Chapitre 2 : Entrées sorties et listes

Audras

Lycée Saint-Just

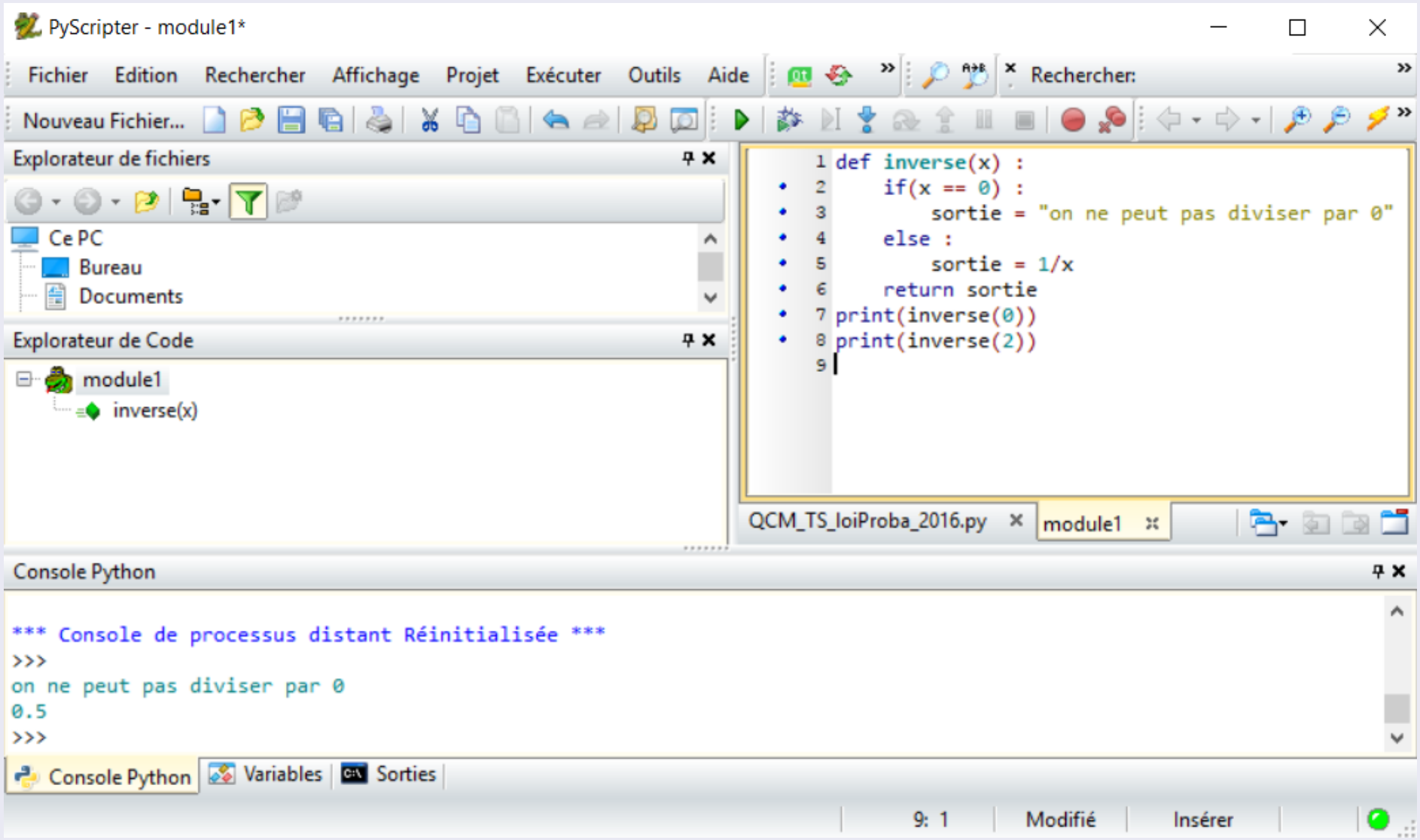
année 2018-2019 ISN

1) Editeur de texte

Définition

Un éditeur de texte permet d'écrire et de sauvegarder ses programmes afin de les exécuter sans avoir à les écrire ligne par ligne dans la console.
EduPython est un environnement de programmation gratuit et open source qui contient un éditeur de texte, une console, un explorateur de fichier et des aides diverses.
Dans l'éditeur on clique sur la page blanche nouveau fichier, on écrit la ligne # -*- coding : utf-8 -*- et on vérifie que l'éditeur est bien en codage du texte utf-8 dans edition format de fichier, on écrit le programme et on l'enregistre avec le suffixe '.py'.
On l'exécute avec la flêche verte Exécuter.
Si EduPython n'est pas disponible sur le réseau alors on peut le faire fonctionner à partir d'une clé USB.
`http://edupython.tuxfamily.org/`

