Formation enseignants

Professeurs du lycée Diderot

oct. 01, 2021

General

CHAPITRE 1

Python pour les pédagogues : pydago

1.1 A propos

Partage pédagogique sur l'enseignement de Python dans le secondaire.

Le projet a démarré en 2018. Il se base sur le programme de mathématiques de Seconde de 2009 et l'enseignement de Python aux élèves de Seconde.

Depuis la rentrée 2019, les programmes du lycée ont changé, Python y est présent de manière officielle. Nous laissons les professeur-e-s de mathématiques ou des autres matières concernées enrichir le projet.

1.2 Participez !

Ce projet est un travail collaboratif initié par des enseignants du lycée Diderot à Marseille. Nous serions ravis de travailler avec vous et toute aide est la bienvenue. Si vous souhaitez participer, lisez notre fichier contributing.

CHAPITRE 2

Formation technique

2.1 1.Quoi ? pour qui ?

Ce document est utilisé pour former les collègues du lycée à l'utilisation de l'environnement python du lycée. Il est téléchargeable en pdf à l'url https://pydiderot.readthedocs.io/_static/formation.pdf. Il est à destination des professeurs n'ayant pas ou peu de connaissances sur le langage python.

L'objectif est d'introduire, à l'aide d'exemples et d'exercices :

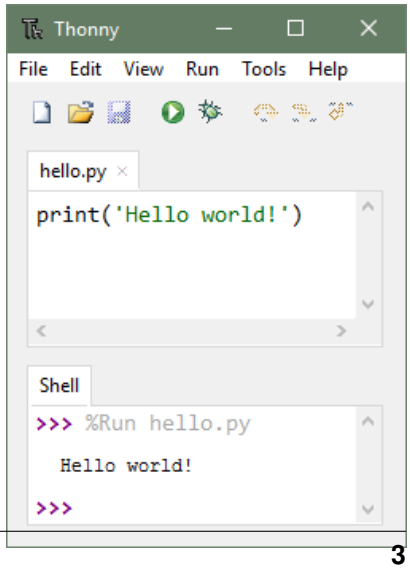
- le langage de programmation
- l'environnement de développement thonny utilisé au lycée
- les librairies propres au lycée Diderot
- quelques constats pratiques d'ordre pédagogique.

2.2 2. Prise en main

A partir du bureau des ordinateurs du lycée : `Logiciels > MATH > _Maths > _RACCOURCIS MATH_ > Thonny python`

Quand on lance Thonny, il y a deux zones :

- la zone du haut où on travaille dans un fichier, qui peut être exécuté quand on le demande, et qui peut être enregistré (sur cette image, le fichier s'appelle « Hello.py »)
- la zone du bas qui est un **shell** : comme un écran de calculatrice, mais qui exécute des commandes Python (on ne peut pas l'enregistrer)(c'est dans cette zone qu'apparaissent les résultats des commandes exécutées depuis un fichier de la zone du haut)



2.2.1 Quelques instructions à essayer dans le shell :

```
3 + 4
print("hello world")
"hello" + "world"
```

2.2.2 Quelques exemples à enregistrer dans un fichier :

```
3 + 4
print("hello world")
"hello" + "world"
```

Pour exécuter le fichier, il faut cliquer sur le bouton  ou appuyer sur *F5*.
Il faut donner un nom au fichier. Le fichier sera sauvegardé par défaut dans : `USER/python`

2.3 3. Quelques trucs sur python

Python est un langage de programmation libre qui fonctionne sur toutes plateformes (Windows, Mac, Linux, Android,. . .). Il à été créé au début des années 90 et baptisé en l'honneur de la série Monty Python's Flying Circus.

Épuré et lisible, Python est utilisé par de nombreuses entreprises et organisations (Google, NASA, . . .), dans la recherche scientifique et est enseigné au lycée, en prépa, et à l'université.

Les projets de nouveaux programmes précisent le choix de python comme langage de programmation utilisé à partir de la seconde.

2.3.1 3.a. Variable

Une des difficulté principale pour les élèves au début (en seconde) est l'utilisation de variables.

Note : Voir *5.b* ci dessous pour des remarques concernant des outils pédagogiques à la notion de variable.

Quelques lignes à tester :

```
x = 4
y = x + 1
print(y)
```

2.3.2 3.b. Typage et ce que provoque l'absence de typage

Les variables peuvent être de différents types : par exemple « entier » `int` ou « chaîne de caractères (texte) » `str` ou encore « nombre à virgule flottante » `float`.

Souvent, on peut suggérer facilement à Python le type utilisé. Par exemple, `a=2` est de type `int`, `a=2.0` est de type `float`, `a='2'` ou `a="2"` est de type `str`.

Si Python ne sait pas trop quel est le type d'une variable, il essaye de faire un choix (par exemple il considère que la variable est de type « texte » `str`) ou alors il renvoie un message d'erreur.

Les messages d'erreur apparaissent en rouge dans le shell. Même s'ils sont indigestes, il faut expliquer aux élèves que dans un message d'erreur il y a deux informations très utiles :

— une explication sur l'erreur (par exemple le mot-clé « typeError » signale qu'il s'agit d'une erreur de typage)
— la ligne du fichier où l'erreur a été rencontrée (en bleu et cliquable pour aller directement au bon endroit du fichier)

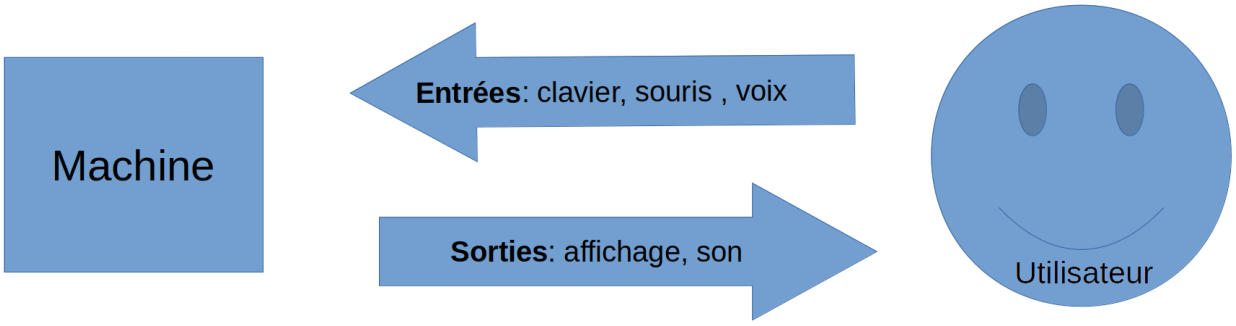
Essayez de faire exécuter ces instructions :

```
3 * '13'
'3' * '13'
'3' + '13'
3 + '13'
3 * 13
3 * 13.0
```

Note :

— pour aller vite dans le shell, on peut rappeler les instructions précédemment tapées en appuyant sur la flèche du haut
— pour mettre un commentaire, il suffit de le précéder d'un #

2.3.3 3.c. Deux fonctions d'entrée / sortie :



— `input()` permet de faire demander par Python à l'utilisateur de d'entrer quelque chose au clavier.
— `print()` permet de demander à Python l'affichage de quelque chose.

