Appendice 2 : modules Python additionnels

Aide-mémoire sur les bibliothèques (modules) utilisées

time

Le module `time` donne accès à quelques fonctions liées au temps.

TI-83 Premium CE Edition Python	TI Nspire CX II-T
`sleep(t)` interrompt l'exécution pendant `t` secondes avant de poursuivre.	
`monotonic()` renvoie la valeur d'un « compteur de temps » (en secondes) permettant de mesurer le temps écoulé.	`clock()` a le même effet.

ti_system

Le module « système » `ti_system` donne accès à quelques fonctions spécifiques de la machine.

TI-83 Premium CE Edition Python	TI Nspire CX II-T
`disp_wait()` permet d'interrompre l'exécution et d'attendre que l'utilisateur appuie sur la touche [annul].	Même résultat (attendre [esc]) avec `while get_key() != "esc":`
`sleep(t)` interrompt l'exécution pendant `t` secondes avant de poursuivre.	
La fonction `recall_list` permet de récupérer le contenu d'une liste du système natif de la calculatrice. L'interface de la fonction `recall_list` est particulière : seules les listes système L_1 à L_6 sont accessibles, et désignées par les chaînes de caractères "1" à "6". Exemple : `uneliste=recall_list("2")`. À noter : tandis que les listes en Python sont indexées à partir de 0, dans l'environnement natif c'est à partir de 1.	On utilise le nom d'une variable (de type liste) déjà définie dans le classeur en cours. Exemple : `uneliste=recall_list("L2")`. À noter : tandis que les listes en Python sont indexées à partir de 0, dans l'environnement natif c'est à partir de 1.
La fonction `store_list` fait l'inverse, copiant une liste dans l'environnement natif de la calculatrice, avec quelques limitations (en taille notamment : au maximum 99 éléments). Exemple : `store_list("2",maliste)`.	On donne un nom de liste arbitraire, qui sera créée dans le système d'exploitation. Exemple : `store_list("L5",[1,2,42])`
La fonction `recall_RegEQ` permet de récupérer les expressions des fonctions calculées comme « régressions » (touche [stats], rubrique CALC, menu E:TracéAjust-Éq). La valeur `s` retournée par `s=recall_RegEQ()` est en fait une chaîne de caractères contenant une expression mathématique dépendant d'une variable nommée `x`. En ayant assigné une valeur à la variable `x`, `eval(s)` renvoie la valeur de l'expression mathématique pour la valeur voulue de la variable `x`.	Pas d'équivalent mais on peut « importer » une fonction Nspire dans l'environnement Python avec `eval_function("mafonction",valeur)` où `mafonction` est une fonction Nspire (d'une seule variable) déjà définie ; par exemple une fonction d'interpolation ...

Appendice 2 : modules Python additionnels

ti_plotlib

Le module `ti_plotlib` donne accès à de nombreuses fonctions de tracé graphique dans une « fenêtre » correspondant à l'écran (de format 3:2) mais avec des coordonnées définies par l'utilisateur.

