Activité : découverte de Python avec Repl.it

Repl.it est un site permettant de programmer en utilisant le language Python (entre autre) sans faire la moindre installation.

Dans ce site un programme est un « repl » et à chaque fois qu'on veut créer un nouveau programme, on commencera par retourner au tableau de bord en cliquant sur le logo du site, puis on cliquera sur le bouton + new repl situé en haut à droite de l'écran.

L'interface est divisée en trois colonnes, la première à gauche correspond à la gestion de fichiers annexes (on ne l'utilisera pas pour cette séquence) , la colonne centrale est la zone dans laquelle on écrira nos programmes (une séquence d'instructions) que l'on pourra exécuter/ lancer en cliquant sur la touche « run ». dans la troisième colonne nous avons la console, c'est un espace de travail, ou chaque commande sera exécutée immédiatement après qu'on ait appuyé sur « entrée ».

Opérations de base :
En plus de nos quatre opérations de base (+, - , /, *) on a :
** : qui sert à écrire les puissance par exemple : 5^3 s'écrira 5**3
// : qui donnera le quotient obtenu par division euclidienne , par exemple 17//3 donnera 5, c'est le quotient de la division euclidienne de 17 par 3
% : qui se lit « modulo » , donnera le reste dans le cadre d'une division euclidienne. Par exemple 17%3 donnera 2 c'est le reste de la division euclidienne de 17 par 3

Assignation :
Régulièrement en programmation, on voudra mémoriser des informations pour pouvoir les utiliser plus tard. On va donc dire à python de prendre un espace mémoire, de lui donner un nom et d'y ranger une valeur.
nombre1 = 5*3-4 indique à python qu'on veut ranger le résultat de 5*3-4 dans un espace mémoire que l'on appellera nombre1
nombre1 est ce que l'on appelle une variable, on y a stocké 11, mais on y mettre d'autres valeurs quand on voudra.

Tester une égalité
Si on tape nombre1 == 11 l'ordinateur va m'écrire True (si on a pas modifié la valeur de nombre1 depuis l'assignation) sinon il écrira False.
« == » sert à dire que l'on veut vérifier l'égalité
Dans le cadre d'un programme pour dire que l'on voudrait que le programme exécute une certaine séquence d'action si une condition on utilisera la commande if.
Elle sera suivie d'une condition et de « : »
Quand on appuiera sur « entrée » l'ordinateur provoquera un décalage quand on ira à la ligne, tous les éléments qui seront écrit de manière décalé ne seront exécutés que si la condition est réalisée.
Exemple :

```
2  if nombre1%3==0 :
3      print ("le nombre est divisible par 3")
4  else :
5      print("le nombre n'est pas divisible par")
```

Séance 2

Premier programme :
On veut faire un testeur de divisibilité, et pour ne pas avoir à taper le code qui suit à la main on pouvait utiliser le projet **diviseur séance 2** :

```
if 217%3==0 :
        print("217  est divisible par 3")
else :
        print("217 n'est pas divisible par 3")
```

```
nombre = 8189
diviseur = 17
if nombre%diviseur==0 :
        print(nombre," est divisible par ",diviseur)
else :
        print(nombre," n'est pas divisible par ",diviseur)
```

en fait c'est deux programmes en un seul.
La première version est super rigide, si on a envie de tester autre chose que la divisibilité de 217 par 3 , on doit changer le code en six endroits, ce qui est long… mais qui pourrait être largement pire dans un programme plus complexe.
La seconde version nous permet de gagner en souplesse, c'est un peu l'équivalent d'une formule en physique ou en maths, chaque quantité est désignée par un nom (ici c'est très explicite : « nombre » et « diviseur ») et pour tester la divisibilité d'autre chose que 8189 par 17 on n'aura que deux modifications à faire. Le reste du code s'appuie non pas sur les valeurs directes mais sur des variables qui vont les contenir.

