Introduction à Python 3.

I La démarche du programmeur.

Programmation.

La programmation consiste à concevoir puis faire exécuter des instructions par un ordinateur dans le but d'effectuer une tâche décidée à l'avance. Cela peut s'apparenter à une recette de cuisine : il faut indiquer tous les ingrédients et toutes les étapes à réaliser pour obtenir le gâteau que nous désirons. Dans un programme nous dirons à l'ordinateur toutes les instructions à effectuer pour qu'il donne le résultat choisi.

Nous distinguerons trois phases dans un programme : entrée, traitement, puis sortie. En entrée l'utilisateur du programme entre de l'information, en traitement c'est l'ordinateur qui passe à l'action et fait les calculs nécessaires pour obtenir le résultat souhaité, enfin, en sortie, l'ordinateur présente le résultat.

Considérons un exemple. Sur un jeu de smartphone en entrée on indique comment lancer l'oiseau avec un lance pierre en glissant le doigt sur l'écran. Lors du traitement le téléphone calcul la trajectoire de l'oiseau. En sortie le téléphone affiche sur l'écran le déplacement de l'oiseau.

Les langages de programmation compris par les ordinateurs.

Les mémoires d'ordinateurs ressemblent à des casiers électromagnétiques et dans chaque petite case du disque dur, soit la machine range une charge électrique soit elle ne met rien.

La méthode mathématique pour modéliser est donc toute trouvée : l'absence de charge électrique est représentée par un 0 et la présence d'une charge électrique est représentée par 1.

Autrement dit le calculateur (microprocesseur) de l'ordinateur ne manipule que des listes de 0 et 1. Ce langage écrit avec les deux seuls chiffres 0 et 1 est appelé le binaire.

Ce langage est appelé un langage bas niveau. Il est directement compréhensible par la machine mais il est hermétique pour un humain.

Des langages de programmation plus simples à manipuler ont donc été développés. Ces langage, qui sont dits de haut niveau, sont maintenant nombreux : C++, Python, Java, etc.

Python désigne à la fois un langage de programmation et un logiciel qui va traduire nos consignes dans un langage compréhensible par la machine.

Le langage Python 3.

Python est un langage textuel. Autrement dit les instructions doivent être écrits sous forme de mots dans un fichier texte que nous appellerons un *script*. Les instructions qui toutes réunies forment un programme sont appelées le *code source du programme*.

II Python comme interpréteur : le shell.

Dans cette partie nous allons communiquer directement avec le logiciel Python. Nous lui donnerons une unique instruction (entrée), il la traduit directement en langage machine (traitement) et il nous affiche le résultat correspondant (sortie).

La fenêtre qui permet de communiquer directement avec Python est appelée un *shell*.

Le shell avec la Ti 83 premium CE Python.

Les différentes opérations.

Variables et affectations.

La variable en programmation n'est pas tout à fait la même chose que la variable mathématique (le x des fonctions). La variable informatique est un emplacement dans la mémoire de l'ordinateur. Sous le nom d'une variable, `a`, différents objets pris en charge par Python 3.

Une variable contient une information. Le fait d'enregistrer quelque chose dans une variable s'appelle une *affectation*. L'instruction d'affectation de Python est noté avec le signe égale (=).

Une variable peut contenir un nombre entier. Essayez :

```
>>> a=3
>>> a
```

`a` est alors un nombre entier comme un autre. Ainsi essayez :

```
>>> a*7
```

Mais il est aussi possible d'enregistrer sous le nom d'une variable un mot :

```
>>> a="abracadabra"
>>> a
```

Les nombres décimaux peuvent de la même façon être enregistré sous le nom d'une variable.

```
>>> a=3.14159
```

Dans les exemples ci-dessus et pour simplifier les choses la variable avait pour nom `a`. mais il est possible de choisir des noms formés de mots avec des chiffres. Quelques contraintes néanmoins :

— le nom de la variable ne doit pas commencer par un chiffre,
— pas d'espace dans le nom de la variable,
— Python étant sensible à la casse : `truc`, `Truc` et `TRUC` sont des variables différentes,
— évitez certains caractères spéciaux.

La convention est d'écrire les noms des variables en minuscules, tout attaché et de choisir le nom le plus court possible mais qui décrive bien le rôle de la variable.

Le type des variables.

Nous voyons que Python 3 accepte de manipuler, sous forme de variable, des objets divers : des nombres et des mots.

Nous dirons que les variables sont de différents *types*.