Instructions	Exemples
`plt.cls()` permet d'effacer l'écran.	`plt.cls()`
`plt.window(xmin,xmax,ymin,ymax)` permet le cadrage de l'affichage.	`plt.window(-2,2,0,5)` ajuste la fenêtre, [–2;2] en abscisses etc.
`plt.autowindow(x-list,y-list)` ajuste le cadrage de l'affichage en fonction d'une liste d'abscisses et ordonnées (afin que tous les points soient visibles)	`plt.autowindow([0,4.5],[1,9.8]))` fixe une fenêtre d'affichage avec des « coins » de coordonnées (0;1) / (4,5;9,8).
`plt.axes("off/on")` permet d'afficher ou non les axes du graphique.	`plt.axes("off")` permet de ne pas afficher les axes du graphique.
TI-83 : `plt.grid(xscal,yscal,type)` affiche la grille en réglant l'écartement sur les axes et le « type » de la grille, valant `"dot"`, `"dash"`, `"solid"` ou `"point"`. **Nspire CX** : Type vaut `"dotted"`, `"dashed"`, ou `"solid"`.	`plt.grid(0.2,0.5,"solid")` pour une grille avec des lignes continues `plt.grid(1,5,"dash")` pour des pointillés.
`plt.color(R,G,B)` fixe la couleur du tracé à partir du code RGB (3 nombres entre 0 et 255 pour les composantes de couleur rouge, vert, bleu de la couleur globale).	`plt.color(255,0,0)` sélectionne une couleur rouge « vif ». `plt.color(0,0,0)` sélectionne le noir.
`plt.show_plt()` finit d'afficher le graphique et se met en attente (appuyer sur annul pour continuer).	`plt.show_plt()`
`plt.text_at(ligne,"texte","left/right/center")` permet d'afficher du texte en définissant la ligne d'affichage, le texte à afficher et le centrage sur la ligne sélectionnée. Il y a 12 lignes d'affichage possibles.	`plt.text_at(12,"hello","left")` affiche sur la ligne 12 à gauche le texte : hello. **Note :** l'arrière-plan des caractères est effacé.
`plt.plot(a,b,"marque")` place sur le graphique un point aux coordonnées (a,b) du style de la « marque ».	`plt.plot(2,3,"+")` affiche **+** au point de coordonnées (2;3).
`plt.scatter(x-list,y-list,"marque")` trace un nuage de points d'après une liste d'abscisses et une liste d'ordonnées, avec le style de la « marque ».	`plt.scatter([1,2],[3,7],"o")` affiche une "bulle" aux points de coordonnées (1;3) et (2;7).
`plt.plot(x-list,y-list,"marque")` trace un nuage de points d'après une liste d'abscisses et une liste d'ordonnées et le style de la « marque », et les segments connectant ces points.	`plt.window(-1.6,1.6,-.1,2)` `plt.plot([-1,0,1,-1], [0,1.732,0,0],"o")` affiche un triangle équilatéral avec des « bulles » aux sommets.
`plt.line(x1,y1,x2,y2,"mode")` trace un segment d'après les coordonnées de ses extrémités et ajoute éventuellement une flèche au bout.	`plt.line(0,0,2,3,"arrow")` trace le vecteur partant de l'origine et allant vers le point de coordonnées (2;3).

Appendice 2 : modules Python additionnels

turtle

La « tortue » peut être imaginée comme un stylet dessinant sur une feuille de papier et se déplaçant suivant des instructions du type « à gauche », « à droite », « en avant », etc. La « taille » de la feuille est de 316×208 pixels, avec un référentiel centré au milieu.

TI-83 Pour utiliser le module Turtle, il faut préalablement télécharger vers la calculatrice le fichier `CE_TURTL.8xv` . Il est possible de mettre ce fichier en « archive » (touche [mém], choix 2 et 1) si la mémoire fait un peu défaut. Ensuite, on doit passer dans l'environnement Python, créer un nouveau script vide et y écrire la ligne d'importation :

```
from ce_turtl import *
```

Dès que cette ligne est écrite, on trouve[1] dans le menu Modul une nouvelle ligne

```
8: ce_turtl …
```

donnant accès à l'ensemble des commandes et fonctions faisant partie de ce module.

Nspire CX Pour utiliser le module Turtle, il faut préalablement le télécharger dans la calculatrice le fichier `turtle.tns`, puis l'ouvrir comme classeur. Des instructions figurent à la page 1.1. Appuyer sur [ctrl] + [►] pour passer au shell Python figurant à la page 1.2, puis sur [menu] et sélectionner "Outils ► 8 installer en tant que module Python".

Une fois l'installation terminée, le message "**Installation terminée**" apparaît, confirmant l'installation[2]. Pour accéder à la programmation Python avec Turtle et aux menus adaptés, il convient de créer un nouveau document : depuis l'écran d'accueil, sélectionner "Nouveau / Ajouter Python / Nouveau...".

À ce stade, on est dans une page d'édition de programme Python. Quand on appuie sur la touche [menu], on peut sélectionner "A: Plus de modules ► 6 Turtle Graphics", les commandes Turtle y sont.

Dans tous les cas, la première ligne du programme consistera à créer une « instance » de Turtle, sous la forme `t=Turtle()` ou `turtle=Turtle()` selon le système et les mises à jour.

