Le langage Python, synthèse, partie 1

La variable

Une variable a un nom et est destinée à contenir une information.
Une variable est associée à un type

La déclaration d'une variable

Une variable est déclarée lorsqu'on précise son nom et on lui donne une valeur (on affecte cette variable de la valeur). Le type sera automatiquement déduit de la valeur.

Le type

On considérera trois catégories de types simples : entier, réel et booléen

L'affectation

Affecter une variable signifie lui donner une valeur.
En python le symbole d'affectation est « = ».
Une affectation va toujours de la droite vers la gauche, on met dans la variable à gauche la valeur qui est à droite.

Exemples de déclarations de variables

En python une déclaration de variable est : son nom suivi de « = » suivi de la valeur

Exemples de déclarations de variables	Commentaires
a=1	On déclare une variable nommée « a » qui reçoit la valeur 1 → le type de a est déduit comme un entier
b=1.0	La variable nommée « b » est affectée de 1.0 (type réel)
c=True	La variable nommée « c » est affectée de True (type booléen)
somme=0	La variable « somme » a un nom évocateur

Afficher du texte

Pour afficher du texte il suffit de mettre la chaîne de caractère entre " " ou entre ' ' et de le précéder du mot clé « print ».

Exemples d'affichage de texte	Commentaires
print "bonjour "	On demande d'afficher la chaîne "bonjour " à l'écran
print 'bonjour '	Effet identique à la ligne précédente
print "bonjour l'inraci "	Une chaîne avec un ' sera plutôt entourée de " "

Affichage mixte (du texte et des variables)

Le texte toujours entre " " ou entre ' ' mais pas pour les variables.
Attention les variables doivent avoir été préalablement déclarées.
La virgule sépare les éléments à afficher.

Exemple d'affichage mixte	Commentaires
cote_max=10	Déclaration d'un entier « cote_max » initialisé à 10
cote=9	Déclaration d'un entier « cote » initialisé à 9
print "Laurent a eu ", cote , "/ ", cote_max	Affichage mixte (« *Laurent a eu 9 / 10* »)

Afficher le type d'une variable

La fonction type() est une fonction préexistante qui renvoie une chaîne de caractère indiquant le type de son argument (ce qu'il y a entre les parenthèses).
Pour afficher le type d'une variable il faut d'abord avoir déclaré cette variable.

Exemple d'affichage du type d'une variable	Commentaires
cote_max=10	Déclaration de l'entier « cote_max »
print type(cote_max)	Affichage du type de « cote_max »

Utilisation d'un délai (exemple manipulant une fonction de « time »)

Pour utiliser des utilitaires qui permettent d'attendre un délai, on écrira au début du programme « import time ».

Exemple d'utilisation d'un délai	Commentaires
import time	On écrit « import time » au début
print "salut"	On affiche « salut »
time.sleep(1)	On attend une seconde
print "tout le monde"	Lorsque la seconde est passée on affiche « tout le monde »

Incrémentation et décrémentation d'une variable

Incrémenter une variable signifie « ajouter 1 à cette variable »
Décrémenter une variable signifie « enlever 1 à cette variable »

Exemple d'incrémentation et de décrémentation	Commentaires
nb1 = 8	Déclaration de nb1
nb2= 5	Déclaration de nb2
nb1= nb1 + 1	Incrémentation de nb1
nb2 = nb2 - 1	Décrémentation de nb2

Incrémentation et décrémentation d'un pas différent de 1

Lorsqu'on ne précise pas le pas, par défaut il s'agit de la valeur 1.
On peut cependant incrémenter (ou décrémenter) une variable un spécifiant un pas.

Exemple d'incrémentation et de décrémentation	Commentaires
nb1 = 8	Déclaration de nb1
nb2= 5	Déclaration de nb2
nb1= nb1 + 10	Incrémentation de nb1 par un pas de 10
nb2 = nb2 - 2	Décrémentation de nb2 par un pas de 2

Utilisation de commentaires

Un commentaire est une explication que le programmeur ajoute qui ne sera pas prise en compte par le compilateur.
Sur une ligne, un commentaire est du texte placé après le caractère « # ».