Essayez de faire exécuter ces instructions :

```
input("tapez quelque chose")
a = input("tapez quelque chose")
a
print(a)
type(a)
```

2.3.4 3.e. Retour sur les problèmes de typage : formule magique

Si vous avez essayé les lignes de code précédentes avec diverses entrées tapées au clavier, vous avez peut-être remarqué que quel que soit ce qu'on tape, Python considère que ce qui vient du `input()` est du type `str`.

Du coup, cela va sans cesse provoquer des erreurs dans les programmes des élèves. Un des moyens d'éviter cela est de leur donner la ligne de `input` sous la forme suivante :

```
a = float(input("tapez quelque chose"))
```

`a` sera alors automatiquement du type « nombre à virgule flottante », ce qui est en général ce qu'on veut.

Exercice : Demander deux nombres à l'utilisateur et afficher leur produit.

2.3.5 3.d Indentation

Pour signaler le début et la fin de blocs de code, on utilise l'indentation. Une indentation correspond à 4 espaces.

```
print("Bonjour!")

for x in range(10):
    # début du sous-block for
    print(x)

# Nous sommes sortis du sous-bloc "for"
print("C'est fini.")
```

```
print("Bonjour!")

for x in range(10):
    # début du sous-block "for"
    print(x)
    # Nous sommes toujours dans le bloc "for"
    print("C'est pas fini.")

# Nous sommes sortis du sous-bloc "for"
print("C'est fini.")
```

Note : L'utilisation du mode « pas à pas » de thonny peut aider à illustrer cette notion (voir *5.c*).

2.4 4. Librairies :

2.4.1 4.a. Généralités :

Python ne charge pas toutes les commandes disponibles lorsqu'on le lance. Si on a besoin d'une commande non chargée, il faut demander le chargement de la librairie.

```
print(sqrt(2))
```

solutions :

1. On importe la fonction `sqrt` de la librairie math. On peut ensuite l'utiliser directement.

   ```
   from math import sqrt
   print(sqrt(2))
   ```

2. Une autre solution qui importe tout le contenu de la librairie math :

```
from math import *
print(sqrt(2))
print(pi)
```

2.4.2 4.b. Quelques exemples avec les librairies « maison »

La librairie entree permet de répondre au problème de typage évoqué précédemment :

```
from entree import *
demander_texte()
a=demander_reel()
print(type(a))
print(a)
```

La librairie repere permet d'afficher facilement une fenêtre graphique munie d'un repère où on fera par exemple afficher la courbe d'une fonction.

```
from repere import *
creer_fenetre()
trace_segment(1,2,3,5)
trace_point(3,3)
```

Un autre exemple, plus élaboré :

```
from math import *
from repere import *
creer_fenetre()
for i in range(100): #i va de 0 à 100
    x = i / 10 - 5 #x va de -5 à 5
    trace_point(x,cos(x))
```

La même chose avec un « while » au lieu du « for » (quel est le plus simple avec les élèves ?) :

```
from math import *
from repere import *
creer_fenetre()
x = -5
while x < 5:  # x va de -5 à 5
    trace_point(x, cos(x))
    x = x + 0.1
```

2.5 5. Python avec les élèves

2.5.1 5.a Prise en main

Voici quelques remarques que nous espérons utiles :

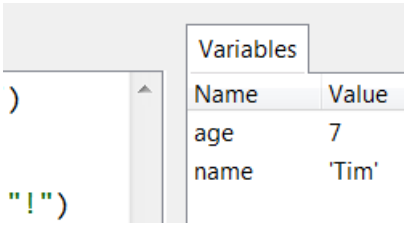
— Les élèves enregistrent leur travail dans le dossier `USER/python/`.
— Lors de la vidéoprojection de thonny, les caractères sont trop petits. On peut utiliser *ctrl* + + ou *ctrl* + *molette* pour zoomer.
— La config est enregistrée automatiquement quand on quitte thonny.
— Pour afficher les numéros des lignes : *Tools* → *Otions* → *Editor* → *Show lines numbers*

2.5.2 5.b Variables

Comme annoncé plus haut, la notion de variable informatique n'est pas du tout évidente à appréhender pour les élèves.

Il peut être bon d'expliquer que concrètement, à chaque nouvelle variable `x`, la machine va créer un espace (une boite) dans la mémoire vive étiquetée par le nom de la variable (`x`). Nous pouvons ensuite lire le contenu de la boite avec `print(x)` et changer ce qu'elle contient avec `x = 3`.

Thonny propose un affichage dynamique du contenu des variables accessible dans le menu *View* → *Variables* qui peut être particulièrement utile dans un cadre pédagogique.



Son utilisation n'est **pas compatible avec l'import de librairies**. En effet, l'onglet affichera le contenu de toutes les fonctions et variables importées ce qui peut ralentir considérablement l'utilisation.

2.5.3 5.c Debug et les différents types de « pas à pas »

Thonny propose un mode débug accessible avec le bouton ou les touches *crtl* + *F5* qui permet l'éxécution du script en mode « pas à pas ». Cela activera le menu debug :

- ou *F6* : Passe au bloc suivant sans rentrer en détail dans le bloc.
- ou *F7* : Rentre dans le bloc sélectionné pour en voir les détails.
- : Sort du bloc sélectionné en remontant vers le bloc parent.
- ou *F8* : Quitte le mode débug et reprend l'exécution du script.

