FICHE DE L'ATELIER 8-9-10 : LOGIQUE et STRING (5 pages)

CODAGE DES ACTIONS CONDITIONNELLES : IF

Tests logiques

Symbole	Signification
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

and	code	ET	: renvoie **True** si les deux conditions sont True
or	code	OU	: renvoie **True** si l'une des deux conditions au moins est True
not	code	NON	: renvoie **True** si la condition vaut False, renvoie **False** si la condition est True
			On note ainsi **not variable** ou **variable!** Pour utiliser le test NON. En gros, cela inverse la sélection

Attention : les instructions à traiter sont comprises par Python à l'aide de la **tabulation**.
Pensez à les faire afficher par Notepad++ dans le menu Affichage – Symboles spéciaux.

Codage du **SI** (IF)

Exemple de test if, elif et else :

```
if nombre > 0:
    print("Le nombre est positif")
    print("Ok")
elif nombre < 0:
    print("Le nombre est négatif")
else:
    print("Le nombre est nul")
```

Exemple de test if imbriqué dans un autre :

```
a=17
if a>0:
    if (a>10 and a<16): # Parenthèses si deux conditions
        print( "C'est pas mal comme note." )
    elif a>= 16:
        print( "C'est bien !" )
    else:
        print( "C'est pas bien !" )
else:
    print("Le prof a dû se tromper, votre note est négative." )
```

Pour tester **la présence d'un caractère** dans une chaîne :

```
chaine =  "ours tigre poule"
for lettre in chaine:    # lettre est un caractère
    if lettre in "AEIOUYaeiouy": # lettre est une voyelle
        print(lettre)
    else:
        print(".")
```

CODAGE DES BOUCLES FOR

Codage de la boucle **POUR** (FOR)

Exemple de boucle FOR numérique utilisant un range:

```
# Pour i partant de 5 et tant que i<30, en augmentant de 10.
for i in range(5,30,10):
    print("Cas ",i) # On obtient 5, 15 et 25.
    print (i," x2 =",i*2)
```

Remarque :
Si on tape range(30), on compte de un en un de 0 à 29, c'est à dire tant que i<30
Si on tape range(5,30), on compte de un en un de 5 jusqu'à 29, c'est à dire tant que i<30.

Avec une chaîne de caractère : **Affichage des caractères un à un**

```
y = "Hello Wordl!"   # On crée le string y
# x est ici un caractère
for x in y:          # Pour chaque caractère x contenu dans y
      print(x)       # Affiche x à l'écran
```

Avec une chaîne de caractère : **Affichage des caractères un à un**

```
y = "Hello Wordl!"     # On crée le string y
nLongueur = len(y)
# x est ici un integer qui correspond au numéro de case
for x in range(nLongueur):  # x est ici le numéro de la case
   print(y[x])
```

Lecture des **éléments** contenus dans une **liste** :

```
y = [1,2,9,8,"ah ah ah",45,(45,45,45)]
for x in y:
      print(x)
```

Affichage des **mots** contenus dans une **phrase**

```
phrase = "Désormais, nous allons pouvoir gérer ..."
mots = phrase.split() # mots est la liste issue du "split"
for mot in mots:      # pour chaque élément mot contenu dans mots
      print(mot)            # on affiche le mot
```

CODAGE DES BOUCLES WHILE

Codage de la boucle **Tant que** (While)

Première méthode pour gérer une note supérieure à 20 :

```
note1=-1
while (note1>20):
    note1 = input("Quelle est votre note (entière) ? ")
    note1=int(note1)
print("Votre note est ", note1)
```

Deuxième méthode pour gérer une note supérieure à 20 :

```
test = True
while test:
    note1 = input("Quelle est votre note (entière) ? ")
    note1=int(note1)
    if (note1>=0 and note1<=20):
        test=False
print("Votre note est ", note1)
```

SORTIR DES ACTIONS

Sortir d'une boucle

L'instruction **break** permet de sortir immédiatement par le bas de la boucle en cours.

L'instruction **continue** permet de revenir immédiatement en haut de la boucle pour continuer le traitement.