Remarque : on peut voir ici que print peut s'utiliser un peu différemment que d'habitude, on peut utiliser cette fonction avec plusieurs arguments, ils seront séparés par des virgules, pour chaque argument on peut mettre du texte entre guillemets ou on peut mettre un nom de variable et là, print affichera non pas le nom de la variable mais ce qu'elle contient. Par exemple, executé tel quel le programme va activer la dernière ligne et écrire **8189 n'est pas divisible par 17** , il a remplacé automatiquement **nombre** par 8189 et **diviseur** par 17.

Pour rendre le programme encore plus souple, c'est-à-dire qu'il va s'adapter aux demandes de l'utilisateur, sans avoir à demander au programmeur de modifier encore une fois de plus son code.

```
nombre=int(input("quel est votre nombre ? "))
diviseur = int(input("quel est votre diviseur ? "))
if nombre%diviseur==0 :
        print(nombre," est divisible par ",diviseur)
else :
        print(nombre," n'est pas divisible par ",diviseur)
```

on n'a modifié que les deux premières lignes. Au lieu d'indiquer quels étaient les nombres à tester à l'intérieur du programme, on a modifié le code pour qu'à chaque exécution le programme demande à l'utilisateur de préciser les nombres qu'il a en tête.
La commande **input(« question »)** pose une question et récupère la réponse de l'utilisateur, mais celle-ci, en l'état n'est pas utilisable pour la suite du programme, en effet le programme ne voit qu'une chaine de caractères qui seront ici des chiffres, il ne comprends pas qu'on vient de lui offrir un nombre, c'est pour cela que la fonction **int(…)** va être utile. Elle convertit le « mot » (la chaine de caractère) répondue et la converti en entier (int est l'abréviation du mot integer, qui veut dire entier en anglais)

Nouveau programme :
A la recherche des diviseurs d'un nombre, détermination de la parité.

Version 1

```
nombre=int(input("quel est votre nombre ? "))
for diviseur in range(1,nombre+1) :
if nombre%diviseur==0 :
        print(nombre," est divisible par ",diviseur)
```

Explications
range(b) crée une liste de tous les entiers allant de 0 à b (exclu)
range(a,b) crée une liste de tous les entiers allant de a (inclus) et b (exclu)
ici diviseur va prendre toutes les valeurs allant de 1 jusqu'à nombre +1 exclu c'est-à-dire allant de 1 à **nombre**
Pas besoin de tester des diviseurs plus grand que le nombre lui-même, ça ne marchera jamais

Maintenant voyons voir en plus si **nombre** est premier, autrement dit s'il a exactement 2 diviseurs :

version 2

```
nombre=int(input("quel est votre nombre ? "))
compteur=0
for diviseur in range(1,nombre+1) :
    if nombre%diviseur==0 :
        print(nombre," est divisible par ",diviseur)
        compteur=compteur+1
print(nombre," a exactement  ",compteur," diviseurs  ")
if compteur==2 :
    print(nombre," est donc un nombre premier ")
```

ici on a utilisé une variable compteur qui partant de 0 a augmenté de 1 à chaque fois qu'on a trouvé un diviseur de **nombre** , et au cas où on n'en ait que 2 on affichera que le nombre étudié est premier.

Séance 3
Découverte de la fonction while
Connectez-vous à votre compte
Aller dans la classe virtuelle : Seconde Maths Kergot 2019-20
Sélectionner l'exercice (assignment) : nombre premier WHILE
Le faire
Sélectionner le projet Devinette

Correction de la séance : https://repl.it/@jkergot/devinette-correction

```
from random import random   #on importe la fonction random() de la librairie
random
nombreE= int(random()*100)+1 # on génère un nombre aléatoire entre 1 et 100
reponse=102 # dans réponse on stocke une réponse fausse
essai=0 # on n'a pas encore joué donc le nombre d'essai est nul
while(reponse!=nombreE): # tant que la réponse est fausse
  reponse=int(input("donne un nombre entre 1 et 100 ")) # on (re)pose la
question
  essai=essai+1 # on augmente le nombre de tentative d'une unité
  if(reponse>nombreE): #on guide le joueur
    print("trop grand")
  if(reponse<nombreE):
    print("trop petit")

print("felicitations tu as trouvé la valeur mystère ", nombreE," en ",essai,"
essais")
# on félicite le joueur
```