Python Cours Basique

Les types de valeurs :

-int

-float

-string (on peut utiliser les triples guillemets pour éviter de faire \')

-bool

Création et affectation de variable :

Variable=valeur #affectation d'une nouvelle variable

Variable2=Variable #affectation d'une nouvelle variable par copie

Affichage avec print:

print(« texte »,variable,« texte »)

Structures conditionnelles :

C'est l'indentation qui fait les structures pas de fermeture de fonctions par {

SI

If condition :

Action

#finSi

Else :

Action

#finSinon

Avec Prédicats

and (&&) /or(||) /not(!=)/is(==)

Les boucles :

TANTQUE

While condition :

Action

#FINTANTQUE

POUR

For elem in sequence : *#de base elem à 1*

Action

#FINPOUR

Les fonctions:

Def nomFonction(paramètres):

Actions

#finFonction

Il peut être utile de récupérer un certain nombre de paramètres que l'on ne connait pas à l'avance.

*Def fonction(*paramètres)*

Cela va récupérer tous les paramètres entrés qui peuvent être de n'importe quel type et les mettre dans une liste.

Importations de modules:

Import module #pour un module préexistant style math en c

From nomPackage.nomFichier import nomFonction #pour une fonction spéciale

*From nomPackage.nomFichier import ** #pour tout le fichier nomFichier

Gestion des exceptions :

Try :

Ce qui est à tester

Except :

Action à faire dans ce cas

Finally :

Sera fait dans tous les cas

#FIN

Lever une exception

raise Exception(« texte »)

Travailler sur les chaines de caractères :

Print(« {},{} ».format(var1,var2)) *#formater une chaine par ordre des variables par défaut*

Print(« {n},{m} ».format(m=2,n=3)) *#formater par ordre donné*

Concaténation :

Ch= « chaine »

Ch+ « ajout de texte en concaténation »

Ch+str(9)+ «ajout de texte après avoir ajouter un nombre pour ce fait il a fallu le cast »

Longueur d'une chaine :

Len(Ch)

Parcours d'une chaine :

Ch[1] *#à partir du début donc ici deuxième case car commence à 0*

Ch[-1] *#à partir de la fin donc avant-dernière case*

La sélection de chaine pour modifier :

Il n'est pas possible de modifier directement une chaine il faut sélectionner ce qui est à modifier

Mot= « salut »

Mot[0 :2]

Mot[:2]

Mot[2 :]

Mot= « b »+Mot[2 :] *#ceci renvoie blut*

Eclater des chaines :

Création de liste à partir d'une chaine

Chaine.split(condition d'éclatement)

Chaine= « salut toi »

Chaine.split(« »)

On a, une liste composée de [« Salut », « toi »]

Souder une chaine à partir d'une liste :

« élément qui soude » .join(liste) OU « élément qui soude » .join([elem,elem2])

Liste=[« Salut », « toi »]

« ».join(liste)

On a « Salut toi » car on a souder avec un espace

Les listes :

Peut contenir n'importe quel type

Création d'une liste :

Nom=list()

Nom=[1,3, «chat »]

Accéder

Nom[indice]

Modifier

Nom[indice]=valeur

Insérer des valeurs :

A la fin de la liste : *nomListe.append(valeur)*

A un indice précis : *nomListe.insert(indice,valeur)*

Concaténation de deux listes :

NomListe.extend(nomListe2) *#modifie NomListe*

NomListe+nomListe2 *#ne modifie pas*

Suppression d'éléments :

Avec un indice : *Del.nom[indice]*

Avec une valeur : *nomListe.remove(valeur)*

Ne retire que la première occurrence de la valeur dans la liste

Parcourir une liste :

```
For elem in enumerate(nomListe):          #ce qui affiche (indice,valeur)
    Print(elem)
#Fin
```

En plus lisible:

```
For i,elem in enumerate(nomListe):
    Print("indice{},valeur{}".format(i,elem))
#fin
```

Compréhension de liste:

Opérations pouvant être effectuées sur les listes pour les trier et afficher les valeurs voules.

New=[opération for elem in liste if condition]

Tuples

Un tuple n'est plus modifiable après déclaration

Déclaration :

nomTuple(val,val2)

nomTuple(val,) *#si on a besoin d'une seule valeur mais peu d'intérêt*

Cela peut aussi être utile avec des fonctions

```
Def fonction(a,b) :
    Action
    Return a,b
#FINFONCTION
```

Récupérer la valeur du tuple de la fonction :

(c,d)=fonction(a,b) *#c=a et d=b*

c=fonction(a,b) *#c=(a,b)* *c contient un tuple*

Les dictionnaires :

C'est un conteneur qui n'a pas d'ordre, on retrouve les éléments à l'aide de mot-clé et non d'indice

Création :

nomDico=dict() OU nom={« clé » :valeur, « clé2 » :valeur}

Instanciation :

Nom[clé]=valeur

Supprimer un ensemble clé/valeur :

nomDico.pop(« clé »)

Cette fonction renvoie la valeur supprimée

Stockage de fonctions dans le dictionnaire :

nomDico[« clé »]=nomFonction

Appel de fonction :

nomDico[« clé »](paramètres de la fonction)

Parcours d'un dictionnaire :

Plusieurs façon de faire :

```
For cle in nomDico.keys():
    Print(cle)
#fin
```

Ici c'est pour un parcours avec clés, attention dans l'affichage ce ne sera pas dans l'ordre rentré puisque qu'il n'y a pas d'ordre définit dans un dictionnaire.

```
For val in nomDico.values():
    Print(values)
#fin
```

Pareil pas d'ordre

Une façon combinée:

```
For cle,valeur in nomDico.items():
        Print("cle{},valeur{}".format(cle,valeur))
#fin
```

On peut comme pour les listes, capturer les paramètres d'une fonction et les mettre dans un dictionnaire, or les paramètres qui seront inclus dans un dictionnaire seront dit nommés

```
Def fonction(*non_nommés,**nommés) :
        Action #on pourrait mettre une fonction d'affichage des différents paramètres
#fin
```

On aura,

Fonction(2,3, « chat »,p=j,m=2)

On aura stockés tous les paramètres certains dans une liste d'autres dans un dictionnaire

Liste[2 , 3 , « chat »]

Dico[« p » : « j », « m » :2]

Les fichiers :

Ouverture d'un fichier avec ses modes :

Open(« nomFic », « mode »)

Les modes sont : r (read)/w (write en écrasant)/a (ajout à la suite)

Fermer le fichier :

Très important pour le bon fonctionnement

nomFic.close()

Lecture du fichier :

Contenu=nomFic.read()

Print(contenu)

Ecrire dans un fichier une chaine de caractères :

nomFic.write(« texte à mettre») #cette fonction renvoie le nombre de caractères écrits

Ecrire dans un fichier autre type :

On doit convertir ce qu'on entre en string en convertissant

nomFic.write(str(2))

Le gestionnaire de contexte :

Il permet de s'asurer que les fichiers ouvert sont bien fermés à la fin des actions pour ne aps causer d'erreurs fatales

With open(« fichier », « mode ») as fic:

    Actions

#fin

Sauvegarder des objets dans fichiers:

Importer pickle

Enregistrer:

With open (« fichier », « w ») as fic:

    P=pickle.Pickler(fic)

    P=dump(objet)

#fin

Récupérer:

```
With open("fichier","r") as fic:
        D=pickle.Unpickler(fic)
        Var=d.load()
#fin
```

Si plusieurs objets alors appeler load plusieurs fois, c'est pour cela qu'il vaut mieux faire un enregistrement par fichier.