Aide-mémoire Algorithmique/Python

Master *"Technologie et Handicap"* : Intensifs 1

Algo	*Python*

Commentaires

(Sans objet en algo)

```
# Tout ce qui suit le caractère # sera
# ignoré par l'interpréteur python
```

Affectation

```
a ← 12
b ← a + 32
s ← "Une␣chaîne␣de␣caractères"

i ← i + 1
```

```
a = 12
b = a + 32
s = "Une␣chaîne␣de␣caractères"

i = i + 1
i += 1
```

Opérateurs

```
Opérateurs arithmétiques entiers
Opérateurs arithmétiques réels
Opérateurs de comparaison
Concaténation de chaînes de caractères
```

```
+ - * / %
+ - * /
< <= == >= > !=
+
```

Lecture/Écriture

```
ecrire("bonjour")




lire a
lire "Tapez␣quelque␣chose:",a
```

```
print "bonjour"
# insère un saut de ligne à la fin

print "le␣résultat␣est", res
# ou res est une variable. La virgule
# sépare les expressions à afficher
# et insère un espace

a=input()
a=input("Tapez␣quelque␣chose:␣")
# la saisie est interprétée,
# e.g. si on saisit 3+4,
# la valeur 7 sera affectée à a
# les chaînes de caractères doivent
# être saisies avec des guillemets

b=raw_input("Tapez...")
# permet de saisir une chaîne
# sans avoir à taper de guillemets.
# quelque soit la saisie, ce sera
# interprété comme une chaîne (même
# si ce sont des nombres)
```

Algo | *Python*

Conditions

```
si expression alors
    instruction a1
    instruction a2
    ...
sinon
    instruction s1
    instruction s2
    ...
finsi
suite du programme
```

```
if expression:
   instruction a1
   instruction a2
   ...
else:
   instruction s1
   instruction s2
    ...
suite du programme
```

S'il n'y a pas de sinon

```
si expression alors
    ...
finsi
suite du programme
```

```
if expression:
   ...
suite du programme
```

Conditions imbriquées

```
si expression1 alors
    ...
sinon si expression2 alors
    ...
sinon si expression3 alors
    ...
sinon
    ...
finsi
suite du programme
```

```
if expression1:
    ...
elif expression2:
    ...
elif expression3:
    ...
else:
    ...
suite du programme
```

Itération

```
tantque condition faire
    instruction 1
    instruction 2
    ...
fait
suite du programme
```

```
while condition:
    instruction 1
    instruction 2
    ...
suite du programme
```

Boucles de type "pour"

```
pour i allantde val1 a val2 faire
  instruction 1
  instruction 2
  ...
fait
suite du programme


pour i allantde 0 a n-1 faire
  instruction 1
  instruction 2
  ...
fait
suite du programme
```

```
for i in range(val1, val2+1):
  instruction 1
  instruction 2
  ...
suite du programme
# range(p,q) renvoie toutes les valeurs
# entre p et q-1

for i in range(n):
  instruction 1
  instruction 2
  ...
suite du programme
# range(n) renvoie toutes les valeurs
# entre 0 et n-1
```

Quelques fonctions prédéfinies Python

Conversions

```
Conversion en chaîne de caractères

exemples (sur l'interpréteur):
>>> str(1)
'1'
>>> str(12.1)
'12.1'
>>> str(12+3)
'15'
>>> a=0
>>> str(a==0)
'True'
>>>


Conversion d'une chaîne en entier

exemples (sur l'interpréteur):
>>> int("12")
12
>>>


Conversion d'une chaîne en réel

exemples (sur l'interpréteur):
>>> float("3.141592654")
3.141592654
>>>
```

```
n=2
ch=str(n)

ch2="le nombre est "+ch
# concaténer directement la chaîne avec
# n planterait car on ne peut pas
# concaténer une chaîne et un nombre
# str() permet de convertir n'importe
# quel type de variable

min=0
max=10
print "["+str(min)+";"+str(max)+"]"
# affiche: [0;100]

s=raw_input("Tape un nombre, stp:")
n=int(s)
# convertit la saisie s en entier
# attention : s doit contenir un nombre
# on verra plus tard comment faire si
# ce n'en est pas un (voir mécanisme
# d'exceptions)

s=raw_input("Tape un nombre, stp:")
f=float(s)
# convertit la saisie s en réel
# attention : s doit contenir un nombre
```

Nombres aléatoires (module `random`)

```
import random
# importe la fonction random

random.randint(min,max)
# ou min, max sont des entiers,
# renvoie un entier aléatoire tiré
# entre min et max (compris)
```

```
# Exemple
import random

alea=random.randint(0,100)
# affecte à la variable alea un entier
# aléatoire entre 0 et 100 compris

print alea
# affiche ce nombre
```

Listes

```
Créer des listes
>>> a=[1,2,3,4]
```
Crée une liste contenant les entiers 1, 2, 3 et 4 et la Place dans la variable a

```
>>> dreels=[1.0, 3.14, 6.12]
```
Place dans dreels une liste de 3 réels

```
>>> voy=['a', 'e', 'i', 'o', 'u', 'y']
```
Place dans voy une liste de caractères

```
>>> vide=[]
```
Place dans vide une liste vide

```
Utiliser
>>> print voy[0]
a
```
Affiche le 1er élément la liste (num 0)

```
>>> print voy[-1]
y
```
Affiche le dernier élément

```
>>> print voy[1:3]
['e', 'i']
```
Affiche la sous-liste comprise entre les indices 1 (compris) et 3 (non compris)

```
>>> print voy[-2:]
['u', 'y']
```
Affiche les 2 derniers

```
>>> a[0]=a[0]+10
```
Ajoute 10 à a[0]

```
>>> print a
[11,2,3,4]
>>> b=a+[5,6,7]
```
Concatène 2 listes et les affecte à b

```
>>> print b
[11,2,3,4,5,6,7]
>>> print len(b)
7
```
Affiche la longueur de la liste b

```
Quelques fonctions
>>> a.append(5)
```
Ajoute 5 à la liste a

```
>>> print(a)
[11,2,3,4,5]
>>> a.extend([6,7,8,9])
```
Ajoute les éléments de la liste [6,7,8,9] à la liste a

```
>>> print(a)
[11,2,3,4,5,6,7,8,9]
>>> c=[0,1,0,1,2,0,1,2,3]
>>> print c.count(0)
3
```
Affiche le nombre d occurrences de 0 dans la liste c

```
>>> c.insert(0,9)
```
Insère un 9 au début de c

```
>>> print c
[9,0,1,0,1,2,0,1,2,3]
>>> c.insert(-1,9)
>>> print c
[9,0,1,0,1,2,0,1,2,9,3]
```
Insère un 9 à l avant dernière position de la liste c

```
>>> c.remove(1)
>>> print c
[9,0,0,1,2,0,1,2,3,9]
```
Enlève le premier 1 de la liste

```
>>> c.reverse()
>>> print c
[9, 3, 2, 1, 0, 2, 1, 0, 0, 9]
```
Retourne la liste c

```
>>> c.sort()
>>> print c
[0, 0, 0, 1, 1, 2, 2, 3, 9, 9]
```
Trie la liste c

```
>>> chaine=str(c)
```
Convertit la liste c en chaîne