/(2*a)
7  endfunction
```

- Par quel appel peut-on stocker le résultat de `racP(1,2,-3)` dans deux variables `u` et `v` ?

On considère maintenant le programme `racPDisp.sci`.

```
1  // racPDisp permet d'afficher les valeurs des
2  // deux racines d'un polynôme pris en paramètre
3  function racPDisp(a, b, c)
4      delta = discrim(a, b, c)
5      xPlus = (-b + sqrt(delta))/(2*a)
6      xMoins = (-b - sqrt(delta))/(2*a)
7      disp("La plus grande racine vaut " + string(xPlus))
8      disp("La plus petite racine vaut " + string(xMoins))
9  endfunction
```

- Combien d'arguments de sortie possède la fonction `racPDisp` ?

- Évaluer `racPDisp(1,2,-3)`. Quel avantage voyez-vous par rapport à la fonction `racP` ? Et quel inconvénient ?

On considère enfin le programme contenu dans le fichier `racPDispInp.sce`.

```
1  // Le script suivant demande la valeur d'un polynôme
2  // et affiche ses deux racines
3  a = input("Entrer la valeur du coefficient a : ")
4  b = input("Entrer la valeur du coefficient b : ")
5  c = input("Entrer la valeur du coefficient c : ")
6  delta = discrim(a, b, c)
7  xPlus = (-b + sqrt(delta))/(2*a)
8  xMoins = (-b - sqrt(delta))/(2*a)
9  disp("La plus grande racine vaut " + string(xPlus))
10 disp("La plus petite racine vaut " + string(xMoins))
```

- Exécuter ce programme à l'aide de la touche `F5` (`fn-F5` sous Mac OS). Qu'obtient-on ?

- Selon vous, quel programme doit-on privilégier si un autre programme nécessite les valeurs des racines calculées ? Expliquer alors le principal inconvénient des programmes `racPDisp` et `racPDispInput`.

IV. Application

- On souhaite écrire un programme qui permet d'afficher la formule explicite d'une suite arithmético-géométrique (u_n) définie par $u_{n+1} = au_n + b$ où les valeurs de a et b sont entrées par l'utilisateur. Écrire un programme qui :
  - × stocke dans une variable `a` un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `b` un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `u0` un nombre entré au clavier par l'utilisateur,
  - × stocke dans une variable `lambda` la solution de l'équation de point fixe associée à la suite,
  - × stocke dans une variable `v0` le calcul `u0 - lambda`,
  - × affiche la phrase :
    `"La formule explicite de la suite considérée est u_n = a^n v0 + lambda"`
    où les variables `a`, `v0` et `lambda` doivent être remplacées par leurs valeurs (utilisation multiple de `string`).

  On appellera `arithmetico_geom.sce` le fichier contenant ce programme.

On pourra utiliser le schéma suivant (à compléter !).

```
1  // dialogue avec l'utilisateur
2  a =
3  b =
4  u0 =
5
6  // calcul de la solution de : x = ax+b
7  lambda =
8  v0 =
9
10 // écriture de la chaîne de caractères à afficher
11 chaine =
12
13 // affichage de la chaîne de caractère précédente
14
15
```