TP5 Les chaînes de caractères, les fichiers

Caractères

Les fonctions Python `ord` et `chr` permettent de passer d'un caractère à sa représentation numérique, et réciproquement.

In [ ]:
```
ord('a')
```

In [ ]:
```
ord('b')
```

In [ ]:
```
ord('z')-ord('a')
```

Q1 Quelle est le numéro du caractère @ ? Celui de 𝄞 ?

In [ ]:
```
chr(97)
```

In [ ]:
```
chr(0x41)
```

Q2 Quel est le numéro du caractère Z en décimal ? Et en hexadécimal ?

Q3 Et pour le caractère α ? Indication : chercher autour de `0xcebe`

Depuis la version 3 de Python, les caractères sont encodés en utf8, ce qui permet d'avoir accès aux accents, mais aussi aux alphabets et caractères du monde entier. Cependant pour d'autres logiciels, en particulier les navigateurs, l'existence de plusieurs encodages possibles reste parfoir un problème.

In [ ]:
```
chr(0x2600)+chr(0x2601)+chr(0x2614)+chr(0x2603)+chr(0x26a1)
```

In [ ]:
```
neko=chr(0x307E)+chr(0x306D)+chr(0x304D)+chr(0x306D)+chr(0x3053)
```

In [ ]:
```
neko
```

Q4 Donner une boucle permettant d'afficher tous les hiragana.

Les chaînes de caractères sont itérables avec `for in :`, comme les listes et les tuples.

In [ ]:
```
maneki='招き猫'
```

In [ ]:
```
for k in maneki:
    print(ord(k))
```

In [ ]:
```
for k in maneki:
    print(hex(ord(k)))
```

Q5 Donner une boucle permettant d'afficher tous les caractères ASCII imprimables, du numéro 32 inclus au numéro 127 inclus. Dans `print`, utiliser le paramètre optionnel `end=''` afin de tout afficher sur une même ligne.

Q6 Que contient la variable `ch` une fois qu'on a exécuté le programme ci-dessous ?

In [ ]:
```
ch=''
for k in range(97,123):
    ch=ch+chr(k)
```

Chaînes de caractères et listes

Les slices sont utilisables sur les chaînes de caractères comme sur les listes.

In [ ]:
```
alphabet='abcdefghijklmnopqrstuvwxyz'
```

In [ ]:
```
alphabet[:6]
```

In [ ]:
```
alphabet[6:]
```

In [ ]:
```
alphabet[::3]
```

Q7 L'instruction `alphabet[a:b:c]` renvoie `'dinsx'`. Quelles sont les valeurs de `a`, `b` et `c` ? Donner toutes les possibilités !

Q8 Quel est le type de l'objet renvoyé par l'instruction ci-dessous ? Quel est sa longueur ?

In [ ]:
```
list(alphabet)
```

In [ ]:
```
'-'.join(list('Python'))
```

In [ ]:
```
'-*-'.join(['Errors','should','never','pass','silently'])
```

In [ ]:
```
'-*-'.join(['','Errors','should','never','pass','silently',''])
```

Q9 Que renvoie l'instruction ci-dessous ?

In [ ]:
```
'#'.join(['Tous','ces','moments','se','perdront','dans',"l'oubli",
          'comme','des','larmes','dans','la','pluie'])
```

In [ ]:
```
import this
```

In [ ]:
```
s='les:mots:sont:séparés:par:un:deux-points'
```

In [ ]:
```
s.split(':')
```

In [ ]:
```
a=s.split(':')
' '.join(a)
```

Les fonctions `split` et `join` sont réciproques l'une de l'autre, et permettent de découper une chaîne en morceaux ou de reconstruire une chaîne à partir de ces morceaux.

L'opérateur `+` permet de concaténer des chaînes, ce qui est souvent utile en lien avec la fonction `str` qui permet de transformer la plupart des objets python en une représentation sous forme de chaîne de caractères.

In [ ]:
```
x=12
a='Le carré de '+str(x)+' est '+str(x**2)
print(a)
```

Q10 Que renvoie l'appel `fonctionmystère(10)` ?

In [ ]:
```
def fonctionmystère(n):
    s=0
    ch=''
    for k in range(n):
        s=s+k
        ch=ch+str(k)
    return s,ch
```

In [ ]:
```
fonctionmystère(10)
```

In [ ]:
```
import binascii,base64
```

La documentation officielle du module binascii (https://docs.python.org/2/library/binascii.html), et celle du module base64 (https://docs.python.org/fr/3/library/base64.html#module-base64).

In [ ]:
```
s=b'Bonjour les amis !'
e=base64.b64encode(s)
```

In [ ]:
```
type(s),type(e)
```

Q11 Quelle est la représentation de l'objet `bytes s en base64` ?