OBJET STRING (chaîne de caractère)

String

C'est un assemblage de caractère. Le premier a le numéro 0.
On peut accéder au contenu de la case i à l'aide de la syntaxe `chaine[i]`

Exemple :

```
chaine = "Bonjour les gens"
print(chaine[0]) # Affiche B
print(chaine[0:10]) # Affiche Bonjour le
print(chaine[2:10]) # Affiche njour le
print(chaine[2:10:2]) # Affiche norl
```

Attention : Un string n'est pas modifiable.
Pour « modifier » une chaîne de caractère, il faut en créer une nouvelle avec le même nom.
Exemple :

```
chaine = "Bonjour"
chaine[0]= "C" #Provoque une erreur car modification impossible.
```

Concaténation

Il s'agit du fait d' « additionner » des chaînes de caractères entre elles.

```
y1 = "Bonjour"
y2 = " à tous"
y3 = y1 + " " + y2    # y3 contient "Bonjour à tous"
```

Fonction **len**()

Renvoie le nombre de caractères contenus dans un string, son « nombre de cases ».

```
nLongueur = len(monTexte)
```

Fonction **int**()

Convertit si possible la chaîne en interger et renvoie le résultat.

```
chaine = "42"
x = int(chaine)    # x contient le nombre 42, pas la chaine « 42 »
```

Méthodes **lower**() et **upper**()

`x.lower()` renvoie x en minuscule si x est une lettre ou une chaîne.
Pour MODIFIER x, il faudrait donc noter x = x.lower().

`x.upper()` renvoie x en majuscule si x est une lettre ou une chaîne.
Pour MODIFIER x, il faudrait donc noter x = x.lower().

Méthode **split**()

Renvoie une liste de « mots » contenu dans une chaîne. Pour lire la liste, voir le FOR.

```
mots = chaine.split()    # le séparateur est l'espace
mots = chaine.split(',') # le séparateur est la virgule
```

Méthode **replace**(old, new)

Remplace la chaîne old par la chaîne new.

```
phrase = phrase.replace(".","") # Remplace les points par « rien »
y1 = y1.replace( "a","*" ) # Remplace les a par des étoiles *
```

Méthode **isprintable**()

Cette méthode renvoie **True** si le caractère est un vrai caractère imprimable, par oppostion à caractère de contrôle.

Méthode **isalpha**()

Cette méthode renvoie **True** si le caractère est alphabétique, **False** sinon.
Si x vaut "abc", alors `x.isalpha()` renvoie **`True`**
Si x vaut "a:c" alors `x.isalpha()` renvoie **`False`**

Méthode **isnumeric()**

Cette méthode renvoie **True** si le caractère est décimal, **False** sinon.
Si x vaut "12", alors `x.isnumeric()` renvoie **`True`**.
Si x vaut "12." alors `x.isnumeric()` renvoie **`False`**.

Méthode **isalnum()**

Cette méthode renvoie **True** si le caractère est alphabétique, **False** sinon.
Si x vaut "abc", alors `x.isalnum()` renvoie **`True`.**
Si x vaut "a5c" alors `x.isalnum()` renvoie **`True`**.
Si x vaut "a5.c" alors `x.isalnum()` renvoie **`False`**.

DEUX EXEMPLES NON TRIVIAUX D'UTILISATION DES TESTS

Gestion des erreurs lors de l'exécution pour traiter correctement les nombres.

Gestion des exceptions dans les erreurs

try :
except *** :

```
# -*-coding:Utf-8 -*
import os
# Acquisition des donnees note 1
test = True
while test:
    note1 = input( "Quelle est votre note (entière) ? " )
    try:
        note1 = float(note1)
    except ValueError:
        note1 = -1
    if (note1>=0 and note1<=20):
        test = False
print("Votre note est ", note1)
os.system("pause")
```

