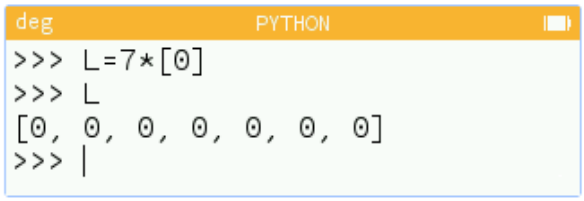
Listes

Une liste en Python est une structure de données qui peut contenir n'importe quel type de données (nombre entier, nombre flottant, chaîne de caractères, liste). Les éléments d'une liste sont rangés et séparés par des virgules. Le 1er élément de la liste L a pour rang 0 ; il est noté L[0].

Déclaration/initialisation :



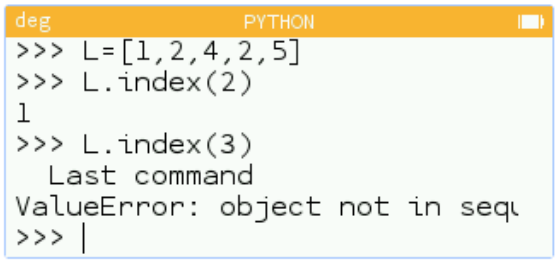
Une liste L peut être simplement déclarée par l'instruction L=list() ou L=[]. Elle existe alors mais ne contient aucun élément. On peut aussi la déclarer et l'initialiser en écrivant explicitement ses éléments

L=[2,3,5,7,11,13,17,19,23,29]

On peut encore déclarer et initialiser une liste avec N éléments égaux à 0 en écrivant L=N*[0].
Il y a bien d'autres façon de déclarer et d'initialiser une liste, par exemple avec la fonction range()

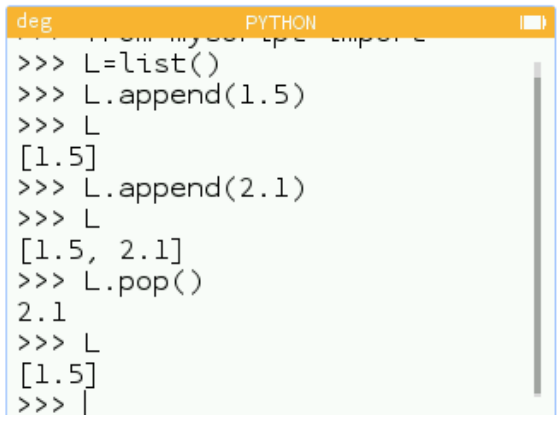
l'instruction L=*list*(*range*(1,10)) conduit à la liste L=[1,2,3,4,5,6,7,8,9]

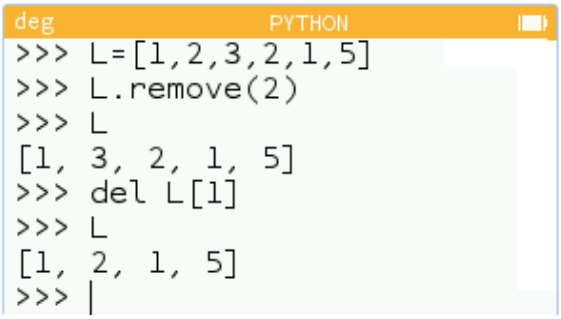
Accès/modification d'un élément d'une liste :



Lorsqu'une liste L a été créée, on peut accéder à l'élément de rang R en écrivant L[R] et le modifier ; en commençant par la fin : L[-1] est le dernier élément, L[-2] l'avant-dernier.
On peut aussi accéder au rang d'un élément : L.index(E) renvoie le 1er rang de l'élément E.
Attention : si il n'y a pas l'élément ou le rang demandé, ces instructions provoquent une erreur !
On peut extraire une liste E d'une liste L à partir d'un rang d jusqu'à un rang f en écrivant E=L[d:f].
Si un des rangs est omis, les valeurs par défaut sont d=0 (le début) et f=len(t) (la fin).
Si, par exemple, L=[12,5,7] alors L[1:2] renvoie [5], L[:2] renvoie [12,5] et L[1:] renvoie [5,7].

Ajout/suppression d'un élément dans une liste :





On peut ajouter un élément E en dernière position dans une liste L en écrivant

L.append(E)

Après L=[], l'instruction L.append(1.5) conduit à avoir L=[1.5]. Si on écrit encore L.append(2.1), on aura L=[1.5,2.1].
La méthode insert(R,E) insère l'élément E au rang R, en décalant tous les éléments suivants d'un rang. Si L=[1,2,3], l'instruction L.insert(2,5) insère l'élément 5 au rang 2 ce qui conduit à avoir L=[1,2,5,3].
Pour supprimer le dernier élément d'une liste L, écrire L.pop(). Cette instruction renvoie l'élément supprimé qui peut alors être utilisé.
L.pop(R) supprime et renvoie l'élément de rang R.
On peut aussi, avec L.remove(E), enlever la 1ère occurrence de l'élément E de la liste L.
L'instruction del L[R] supprime L[R], l'élément de rang R, de la liste L.