Voici par exemple un programme affichant la table de multiplication par 7 dont la vidéoprojection en mode débug peut avantageusement illustrer le fonctionnement d'une boucle et des variables :

```
for x in range(10):
    y = 7 * x
    print(y)

print("Et voila!")
```

2.5.4 5.d. Gestion salle info

Ce qui ne marche pas : parler à un groupe d'élèves devant leur écran.

Ce qui peut marcher :

- faire se lever tout le groupe, venir devant le tableau et parler avec le vidéo-projecteur
- verrouiller tous les écrans (ça peut se faire avec le logiciel iTALC (il y a souvent un ou deux écrans qui ne se verrouillent pas !), donner les consignes, puis déverrouiller
- envoyer son écran dans les écrans de tous les élèves (avec la commande « démo » du logiciel iTALC), parler en montrant à l'écran en même temps, puis arrêter la démo (les élèves sont encore bluffés par cette manip pour le moment :-)
- annoncer au groupe « je vais vous envoyer l'écran de -nom d'élève- » (avec la commande clic droit > « laisser faire une démo » du logiciel iTALC), laisser l'élève faire sa démo (en l'incitant à parler de façon compréhensible), puis arrêter la démo (je ne sais pas comment faire alors je verrouille / déverrouille !)

2.6 6. un peu plus loin :

2.6.1 6.a Questions fréquentes

— comment peut-on (les élèves ou nous) télécharger thonny pour chez nous ?

Il suffit de se rendre sur https://pydiderot.readthedocs.io

— comment peut-on demander à ajouter une fonction dans les librairies ?

Ouvrir une issue sur https://github.com/cspaier/pydiderot

2.6.2 6.b. Exemples d'activités

2.6.2.1 Le milieu

```
xA = float(input("Abscisse de A ? "))
yA = float(input("Ordonnée de A ? "))
xB = float(input("Abscisse de B ? "))
yB = float(input("Ordonnée de B ? "))
xM = (xA + xB) / 2
yM = (yA + yB) / 2

print("Coordonnées du milieu : (" + str(xM) + " ; " + str(yM) + ")")
```

exo : le refaire et l'améliorer en ajoutant de quoi afficher la distance AB

Remarque on peut demander exactement la même chose à une classe à condition d'écrire au tableau la « formule magique » ou d'utiliser la libraire entree.

2.6.2.2 Ca bouge !

Voici un exemple utilisant la librairie graphique où une balle traverse l'écran en diagonale.

```
# On importe la librairie
from graphique import *
# Nous aurons également de la librairie time
import time

# On initialise les coordonnées du point au coin haut gauche de la fenêtre
x = 0
y = 0
# On initialise les coordonnées du vecteur vitesse
v_x = 1
v_y = 1

# On créé la fenêtre graphique de taille 200 x 300
creer_fenetre()

# Boucle principale
while 1:
    # Il est important d'appeler  la fonction evenements() qui gère la fermeture de␣
↪la fenêtre
    evenements()

    # Trace un cercle au coordonnées (x,y)
```

(suite sur la page suivante)

(suite de la page précédente)

```
    trace_cercle(x, y)
    # Attend un dixième de secondes
    time.sleep(0.1)
    # Efface le cercle
    trace_cercle(x, y, couleur=blanc)
    # Ajoute le vecteur vitesse aux coordonnées du point
    x += v_x
    y += v_y
```

a. Modifier le code pour que la balle traverse l'autre diagonale.

b. Faire en sorte que la balle rebondisse sur les bords de l'écran.

c. Faire en sorte que l'utilisateur déplace la balle avec la sourie ou l'écran. On pourra faire un `print(evenements())` pour explorer la gestion des événements.

CHAPITRE 3

Formation Python 2 : échanges sur le plan pédagogique

3.1 1. Quoi ? pour qui ?

Ce document est utilisé en formation interne au lycée, pour partager nos expériences avec l'environnement python du lycée. Il est téléchargeable en pdf à l'url https://pydiderot.readthedocs.io/_static/formation2.pdf.

Il fait référence à des documents mutualisés en interne par les professeurs du lycée Diderot, Marseille. Ces documents ne sont pas forcément publics.

L'objectif est de :

- partager des idées d'activités sur l'algorithmique (une synthèse des activités mutualisées par les collègues est proposée)
- partager les difficultés que rencontrent nos élèves dans l'apprentissage de l'algorithmique
- partager les difficultés techniques que nous rencontrons ou que rencontrent nos élèves dans l'utilisation de l'environnement pydiderot
- mettre en place une progression commune sur l'algorithmique

3.2 2. Synthèse des activités mutualisées

Les documents qui se trouvent dans : `Atrium > Mes sites > Professeurs de maths > Documents > Python > Mutualisation` se répartissent ainsi : il y a des documents généraux, des fiches techniques, des activités à faire avec les élèves (certaines avec Python, d'autres sans Python).

3.2.1 1. Documents généraux

- une liste d'idées d'algorithmes qui peuvent être en lien avec les programmes
  fichier « progression algo en parallèle.odt », Loïc
- quelques trames de séances très basiques sur la prise en main, les opérations de base, une première fonction, le début avec les listes (Python)
  fichier « séances_python.odt », Loïc

— des remarques sur l'introduction progressive de la notion de variable, avec quelques idées d'activités
fichier « progression début algo », Paul

3.2.2 2. Fiches techniques

— une fiche à donner aux élèves, reprenant la liste des commandes disponibles, avec à chaque fois une description et un exemple. Cette fiche correspond à un petit environnement javascript, du genre de celui visible ici, et repris en blockly ici. Il faudrait la refaire pour pydiderot. Il serait sans doute judicieux également de créer un environnement Blockly qui mime pydiderot, de façon à pouvoir basculer facilement de l'un à l'autre, notamment en classe de seconde…
fichier « Liste instructions python.pdf », Paul

3.2.3 3. Activités avec Python

— deux activités sur la recherche du maximum d'une fonction (boucles while en python et utilisation de la librairie « repère »)
fichier « 2nde act algo fonction max.odt », Paul
— un TP de 1S avec boucle for, boucle while et librairie « repère »
fichier « TP_1S.odt », Clément
— une activité de 1STI2D (prise en main de Python, boucle for, à réécrire avec la nouvelle version de pydiderot)
fichier « 1STI2D_La fusée dans l'espace.pdf »,Loïc
— une activité sur les suites en 1S qui donne lieu a un travail sur les boucles
fichier « Activités suites et boucles.pdf », Loïc

