Plan Régional de Formation
Mme Larrouturou- M. Schavsinski

Initiation et enrichissement sur Python

I) Repl.it

Un moyen très simple de travailler Pythn avec les élèves est d'utiliser la plate-forme Repl.it.
C'est une plate-forme d'échanges capable d'interpréter de nombreux languages (Java, Javascript, C, C++, Go, Ruby, ...) dont Python 3 . Il suffit de vous créer un compte, d'envoyer une invitation à vos élèves qui s'inscrivent eux aussi puis de créer un nouveau devoir pour pouvoir corriger chaque élève en commentant son travail.

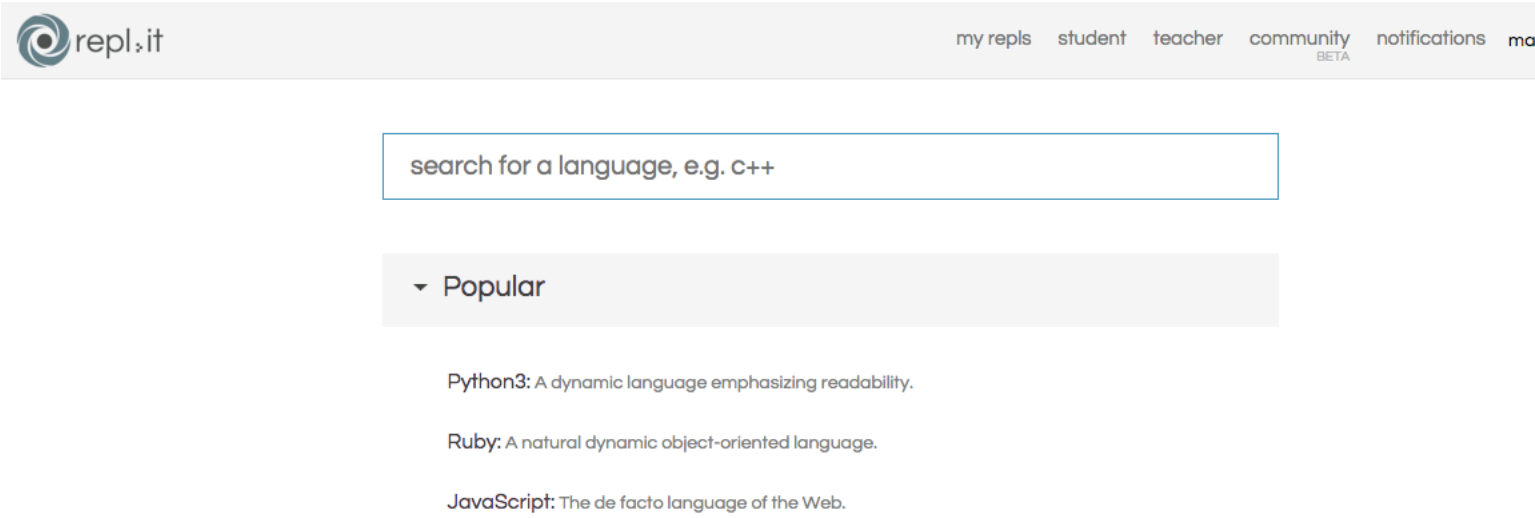
Vous avez surtout la possibilité de pouvoir faire tourner chaque programme soumis sans avoir à installer quoique ce soit sur l'ordinateur sur lequel vous consultez le travail.

C'est un outil vraiment très pratique qui contient même des modules d'autocorrection.

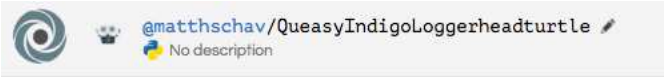
Il y a quelques limitations comme les bibliothèques pygame et tkinter qui ne peuvent être utilisées faute de fenêtre graphique.
Mais, pour le reste de l'algorithmique pure et dure, c'est vraiment une application très confortable malheureusement limitée à 30 élèves par prfo dans sa version graphique.
Vous avez bien sûr possibilité de l'utiliser pour vos propres programmes.

II) On lance Repl.it ?

Une fois votre compte créé, vous lancez un nouveau programme en allant sur MyRepls et New REpl.



Choisissez Python 3.0, donnez un nom à votre programme en cliquant sur le crayon en haut



et c'est parti !