Longueur d'une liste :

Très utile pour éviter les erreurs de rang, l'instruction len(L) donne la longueur de la liste L (son nombre d'éléments). Avec la liste J=["Lu","Ma","Me","Je","Ve","Sa","Di"], l'instruction len(J) renvoie 7. Si j'ai une liste de listes, par exemple M=[[0,0],[0,1],[1,0]], l'instruction len(M) renvoie 3 (c'est la longueur de la liste M), par contre len(M[0]) renvoie 2 (c'est la longueur de la liste M[0]).

Test de la présence d'un élément dans une liste :
Le mot-clé in permet de tester la présence d'un élément dans une liste. N in L renvoie True si N est un élément de la liste L et False sinon. On peut tester si un élément N n'est pas dans la liste L en écrivant N not in L.
L'instruction L.count(E) donne le nombre d'occurrences d'un élément E dans une liste L.

```
deg                PYTHON
>>> L=[2,3,5,11]
>>> 5 in L
True
>>> N=7
>>> if N not in L:L.append(N)
>>> L
[2, 3, 5, 11, 7]
>>> L.sort()
>>> L
[2, 3, 5, 7, 11]
>>>
```

Tri d'une liste :
Une liste L peut être triée dans l'ordre alphanumérique croissant avec l'instruction L.sort(). On obtiendra le tri dans l'ordre inverse en utilisant la méthode reverse().

Parcours d'une liste :
L'instruction for x in L : est une déclaration de boucle bornée qui donne successivement à x la valeur des éléments de la liste L. Si P=1, l'instruction

for x in [2,3,5,7] : P*=x

conduit à P=210 (la syntaxe P*=x condense P= P*x).
On peut parcourir une liste pour en créer une autre avec ce type de déclaration qui conduit à L=[4,9,25,49]

L=[n*n for n in [2,3,5,7]]

On peut même effectuer un filtrage avec une condition L=[n*n for n in [1,2,3,4,5] if n%2==1] conduit à L=[1,9,25]

```
deg                PYTHON
>>> P=1
>>> for x in [2,3,5,7]:P*=x
>>> P
210
>>> L=[n**2 for n in [2,3,5,7]]
>>> L
[4, 9, 25, 49]
>>>
```

Choix d'un élément aléatoire dans une liste :
Cette possibilité est offerte par le module random qu'il faut donc importer au préalable par l'instruction from random import choice (ou import *)
Une liste L contenant au moins deux éléments, on en tire un N au hasard en écrivant N=choice(L).
Si L=[1,10,100,1000] alors choice(L) renvoie un élément au hasard de L.

Exercices :

1) Écrire une fonction qui supprime les doublons d'une liste : [1,2,2,1,2,1] est changée en [1,2].
2) Écrire un programme qui renvoie une main de N cartes prises au hasard dans un jeu de 32 cartes. Par exemple, avec N=2, on devrait pouvoir obtenir [As de Trèfle, Dix de Pique].
3) Écrire un programme qui affiche un petit calendrier de la semaine lorsqu'on lui fournit en argument la date du dimanche. Si on entre 29,4,18 (29 avril 2018), ce programme doit nous afficher Lundi 30 avril 2018, Mardi 1 mai 2018, …, Dimanche 6 mai 2018.
4) Écrire un programme qui détermine la liste des nombres premiers inférieurs ou égaux à un entier *n* donné. Selon la méthode du crible d'Ératosthène, on peut supprimer d'une liste contenant les entiers entre 2 et n tous les multiples de 2, puis supprimer tous les multiples du nombre suivant 2 (sauf erreur, cela doit être 3), etc.
5) En partant de l'écriture irréductible d'un nombre rationnel donnée par deux nombres *a* et *b* (le numérateur et le dénominateur de la fraction), écrire une fonction qui donne la nature du nombre $\frac{a}{b}$ (entiers, décimal non entier ou rationnel non décimal) et la suite des chiffres qui se répète dans l'écriture décimale d'un rationnel non décimal. Si on entre 1 et 7, la fonction doit renvoyer 142857. Si le nombre est décimal, la fonction donne son écriture décimale.
6) Un problème ancien : Josèphe fut amené à se disposer en cercle avec quarante personnes, sachant qu'en comptant de trois en trois à partir de l'un d'eux, ils devaient se supprimer mutuellement. Le premier à être supprimé est rangé en position 3 et il se fait tuer par celui qui est rangé en position 6 ; celui-ci est le suivant à être supprimé, par celui en position 9.

Ainsi de suite, progressivement, toute la troupe est appelée à s'autodétruire. À quelle place doit se ranger Josèphe pour être le dernier et devoir ainsi se supprimer lui-même ?

