

Tableau récapitulatif des instructions en Python

En Python, pour faire appel à des modules supplémentaires avec des commandes particulières on doit l'indiquer avec des instructions **en début du programme** du type :

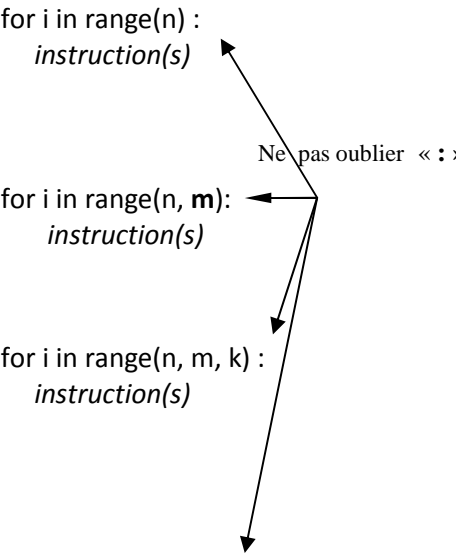
from math import* : pour des fonctions mathématiques telles que « pi » ou « cosinus »,...

from random import* : pour faire un calcul aléatoire

from tkinter import * : pour créer des interfaces graphiques

Si vous programmez sous EduPython, vous pouvez également appeler la bibliothèque lycée par l'appel **from lycee import ***, qui comprend toutes les bibliothèques citées plus haut.

Type d'instruction	Ecriture en Python	Exemples
Pour afficher un message à l'écran	Print("message")	Print("Bonjour !")
Entrer une valeur au clavier pour une variable X, mais précédée par un message * si X est une chaîne de caractères : * si X est un entier : * si X est un réel :	X=input("message") X=int(input("message")) X=float(input("message"))	X=int(input("Entrer un nombre entier : "))
Afficher le résultat d'une variable X à l'écran	print (X)	print (X)
Afficher le résultat d'une variable X à l'écran, mais précédée ou suivie de messages	print ("message n°1" , X , "message n°2")	print ("la longueur est de " , X , " cm")
Affecter dans l'algorithme une valeur à une variable A	A=valeur A=formule	A=5 le programmeur affecte 5 à A A=X+2 le programmeur affecte à A la valeur de X à laquelle on ajoute 2 A=A+1 le programmeur affecte à A la précédente valeur de A à laquelle on ajoute 1
Nombres au hasard : un réel $\in [0 ; 1[$ un réel $\in [a ; b[$ un entier $\in [a ; b]$	random() from random import* a+(b-a)*random() from random import* randint(a,b) from random import*	random() 3+5*random() donne un réel $\in [3 ; 8[$ randint(5,13) donne un entier $\in [5 ; 13]$
Opérations particulières : Division euclidienne de A par B Reste de la division euclidienne de A par B A^n \sqrt{A} partie entière de A	A//B A%B A**n sqrt(A) from math import* floor(A) from math import*	X//3 X%3 X**2 calcule le carré de X sqrt(X) calcule la racine carrée de X floor(pi) donne 3 : partie entière de Pi
Tests pour les conditions des boucles : A=B ? A≠B ? A≤B ? A≥B ? A<B<C ?	A==B A!=B A<=B A>=B A<B<C ou bien A<B and B<C	Effectue des instructions si : A==2 A est égal à 2 A!=2 A est différent de 2 A<=2 A est inférieur ou égal à 2 A>=2 A est supérieur ou égal à 2 2<A<5 A est compris entre 2 et 5 exclus
Instruction conditionnelle Si condition alors instructions ou bien Si condition alors instructions Sinon instructions	if conditions : instructions ou bien if conditions : instructions elif conditions : instructions else : instructions	Exemple : if X>0: print "le nombre X est positif" elif X<0 : print "le nombre X est négatif" else : print "le nombre 'X' est nul" ce programme permet de savoir si un nombre est positif , négatif ou nul.