3.2.4 4. Activités sans Python

— une activité sans ordinateur, pour mettre en évidence la notion de boucle et celle de variable
fichier « act debranchee boucle et variable (1).odt », Paul
— une activité sur la découverte de la dérivée de la fonction carrée en réalisant l'enveloppe d'une famille de droites (boucle "for" *préparée* dans un outil Wims)
fichier « 1STD2A act tangentes fonction carrée.odt », Paul
— une activité de 2nde pour utiliser les coordonnées (variable, boucle, code à modifier facilement, mais compliqué à écrire pour les élèves)(Javascript)
fichier « 2nde act algo enonce (javascript).odt », Paul
— une activité de seconde sur les vecteurs (coordonnées de vecteurs pour faire rebondir une balle en mouvement sur les parois d'un rectangle)(blockly)
fichier « act 2nde vecteurs blockly.odt », Paul
— un exercice d'entraînement où il faut créer du code blockly
fichier « 2nde axo algo blocklypouraujourdhui.odt », Paul
— une activité plus difficile où il faut utiliser les intervalles et la logique pour programmer une fonction qui détecte si deux rectangles s'intersectent ou non (pseudo-code / javascript)
fichier « 2nde act algo jeu video intervalles.odt », Paul
— Quelques idées pour la TSTD2A (javascript / pseudo-code)
voir ici
— Quelques idées pour la seconde (javascript)
voir ici

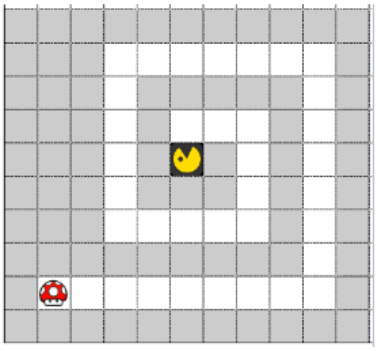
3.3 3. Difficultés

La notion de variable pose des problèmes aux élèves. Son introduction doit être considérée comme délicate : attention à ne pas croire que c'est facile pour eux, sinon on risque des désillusions.

Ceci dit, différentes utilisations existent : `x=5` est plus simple à comprendre pour les élèves que `x=x+1` ou que `for j in range(5)...`

Pour y aller en douceur, on peut par exemple faire une première activité « débranchée » (mais cela peut sembler inutile lorsqu'ils ont déjà travaillé ce genre de choses au collège...).

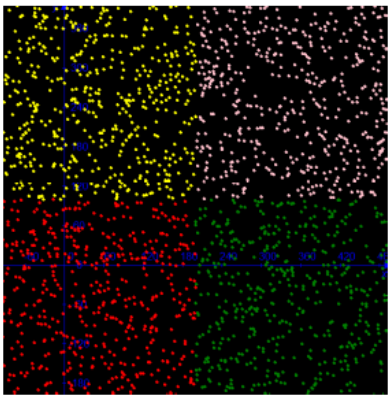
Un exemple d'activité débranchée où le codage de `j=j+1` s'impose :



On peut aussi faire un travail préalable en maths (avant ou en parallèle du travail en algo) : faire une ou deux études de fonctions où les variables sont modélisées par des curseurs GeoGebra par exemple...

Remarques générales : la notion de boucle pose moins de problème aux élèves que la notion de variable. On peut très bien faire des boucles sans variable, par exemple en blockly ou en « débranché ». L'instruction est alors du type « répéter 5 fois... ». Les élèves comprennent très bien l'intérêt de demander à l'ordinateur de répéter plusieurs fois les mêmes instructions.

Un exemple d'image qu'on peut générer avec une fonction « entier_aleatoire() » et des boucles de type « répéter 1000 fois ». Aucune variable n'est nécessaire.



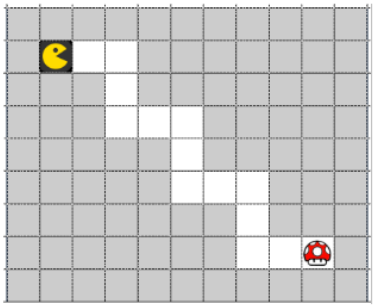
Cependant, en Python, difficile de faire une boucle sans variable. On a essentiellement le choix entre « for j in range(5) » et « while j<5 : ... j=j+1 ». Dans les deux cas, le travail sur la variable « compteur » j s'impose.

Une autre remarque : ce travail a un coût pour les élèves. Si on présente par exemple les quelques lignes de syntaxe permettant de faire une boucle « while », ces lignes ne paraissent pas naturelles. De plus, leur mémorisation par les élèves ne va pas du tout de soi. Enfin, ils ne comprennent pas forcément très vite à quel point ces quelques lignes

permettent de résoudre un GRAND nombre de problèmes. Tous les ingrédients sont donc réunis pour que les élèves décrochent.

Propositions de remèdes : aller progressivement, faire faire prendre conscience de l'utilité de la structure de boucle while…

Voici un exemple d'activité débranchée pour mettre en évidence l'intérêt de faire une boucle. Cependant, on n'a pas besoin spécialement d'une boucle « while » !



Une idée : programmer une fonction « repeter(n,f) » qui répète n fois la fonction f (cela nécessite de commencer très tôt la notion de fonction en algorithmique mais ce n'est sans doute pas grave).

On utilise plusieurs fois cette fonction.

Ensuite, on s'arrange pour proposer des problèmes dans lesquels on ne sait pas à l'avance le nombre de fois où il faut répéter et/ou des problèmes où la vraiable « compteur » est utile. Cela permet alors de proposer la boucle while.

Un peu de comparaison entre les deux types de boucles :

3.3.1 Boucle « while »

Exemple simple :

```
x=0
while x<3:
    print(x)
    x=x+1
print('fini !')
```

L'utilisation de cette boucle nécessite d'expliquer `x=x+1`

Elle nécessite de traduire en langage python « tant que j < 5, il faut répéter ….. fin. »

Le « il faut répéter » se traduit par « deux points » !

Le « fin » se traduit par une absence de tabulation ! (Donc il faut expliquer ce qu'est une tabulation…)

3.3.2 Boucle « for »

Exemple simple :

```
for x in range(3):
    print(x)
print('fini !')
```

Cette boucle nécessite d'expliquer le range()...