7) On effectue une collection d'objets choisis au hasard parmi N objets numérotés de 1 à N en cherchant à obtenir la collection complète. Par exemple, on tire au hasard un chiffre (entre 0 et 9) jusqu'à ce qu'on obtienne les 10 chiffres au moins une fois chacun. Simuler cette situation avec un programme qui prend N en argument et qui renvoie une liste contenant le nombre d'éléments de chaque sorte. Si on lance cette fonction avec N=10, on doit pouvoir obtenir quelque chose comme [3,5,1,2,1,2,5,3,1,1] qui montre qu'on a obtenu le 1er élément 3 fois, le 2ème 5 fois, … et le 10ème 1 fois.

8) On veut parcourir l'ensemble des rationnels strictement positifs en rangeant les fractions $\frac{a}{b}$ dans l'ordre de la somme $s=a+b$ et, pour une même valeur de *s*, dans l'ordre de *a*. L'idée est d'aller jusqu'à une certaine fraction $Q=\frac{A}{B}$ et d'afficher le nombre *N* et le pourcentage *p* des fractions irréductibles inférieures ou égales à *Q*. Avec $Q=\frac{2}{5}$, les 17 fractions rangées jusqu'à *Q* étant $\frac{1}{1}$, $\frac{1}{2}$, $\frac{2}{1}$, $\frac{1}{3}$, $\frac{2}{2}$, $\frac{3}{1}$, $\frac{1}{4}$, $\frac{2}{3}$, $\frac{3}{2}$, $\frac{4}{1}$, $\frac{1}{5}$, $\frac{2}{4}$, $\frac{3}{3}$, $\frac{4}{2}$, $\frac{5}{1}$, $\frac{1}{6}$ et $\frac{2}{5}$ parmi lesquelles 4 sont simplifiables ($\frac{2}{2}$, $\frac{2}{4}$, $\frac{3}{3}$ et $\frac{4}{2}$), on doit obtenir $N=17-4=13$ et $Q=\frac{13}{17}\approx 76{,}5\,\%$. Jusque là, on peut se passer des listes mais on souhaite obtenir l'affichage des fractions irréductibles dans l'ordre numérique croissant.

9) Voici l'énigme MU de Hofstadter[1] : on dispose de MI. À l'étape suivante, on ajoute toutes les chaînes que l'on peut obtenir par une des quatre règles suivantes : ①-Si on possède une chaîne se terminant par I, on peut lui ajouter U à la fin ; ②-Si on possède la chaîne Mx, on peut lui ajouter x pour obtenir la chaîne Mxx ; ③-Si on possède une chaîne contenant III, on peut remplacer III par U ; ④-Si une chaîne contient UU, on peut supprimer ce UU. L'énigme est la suivante : va t-on, à une étape quelconque, obtenir la chaîne MU ? Proposer un programme qui détermine les chaînes s'ajoutant à la collection à chaque étape et qui s'arrête quand MU est trouvé (cela peut ne jamais arriver). Vérifier qu'à l'étape 1 on obtient les chaînes MIU et MII et à l'étape 2 obtient MIUIU, MIIU et MIII.

10) Vous connaissez le code César : on commence par créer un alphabet de substitution en décalant toutes les lettres d'un certain nombre constituant la clé du code. On peut alors appliquer la substitution à chaque lettre du message à encoder ou bien, en sens inverse, on peut décoder un message. Par exemple, XKJFKQN code le message BONJOUR avec une clé de 4 (E code A, F code B, etc.). Le message suivant XQEBX MUEMZ FQDUQ EXQEB XGEOA GDFQE EAZFX QEYQU XXQGD QE a été encodé avec un code César mais on a perdu la clé. Écrire un programme qui puisse le décrypter. On peut utiliser le principe suivant : en français, la lettre la plus fréquente étant le E, il suffit de déterminer la lettre la plus fréquente pour savoir comment est codé le E et en déduire la clé et le message décodé. Si celui-ci n'a pas de sens, on peut essayer la 2ème lettre la plus fréquente.

D'autres informations sur les listes et sur Python en général ?

- L'aide de dans la console Python d'IDLE (pas dans la Numworks) est bien utile : taper help([]) ou help(list).
- L'ouvrage incontournable de Gérard Swinnen qui est libre : on peut le télécharger gratuitement (versions les plus récentes aux formats *pdf* et *odt*) ou en obtenir une version papier (chez Eyrolles).
- Le manuel de Vincent Le Goff *Apprenez à programmer en Python. Développer en Python n'a jamais été aussi facile!* cours complet existant en version imprimée (OpenClassrooms, collection Livre du Zéro).
- Apprendre la programmation Python
- Le tutoriel de Kamel Naroun

1 Douglas Hofstadter, Gödel, Escher, Bach, les Brins d'une Guirlande Éternelle, Dunod 2000, pp.38-47.