<p>Boucle itérative</p> <p>Exécute en boucle la (ou les) instruction(s) n fois</p> <p>Exécute en boucle la (ou les) instruction(s) pour une variable allant de n à $m - 1$</p> <p>Exécute en boucle la (ou les) instruction(s) pour une variable allant de n à $m - 1$ avec un pas de k.</p> <p>Exécute en boucle la (ou les) instruction(s) pour chaque caractère de la chaîne de caractères ch</p>	 <pre> for i in range(n) : instruction(s) for i in range(n, m): instruction(s) for i in range(n, m, k) : instruction(s) for caractere in ch : instruction(s) </pre>	<p>Exemple :</p> <pre> S=0 for i in range(1;11) : S=S+i print "la somme S est égale à ",S </pre> <p><i>ce programme permet de calculer la somme des 10 premiers entiers</i></p>
<p>Boucle conditionnelle</p> <p>Tant que <i>condition</i> instructions</p>	<pre> while condition : instructions </pre> <p>Ne pas oublier « : »</p>	<pre> X= -1 while X<0 : X=input("saisir un nb positif") </pre> <p><i>Ce programme permet d'obliger à saisir un nombre positif</i></p>
<p>Fonction</p> <p>Une fonction peut avoir aucun, un ou plusieurs paramètres. Une fonction peut renvoyer un résultat (avec l'instruction return). Celui-ci est alors réutilisable dans un programme ou une autre fonction.</p>	<pre> def fonction(param1,param2, etc.) : instruction(s) return resultat </pre>	
<p>Le type Liste</p> <p>Affecter une liste vide à une variable</p> <p><i>Rappel : une liste est un tableau à une ligne et n colonnes n étant la longueur de la liste.</i></p> <p><i>Chaque élément est indexé de 0 à n-1</i></p> <p>Récupérer la longueur de la liste</p> <p>Supprimer un élément de la liste</p> <p>Concaténer deux listes</p> <p>Ajouter un élément à la fin de la liste</p> <p>Insérer au rang indiqué un élément</p> <p>Supprimer un élément de la liste</p> <p>Supprimer un élément à un rang donné</p> <p>Récupérer le rang (=l'index) de l'élément souhaité</p> <p>Trier une liste par ordre croissant ou alphabétique</p> <p>Inverser l'ordre des éléments d'une liste</p>	<pre> liste=[] l = len(liste) del(liste[élément]) liste1 + liste2 liste.append(élément) liste.insert(rang ,élément) liste.remove(élément) liste.pop(rang) liste.index(élément) liste.sort() attention chaque élément de la liste doit être de même type ! liste.reverse() </pre>	<p><i>A tester pour vous ré-entraîner aux listes</i></p> <pre> s=['lundi' , 'mardi' , 230 , 'dimanche'] print(len(s)) del(s[2]) s s.append('samedi') s s.insert(3,29) s s.remove('lundi') s s.pop(3) s s.index('mardi') s s.index('jeudi') s s.reverse() s s.sort() </pre>

Le type String

Une donnée de type **string** peut se définir comme une suite quelconque de caractères. On peut la délimiter par des simples quotes '...' ou par des doubles quotes « ... ».

Une chaîne de caractères peut alors être considérée comme une suite de caractères indexés de 0 à $n-1$, n étant la longueur de la chaîne.

- * Les fonctions **len** ou **+** (pour concaténer) sont les mêmes que pour les listes.
- * Les chaînes de caractères sont non modifiables (l'instruction `chaine[ij] = 'a'` est impossible)
- * On peut comparer des chaînes de caractères.

Fonctions qui peuvent également vous servir : (notons ch notre chaîne de caractère)

`ch.find(sch)` : cherche la position d'une sous-chaîne sch dans la chaîne

`ch.count(sch)` : compte le nombre de sous-chaînes sch dans la chaîne

`ch.lower()` : convertit une chaîne en minuscules

`ch.upper()` : convertit une chaîne en majuscules

`float(ch)` convertit la chaîne ch en un nombre réel (*float*) (pour fonctionner il faut que la chaîne représente bien un tel nombre)

`int(ch)` convertit la chaîne ch en un nombre entier (idem)

`ord(ch[caractere])` : permet de savoir la valeur ASCII d'un caractère de la chaîne.

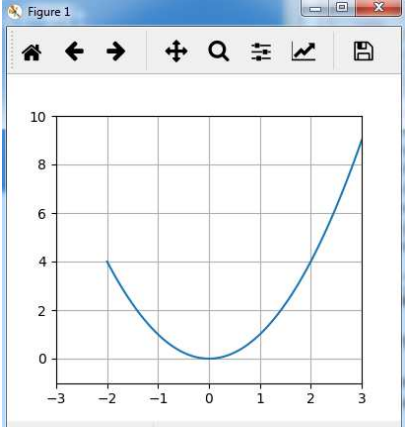
Représentation graphique d'une fonction

Pour tracer la courbe représentative d'une fonction, on fera l'appel de 2 bibliothèques :

- La bibliothèque **numpy** que l'on peut renommer avec l'alias np – qui permet de faire du calcul scientifique
- La bibliothèque **matplotlib.pyplot** que l'on peut renommer avec l'alias plt – qui permet de tracer les courbes

Si vous programmez **avec Edu Python**, la bibliothèque numpy est intégrée au module lycée proposé donc vous pouvez remplacer l'appel classique « `import numpy as np` » par « `from lycee import *` ».

Toutefois ce n'est pas le cas du module matplotlib. Vous devrez en faire l'appel ! (voir exemple ci-dessous)

Définir les dimensions du repère : Axe des abscisses de val1 à val2 Axe des ordonnées de val3 à val4	<code>plt.axis([val1,val2,val3,val4])</code>	<p>Exemple :</p> <pre>from lycee import * import matplotlib.pyplot as plt x=np.linspace(-2,3,100) y=x**2 plt.axis([-3,3,-1,10]) plt.plot(x,y) plt.grid() plt.show()</pre> 
Affiche un quadrillage	<code>plt.grid()</code>	
Définit l'intervalle [val1 ; val2] dans lequel x varie et le nombre de points calculés par tracer la courbe	<code>np.linspace(val1, val2, nb_points)</code>	
Trace les points de coordonnées x et y	<code>plt.plot(x,y)</code>	
Permet l'affichage de la courbe	<code>plt.show()</code>	