Propriété


PyScripter - module1*
Fichier Edition Rechercher Affichage Projet Exécuter Outils Aide
Rechercher:
Nouveau Fichier...
Explorateur de fichiers
Ce PC
Bureau
Documents
Explorateur de Code
module1
inverse(x)
1 def inverse(x) :
2 if(x == 0) :
3 sortie = "on ne peut pas diviser par 0"
4 else :
5 sortie = 1/x
6 return sortie
7 print(inverse(0))
8 print(inverse(2))
9
QCM_TS_loiProba_2016.py
module1
Console Python
*** Console de processus distant Réinitialisée ***
>>>
on ne peut pas diviser par 0
0.5
>>>
Console Python
Variables
Sorties
9: 1
Modifié
Insérer

2) Les entrées / sorties

Définition

Afficher des données, imprimer des données, émettre des sons, montrer un graphique, une vidéo, faire une interface graphique, enregistrer un fichier, tout cela constitue les sorties.
L'interaction avec l'utilisateur, demander une valeur, cliquer sur un bouton, capter un son, une image, lire dans un fichier, tout cela constitue les entrées.

Propriété

- Ecrire dans la console :

```
>>> x=3
>>> print("x = ",x)
x =  3
```

- Entrer ou demander, dans un pop-up, un texte ou une valeur et l'affecter à une variable :

```
>>> monText = input("entrer un texte : ")
>>> monDecimal = float(input("entrer un nombre : "))
>>> monEntier = int(input("entrer un nombre entier : "))
```

input() utilise str() pour retourner un type str.

3) Les listes

Définition

Une liste est une collection ordonnée d'objets. On peut avoir une liste de nombres, de textes, de variables, de fonctions, de listes,...
Son type est list, on note une liste avec des crochets et le séparateur est la virgule :
liste = [2, 1, 0.5] ou liste=list([2, 1, 0.5])
Pour accéder à un élément on indique sa place dans la liste en démarrant à 0 : liste[0] retourne 2 ; liste[2] retourne 0.5
liste[:] retourne la liste ; liste[1:3] retourne [1,0.5]
On peut ajouter un élément en fin de liste avec liste.append(elt).
On peut enlever et obtenir le dernier élément avec liste.pop().
On peut concaténer deux listes, c'est à dire en faire une nouvelle à partir d'anciennes.
On obtient la longueur d'une liste avec la fonction len(liste).

Propriété

```
>>> liste = [2,1,0.5]
```

Pour ajouter un élément en fin de liste :

```
>>> liste.append(0.33)
>>> liste
[2, 1, 0.5, 0.33]
```

Pour enlever et obtenir le dernier élément :

```
>>> liste.pop()
0.33
>>> liste = [2,1,0.5]
```

On peut concaténer deux listes :

```
>>> ajout = [-0.5,-1,-2]
>>> nouvListe=liste + ajout
>>> nouvListe
[2, 1, 0.5, -0.5, -1, -2]
```

, parcourir une liste :

```
>>> liste=[4,5,6]
>>> for i in range(len(liste)):
...     print(str(liste[i])+"x7="+str(7*liste[i]),end=" ")
...
4x7=28 5x7=35 6x7=42
```

4) Type mutable et type immuable

Définition

On ne peut pas changer la valeur type immuable alors qu'on peut changer la valeur d'un type mutable.
Les types de base int, float, str sont immuables alors que le type list est mutable. Par exemple, quand on ajoute 0.33 à liste, on mute liste alors que si on ajoute 1 à 2 on obtient 3, 3 n'est pas une mutation de 2, c'est un autre nombre.
Un type immuable n'a qu'une seule étiquette alors qu'un type mutable peut en avoir plusieurs. On peut modifier une liste dans une fonction sans l'avoir déclarer globale car le paramètre de la fonction est une autre étiquette de la même liste. On peut de cette façon placer les variables du programme dans des listes au lieu d'utiliser une multitude de variables et d'utiliser global var1,var2,var3,....

