2017-2018

Méthodologie de la Recherche en Langue et Informatique

TD1: manipulations de base en Python

Gaël Lejeune, Sorbonne Université

Inspiré de Agnès Delaborde 2015-2016

---

1 Installation et rappels

Pour les consignes d'installation, référez vous au module de Remise à Niveau disponible sur `http:www.lejeunegael.fr/cours.html` puis dans quelques jours sur Moodle :

- `https://moodle.paris-sorbonne.fr/` puis "Sciences Humaines" et "Informatique Appliquée"
- ou directement : `https://moodle.paris-sorbonne.fr/course/index.php?categoryid=411`

Vous devez commencer à être familier des notions suivantes:

- Instruction
- Variable
- Type, typage
- Affectation
- Opération, opérateur, opérande

Si ce n'est pas le cas, c'est le moment de vous manifester !

2 Les chaînes de caractères

2.1 Affichage et manipulations de base

phrase1, phrase2 et phrase3 sont des chaînes de caractères. Saisissez les puis affichez les.

```
phrase1 = "les oeufs durs."
phrase2 = "Oui, répondit-il,"
phrase3 = "j'aime bien"
```

Testez différents affichages :

```
print(phrase2, phrase3, phrase1)

print(phrase2)
print(phrase3)
print(phrase1)

print(phrase2+"--"+ phrase3+"--"+phrase1)
```

Cette dernière opération s'appelle **concaténation** de chaînes de caractères. On voit donc, à nouveau, que l'opérateur "+" ne se comporte pas de la même manière selon les opérandes qu'il affecte.

2.2 Formatage de chaînes

Testez l'affichage suivant où l'on sépare les chaînes et les traitements qui leur sont appliqués.

```
print("%s--%s--%s"%(phrase2, phrase3, phrase1))

print("{}--{}--{}".format(phrase2, phrase3, phrase1))

# la méthode format permet des manipulations plus riches :
produit = "café"
prix = 0.456
print("Les %s coûtent %.2f euros" % (produit, prix))
```

Modifiez les déclarations des variables et ajoutez où vous souhaitez les caractères suivants : $\backslash n$ et $\backslash t$. Par exemple :

```
print(phrase2+"\t"+ phrase3+"\n"+phrase1)

print("%s\t%s\n%s"%(phrase2, phrase3, phrase1))

print("Le %s coûte \n\t %.2f euros sans sucre\n\t
 idem avec" % (produit, prix))
```

Dans l'exemple ci-dessous, notez la présence de l'antislash avant l'apostrophe. Testez cet exemple en omettant l'antislash :

```
toto = "N\'est-ce pas ?"
print(toto)
```

Testez maintenant cet exemple, en mettant tout d'abord l'antislash, puis sans le mettre :

```
toto = 'N\'est-ce pas ?'
print(toto)
```

" et ' sont équivalents, et sont des caractères spéciaux délimitant des chaînes de caractères, ici ils aident PYTHON à faire la délimitation (*parsing*) des éléments composant les instructions. On peut **protéger les caractères spéciaux** au sein d'une chaîne de caractère en utilisant l'antislash, on dit aussi qu'on les déspécialise. Si PYTHON ne parvient pas à délimiter, on a de bonnes chances d'avoir une *syntax error*.

2.3 Une chaîne de caractères en tant que collection ordonnée d'éléments

Saisissez cet exemple. Que concluez-vous sur la façon dont Python gère les chaînes de caractères ?

```
ch = "Stéphanie mange."
print(ch[0], ch[3])
```

Il est possible d'obtenir la longueur d'une chaîne avec la fonction len(). Affichez la longueur de la chaîne `ch` :

```
print(len(ch))
```

Affectez maintenant ces valeurs aux variables ch et n. De quel type sont ces deux variables ? Que constatez-vous ?

```
ch = "8000"
n = 45
print(ch+n)
```

Il est alors nécessaire de convertir la chaîne de caractère en entier. On utilise pour cela la fonction int(). Dans l'exemple ci-dessous, on affecte une nouvelle valeur à la variable ch. Testez ce code :

```
ch = int(ch)
print(ch+n)
```

Tentons maintenant de convertir la chaîne "bonjour" en un entier. Entrez le code ci-dessous. Que se passe-t-il ?

```
ch = int("bonjour")
print(ch)
```

Une variable peut être convertie, par exemple, en integer avec la fonction int(), en chaîne de caractères avec str() (str signifiant string), en flottant avec float().

Certaines opérations sur des variables de types différents sont cependant acceptées par Python. Par exemple, saisissez ce code :

```
s = "bonjour"
n = 5
print(s*n)
```

3 Les listes : des éléments ordonnés

L'exemple ci-dessous présente une liste nommée "jour". Saisissez l'exemple à la console.

```
jour = ["lundi", "mardi", "mercredi", 1800,
20.357, "jeudi", "vendredi"]
print(jour)
```

Une liste peut contenir différents types de variables. Maintenant, affichons l'élément à l'indice 2 de la liste (voir exemple). Que constatons-nous ?

```
print(jour[2])
```

Une liste en Python commence par l'indice 0.

