Les bases du Python 3

A.1 Interface

En lançant l'application IDLE (Python GUI) une fenêtre d'exécution *Python Shell* s'ouvre. Vous pouvez y tester des lignes de commande et c'est là que les résultats de vos programmes s'afficheront.

En chargeant un programme (menu Fichier) ou en créant un nouveau document, une fenêtre d'édition apparaît. C'est dans celle-ci que vous pourrez taper vos programmes. **Ces derniers doivent être sauvegardés avec l'extension ".py"** pour être reconnus par l'éditeur et profiter de la coloration syntaxique.

A.2 Commentaires

Les commentaires doivent être précédés du symbole #. La fin de ligne à partir de ce symbole est alors ignorée.

A.3 Structure générale

La structure d'un programme écrit en *Python* doit ressembler à ceci :

structure type en Python 3

```
# ALGORITHME      ...
# ROLE   ...
# ENTREES         ...
# SORTIES         ...
# VARIABLES       noms
: types
# DEBUT
...
instructions      # commentaire
# ===== titre
instructions      # commentaire
...
# FIN
```

A.4 Affectation

L'affectation de *NomVariable* se fait grâce au symbole = suivant le schéma : *NomVariable = Valeur*

A.5 Chaînes de caractères

Doit être placée entre ' (et si une apostrophe figure dans la chaîne on la notera \') ou encore entre ". Un retour à la ligne peut être provoqué en insérant \n dans la chaîne.

A.6 Saisie

input(*message*) affiche le contenu de *message*, attend une saisie au clavier validée par *entrée* et renvoie la saisie **sous forme d'une chaîne de caractères**.

Par exemple : *Nom = **input**('Quel est votre nom ? ')*

Si la valeur de la saisie doit être stockée dans une variable entière (respectivement réelle) on lui appliquera la fonction **int** (respectivement **float**).

Par exemple : *Age = **int**(**input**('Quel est votre âge ? '))*

A.7 Affichage

print(*Arg1* , *Arg2* , ...) affiche les valeurs des arguments (contenus de variables, chaînes ou nombres) en les séparant par un espace. En ajoutant en dernier argument **sep**=*chaîne* , c'est la chaîne spécifiée qui séparera l'affichage des arguments.

Par exemple **sep**='*' les séparera par une étoile , **sep**='' (chaîne vide) les accolera, **sep**='\t' insèrera une tabulation entre chaque et **sep**='\n' les écrira en passant à la ligne à chaque fois.

De la même façon, en ajoutant en dernier argument **end**=*chaîne* vous pouvez décider comment l'affichage doit se comporter à la fin de l'instruction **print** et, par exemple, l'empêcher d'aller à la ligne pour exécuter l'affichage suivant (en mettant la chaîne vide ou d'un espace).

Opérations et fonctions élémentaires en Python 3

B.1 Opérations et opérateurs de base

Symbole	Signification	Exemple	Valeur renvoyée
+	addition	**3 + 2**	5
−	soustraction	**3 − 2**	1
*	multiplication	**3 * 2**	6
/	division	**3/2**	1.5
**	exponentiation	**3 **2**	9
//	quotient entier (div)	**17//3**	5
%	reste entier (mod)	**17%3**	2
==	égalité	(7 == 2)	False
!=	non égalité	(7 != 2)	True
< ; >	comparaison stricte	(5 < 4)	False
>= ; <=	comparaison large	(2+2 <= 4)	True
+	concaténation	'abc' + 'def'	'abcdef'
and	ET logique	(3 > 0) and (1 < 0)	False
or	OU logique	(3 > 0) or (1 < 0)	True
not	NON logique	not(3>0)	False

B.2 Fonctions

Dans les exemples, la variable **X** est de type chaîne et est affectée par 'BonjouR'. Les fonctions annotées de (*) [1] ou de (**) [2] nécessite le chargement d'un module spécial en début de programme.

Fonction	Signification	Type	Exemple	Renvoie
round(*réel* , *entier*)	arrondi	réel	round(3.2572 , 2)	3.26
abs(*réel*)	valeur absolue	réel	abs(-3.25)	3.25
random()	nombre aléatoire dans **[0; 1[** (**)	réel		
floor(*réel*)	partie entière (*)	entier	floor(**−2.3**)	**−3**
exp(*réel*)	exponentielle (*)	réel	exp(**−1.2**)	**0.301...**
log(*réel positif*)	logarithme népérien (*)	réel	log(**0.9**)	**−0.105...**
sqrt(*réel positif ou nul*)	racine carrée (*)	réel	sqrt(**3.7**)	**1.923...**
int(*chaîne*)	conversion en nombre entier	entier	int('235')	235
float(*chaîne*)	conversion en nombre réel	réel	float('23.5')	23.5
str(*nombre*)	conversion en chaîne	chaîne	str(3/2)	'1.5'
chr(*code ASCII*)	code ASCII → caractère	chaîne	chr(97)	'a'
ord(*caractère*)	caractère → code ASCII	entier	ord('a')	97
len(*chaîne*)	nombre de caractères	entier	len(X)	7
chaîne.upper()	mise en majuscule	chaîne	X.upper()	'BONJOUR'
chaîne.lower()	mise en minuscule	chaîne	X.lower()	'bonjour'
chaîne[***n***]	caractère de rang ***n***	chaîne	X[**2**]	'n'
chaîne[***n*** **:** ***p***]	extrait du rang ***n*** au rang ***p*** − 1	chaîne	X[**1 : 5**]	'onjo'
chaîne[***n*** **:**]	extrait à partir du rang ***n***	chaîne	X[**4 :**]	'ouR'
chaîne[**:** ***p***]	les ***p*** premiers caractères	chaîne	X[**: 3**]	'Bon'
chaîne[**−*****n*** **:**]	les ***n*** derniers caractères	chaîne	X[**−2 :**]	'uR'
chaîne.find(*ch*)	position d'apparition de *ch*	entier	X.find('o')	1
			X.find('ur')	− 1
chaîne.find(*ch*,***n***)	cherche à partir du rang ***n***	entier	X.find('o',2)	4
chaîne.find(*ch*,***n***,***p***)	cherche du rang ***n*** au rang ***p*** − 1	entier	X.find('o',2,4)	− 1