La difficulté peut-être contournée dans un premier temps par l'intruction suivante :

```
for x in 1,2,3,4,5,6,7,8:
   U=U+2
```

en calculant seulement le U au rang 8.

3.3.3 Boucles et suites

Observations après l'activité boucle et suite (cf. plus haut). Il y a beaucoup de variables à considérer pour une suite Un. Les valeurs et rang initiaux, les valeurs et rang finaux désirés et un compteur de boucle : tous les rangs intermédiaires qu'il faut calculer.

Le "n" dans Un en math joue à la fois le rôle de rang final et de compteur de la boucle...

3.3.4 Difficultés liées au langage

Python est un langage assez naturel et souple à utiliser, mais il y a quand même pleins de choses plus ou moins implicites chez les programmeurs et qui doivent être expliquées et travaillées avec les élèves. Petite liste :

Il faut expliquer le fait qu'il y a du code qu'on écrit en général dans un fichier et qu'il y a ensuite l'exécution du code.

Que l'indentation a du sens et qu'il ne faut pas indenter n'importe comment.

Que les commandes sont suivies de parenthèses, sans espace (comme par exemple `print('coucou')`)

Expliquer aussi les différents types (ou au moins les marqueurs « guillemets » pour le type `string`)

Que la fonction `input()` renvoie un string lorsqu'on lui passe un nombre et qu'il faut le convertir avec `int()`.

Exemple :

```
n=input("donner un nombre : ")
n=int(n)
```

3.3.5 Difficultés liées aux librairies créées spécialement pour pydiderot

Pydiderot contient le langage Python, l'IDE Thonny, quelques librairies couramment utilisées en Python et quelques librairies spécialement créées pour faciliter le codage à des élèves de lycée. ces dernières librairies ont été créées par d'autres et un peu modifiées pour le lycée Diderot, ou parfois créées spécialement pour le lycée Diderot.

Elles induisent fatalement quelques problèmes (mais en même temps elle permettent bien sûr de coder facilement des trucs intéressants et normalement inaccessibles pour des débutants !). Petite revue :

Il est difficile de devoir sans cesse réécrire au tableau les noms des commandes. De plus, les nommages ne sont pas forcément ultra-judicieux.

Conclusion : d'où la proposition ci-dessus de créer un document pdf avec les différentes commandes + pour l'année prochaine, renommer les commandes. Le plus cohérent avec python existant : nommer les commandes en anglais + conserver la même structure par exemple au lieu de `creer_fenetre()` et `trace_point()`, mettre : `window()` et `point()`.

Autre proposition : on peut multiplier les fonctions (faire des alias) : faire une fonction `fenetre()` ET une fonction `window()` qui font la même chose.

3.3.6 Difficultés techniques

Les plantages de la nouvelle version de pydiderot : aucun réel plantage n'a été détecté, par contre une difficulté liée sans doute à l'installation sur le réseau du lycée revient fréquemment. Thonny se met à « ramer ». Dans ce cas, il suffit de le fermer et de le réouvrir. Le travail est automatiquement enregistré, donc rien n'est perdu. Ca fait juste un peu « bricolage »…

3.4 4. Progression

— fonction (expliquer les différences avec les fonctions en maths)
— boucle (sans variable –> « répéter n fois »)
— variable (en liaison avec les fonctions d'une variable en maths)
— boucle avec variable, mais simple (que des while avec la variable compteur qui est utilisée de façon hyper standardisée ?)
— condition « if… » (pas de difficulté our les élèves ?)

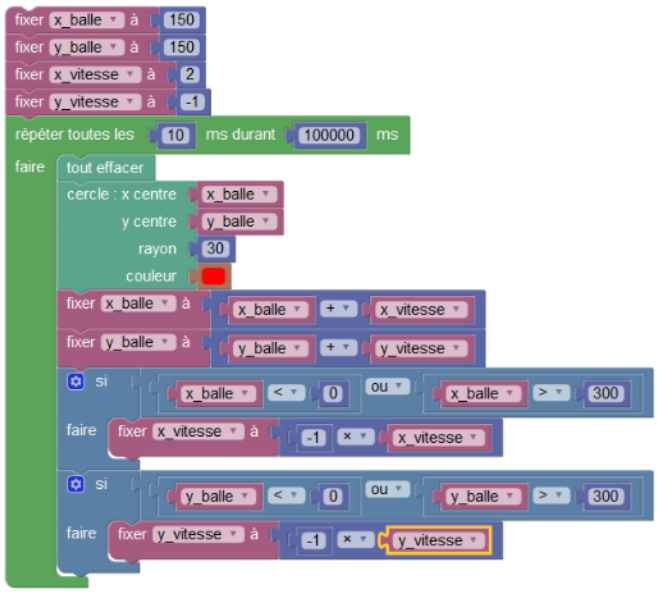
— notion de liste
— boucle « for »
— boucles avec variables plus compliquées (typiquement : boucles pour résoudre les problèmes sur les suites, dans lesquels il y a aussi bien un travail possible sur l'indice de la suite que sur la valeur du terme de la suite)
— boucle avec un « append » (instruction qui ajoute un élément à une liste)
Terminale :
— rien de plus, retravail de ces notions

3.5 5. Les consignes dans nos activités

Quand on lit tout ce qui a été mutualisé, on peut regrouper les activités en fonction (par exemple) du type de consigne.

Une consigne très bête, mais parfois très utile : *recopiez ces lignes de code et faites les s'exécuter sur votre machine !*

Une autre solution plus élégante :



Une consigne un peu différente : *recopiez puis analysez*

1. Commencez par recopier exactement dans votre fichier l

```
from ZG import *

zonegraphique()
axes()

affichage()
```

2. Enregistrez votre fichier. La première ligne `from ZG in` facilement un autre fichier nommé `ZG.py`. Il faut absol quand on veut faire des graphiques.