Propriété

```
1  def neChangePas(a):
2      a=a+1
3
4  def changeList(l):
5      l[0]=l[0]+1
6
7  a=2
8  print(a,end='')
9  neChangePas(a)
10 print(" égale ",a)
11
12 liste=[a]
13 print(liste,end='')
14 changeList(liste)
15 print(" mute en ",liste)
```

```
2 égale  2
[2] mute en  [3]
>>>
```

5) Un autre type mutable : le dictionnaire

Définition

Un dictionnaire est de type dict, il associe une clé de type non mutable à une valeur de type quelconque ou à une fonction.

```
>>> mars={"x":120.2,"y":-56.2,"Vx":-23.5,"Vy":53}
>>> mars["x"]
120.2
>>> mars["Vy"]
53
>>> mars["x"]+=50
>>> mars["x"]
170.2
```

Cela permet de stocker des données comme avec une liste mais en plus explicite.

Remarque : $+=$ signifie 'ajouter', il y a de même $*=$ pour 'multiplier par' ainsi que $-=$ et $/=$.

6) Deux autre type immuable : booléen et n-uplet

Définition

Le type bool ne prend que deux valeurs : True et False, c'est la valeur d'une condition. Dans une condition un objet non vide est converti en True et s'il est vide ou 0 alors en False .
(1,2,3) est de type tuple, (1,) est un tuple à un élément, on peux parfois omettre les parenthèses, return a,b et return (a,b) sont identiques. Par contre f((a,b)) et f(a,b) ne sont pas identiques, dans le premier cas, f attends un argument et f en attends deux dans le second cas.

```
1 def f(): return 1,2
2 def g(): return (1,2)
3 print( type( f() ) is type( g() ) )
```

7) Exemple : construire une table de valeurs de la fonction inverse

```
1  def tabInv(a,b,pas):
2      """il faut 0<a<b ou a<b<0 et pas>0, retourne un tableau
3      de valeur de x--> 1/x sur [a,b]"""
4      x=a
5      absc=[]
6      ordo=[]
7      while (x <= b):
8          absc.append(x)
9          ordo.append(round(1/x,2))
10         x+=pas
11     return absc, ordo
12 absc, ordo = tabInv(-5,-0.5,0.5)
```

```
>>> absc
[-5, -4.5, -4.0, -3.5, -3.0, -2.5, -2.0, -1.5, -1.0, -0.5]
>>> ordo
[-0.2, -0.22, -0.25, -0.29, -0.33, -0.4, -0.5, -0.67, -1.0, -2.0]
```

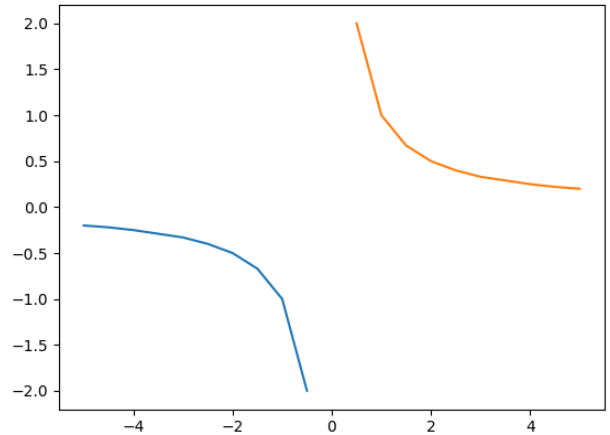
8) Exemple : construire une courbe représentative de la fonction inverse

Propriété

On peut utiliser les bibliothèques 'matplotlib.pyplot' et 'numpy' qui sont incluses dans EduPython. On est alors obligé de tracer les deux branches hyperboliques en deux fois.
On place les abscisses dans une liste comme 'absc' et les ordonnées dans une autre liste comme 'ordo'. On trace et on recommence.

Figure

```
>>> import numpy
>>> import matplotlib.pyplot as plt
>>> absc,ordo=tabInv(-5,-0.5,0.5)
>>> plt.plot(numpy.array(absc),numpy.array(ordo))
[<matplotlib.lines.Line2D object at 0x0149A750>]
>>> absc,ordo=tabInv(0.5,5,0.5)
>>> plt.plot(numpy.array(absc),numpy.array(ordo))
[<matplotlib.lines.Line2D object at 0x0149A710>]
>>> plt.show()
```



Remarque : plt est un pseudo pour matplotlib.pyplot