Exemples de commentaires	Commentaires
#mini programme	Les commentaires rendent le code plus lisible.
nb1 = 8 #déclaration de nb1	On peut placer un commentaire sur une ligne seule ou
nb1= nb1 + 1 #incrémentation de nb1	placer un commentaire pour détailler une ligne

Les instructions Lire et Ecrire (en général les fonctions « input » et « print)

L'instruction Ecrire signifie que le programme renvoie à l'utilisateur une information. Cette information est en général le contenu d'une variable. Le plus souvent l'instruction Ecrire se fera par l'intermédiaire de la fonction « print ».
L'instruction Lire signifie que le programme, pendant son exécution, doit faire l'acquisition d'une donnée extérieure. En général la donnée extérieure sera fournie par l'utilisateur par l'intermédiaire du clavier, on utilisera la fonction « input ».

Exemple de lecture et d'écriture	Commentaires
#somme de deux nombres	Commente l'énoncé de l'exercice
nb1 = 0 #déclaration de nb1	
nb2 = 0 #déclaration de nb2	
somme = 0 #déclaration de somme	
nb1= input("introduisez nb1 SVP: ") #Lire nb1	L'utilisateur voit le texte "introduisez nb1 SVP: " et le programme attend sa donnée pour l'affecter à nb1
nb2= input("introduisez nb2 SVP: ") #Lire nb2	L'utilisateur voit le texte "introduisez nb2 SVP: " et le programme attend sa donnée pour l'affecter à nb2
somme = nb1+nb2	La variable somme est affectée du résultat de l'opération
print "la somme de ",nb1, " et ",nb2, " = ",somme	L'affichage renvoie le résultat à l'utilisateur (Ecrire)

Les opérations arithmétiques

Les opérations sont formées à partir d'opérateurs. Les opérateurs arithmétiques sont :

+, -, *, / et permettent de constituer des opérations comme par exemple : (nb1+nb2), (nb1-1), (10*nb2), (nb1/nb2),…

Les opérations sont constituées d'opérateurs mais aussi de variables et/ou constantes.
Le résultat de l'opération peut servir à une affectation (exemple : somme =nb1+nb2).
La priorité des opérateurs est la priorité habituelle. On peut aussi ajouter des ( ).

Exemple d'utilisation d'opérations	Commentaires
#tests sur les opérateurs arithmétiques	Commente l'énoncé de l'exercice
nb1 = 0	
nb2 = 0	
somme = 0	
difference=0	
produit =0	
quotient=0	
nb1= input("introduisez nb1 SVP: ") #Lire nb1	Lire
nb2= input("introduisez nb2 SVP: ") #Lire nb2	Lire
somme = nb1+nb2	Affectation d'une opération
difference = nb1-nb2	Affectation d'une opération
produit = nb1*nb2	Affectation d'une opération
quotient = nb1/nb2	Affectation du quotient (entier)
print "les nombres sont ", nb1, " et ", nb2	Réaffichage des nombres lus
print "somme : ", somme, " et difference : ", difference	Ecrire somme et différence
print "produit : ", produit, " et quotient : ", quotient	Ecrire produit et quotient

Exemple avec décompte

On peut faire un décompte en utilisant des délais et une décrémentation.
Ici notre exemple fonctionne mais devra être amélioré à l'aide d'une structure répétitive.