On peut parcourir un mliste de la manière suivante :

```
for toto in jour:
  print(toto)
```

On peut faire des vérifications sur les types en même temps :

```
for toto in jour:
  print(toto)
  print(type(toto))
```

Manipuler une liste

On peut également assigner une nouvelle valeur à un élément de la liste. Entrez l'exemple ci-dessous, et affichez le contenu de la liste pour vérifier que votre modification a bien été prise en compte.

```
jour[3] = "Juillet"
```

Les fonctions len(liste) et del(liste[indice]) permettent respectivement d'obtenir la longueur de la liste (i.e. le nombre d'éléments la constituant), et de supprimer un élément à une position définie de la liste.

```
len(jour)
del(jour[4])
print(jour)
```

Que se passe-t-il si l'on tape l'exemple ci-dessous ?

```
del(jour[9])
```

La méthode append permet d'ajouter un élément à la liste. Ajoutez un élément comme dans l'exemple ci-dessous, puis affichez la liste. Où le nouvel élément a-t-il été placé ?

```
jour.append('samedi')
```

Manipuler et visualiser des données

4 Interaction avec l'utilisateur

La commande input() met en attente le script. La suite ne sera exécutée que lorsque l'utilisateur aura saisi une entrée. a) Saisissez ce code dans votre console. On note la présence de l'opérateur puissance **.

```
print('Veuillez entrer un nombre positif quelconque :')
nn = input()
print('Le carré de', nn, 'vaut', nn**2)
```

Que se passe-t-il ? Comment corriger cette erreur ?
L'exemple ci-dessous est une façon plus réduite d'écrire le même code :

```
nn = input('Veuillez entrer un nombre positif quelconque : ')
print('Le carré de', nn, 'vaut', nn**2)
```

Dans l'exemple ci-dessous, on demande à l'utilisateur d'entrer son prénom entre guillemets. Que se passe-t-il si l'on ne met pas ces guillemets ?

```
prenom = input('Entrez votre prenom (entre
guillemets) : ')
print('Bonjour, ', prenom)
```

5 Factorisation : bibliothèques, scripts et fonctions

Les bibliothèques

Une bibliothèque (appelée également "librairie" par anglicisme) regroupe une quantité de fonctions et de variables utiles. Faire appel à une bibliothèque permet de ne pas avoir à réécrire à chaque fois le contenu d'une fonction souvent utilisée.

Recopiez l'exemple ci-dessous. La première ligne peut se lire "depuis la bibliothèque math, importe tout".

```
from math import *
nombre = 121
print("racine carree de",nombre,"=",sqrt(nombre))
angle = pi/6
print("sinus de", angle,"radians","=",sin(angle))
```

Quels sont les deux fonctions de la bibliothèque math utilisées ? A quelle variable fait-on également appel ?

Plutôt que de charger toute la bibliothèque, on peut également charger uniquement les variables et fonctions dont on se sert (on lit : "depuis la bibliothèque math, importe pi, sqrt et sin") :

```
from math import pi, sqrt, sin
```

Création d'un script

Testez et sauvegarder le code suivant dans un fichier texte et envoyez le à gael.lejeune@sorbonne-universite.fr

```
#####################################
# Importation de fonctions externes
# Fonctions créées
#####################################
# Corps principal du programme :
```

```
phr = input("Veuillez entrer une phrase : ")
cch = input("Entrez le caractère à compter : ")
nc = i = 0
while i < len(phr):
  if phr[i] == cch:
    nc = nc + 1
  i = i + 1
print("La phrase contient", nc, "caractères", cch)
```

Les fonctions en Python

Étudiez la fonction ci-dessous. Ne la codez pas maintenant.

```
def table7():
  n = 1
  while n <= 10 :
    print(n * 7),
    n = n + 1
```

Repérez le nom de la fonction dans ce script ?

Que fait-elle ? Repérez le type de structure utilisée au sein de cette fonction.

Sur papier, notez dans un tableau les valeurs successives de n durant l'exécution de la boucle, en faisant tourner le script à la main (i.e. sans utiliser la machine). Voici ce que vous devez noter :

- Valeur initiale
- Valeur maximale
- Valeurs successives

Copiez le script dans un nouveau script python (que vous nommerez table7.py). Dans ce script, ajoutez un appel à cette fonction (voir ci-dessous) :

```
table7()
```

En reprenant le script ci-dessus, comment faire pour appeler trois fois de suite la fonction table7 ? Proposez deux façons.

Écrivez un script (dans un fichier nommé epeler.py) permettant d'afficher un à un les caractères de la chaîne "Demain, il va pleuvoir.". Ce script devra apparaître au sein d'une fonction nommée " epeler_mot ".