— Les nombres entiers sont appelés des *entiers* (en anglais *integer*),
— Les nombres décimaux sont appelés des *flottants* (en anglais *float*) car avec l'écriture scientifique la virgule peut être déplacée.
— Les *chaînes de caractères* (en anglais *string*) sont des mots ou des phrases. Elles sont notées en guillemets.

Les différents types permettent à Python de traiter différemment les mots et les nombres.

Que donne l'addition de deux mots ?

```
>>> a="Sal"
>>> b="ut"
>>> a+b
```

Pour afficher le contenu d'une variable nous avons entré le nom de la variable suivit d'entée. Cependant il existe une instruction pour demander l'affichage (notamment dans un programme) il s'agit de `print()`. Essayez :

```
>>> print(a)
```

Exercice 1.

Testez les instructions suivantes.

```
>>> r,pi=12,3.14159
>>> s=pi*r**2
>>> print(s)
>>> print(type(r),type(pi),type(s))
```

Quelle est l'utilité de la fonction type ?

III Premiers codes sources : fonctions.

Avec la Ti 83 premium CE Python.

Dans le script écrivez le programme de calcul suivant :

```
a=3.2
b=4
print("Reponse:",a+b)
```

Demander à la calculatrice d'exécuter le script (de le lire et de l'interpréter).

La fonction dans Python.

La convention est de présenter les programmes en Python si possible sous forme de *fonctions*. Une fonction est un bloc d'instructions qui porte un nom particulier et qui peut (ou non) dépendre du choix de certaines variables.

Le script précédent s'écrira sous forme de fonction :

```
def addition():
    a=3.2
    b=4
    print("Reponse:",a+b)
```

La création de la fonction commence forcément par
— l'instruction `def`
— suivie du nom de la fonction, ici `addition`,

— de parenthèses,
— des deux points,
— puis, en passant à la ligne d'une indentation (alinéa).

En langage Python l'indentation indique que les instructions font partie du même bloc. Les trois instructions font partie de la même fonction.

Lorsque nous demandons à la calculatrice d'exécuter le script nous obtenons l'affichage désiré.

Les fonctions sont précisément appelées fonctions car elles peuvent dépendre de paramètres, appelés *arguments* (« en fonctions des arguments ») qui sont alors placés entre les parenthèses et séparés les uns des autres par des virgules.

Exercice 2. ♥

1. Programmez la fonction suivante :

```
def theo(a,b,c):
    print("D'une part: ",a**2+b**2," d'autre part: ",c**2)
```

2. Testez-la (remarquez les chevrons donc il faut utiliser le Shell) en faisant :

(a) `>>> theo(2,3,4)`

(b) `>>> theo(3,4,5)`

(c) Expliquez dans quelle situation (en géométrie) cette fonction peut être utile.

Remarquons que les fonctions que nous définissons sont semblables aux instructions déjà définies dans Python : `print()`, `type()`, etc.

Pour se conformer à la présentation classique des fonctions en Python le résultat doit être l'instruction `return` comme dans les exercices suivants.

Exercice 3. Application.

Voici une fonction écrite en Python.

```
def trin(a):
    return(a**2-a+1)
```

1. Combien d'arguments cette fonction possède-t-elle ? Lesquels ?
2. Que renvoie la fonction `trin` lorsqu'on tape les instructions suivantes ?

   (a)
   ```
   >>> trin(2)
   ```

   (b)
   ```
   >>> trin(5)
   ```

   (c)
   ```
   >>> trin(2.1)
   ```

Exercice 4. Application.

On considère la fonction vitesse ci-dessous.

```
def vitesse(distance,temps):
    return(distance/temps)
```

1. Combien d'arguments cette fonction possède-t-elle ?
2. On suppose que la distance est exprimée en kilomètre et le temps en heure. On a écrit dans la console l'instruction suivante.

   ```
   >>> vitesse(120,1.5)
   ```

   Donnez le résultat et interprétez-le.
3. Écrivez un script dont les arguments sont une distance d parcourue à une vitesse v et qui renvoie le temps correspondant.

Dans l'exercice suivant nous voyons un commentaire qui explique ce le rôle de la fonction comme une chaîne de caractère (donc ente guillemets). Ce commentaire ne participe pas vraiment du programme. Nous verrons ultérieurement que ce commentaire est quand même lu et utilisé par le logiciel Python.

Exercice 5. Application.