Exemple de décompte sans structure répétitive	Commentaires
import time	
nb_sec_rest=3	On répète trois fois :
print "fini dans ", nb_sec_rest, " s"	
time.sleep(1)	print "fini dans ", nb_sec_rest, " s"
nb_sec_rest= nb_sec_rest – 1	time.sleep(1)
print "fini dans ", nb_sec_rest, " s"	nb_sec_rest= nb_sec_rest – 1
time.sleep(1)	
nb_sec_rest= nb_sec_rest - 1	
print "fini dans ", nb_sec_rest, " s"	
time.sleep(1)	
nb_sec_rest= nb_sec_rest - 1	
print "FINI"	

La structure de choix simple

Lors de l'exécution du programme, il est possible d'examiner une condition afin d'exécuter un bloc d'actions que si cette condition est vraie.
Nous dirons qu'un bloc d'actions est simplement une suite successive de lignes.
La condition à examiner se trouve entre le mot clé « if » et « : », le bloc d'actions à exécuter est indenté (c'est-à-dire décalé vers la droite).

Exemple d'utilisation de structure de choix	Commentaires
a = 55	
b = input("un entier SVP:")	Lire b
if b>a:	Pendant l'exécution, la condition (b>a) est examinée
print "1ere ligne du bloc d'actions"	← Ce n'est que lorsque la condition est vraie
print b," est plus grand que ",a	← que le bloc d'actions (ici 3 lignes) est exécuté
print "derniere ligne du bloc d'actions"	
print "sans indentation, plus dans le bloc d'actions "	Ligne toujours affichée (peu importe la condition)

La structure de choix simple – Définir un bloc d'actions en cas de condition fausse

Nous avons vu que la structure de choix permet de préciser (en indentant) les lignes d'actions à exécuter lorsque la condition est examinée comme vraie au moment de l'exécution. Pour rappel la condition à examiner se trouve entre le « if » et le « : ».
Cependant il est également possible de définir un autre bloc d'actions à exécuter lorsque la condition est examinée comme fausse, pour ce faire on utilise le mot clé « else ».
Après le « else » on met « : » pour montrer que va commencer le bloc d'actions lié à une condition fausse ; toutes les lignes du bloc d'actions doivent être indentées.

Exemple d'utilisation du « if » et du « else »	Commentaires

a = 55
b = input("un entier SVP:")
if b>a:
  print "1ere ligne du bloc d'actions du if"
  print b," est plus grand que ",a
  print "derniere ligne du bloc d'actions du if "
else:
  print "1ere ligne du bloc d'actions du else"
  print b," est plus petit ou égal à ",a
  print "derniere ligne du bloc d'actions du else"
print "sans indentation, plus dans le else "
	Lire b
	Pendant l'exécution, la condition (b>a) est examinée
	
	← Ce n'est que lorsque la condition est vraie
	← que ce bloc d'actions (ici 3 lignes) est exécuté
	Attention plus de condition après le mot else
	
	← Ce n'est que lorsque la condition est fausse
	← que ce bloc d'actions (ici 3 lignes) est exécuté
	Ligne toujours affichée (peu importe la condition)

Constituer une condition

Une condition est ce qui devra être examiné lors de l'exécution du programme afin de déterminer si elle est vraie ou fausse. Après examen, une condition est donc un booléen (vrai ou faux). Souvent une condition apparaît comme une opération dont le résultat est un booléen (vrai ou faux) ; Les opérateurs de comparaison le montrent bien.
Les opérateurs de comparaison sont :

< , <= , > , >= , == , !=

Exemples de conditions utilisant des opérateurs de comparaison :

(nb1<nb2), (nb1<=1), (10>nb2), (nb1>=nb2), (nb1==nb2), (nb1 !=nb2), …

Les conditions ci-dessus sont constituées d'opérateurs de comparaison mais aussi de variables et/ou de constantes. Les parenthèses ne sont en général pas nécessaires.
L'opérateur de comparaison « == » signifie «est la même chose que », à ne pas confondre avec l'opérateur d'affectation « = ».
L'opérateur de comparaison « != » signifie « est différent de ».

Utilisation d'une condition au sein d'une structure de choix

Comme montré un peu plus tôt, il est apparu le besoin d'utiliser une condition avec la structure de choix. Nous verrons plus loin qu'il existe d'autres structures qui utilisent aussi une condition (comme la structure répétitive « tant que » par exemple).