Comme pour tout nouveau pas dans un language de programmation, votre première commande sera d'afficher Hello World !
Saisissez :

```
print("Hello world!")
```

et cliquez sur Run (le triangle classique).
Bravo !

III) Quelques éléments de syntaxe de Python

1) affectation

La base de la programmation est de pouvoir accéder à la mémoire de l'ordinateur, plus exactement à des cases mémoire qu'on appelle des **variables** et auxquelles on donnera des noms.
On va pouvoir donner des valeurs à ces variables, ce qu'on appelle une affectation.

Par exemple, nous programmons un jeu de voiture et on veut pouvoir contrôler la vitesse du véhicule à l'aide d'une touche.
On va affecter une variable qu'on peut appeler *vitesse*, et on change la valeur de la variable quand on appuie sur la touche.
Pour affecter dans Python, on utilise le =
Par exemple, on saisira vitesse=5 pour mettre la valeur 5 dans la variable *vitesse*.

2) affichage d'un message ou une variable

- Si on veut afficher un message, on doit saisir *print("ici vous tapez votre message")*
- Si on veut afficher le contenu d'une variable (par exemple *vitesse*), on doit saisir *print(vitesse)*
- Si on veut mélanger un affichage de message et du contenu de la variable(par exemple *vitesse*), on doit saisir : *print("vous roulez à ",vitesse)*

3) demander à l'utilisateur la valeur qu'il veut mettre dans une variable

L'instruction à saisir pour demander à l'utilisateur une valeur pour la variable *niveau* par exemple est :

niveau=eval(input("donnez le niveau auquel vous souhaitez jouer"))

4) les boucles

Les boucles sont des instructions à utiliser quand on veut que le programme effectue une série d'instructions plusieurs fois.

- Si on sait exactement combien de fois on veut les faire , on utilise la **boucle Pour**

  Par exemple, pour monter un escalier de 10 marches, on saisirait :
  Pour la variable *marche* allant de 1 à 10, on fait :
  — une montée du genou droit de 20 cm
  — on pose le pied
  — une montée du genou gauche de 20 cm
  — on pose le pied

  La syntaxe est la suivante :

  for i in range(N) :
  Bloc de toutes les instructions de la boucle à faire N fois avec le décalage devant

  Par exemple
  for i in range(10) :
  hauteurgenoudroit = hauteurgenoudroit +20
  on pose le pied droit
  hauteurgenougauche = hauteurgenougauche +20
  on pose le pied gauche
  print("ah, ça y est, je suis arrivé en haut de l'escalier")

- Si on veut que les instructions se fassent jusqu'à qu'il y ait un évènement , on utilise la **boucle Tant que**

  Par exemple, pour remplir une tasse à ras-bord, au centilitre près :

Tant qu'il reste de la place dans la variable *tasse* :
— tu mesures la place restante dans la variable *tasse*
— tu attrapes la cafetière
— tu verses un centilitre de café dans la *tasse*
— tu reposes la cafetière

La syntaxe est la suivante :

while *condition* :
Bloc de toutes les instructions de la boucle avec le décalage devant

5) les tests

Cette instruction consiste à ne déclencher une série d'instructions qu'à une certaine condition

Par exemple :

Dans notre jeu vidéo, si la vitesse dépasse 400,
— faire changer la couleur de la voiture
— attendre une seconde
— afficher une explosion à la place de la voiture

La syntaxe est la suivante :

if *condition* :
Bloc de toutes les instructions du test avec le décalage devant

IV) Nos premiers exercices en Python

Exercice 1 (Affectation ,entrée , sortie)

Ecrire un programme qui invite l'utilisateur à entrer un nombre entier, puis qui affiche le carré de ce nombre

Exercice 2 (Affectation ,entrée , sortie)

Ecrire un programme qui invite l'utilisateur à entrer deux notes l'une après l'autre, puis qui affiche la moyenne de ces deux notes

Exercice 3 (Affectation ,entrée , sortie)

Ecrire un programme qui invite l'utilisateur à entrer les coordonnées de 2 points , puis qui affiche la distance entre ces deux points.

Exercice 4 (Test)

Ecrire un programme qui invite l'utilisateur à entrer les coordonnées de 3 points , puis qui nous affiche si le triangle est isocèle ou équilatéral.

Exercice 5 (Test)

Ecrire un programme qui invite l'utilisateur à entrer trois nombres , puis qui nous les affiche dans l'ordre croissant.

