LISTES

Voir le cours en ligne pour plus de détails.

Python 1: Chaînes de caractères - type str

```
texte = ''                  # texte vide
poisson = 'thon'
print(poisson)              # affichage
len(poisson)                # longueur = 4

serpent = a + poisson       # concaténation
3*poisson

# test d'appartenance
poisson in serpent          # True
serpent in poisson          # False
```

- délimitées par des guillemets simples ou doubles.
- une chaîne est *non-mutable* : on ne peut pas la modifier.

Python 2: Listes - type list

```
listeVide=[]                # liste vide
liste = [1,'deux',3]
print(liste)                # affichage
len(liste)                  # longueur = 3

liste += ['intrus',7]       # concaténation
2*liste      # 2 fois la liste bout à bout

# test d'appartenance
1 in liste                  # True
[1,'deux'] in liste         # False
```

- suite finie d'éléments indicés par $[\![0, n-1]\!]$
- une liste est *mutable* : on peut la modifier.

Les listes se manipulent comme des chaînes de caractères mais sont modifiables.

Python 3: conversion

```
liste = list('texte')      # 'texte' en liste
liste = list(range(0,10)) # liste de 0 à 9
```

Indexation et slicing

Fonctionne comme pour le range pour extraire des sous-listes (ou sous-chaînes de caractères).

Syntaxe	Signification
`liste[p]`	élément d'indice p de la liste.
`liste[p:q]`	sous-liste des éléments d'indices p à $q-1$
`liste[p:q:r]`	sous-liste des éléments d'indices p à $q-1$ par pas de r.
`liste[:q]`	sous-liste des éléments d'indices 0 à $q-1$
`liste[p:]`	sous-liste des éléments d'indices p à la fin.
`liste[:]`	liste complète.
`liste[::-1]`	liste lue à l'envers.

Python 4: slicing

```
liste[0]       # élément 0 de la liste
liste[-1]      # dernier élément
liste[1:4]     # éléments de 1 à 4 (exclu)
liste[3:]      # éléments à partir de 3
liste[:4]      # éléments jusqu'à 4 (exclu)
liste[0:4:2]   # en comptant de 2 en 2
liste[::-1]    # liste miroir

liste[0:2]=['a',2] # modif les deux 1ers élt
liste[2:2]=[1,3] # insère [1,3] en indice 2
```

Python 5: Modification en place

```
# la variable liste est modifiée
del liste[1]       # supprime élément 1
liste.append(a)    # ajouter élément a en fin
liste.pop()        # retire dernier et renvoie

# Compléments
liste.pop(3)       # idem pour l'indice 3
liste.remove(a)    # retire 1re occur. de a
liste.insert(4,a) # insère a en position 4
liste.sort()       # ordonne la liste
liste.reverse()    # miroir de  la liste
```

Python 6: Chercher dans les listes

```
a in liste       # teste l'appartenance
liste.index(a)   # indice 1ère occurrence de a
liste.count(a)   # nombre d'occurrences de a
```

Python 7: copies de listes

```
copie = liste.copy()  # 'vraie' copie
copie2 = liste[:]     # idem
```

Python 8: Listes en compréhension

```
[x**2 for x in range(1,100)]
# liste des carrés des entiers de 1 à 99

[x**2 for x in range(1,100) if x%2==1]
# carrés des entiers impairs de 1 à 99

[x*y for x in range(10) for y in range(3,5)]
# l'ordre des for est important
```

ALGORITHMES À CONNAÎTRE

Python 9: Recherche d'éléments

```
def appartient(a,liste):
    for i in liste:
        if a == i:
            return True
    return False

def indice(a,liste):
    for i in range(len(liste)):
        if a == liste(i):
            return i
    raise ValueErrort(str(a)+" not in list")

def compter(a,liste):
    n = 0
    for i in liste:
        if a == i:
            n += 1
    return n
# ou
def compter(a,liste):
    n = 0
    for i in liste:
        n += (a == i)
    return n
```

Python 10: Statistiques

```
# somme des éléments d'une liste de nombres
def somme(liste):
    s = 0
    for k in liste:
        s += k
    return s

# maximum d'une liste de nombres
def maximum(liste):
    m = liste[0]
    for i in liste[1:]:
        if i > m:
            m = i
    return m

# (premier) indice du maximum d'une liste de
    nombres
def maxIndice(liste):
    m = 0
    for i in range(1,len(liste)):
        if liste[i] > liste[m]:
            m = i
    return m

# moyenne pondérée arrondie à 0,1
def moyPonderee(notes,coeffs):
    s,c = 0,0
    for i in range(len(notes)):
        s += notes[i]*coeffs[i]
        c += coeffs[i]
    return round(s/c,1)
```

Python 11: Extraction et compréhension

```
# sous-liste des éléments > M
def listeSup(liste,M):
    res = []
    for i in liste:
        if i > M:
            res.append(i)
    return res
# idem par compréhension
def listeSup(liste,M):
    return [i for i in liste if i>M]
# liste de 20 nombres aléatoires entre 1 et 6
[randrange(1,7) for i in range(20)]
```

Python 12: Tris de liste avec effets de bord

```
def triSelection(liste):
    for p in range(len(liste)):
        minIndice = p
        for i in range(p+1, len(liste)):
            if liste[i] < liste[minIndice]:
                minIndice = i
        liste[p],liste[minIndice] = liste[
            minIndice],liste[p]
    return liste

def triInsertion(liste):
    for i in range(len(liste)):
        j = i
        while j>0 and liste[j] < liste[j-1]:
            liste[j],liste[j-1] = liste[j-1],
                liste[j]
            j = j-1
# ou
def triInsertion2(liste):
    for i in range(len(liste)):
        j = i
        carte = liste.pop(i)
        while j>0 and carte < liste[j-1]:
            j = j-1
        liste.insert(j,carte)

def triBulles(liste):
    chaos = True
    while chaos:
        chaos = False
        for i in range(len(liste)-1):
            if liste[i] > liste[i+1]:
                liste[i],liste[i+1]= liste[i
                    +1],liste[i]
                chaos = True
#ou
def triBulles2(liste):
    fin = len(liste)
    while fin > 1:
        last = 0
        for i in range(fin-1):
            if liste[i] > liste[i+1]:
                liste[i],liste[i+1]= liste[i
                    +1],liste[i]
                last = i+1
          fin = last
```