Voici quelques commandes fournies grâce à l'importation du module `turtle` :

- `t.clear()` permet d'effacer l'écran.
- `t.home()` permet de remettre la tortue au centre de l'écran et dirigée vers la droite (azimut 0°).
- `t.penup()` ou `t.pendown()` permet de lever ou baisser le stylet.
- `t.goto(a,b)` permet de se positionner aux coordonnées (a,b) (en pixels).
- `t.show()` « montre» la tortue et se met en attente d'un appui sur la touche [annul].
- `t.left(d)` et `t.right(d)` permettent de faire « tourner » la tortue (elles prennent comme paramètre un angle `d` en degrés).
- `t.forward(p)` permet de faire « avancer » la tortue d'un nombre `p` de pixels.
- `t.circle(r)` permet de tracer un cercle de rayon `r` pixels, la tortue étant au centre.
- `t.dot(s)` permet de tracer un disque de rayon `s` pixels.
- `t.color(r,g,b)` (ou `t.pencolor(r,g,b)` selon le système et les mises à jour) permet de fixer la couleur des traits en composantes de rouge, vert, bleu. Le code RGB permet de définir la couleur

1 Sous réserve d'avoir mis à jour la calculatrice avec un système 5.7 au moins.

2 Une fois installé, le fichier du module Turtle est déplacé dans le dossier Pylib, dans le dossier Pylib.

d'un pixel à l'aide de trois nombres entiers compris entre 0 et 255 décrivant des intensités lumineuses pour les composantes rouge, verte et bleue de la lumière (0 indiquant une absence de lumière).

- `t.pensize(p)` permet d'avoir des tracés de largeur fine (p=0), moyenne (p=1) ou épaisse (p=2).
- `t.speed(v)` permet d'avoir des tracés lents (v=0) ou rapides (v=1 ou v=10 selon le système).
- `t.setheading(d)` permet de « diriger » la tortue dans l'azimut (ou direction) `d`, `d` étant un angle en degrés compté à partir de l'horizontale dans le sens trigonométrique.
- `t.position(v)` renvoie un couple (a,b) donnant la position de la tortue (en pixels).
- `t.heading(v)` renvoie la valeur de l'azimut de la tortue (en degrés).

ti_draw

Ce module donne accès à des fonctions de tracé graphique « ponctuel » (bitmap) à l'écran avec des coordonnées « absolues » (pixels).

Instructions	Exemples
Sur l'écran de la calculatrice il y a des colonnes et des lignes de pixels. Les pixels de coordonnées x et y sont repérés à l'aide de ces lignes et colonnes, avec $0 \leq x \leq 319$ et $0 \leq y \leq 209$. Le « coin » en haut à gauche a les coordonnées (0;0).	On commence par effacer l'écran : `clear()`
`set_color(R,G,B)` fixe la couleur pour les traits d'après des intensités de rouge, vert, jaune (comprises entre 0 et 255, 255 étant l'intensité la plus forte).	`set_color(128,128,128)` choisit un « gris moyen » et `set_color(255,0,0)` choisit un rouge « vif ».
`plot_xy(x,y,type)` permet d'allumer le pixel aux coordonnées $(x;y)$ sur l'écran de la calculatrice avec un « type » de point compris entre 1 et 13 (plus ou moins gros).	`plot_xy(159,105,7)` allume le pixel du centre de l'écran dans la couleur choisie.
`draw_line(x1,y1,x2,y2)` trace un segment d'après les coordonnées de ses extrémités.	`draw_line(0,209,319,0)` trace la diagonale montante de l'écran.
`draw_circle(x,y,r)` trace un cercle d'après les coordonnées de son centre et son rayon.	`draw_circle(159,104,105)` trace le plus grand cercle entier possible.
`fill_circle(x,y,r)` trace un disque (cercle plein) d'après les coordonnées de son centre et son rayon.	`fill_circle(212,104,105)` trace le plus grand disque possible, calé à droite.
`draw_text(x,y,"texte")` trace un texte commençant aux coordonnées indiquées.	`draw_text(0,0,"TITRE")`
TI-83 : `show_draw()` finit le dessin et se met en attente de l'appui sur annul.	