Exercice 6 (Test)

Ecrire un programme qui invite lutilisateur à entrer les coordonnées de 3 points , puis qui nous affiche si le triangle est rectangle.

V) Notre premier jeu : le juste prix

Vous allez devoir créer un jeu qui donne 10 chances à l'utilisateur pour trouver le prix secret d'un objet entre 0 et 100 dollars (choisi au hasard).
La syntaxe pour choisir le prix-mystère au hasard est issu d'une bibiothèque , il faudra donc mettre l'instruction *import random* dès le début du programme.

Puis ensuite, au bon endroit dans votre programme :
prix-mystère = random.randint(0,100)

C'est parti sur Repl.it !

VI) Les divers types de données

Quand on travaille avec nos variables, Python ne sait pas à priori ce que c'est ... Est-ce une image ? Un son ? Un fichier ? Du texte ? Un nombre ?
Or il a besoin de savoir ce que c'est pour travailler avec. Il faut donc lui décrire le type de données :

- **int** désigne un entier
- **float** désigne les autres nombres.
- **true/false** désigne un booléen, c'est à dire une variable qui n'a que deux états (blanc/noir, homme/femme, ...)
- Mais il y a aussi les caractères (lettres, chiffres, symboles) et les chaînes de caractères. Dans Python, pour signifier un caractère ou une chaîne de caractères, on le met entre ”

VII) Fonctions sur les chaînes de caractères

Nous allons travailler avec ces chaînes aujourd'hui donc il nous faut connaître les fonctions qui nous permettent de les manipuler.

1) premières fonctions

Commande	Effet
ch=**input**(*quest*)	Ouvre une fenêtre contenant le texte *quest* et attend la saisie d'un texte qui sera stocké dans la variable *ch*.
len(*ch*)	Renvoie la longueur de la chaîne *ch*
ch[i]	Renvoie le caractère situé au rang i dans la chaîne. Attention ! la 1ère lettre est au rang 0

Exercice 1 :
Ecrire un programme qui demande une lettre et qui compte le nombre de fois où cette lettre apparaît dans la phrase " Belote, rebelote et dix de der !"

2) Extraire et transformer une chaîne de caractères

Commande	Effet
mots[*debut* :*fin*]	Renvoie la partie de la chaîne stockée dans *mot* comprise entre les caractères à la position *debut* (inclus) et celui a la position *fin* (exclu)
mot1+*mot2*	Colle les deux chaînes de caractères *mot1* et *mot2*. On dit qu'on concatène les deux chaînes.
mots.**upper()**	Renvoie la chaîne *mots* en majuscule.
mots.**lower()**	Renvoie la chaîne *mots* en minuscule.
str(*nb*)	Transforme le nombre *nb* en une chaîne de caractère
eval(*ch*)	Transforme la chaîne de caractère *ch* en un nombre

3) Rechercher et remplacer

Commande	Effet
l **in** *(ch)*	Renvoie vrai si la lettre *l* apparaît dans la chaîne *ch* et faux sinon
ch.**count**(*texte*)	Compte le nombre de fois où *texte* est présent dans la chaîne *ch*
ch.**replace**(*txt1*,*txt2*)	Remplace par *txt2* à chaque fois qu'il trouve *txt1* dans la chaîne *ch*
ch.**find**(*texte*)	Indique la position où se trouve *texte* dans la chaîne *ch*. Renvoie -1 si *texte* n'apparaît pas

Exercice 2 :

L'ordinateur demande un mot à l'utilisateur puis l'écrit à l'envers

Exercice 3 :

L'ordinateur réalise un cryptage : il met les lettres successives du mot clé "serpent" entre chaque lettre de la phrase saisie par l'utilisateur, tout en laissant les espaces.

VIII) Notre deuxième jeu : Le Motus - un projet plus long

L'ordinateur contient une liste de mots à 6 lettres dans une liste et en choisit un au hasard.

Il demande ensuite à l'utilisateur de trouver ce mot en un certain nombre de tentatives.

A chaque proposition de l'utilisateur, le programme reprend le mot saisit et met une lettre en majuscule si la lettre est à sa bonne place, une lettre en minuscule si elle apparaît dans le mot mais pas à sa place et une étoile à la place d'une lettre qui n'apparaît pas dans le mot.

Il affiche Bravo si on gagne et perdu sinon.