Exemples de conditions au sein de structures de choix	Commentaires

cote = input("votre cote sur 10 SVP:")
if cote==10:
  print "bravo !"
  print "vous avez la cote max !"
print "je suis après la structure "
	Lire cote
	Si cote est la même chose que 10
	Alors afficher "bravo !"
	  afficher "vous avez la cote max !"
	afficher "je suis après la structure "

cote = input("votre cote sur 10 SVP:")
if cote !=10:
  print "snif !"
  print "vous n'avez pas la cote max !"
else :
  print "vous avez la cote max !"
print "je suis après la structure "
	Lire cote
	Si cote est différent de 10
	Alors afficher "snif !"
	  afficher "vous n'avez pas la cote max !"
	sinon
	  afficher "vous avez la cote max !"
	afficher "je suis après la structure "

Les opérateurs logiques

Les opérateurs logiques permettent de constituer des conditions plus complexes.
Les opérateurs logiques sont : and (ET) or (OU) not(NON)
Les opérateurs logiques permettent de constituer une condition générale à partir d'autres conditions. Par exemple la condition (a<=b and a<=c) est constituée de :
la condition a<=b , l'opérateur logique « and » , la condition a<=c
L'opérateur logique « and » constitue une condition qui sera vraie si et seulement si les deux autres conditions qui s'y rapportent sont vraies toutes les deux.
L'opérateur logique « or» constitue une condition qui sera vraie dès qu'une des deux autres conditions qui s'y rapportent est vraie (ou même le deux).
L'opérateur logique « not » ne se rapporte qu'à une seule condition.
Si cette dernière est « true », « not true » renvoie false, sinon « not false » renvoie true.
Les conditions constituées à l'aide d'opérateurs logiques sont en général utilisées au sein d'une structure (comme la structure de choix, la structure répétitive tant que,…).

Exemples d'utilisations d'opérateurs logiques	Commentaires

a = input("veuillez introduire le nombre a :")
b = input("veuillez introduire le nombre b :")
c = input("veuillez introduire le nombre c :")
if a<=b and a<=c:
  print a, " est le plus petit"
else:
  print a, " n'est pas le plus petit"
	Lire a
	Lire b
	Lire c
	Si a<=b et que a<=c:
	Alors afficher a, " est le plus petit"
	Sinon
	  afficher a, " n'est pas le plus petit"

cote = input("votre cote sur 10 SVP:")
if not cote ==10:
  print "snif !"
  print "vous n'avez pas la cote max !"
else :
  print "vous avez la cote max !"
print "je suis après la structure "
	Lire cote
	Si on n'a pas cote qui est la même chose que 10
	Alors afficher "snif !"
	  afficher "vous n'avez pas la cote max !"
	sinon
	  afficher "vous avez la cote max !"
	afficher "je suis après la structure "

Les structures imbriquées

Nous avons vu que la structure de choix examine une condition et exécute ou non un bloc d'actions lorsque la condition est vraie. Appelons « bloc_V » ce bloc d'actions.
Nous avons vu également qu'avec le mot clé « else » il est possible d'exécuter un autre bloc d'actions lorsque la condition est fausse. Appelons « bloc_F » ce bloc d'actions.
Nous pouvons alors résumer le principe de la structure de choix (avec un else) comme :

Descriptif de la structure de choix (avec un else)	Commentaires

if condition :
  bloc_V
else:
  bloc_F
print "je suis après la structure "
	Si condition est vraie Alors
	  bloc_V
	Sinon
	  bloc_F
	afficher "je suis après la structure "

Ci-dessus « condition » représente la condition qui sera examinée et qui correspondra à vrai ou faux. Les deux blocs d'actions bloc_V et bloc_F représentent une suite successive de lignes. Il peut s'agir de 2 ,10 ou 1000 lignes. Un bloc d'actions peut même contenir lui-même une ou plusieurs autres structures. Il faut juste s'assurer que tout bloc d'actions contienne un nombre entier de structures (et pas de structures non achevées).
Lorsqu'on a une structure dans une structure on appelle ça des structures imbriquées.