3. Lancer votre programme en cliquant sur l'icône 

4. Que se passe-t-il ?

5. Enlevez la ligne `axes()`
   Relancer le programme. Qu'observez-vous ?

Une autre consigne, qui marche super bien mais qui n'apprend sans doute pas réellement aux élèves à coder : *modifiez ces lignes de code pour qu'on obtienne tel ou tel résultat* (Les élèves aiment bien et ils arrivent assez vite au résultat demandé en tâtonnant. Cependant : ils s'inventent des tas de « théorèmes élèves » sur le code qu'ils manipulent (c'est à dire qu'ils le comprennent souvent assez mal). Par ailleurs, si on leur demande par la suite de coder un petit programme à partir de zéro, c'est assez catastrophique si ça n'a pas été travaillé également de façon approfondie.

```
//les dimensions de la fenêtre du jeu sont : 300x3
var x = 150;
var y = 150;
var h = 1;

function draw() {
  //on efface tout
  clear();
  //on dessine un disque rouge centré en (x,y)
  circle(x, y, 10,'red');
  //on modifie la valeur de y
  y = y+h;
}

init(); //on initialise le jeu
loopdraw(); //on répète la fonction "draw" indéfin
```

Vous pouvez vous amuser à modifier le code ci-dess
1- Comment placer la balle en haut à gauche ?
2- Comment faire descendre la balle ?
3- Comment faire se déplacer la balle horizontaleme
4- Comment changer la couleur et le rayon de la bal
5- Comment accélérer la balle ?

Un exemple de consigne totalement non technique (là, au contraire, il s'agit vraiment de codage !) :

Un exemple de type « mini projet » où la consigne est très libre. Cependant, le guidage provient du fait que les élèves vont essentiellement copier les exemples qui ont été travaillés précédemment.

Le premier obtient de sa banque un placement au taux de $4,00\%$. Chaque année il prélève 3600€ (soit l'équivalent de 300€ par mois).

Le deuxième prélève 1200€ par an. Le compte n'est pas rémunéré.

Le troisième, quant à lui, prélève chaque année $3,00\%$ de l'héritage restant de l'année précédente. Le compte est rémunéré à $1,25\%$.

- De quels montants disposeront les trois frères après 5 ans, 10 ans, 20 ans.
- Étudier le comportement des trois placements (le montant placé augmente-t-il ? , baisse-t-il ? , stagne-t-il ?).

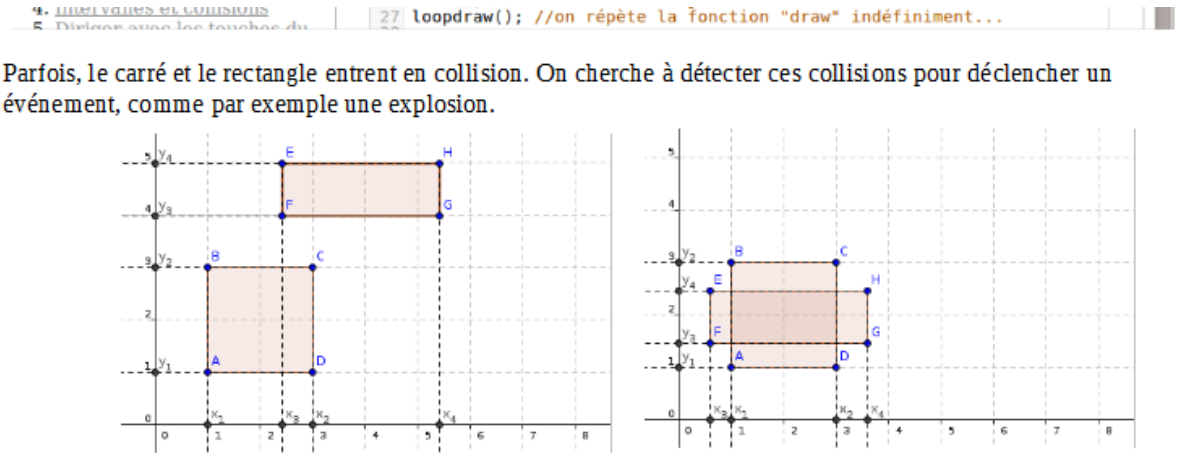
2nde | Activité | Algorithmique.

Vous allez commencer un projet de programmation : décrivez en quelques lignes ce que vous voulez faire.
Vous pouvez facilement :
- afficher et faire se déplacer à l'écran un disque ou un rectangle coloré, ou encore une image.
- piloter un déplacement avec certaines touches du clavier
- utiliser des conditions « si...alors...sinon »

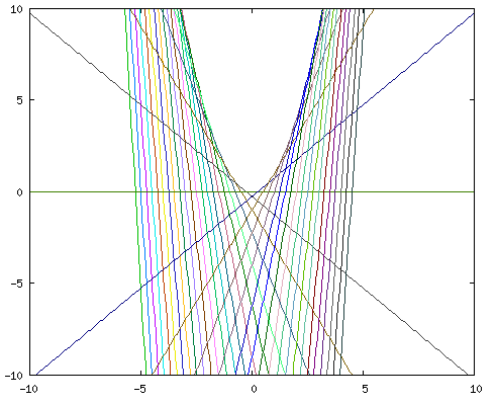
3.6 6. Le dialogue entre les maths et l'algorithmique dans nos activités

Quand on lit tout ce qui a été mutualisé, on peut aussi regrouper les activités en fonction de l'objectif qui est présenté aux élèves.

Ici, un problème d'algorithmique devient le prétexte pour une étude mathématique qui utilise la notion d'intervalle de façon un peu inhabituelle :



Ici, on veut tester (en 1STD2A, sans avoir les moyens de le justifier mathématiquement) que la dérivée de la fonction carré est x->2x. Pour cela, on trace à l'aide d'une boucle for pleins de tangentes en calculant leurs coefficients directeurs grâce à cette fonction. Le résultat est une famille de droites faisant apparaître clairement la parabole de la fonction carré. C'est à la fois beau et convaincant.


10
5
0
−5
−10
−10
−5
0
5
10

.