Complétez la fonction `sec` suivante afin qu'elle convertisse en secondes une durée exprimée en heure, minute et seconde (l'instruction `return` renvoyant la valeur calculée).

```
def sec(h,m,s):
    "conversion en secondes"
    return ...
```

Exercice 6. Application.

La fonction `cylindre` ci-dessous calcule et renvoie le volume d'un cylindre lorsqu'on entre son rayon `R` et sa hauteur `h`.

```
from math import *
def cylindre(R,h):
    return(pi*R**2*h)
```

1. À quoi sert l'instruction de la première ligne ? Pourquoi est-elle nécessaire ?
2. Programmez et modifiez le script précédent pour qu'il renvoie le volume d'un cylindre arrondi à 10^{-3}. Pour cela vous utiliserez la fonction avec deux arguments déjà programmée dans Python : `round`(*nombre,nombre de chiffres*).

Exercice 7. Application.

```
from random import *
def fonction():
    return(randint(1,6))
```

1. Combien d'arguments cette fonction possède-t-elle ?
2. Utilisez une dizaine de fois cette fonction dans le shell puis proposez une situation dans laquelle elle pourrait servir.
3. Modifiez le précédent script de façon qu'il simule le lancer d'une pièce de monnaie parfaitement équilibrée.

Les différentes versions d'un programme : une lente maturation.

IV Contrôle du flux d'exécution.

Séquence d'instructions.

Les instructions d'un programme s'exécutent dans l'ordre ou elle ont été écrites (de haut en bas). Une succession d'instructions qui doivent être effectuées les unes à la suite des autres est appelée une *séquence*.

Exercice 8.

Exécutez et comparez les résultats des deux fonctions avec $a = 7$ et $b = 3$:

```
def echangA(a,b):
    a=b
    b=a
    print(a,b)
```

```
def echangB(a,b):
    b=a
    a=b
    print(a,b)
```

Cependant dans un programme toutes les instructions ne doivent pas forcément être exécutées. Pensons à un jeu vidéo : le monstre ne doit surgir de sa cachette que si l'avatar à déclenché le piège. Nous allons voir des blocs d'instructions qui permettent de choisir les lignes de codes à exécuter.

Instruction conditionnelle.

Version Scratch :
« Si l'avatar déclenche le piège alors ... » est un exemple d'instruction conditionnelle. Nos conditions seront en général plus mathématique. Pour programmer cela nous utiliserons le bloc conditionnel suivant :

```
if condition:
    instruction1
    instruction2
```

Remarquons certains points communs avec l'écriture d'une fonction.
- Le bloc d'instruction conditionnel commence toujours par `if`.
- La `condition` sera le plus souvent une phrase mathématique (une égalité par exemple) et l'ordinateur devra vérifier si cette phrase est vraie ou fausse.
- La ligne du `if` se termine par deux points.
- Pour indiquer que les instructions sont dans le bloc conditionnel elles sont toutes indentées de la même façon (décalage par rapport au bord gauche). Nous pouvons considérer que le « alors » qui suit normalement un « si » en français est ici remplacé par une indentation.

Exercice 9.

Voici un exemple de fonction utilisant une exécution conditionnelle :

```
def signe(a):
    if a>0:
        text="a est strictement positif"
    return(text)
```

1. Testez la fonction `signe` avec différentes valeurs de a.
2. Modifier la fonction `signe` comme suit :

```
def signe(a):
    if a>0:
        text="a est strictement positif"
    else:
        text="a est negatif"
    return(text)
```

   Comment traduiriez-vous l'instruction `else` ?
3. Complétez cette nouvelle modification de la fonction `signe`.

```
def signe(a):
    if a>0:
        text="a est strictement positif"
    elif a<0:
        text="a est negatif"
    else:
        text="a est ..."
    return(text)
```

   Expliquez le rôle de `elif`.

Opérateurs de comparaison.

Pour exprimer des conditions nous aurons souvent besoin de comparer des variables. Voici des outils comparaisons classiques :

```
x==y    #x est egale y
x!=y    #x est different de y
x>y     #x est strictement plus grand que y
x<y     #x est strictement plus petit que y
x>=y    #x est superieur (ou egale) a y
x<=y    #x est inferieur (ou egale) a y
```

Remarquons que pour vérifier si deux variables sont égales nous ne pouvons pas utiliser simplement le signe égale car il sert déjà pour l'instruction d'affectation.

Exercice 10.