Descriptif de la structure de choix (avec un else)	Commentaires

if a>=0 :
  if a==0 :
  print "a=0"
  else :
  print "a>0"
else :
  print "a<0"
	Si a>=0 alors
	  Si a est la même chose que 0 Alors
	  afficher "a=0
	  Sinon
	  afficher "a>0"
	Sinon
	  afficher "a<0"

Un if dans un else, une notation abrégée.

Le bloc d'actions associé à un « else » est un ensemble de lignes successives.
Si ce bloc d'actions doit commencer par une structure de choix, au lieu d'écrire « else : if … » on peut utiliser une notation abrégée notée « elif… ».

Un if dans un else, notation longue	Un if dans un else, notation abrégée

a = input("un entier SVP :")
if a>0 :
  print "positif"
else :
  if a==0 :
  print "nul"
  else :
  print "négatif"
	a = input("un entier SVP :")
	if a>0 :
	  print "positif"
	elif a==0 :
	  print "nul"
	else :
	  print "négatif"

Les deux exemples ci-dessus sont similaires (fonctionnement identique) cependant la version à droite est plus concise et n'exige pas deux niveaux d'indentation.

La structure répétitive « tant que »

Comme pour la structure de choix, la structure « tant que » fonctionne avec une condition. La « tant que » est associée à un bloc d'actions qui sera exécuté tant que la condition est vraie. Il s'agit d'une structure répétitive car le bloc d'actions associé à la « tant que » est susceptible d'être répété si la condition continue à être vraie. Contrairement à la structure de choix la condition peut être examinée plusieurs fois.
A la première exécution de la « tant que » la condition est examinée, si elle est fausse on sort de la structure et on passe à la suite (cas particulier où le bloc d'actions n'aura jamais été exécuté), si elle est vraie on effectue une fois le bloc d'actions et on réexamine la condition ; si au deuxième examen de la condition celle-ci est toujours vraie, on exécute pour la deuxième fois le bloc d'actions, on réexamine la condition et ainsi de suite. Tant que la condition sera vraie on va répéter le bloc d'actions, on ne sortira de la structure « tant que » que lorsque la condition sera examinée comme fausse (ou qu'une instruction break a eu lieu, voir plus tard).

Descriptif de la structure répétitive « tant que »	Commentaires

while condition :
  bloc
print "je suis après la structure "
	Tant que condition est vraie, répéter
	  bloc
	afficher "je suis après la structure "

Exemple d'utilisation de la "tant que"	**Commentaires**

#Attente de Lire la valeur 5
nb = input("Veuillez introduire un nombre entre 1 et 10")
while nb !=5 :
  print nb, "n'est pas le nombre attendu, recommencez"
  nb = input("Veuillez introduire un nombre entre 1 et 10")
print "après la structure la condition est fausse"
print "votre nombre (",nb, ")vaut bien la valeur 5"
	
	Lire nb
	tant qu'on n'a pas nb qui vaut 5 répéter
	  afficher "…, recommencez"
	  Lire nb
	afficher "… la condition est fausse"
	afficher "… vaut bien la valeur 5"

La « tant que » pour répéter un bloc d'actions un nombre de fois donné

Il est possible d'utiliser la « tant que » pour répéter un bloc d'actions un nombre de fois donné. Nous avions vu un exemple de décompte (page 3) qui montrait clairement que l'on répétait trois fois un bloc d'actions. Il vaut mieux dans ce cas utiliser une structure répétitive. L'exemple ci-dessous utilise la variable de décompte au sein de la condition.