CHAPITRE 4

Formation Python 3 : échanges équipe de maths / équipe de physique

4.1 1. Aspects techniques

4.1.1 Python

Langage multiplateforme ordinateur, calculatrice, microcontroleur

4.1.2 IDE Environnement de développement

Un IDE réunit :

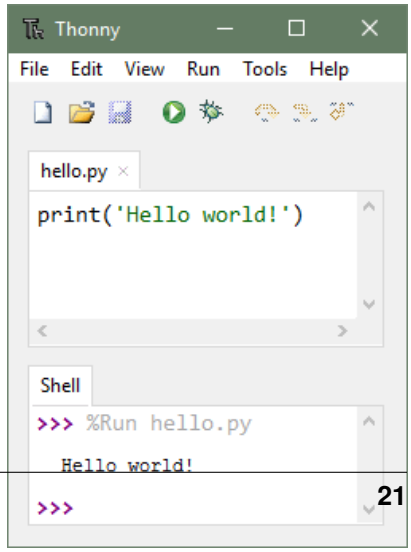
- un traitement de texte (à coloration syntaxique)
- une console (accès direct)
- un lancement de python

4.1.3 L'IDE Thonny

A partir du bureau des ordinateurs du lycée : `Logiciels > Python > Thonny`

Quand on lance Thonny, il y a deux zones :

- la zone du haut où on travaille dans un fichier, qui peut être exécuté quand on le demande, et qui DOIT être enregistré (sur cette image, le fichier s'appelle « Hello.py »)
- la zone du bas qui est un **shell** (ou **console**) : comme un écran de calculatrice, mais qui exécute des commandes Python (on ne peut pas l'enregistrer). C'est dans cette zone qu'apparaissent les résultats des commandes exécutées depuis un fichier de la zone du haut.



4.1.3.1 Autres IDE accessibles du lycée

— Edupython : connue et souvent montré en démo
— Mu : Pour programmer certains microcontroleurs

4.1.3.2 Quelques instructions à essayer dans la console :

```
3 + 4
print("hello world")
"hello" + "world"
```

4.1.3.3 Quelques exemples à enregistrer dans un fichier :

```
3 + 4
print("hello world")
"hello" + "world"
```

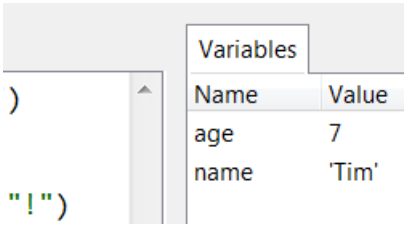
Pour exécuter le fichier, il faut cliquer sur le bouton  ou appuyer sur *F5*. Il faut donner un nom au fichier. Le fichier sera sauvegardé par défaut dans : `U:\Documents`

4.1.3.4 Quelques remarques pour l'utilisation avec les élèves :

— Lors de la vidéoprojection de thonny, les caractères sont trop petits. On peut utiliser *ctrl* + + ou *ctrl* + *molette* pour zoomer.
— La config est enregistrée automatiquement quand on quitte thonny.
— Pour afficher les numéros des lignes : *Outils* → *Otions* → *Editeur* → *Afficher les numéros de ligne*
— La notion de variable informatique n'est pas du tout évidente à appréhender pour les élèves. Il peut être bon d'expliquer que concrètement, à chaque nouvelle variable `x`, la machine va créer un espace (une boite) dans la mémoire vive étiquetée par le nom de la variable (`x`). Nous pouvons ensuite lire le contenu de la boite avec `print(x)` et changer ce qu'elle contient avec par exemple `x = 3`.

´ Pour l'affichage dynamique du contenu des variables :

Affichage → *Variables* (A désactiver lorsqu'on importe de grosses librairies car sinon l'onglet affichera le contenu de toutes les fonctions et variables importées, ce qui peut ralentir l'utilisation.)



— Pour exécuter en mode débug : bouton  ou touches *crtl* + *F5* qui permet l'éxécution du script en mode « pas à pas ». Cela activera le menu debug : 

4.1.3.5 Librairies

Python possède de très nombreuses librairies. Ce sont des bout de programmes déjà écrits qui permettent d'utiliser des fonctions parfois très complexes.

par exemple la librairie mathématiques

```
import math
print(math.sqrt(9))
```

Il y a plusieurs librairies classiques déjà installées : numpy, scipy, matplotlib, sympy, pygame

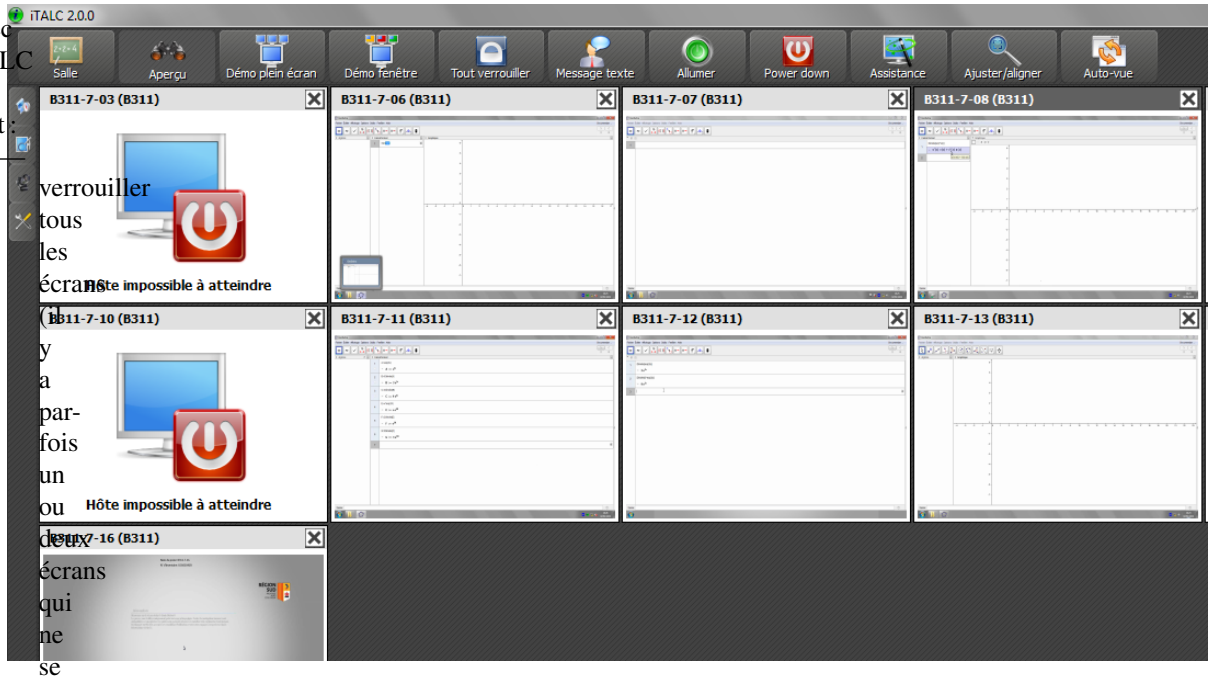
Pour voir la liste de celles qui sont installées au lycée, vous pouvez aller voir dans : `C:\Python3.7\Lib\site-packages`.