La documentation Python sur les bytes (https://docs.python.org/fr/3/library/stdtypes.html#bytes), l'article wikipedia sur base64 (https://fr.wikipedia.org/wiki/Base64).

La représentation d'un texte dans un fichier

Python utilise deux types de fichiers, les fichiers textes qui contiennent des caractères, et les fichiers binaires, qui contiennent des bytes. Un caractère peut être codé par plusieurs octets. Pour travailler avec un fichier, on commence par l'ouvrir avec la fonction `open` . Cela signifie qu'on fournit à Python le chemin d'accès et le nom du fichier. Ensuite àon peut lire ou écrire dans le fichier, suivant le mode sous lequel on a ouvert le fichier. Il faut ensuite penser à fermer le fichier, avec la fonction `close` , pour que le fichier soit écrit sur le disque.

In [ ]:
```
fichier=open('texteJB.txt','w')   #le fichier texte.txt est ouvert en mode écr
iture 'w'
                                  #dans le répertoire courant

# attention, appeler le fichier texteJB si vous vous appelez John Bercow.
# penser à mettre vos propres initiales pour éviter que tout le monde écrase l
e fichier

ligne=input()
while ligne:                      #une chaîne est considérée True si elle n'est
pas vide
    fichier.write(ligne+'\n')     #le caractère '\n' est un passage à la ligne
    ligne=input()

fichier.close()
```

Le mode `'r'` est le mode lecture. La construction `with as` permet de ne pas avoir à fermer le fichier explicitement, le fichier sera fermé automatiquement à la fin du bloc indenté. Le fichier texte est un itérable : avec `for in` la variable de boucle prendra comme valeur chaque ligne du fichier.

In [ ]:
```
with open('texteJB.txt','r') as f:
    for ligne in f:
        print(ligne)
```

Pour afficher le contenu d'un fichier, on peut utiliser une vidange hexadécimale (un hexdump en Anglais). La commande `hd` sert précisément à cela sous un système Unix/Linux. Sous les autres systèmes, moins performants, il faut ajouter un programme extérieur comme par exemple HxD (https://mh-nexus.de/en/hxd/).

In [ ]:
```
!hd Documents/Curie19/NSI/texte1.txt
```

In [ ]:
```
base64.encode('texte1.txt','texte64.txt')
```

Q12 Quelle est la longueur de chacun des deux fichiers `texte1.txt` et `texte64.txt` ?

Exercices

Q13 Ecrire une fonction `somme` qui prend une chaîne d'entiers séparés par des `+` et qui calcule la somme (utiliser `split`).

In [ ]:
```
def somme(ch):
    liste=...
    s=0
    for k in liste:
        s=...
    return

assert(somme('12+23+35')==70)
assert(somme('1+2+3+4+5+6+7+8+9+10')==55)
```

Q14 Ecrire une fonction `ADN` qui vérifie qu'une chaîne correspond au code génétique formé des quatre lettres ATCG.

In [ ]:
```
def ADN(s):
    for k in s:
        if :
            return False
    return True
```

Q15 Ecrire une fonction `complément` qui prend une chaîne représentant un brin d'ADN et qui renvoie une chaîne représentant le brin complémentaire : A, T, C et G sont complémentaires respectivement de T, A, G et C.

Q16 Ecrire une fonction `prefixes` qui imprime les préfixes d'une chaîne. P.ex `prefixes('Python')` doit afficher

P

Py

Pyt

Pyth

Pytho

Python

Q17 Ecrire une fonction `suffixes` .

Ecrire dans un fichier csv

Le format Comma Separated Values est un format qui permet de stocker des lignes contenant des champs séparés par des virgules (Comma en Anglais). Ce format est typiquement utilisé par les tableurs.

Nous allons calculer des valeurs et les ranger dans un fichier qu'on pourra ensuite ouvrir avec un tableur.

In [ ]:
```
f=open('dataJB.csv','w')
for k in range(20):
    f.write(str(k)+','+str(100*0.9**k)+'\n')
f.close()
```

Q18 Comment s'appelle la suite de la deuxième colonne ? (Ceux qui n'ont pas pris spé maths, demandez à ceux qui savent) Ouvrir le fichier avec un tableur, puis représenter la suite par un nuage de points.

Q19 Ecrire un programme qui écrit dans `carreJB.csv` les nombres de -2 à 3 par pas de 0,1 dans la première colonne, et leurs carrés dans la deuxième colonne.

Q20 Décrire ce que fait le programme ci-dessous.

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
abscisses=[]
ordonnées=[]

x=-2
h=0.1
while x<3:
    abscisses.append(x)
    ordonnées.append(x*x)
    x=x+h
```

In [ ]:
```
plt.plot(abscisses,ordonnées,label='Ligne brisée $y=x^2$')
plt.title("Une parabole")
plt.axis([-2,3,0,10])
plt.axis('equal')
plt.grid()
plt.legend()
```

In [ ]:
```

```