INFO-H-100 - Informatique

Séance d'exercices 4
Introduction à Python
Chaînes de caractères et itérations.

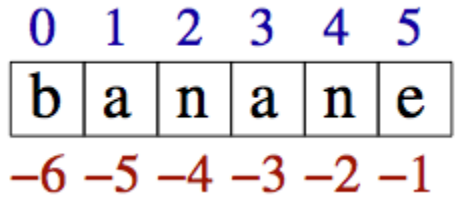
Université libre de Bruxelles
École polytechnique de Bruxelles

2013-2014

Chaîne de caractères

En Python, une chaîne de caractères est une séquence de caractères. Une séquence est une collection ordonnée.

Une chaîne de taille n est indicée de 0 à $n-1$ et de -1 à $-n$.



On peut accéder à chaque caractère à l'aide des crochets.

```
>>> word = 'banane'
>>> word[1]
'a'
>>> word[-2]
'n'
>>> type(word[3]) #un caractere est une chaine de 1 caractere
<type 'str'>
>>> word[6]
'IndexError: string index out of range
```

Tranches de chaînes

Un segment d'une chaîne est appelé une tranche (*slice*). On peut accéder à une tranche à l'aide de l'opérateur `[n:m]` où :

- `n` est l'indice du premier caractère de la tranche (inclus)
- `m` est l'indice du dernier caractère de la tranche (exclu)

Si `n` est omis, la tranche commence à l'indice 0 et si `m` est omis, la tranche se termine à la fin de la chaîne.

```
>>> word = 'banane'
>>> word[1:4]
'ana'
>>> word[:5]
'banan'
>>> word[3:]
'ane'
>>> word[:]    #copie toute la chaine
'banane'
```

Opérations sur les chaînes

```
>>> word = 'banane'
>>> len(word)          #longueur
6
>>> word.upper()       #copie la chaine en majuscules
'BANANE'
>>> word.find('n')     #indice de la premiere apparition de n
2
>>> 'BANANE' == word.upper()
True
>>> 'avoir' < 'etre' #comparaison (ordre ascii)
True
>>> 'a' in 'banane'  #vrai si 'a' est dans 'banane'
True
>>> word.islower()     #vrai si la chaine est en minuscule
True
```

Consultez la documentation pour trouver d'autres opérations.

Une chaîne est une séquence immuable.

En Python, une chaîne est immuable (*immutable*), c'est-à-dire que l'on ne peut pas la modifier.

```
>>> message = "bienvenue"
>>> message[0] = 'B'
TypeError: 'str' object does not support item assignment
```

Solution : créer une nouvelle chaîne de caractères :

```
>>> message = 'B' + message[1:]
>>> print(message)
Bienvenue
```

Table ASCII

En informatique, un caractère est représenté par un entier qu'on appelle son code :

```
>>> ord('a')
97
>>> chr(65)
'A'
```

Arithmétique :

```
>>> def lower_to_uppercase(c):
        dec = ord('A') - ord('a')
        return chr(ord(c)+dec)

>>> lower_to_uppercase('e'):
'E'
>>> ord('9') - ord('0')
9
```

Comparaison :

```
>>> 'avoir' < 'etre'
True
>>> 'avoir' < 'Etre'
False
```

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Itérations : boucle `for`

La structure itérative `for` permet de parcourir une séquence.

```
for variable in sequence:
    intructions
```

Signification : pour chaque valeur dans la séquence, exécuter les instructions.

```
>>> string = 'spam'
>>> for character in string:
      print(character)

s
p
a
m
```

```
>>> for i in range(1,5):
...     print(i, end=" ")
...
1 2 3 4
```

```
>>> string = 'spam'
>>> for i in range(len(string)):
      print(string[i])

s
p
a
m
```

```
>>> s = ''
>>> for l in 'ham':
...     s += lower_to_uppercase(l)
...
>>> print(s)
HAM
```

Créer et importer un module

En Python, un module est un fichier qui contient du code (variables, fonctions, classes) que l'on peut importer.

Exemple :

- Module `circle.py`

```
import math

def circonference(radius):
  return 2 * math.pi * radius

def area(radius):
  return math.pi * radius ** 2
```

- Un programme Python (dans le même répertoire) :

```
import circle

print(circle.area(2))
```

```
from circle import area

print(area(2))
```

Exercices