Si vous voulez demander l'installation d'une librairie supplémentaire, faites un ticket (`Outils Profs > Support Informatique`). On peut aussi gérer les paquets à partir de Thonny (*Outils → Gérer les paquets*).

4.1.4 iTALC

Ce qui ne marche pas : parler à un groupe d'élèves qui sont chacun devant leur écran.

Le petit outil iTALC (dans une salle informatique, accessible depuis : `Outils Profs > iTALC`) permet de mieux gérer les moments où on réclame l'attention des élèves.



Avec iTALC on peut :

— verrouiller tous les écrans (il y a parfois un ou deux écrans qui ne se verrouillent pas !)

— envoyer le contenu de son écran vers les écrans de tous les

élèves (avec la commande « démo »)

— annoncer au groupe « je vais vous envoyer l'écran de - nom d'élève- » (avec la commande clic droit > « laisser faire une démo »)

4.1.5 Pydiderot : une organisation hébergée sur Github

4.1.5.1 librairies de pydiderotlibs :

Pour faire des graphiques même simples, les manuels utilisent les bibliothéques numpy et matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
# U et I sont à compléter par les élèves
# Tableau entre crochets, nombres séparés d'une virgule
U = np.array([-5.0,-4.0,-3.0,-1.0,0,1.0,2.0,3.0,4.0,5.0])
I = np.array([-0.052,-0.039,-0.028,-0.011,0,0.01,0.021,0.032,0.039,0.05])

#Tracé de la courbe
plt.grid()
plt.title('Caractéristique tension-intensité')
plt.xlabel('I(en A)')
plt.ylabel('U(en V)')
plt.plot(I,U,'+',label= 'Points issus de la mesure')
```

(suite sur la page suivante)

(suite de la page précédente)

```
plt.legend()

# Effectuer une regression linéaire et la tracer
Umod = np.polyfit(I,U,1)
print('Um =', round(Umod[0],2),'x','I','+',round(Umod[1],4))

Reg = np.poly1d(Umod)
x = np.linspace(1.2*min(I), 1.2*max(I))
y = Reg(x)
plt.plot(x, Reg(x))

plt.show()
```

Il existe des librairies qui permettent de simplifier cette écriture. En particulier la librairie **pydiderotlibs** :

```
from pydiderotlibs.repere import *
fenetre()

i=0
while i<10:
    point(i,i**2)
    i=i+1
```

Tout un tas de ressources sont disponibles sur un compte Github qui a été développé par l'équipe de maths (à la suite de Clément Spaier). https://github.com/Pydiderot Vous pourrez y trouver :

- les notes de cette pettie formation en allant dans pydago https://pydago.readthedocs.io
- les librairies « maison » en allant dans pydiderotlibs https://pydiderotlibs.readthedocs.io
- des liens pour télécharger thonny en allant dans pydiderotIDE https://pydiderotide.readthedocs.io

4.2 2. Aspects pédagogiques

Proposition de progression commune concernant l'algo au lycée (à re-discuter...)

4.2.1 Seconde :

- fonction (expliquer les différences avec les fonctions en maths)
- boucle sans variable –> « répéter n fois »
- variable (en liaison avec les fonctions d'une variable en maths)
- boucle avec variable, mais simple (seulement des boucles « while » avec la variable compteur qui est utilisée de façon hyper standardisée)
- condition « if... » (pose peu de difficulté avec les élèves)

4.2.2 Première :

- notion de liste
- boucle « for »
- types string, integer, float...
- boucles avec variables plus compliquées (typiquement : boucles pour résoudre les problèmes sur les suites, dans lesquels il y a aussi bien un travail possible sur l'indice de la suite que sur la valeur du terme de la suite)
- boucle avec un « append » (instruction qui ajoute un élément à une liste)

4.2.3 Terminale :

— rien de plus, retravail de ces notions

4.2.4 Exemples en classe de seconde :

Il faut échanger sur le type d'utilisations qui sont envisagées en maths et en physique.

4.2.4.1 Afficher un repère avec la librairie pydiderotlibs :

```
from pydiderotlibs.repere import *

fenetre()
```

Remarque : l'instruction

```
window()
```

est un alias de fenetre().

A la souris, on peut déplacer la fenêtre ; zoomer ; zoomer suivant un seul axe de cooordonnées.

Il y a un lien dans la console pour accéder à la documentation de la librairie.

4.2.4.2 Une fonction qui n'utilise presqu'aucune variable :

```
from pydiderotlibs.repere import *
from random import randint
def flocon():
    x=randint(-100,100)/10
    y=randint(-100,100)/10
    point(x,y,'pink',2)

fenetre()
flocon()
```

4.2.4.3 Une boucle utilisant la fonction précédente et utilisant une variable, mais n'utilisant aucune liste :

```
i=0
while i<1000 :
    flocon()
    i=i+1
```

4.2.5 Exemples en classe de première :

perso je n'ai rien à proposer...

On peut espérer qu'avec un peu de travail les élèves puissent écrire à partir de zéro du code ressemblant à cela :

```
u=5
for i in range(11):
    u=u*3
    print('u_'+str(i)+' = '+str(u))
    i=i+1
```

Les difficultés ici sont :

- utilisation de variables
- utilisation d'une boucle (avec indentation)
- utilisation de liste (dont la première valeur est 0 et pas 1)
- typage des objets (types integer ou float, type string)

Donc ça fait plusieurs difficultés…

Il y a une discussion sur des difficultés que peuvent rencontrer les élèves ici : https://pydago.readthedocs.io/formation/enseignants2.html#difficultes

Dans la librairie pydiderotlibs, il y a une boucle sans variable (avec toutes les limitations que cela implique, mais qui peut rendre service dans certains cas, avec des élèves débutants) :

```
repeat(f,10)
```

(Ici on demande de répéter 10 fois la fonction f.)

4.2.6 Voir le fichier avec tous les programmes de 2de

4.2.7 Les programmes python de Physique Chimie par l'académie de Guyane

https://physique-chimie.dis.ac-guyane.fr/Banque-de-programme-python-pour-le-lycee.html