Programmez et testez la fonction suivante avec des valeurs entières de a. (% est l'opérateur qui donne le reste dans la division euclidienne.)

```
def parite(a):
    if a%2==0:
        text="a est un nombre ..."
    else:
        text="a est un nombre ..."
    return(text)
```

Complétez ce programme une fois que vous aurez compris son utilité.

Exercice 11.

Dans un parc d'attraction qui possède des manèges un peu sportifs, le prix de l'entrée dépend de la taille et de l'âge de la personne. Pour des raisons de sécurité, une personne mesurant moins de 1,30 mètre ne peut accéder à tous les manèges. Elle paie alors moins cher l'entrée du parc.
La fonction ci-dessous détermine le prix que doit payer une personne selon sa taille et son âge.

```
def prix(taille,age):
    if taille<1.30:
        entree=age*0.5
    else:
        entree=age*1.1
    return(entree)
```

1. Combien d'arguments la fonction possède-t-elle ? Lesquels ?
2. En utilisant cette fonction, déterminez le prix que va payer quelqu'un qui a 18 ans et qui mesure 1,82 mètre.
3. Combien va payer un enfant de 8 ans et qui mesure 1,20 mètre ?
4. Modifier la fonction pour que l'entrée soit gratuite pour les personnes moins de 90 cm.

Exercice 12. Application.

Un site de développement de photographies affiche les tarifs suivants :

- de 1 à 25 tirages : 0,15 € par photo et 3 € de frais de port,
- de 26 à 70 tirages : 0,10 € par photo et 4 € de frais de port,
- au-delà de 70 tirages : 0,05 € par photo et 7 € de frais de port.

Complétez le script suivant de façon à ce que la fonction renvoie le total à payer en fonction du nombre de tirages demandés.

```
def prix(photos):
    "Total a payer pour l'internaute"
    if photos<=25:
        total= 0.15*...
    elif 26<=photos and photos<=70:
        total=0.10*...
    else:
        ...
    return(total)
```

V Instructions répétitives.

L'informatique a pris en charge les calculs numériques mais aussi bon nombre de tâches répétitives et donc fastidieuses. Il est donc naturel que dans les instructions élémentaires d'un langage de programmation se retrouve des instructions permettant de répéter des commandes.

Pour Python nous en verrons deux : la boucle « for » qui permet de répéter un nombre précis de fois et la boucle « while » qui répète autant que nécessaire.

La boucle bornée « for ».

Voici la forme générale du boucle for :

```
for variable in range():
    instruction1
    instruction2
```

Voici trois exemples un peu différents de fonction additionnant des suites d'entiers. Essayez-les puis construisez les tableaux d'état des variables de ces programmes.

```
def addentier1():
    S=0
    for i in range(3):
        S=S+i
    return(S)
```

```
def addentier2():
    S=0
    for i in range
   (3,5):
        S=S+i
    return(S)
```

```
def addentier3():
    S=0
    for i in range(1,6,2):
        S=S+i
    return(S)
```

Exercice 13.

1. Programmez, testez puis expliquez le rôle de la fonction ci-dessous :

```
def fonc():
    t=0
    for i in range(0,10):
        t=t+7
        print(t)
```

2. Modifiez la précédente fonction pour qu'elle remplisse la même rôle mais pour d'autres valeurs.

Exercice 14.

Le coût de production d'un produit, en milliers d'euros, est donné par la fonction mathématique

$$C(x) = \frac{1}{6}x^3 - \frac{5}{2}x^2 + 13x,$$

où x est la quantité fabriquée en tonnes. l'entreprise peut fabriquer entre 0 et 100 tonnes de ce produit. Chaque tonne est vendue 6 000 €.

1. Complétez l'algorithme ci-contre afin qu'une fois la quantité x saisie, l'algorithme calcule et affiche le coût de production correspondant, en milliers d'euros.

```
def cout(x):
    return ...
```

2. Exprimer la recette $R(x)$ en milliers d'euros pour x tonnes vendues.
3. Montrez que pour x un nombre compris entre 0 et 100, le bénéfice réalisé pour la vente de x tonnes est, en milliers d'euros :

$$B(x) = -\frac{1}{6}x^3 + \frac{5}{2}x^2 - 7x.$$

4. Quel est le rôle de la fonction suivante ?

```
def benef():
    x=0
    for i in range(1,100):
        x=x+1
        b=-1/6*x**3+5/2*x**2-7x
    return(b)
```

La boucle non-bornée « while ».