Fonctions avec paramètres

Recopiez l'exemple ci-dessous, et enregistrez-le dans un nouveau fichier table.py.

```
def table(base):
  n = 1
  while n <11 :
    print(n * base)
    n = n +1
```

Que fait cette fonction ? Quelle est l'utilité du paramètre base ?
Appelez cette fonction en lui passant en paramètre la valeur 10.
Ajoutez le code ci-dessous à la suite de votre script, puis exécutez-le.

```
a = 1
while a<=20:
  print("\nPasse numero",a)
  table(a)
  a = a + 1
```

A quoi sert cette boucle while ? Que fait le script ? Adaptez ce code afin qu'il n'affiche que les tables de multiplication de 5 à 10.

Éditez maintenant le script epeler.py afin que la fonction epeler_mot prenne en paramètre une chaîne de caractères.

Dans votre script, réalisez deux appels à la fonction epeler_mot, en prenant une fois en paramètre la chaîne s1, puis s2 (voir ci-dessous).

```
s1 = "Demain, il va pleuvoir."
s2 = "Aujourd'hui, il fait beau."
```

Dans un nouveau script (tableMulti.py), copiez le code suivant :

```
def tableMulti(base, debut, fin):
  print("Fragment de la table de multiplication par", base)
  n = debut
  while n <= fin :
    print(n, 'x', base, '=', n * base)
    n = n +1
```

Appelez cette fonction de façon à afficher la table de multiplication de 5, de 1 à 10.

Ajoutez le code ci-dessous et exécutez-le. Afin de vous entraîner à la compréhension d'un code, écrivez le tableau des valeurs successives de t, d et f.

```
t, d, f = 11, 5, 10
while t<21:
  tableMulti(t,d,f)
  t, d, f = t+1, d+3, f+5
```

Créez un nouveau script (appelé stop_lettre.py). Dans ce script, testez une chaîne de caractères en l'analysant caractère par caractère. Chaque caractère sera affiché, sauf s'il correspond à un caractère interdit. Par exemple, si nous analysons la chaîne "Rien ne va plus maintenant." et que la lettre interdite est "a", le script affichera ceci :

R i e n
n e
v
p l u s
m i n t e n n t.

Pour ce faire, Vous utiliserez impérativement une fonction qui prendra deux paramètres : la chaîne de caractères à analyser, et le caractère interdit.

Adaptez le script stop_lettre.py afin qu'il s'interrompe lorsqu'il rencontrera le caractère interdit (plutôt que de continuer à afficher). Vous aurez peut-être besoin de la commande break (qui interrompt une boucle).

R i e n
n e
v

Recopiez ce code dans un nouveau script :

```
def politesse(nom, vedette ='Monsieur'):
  print("Veuillez agréer,", vedette, nom, ", mes salutations distingu
```

Appelez cette fonction avec pour seul paramètre "Dupont", et ensuite avec les paramètres ("Dupont", "Mademoiselle"). Testez une syntaxe alternative pour la deuxième ligne, le " formatage de chaîne " :

```
print("Veuillez agréer,%s,%s, mes salutations distinguées."%(vedette,
```

Sur le même principe, écrivez une fonction qui affiche le résultat de la multiplication entre deux paramètres. Si le deuxième paramètre n'est pas fourni lors de l'appel à la fonction, alors par défaut il vaut 10.

Portée des variables

Dans un nouveau script, copiez et exécutez le code ci-dessous :

```
def f(s):
  print(s)
f("bonsoir")
```

Notez que l'on peut passer en paramètre directement la valeur d'une variable lors d'un appel à une fonction (f("bonsoir")). Que se passe-t-il si l'on ajoute à la suite de ce code print(s) ? Notez qu'il se passe la même chose si l'on cherche à connaître la valeur de s à la console.

Exécutez le script tableMulti.py, et tentez à la console d'afficher la valeur de la variable base.

```
print(base)
```

Pourquoi recevez-vous une erreur ?

Repérez pour chaque variable utilisée dans ce code si elle est locale à une fonction, ou globale.

```
def mask():
  p = 20
  print("Au sein de la fonction :",p, q)
  p, q = 15, 38
print("A l'exterieur de la fonction :",p,q)
mask()
```

A chaque endroit où sont citées les variables p et q au sein du code, indiquez si elles sont globales ou locales.

Fonctions avec retour

```
def f(n):
  n = n*2
  return n

resultat = f(3)
print(resultat)
```

Une façon de rédiger le code de manière plus condensée serait :

```
def f(n):
  return n*2
print(f(3))
```

Copiez ce code et exécutez-le. vérifiez que le code est bien indenté. Affichez ensuite le résultat de table(9).

```
def table(base):
  result = []
  n = 1
  while n < 11:
    b = n * base
  result.append(b)
  n = n +1
  return result
```