Lecture des codes couleurs des pixels d'une image RGB :

```
# On importe la classe Img
from PIL import Image as Img
# On crée un objet-image à partir d'un fichier-image
monImage=Img.open( "linux.jpg" )
# On va chercher la largeur de l'image
nLargeur = monImage.width
# On va chercher la hauteur de l'image
nHauteur = monImage.height
# Pour y variant de 0 jusqu'à nHauteur exclus
for y in range(nHauteur) :
    for x in range(nLargeur): # x variant de 0 jusqu'à nLargeur
        # Place dans r,g et b les RGB du pixel (x,y) de monImage
        r,g,b = monImage.getpixel((x,y))
# On est en dehors de la boucle, on reprend le fil du programme
```

FONCTIONS APPLICABLES SUR LES CARACTERES

Méthode **ord**()	Renvoie le code numérique d'un caractère. `print(ord(x))  # renvoie 65 si x==A ou 97 si x==a`
Méthode **chr**()	Renvoie le caractère associé au code numérique `print(chr(x))  # renvoie A pour si x ==65 ou a si x == 97`

FONCTIONS PERMETTANT DE SUIVRE LES VARIABLES

Fonction **id**(var)	Renvoie l'adresse mémoire de la variable var.
Fonction **type**(var)	Renvoie le type des données stockées dans la variable var. <class 'str'> pour une chaîne de caractère. <class 'int'> pour un entier. <class 'float'> pour un réel.

Fonction **print ()**	Par défaut, le print provoque un passage à la ligne. Si on ne veut pas passer à la ligne, il faut utiliser : `print(lettre, end="")`

TABLE ASCII

0	NUL	*Null* (nul)
1	SOH	*Start of Heading* (début d'en-tête)
2	STX	*Start of Text* (début de texte)
3	ETX	*End of Text* (fin de texte)
4	EOT	*End of Transmission* (fin de transmission)
5	ENQ	*Enquiry* (demande)
6	ACK	*Acknowledge* (accusé de réception)
7	BEL	*Bell* (sonnerie)
8	BS	*Backspace* (espacement arrière)
9	HT	*Horizontal Tab* (tabulation horizontale)
10	LF	*Line Feed* (saut de ligne)

11	VT	*Vertical Tab* (tabulation verticale)
12	FF	*Form Feed* (saut de page)
13	CR	*Carriage Return* (retour chariot/retour à la ligne)
14	SO	*Shift Out* (code spécial)
15	SI	*Shift In* (code standard)
16	DLE	*Data Link Escape* (échappement en transmission)
17	DC1	*Device Control 1* à *4* (contrôle de périphérique)
18	DC2	
19	DC3	
20	DC4	
21	NAK	*Negative Acknowledge* (accusé de réception négatif)

22	SYN	*Synchronous Idle* (attente synchronisée)
23	ETB	*End of Transmission Block* (fin de bloc de transmission)
24	CAN	*Cancel* (annulation)
25	EM	*End of Medium* (fin de support)
26	SUB	*Substitute* (remplacement)
27	ESC	*Escape* (échappement)
28	FS	*File Separator* (séparateur de fichier)
29	GS	*Group Separator* (séparateur de groupe)
30	RS	*Record Separator* (séparateur d'enregistrement)
31	US	*Unit Separator* (séparateur d'unité)

32	Space	52	4	72	H	92	\	112	p
33	!	53	5	73	I	93	]	113	q
34	"	54	6	74	J	94	^	114	r
35	#	55	7	75	K	95	_	115	s
36	$	56	8	76	L	96	`	116	t
37	%	57	9	77	M	97	a	117	u
38	&	58	:	78	N	98	b	118	v
39	'	59	;	79	O	99	c	119	w
40	(	60	<	80	P	100	d	120	x
41	)	61	=	81	Q	101	e	121	y
42	*	62	>	82	R	102	f	122	z
43	+	63	?	83	S	103	g	123	{
44	,	64	@	84	T	104	h	124	|
45	-	65	A	85	U	105	i	125	}
46	.	66	B	86	V	106	j	126	~
47	/	67	C	87	W	107	k	127	DEL
48	0	68	D	88	X	108	l		
49	1	69	E	89	Y	109	m		
50	2	70	F	90	Z	110	n		
51	3	71	G	91	[	111	o		