Exemple de décompte avec structure répétitive	Commentaires

import time
nb_sec_rest=3
while nb_sec_rest>0:
  print "fini dans ", nb_sec_rest, " s"
  time.sleep(1)
  nb_sec_rest= nb_sec_rest – 1
print "FINI"
	
	On répète trois fois :
	
	print "fini dans ", nb_sec_rest, " s"
	time.sleep(1)
	nb_sec_rest= nb_sec_rest – 1

Une idée intéressante lorsqu'on veut répéter un bloc d'actions un nombre connu de fois est d'utiliser une variable dont le rôle est de compter le nombre de fois que le bloc d'actions a déjà été répété. On peut alors demander que « tant qu'on n'a pas atteint le nombre de répétitions souhaitées on répète le bloc d'actions », on utilise alors une structure répétitive « tant que » qui va utiliser une condition créée à partir de notre variable « compteur » et du nombre de répétitions souhaitées.
La condition sera par exemple (nb_repet_actu < nb_repet_a_faire) où :

nb_repet_actu	est la variable qui indique le nombre de répétitions qui a déjà eu lieu actuellement
nb_repet_a_faire	est une variable ou une constante qui indique le nombre total de répétitions que l'on souhaite faire.

Pour que cette façon de faire fonctionne, il faut s'assurer que la variable *nb_repet_actu* soit initialisée à 0 et qu'elle soit incrémentée à chaque exécution du bloc d'actions.

Exemples de répétitions connaissant le nombre d'itérations	Commentaires

#Afficher bonjour 10 fois de suite
nb_repet_actu =0
while nb_repet_actu < 10 :
  print "bonjour !"
  nb_repet_actu = nb_repet_actu + 1
	On initialise nb_repet_actu à 0
	Tant qu'on n'a pas atteint 10 répétitions
	  Afficher "bonjour !"
	  « mettre ajour la variable nb_repet_actu »

#Afficher bonjour n fois de suite (n introduit par l'utilisateur)
nb_repet_a_faire = input("Combien de répétitions SVP ? ")
nb_repet_actu =0
while nb_repet_actu < nb_repet_a_faire :
  print "bonjour !"
  nb_repet_actu = nb_repet_actu + 1
	Lire *nb_repet_a_faire*
	On initialise nb_repet_actu à 0
	Tant qu'on n'a pas atteint *nb_repet_a_faire*
	  Afficher "bonjour !"
	  « mettre ajour la variable nb_repet_actu »

Cependant il existe une structure répétitive plus adaptée que la « tant que » lorsqu'on connaît le nombre de répétitions, il s'agit de la boucle FOR. Cette boucle utilise d'ailleurs le principe d'une variable « compteur » initialisée au début, mise à jour à chaque tour de boucle et testée régulièrement pour assurer le nombre de répétitions souhaitées.

Utilisation de l'instruction « break »

L'instruction break permet de sortir de la structure englobante la plus proche qui ne soit pas une structure de choix. On peut très bien avoir une boucle « tant que » classique où tant que la condition est vraie on répète le bloc d'actions et dés que la condition devient fausse on sort de la boucle. Cependant on peut prévoir une façon de sortir de la boucle anticipativement grâce à l'instruction break.

Exemple d'utilisation du break au sein d'une tant que	Commentaires

#Somme de 5 chiffres introduits par l'utilisateur
print "un nombre n'est pas toujours un chiffre"
chiffre=0
somme=0
nb_repet_actu =0
while nb_repet_actu < 5 :
  chiffre = input("un nombre positif à un seul chiffre SVP : ")
  if chiffre>=0 and chiffre>9 :
  print "vous avez introduits plusieurs chiffres"
  break
  somme=somme + chiffre
  nb_repet_actu = nb_repet_actu + 1
print "je suis après la structure "
if nb_repet_actu == 5 :
  print "la somme des 5chiffres vaut : ", somme
	L'utilisateur doit introduire 5 fois de suite un nombre à un seul chiffre.
	On utilise alors une structure tant que et la gestion de la variable *nb_repet_actu* pour assurer 5 répétitions.
	Cependant l'exécution de la boucle peut s'interrompre anticipativement si l'utilisateur introduit un nombre à plusieurs chiffres.
	
	Une fois que nous sommes sortis de la boucle, la variable *nb_repet_actu* permet de déterminer si nous en sommes sortis anticipativement ou non.