1. mettre en début de programme l'instruction : from math import *
2. mettre en début de programme l'instruction : from random import *

Structures de contrôle élémentaires en Python 3

C.1 Traitement conditionnel simple

```
...
if ( condition ) :
    ...
    instructions à réaliser si la condition est remplie
    ...
# end if
...
```

C.2 Traitement conditionnel étendu

```
...
if ( condition ) :
    ...
    instructions à réaliser si la condition est remplie
    ...
else :
    ...
    instructions à réaliser si la condition n'est pas remplie
    ...
# end if
...
```

C.3 Boucle Tant que

```
...
while (condition ) :
    ...
    instructions à réaliser dans la boucle
    ...
# end while
...
```

C.4 Boucle Pour

```
...
for variable de comptage in range(valeur départ,valeur d'arrivée + 1) :
    ...
    instructions à faire pour chaque valeur
    ...
# end for variable de comptage
...
```

Attention :

Si vous voulez que votre variable ***i*** de comptage aille de 5 à 10, il faudra mettre : **for *i* in range(5,11)**.

Les tableaux en Python 3

D.1 Tableau à une dimension

D.1.1 Déclaration

Un tableau à une dimension peut être initialisé de différentes façons :

L'instruction $T = []$ crée une variable tableau T à 1 dimension de taille 0.

L'instruction $T = [2, 7, 0]$ crée un tableau T à 1 dimension de taille 3 donc les contenus des cellules sont initialisés respectivement à 2, 7 et 0.

Pour créer un tableau T à 1 dimension de taille n dont tous les contenus sont initialisés à x, les instructions suivantes sont équivalentes :

$T = [x, x, \cdots, x]$ $\qquad$ $T = [x] * n$ $\qquad$ $T = [x \text{ for } i \text{ in range}(0, n)]$

D.1.2 Accès aux données

La fonction $\mathbf{len}(T)$ renvoie la taille du tableau T (nombre de cellules)

Pour $0 \leqslant i \leqslant \text{len}(T) - 1$, l'accès au contenu de la cellule de rang i du tableau T se fait en utilisant la variable $T[i]$ (que ce soit pour utiliser ou changer son contenu).

D.1.3 Affectation dynamique

L'instruction $T.\mathbf{append}(x)$ ajoute une cellule au tableau T en y mettant la valeur x. La taille du tableau est donc augmentée de 1.

D.2 Tableau à deux dimensions

D.2.1 Déclaration

Un tableau T à deux dimensions de taille $n \times p$ (n lignes et p colonnes) se déclare comme un tableau à une dimension de taille n dont les composantes sont des tableaux à une dimension de taille p.

Un tableau à deux dimensions peut être initialisé en y affectant directement le contenu de ses cellules :

Par exemple l'instruction $T = [[2, 3], [5, 6], [8, 9]]$ crée le tableau T à deux dimensions, 3 lignes et 2 colonnes, possédant donc 6 cellules. Le contenu de la cellule qui se trouve à la ligne d'indice 2 et à la colonne d'indice 0 valant 8.

Pour initialiser un tableau T de n lignes et p colonnes en mettant la valeur x dans toutes ses cellules, on écrira l'instruction suivante : $T = [[x \text{ for } j \text{ in range}(0, p)] \text{ for } i \text{ in range}(0, n)]$

D.2.2 Accès aux données

Pour un tableau T à 2 dimensions, l'expression $\mathbf{len}(T)$ renvoie le nombre de lignes du tableau T.

$T[i]$ représente la ligne d'indice i. C'est un tableau à 1 dimension dont la taille est le nombre de colonnes de T. L'expression $\mathbf{len}(T[0])$ renvoie donc le nombre de colonnes du tableau T.

La variable $T[i][j]$ représente la cellule de la ligne d'indice i et de la colonne d'indice j.

D.2.3 Affectation dynamique

L'affectation dynamique (un "append") sur un tableau à 2 dimensions est risquée !

En effet, si vous voulez lui ajouter une ligne, il faudra que l'ajout transmette un tableau à 1 dimension dont la taille correspond au nombre de colonnes du tableau. Par exemple si le tableau T comporte p colonnes et qu'on veut lui ajouter une ligne de "x", on écrira :

$$T.\text{append}([x] * p)$$

Et si vous voulez lui ajouter une colonne, il faut cette fois ajouter une cellule par ligne et il faudra utiliser une boucle. Par exemple si T contient n lignes et qu'on veut lui ajouter une colonne de "x", on écrira :

$$\text{for } i \text{ in range}(0, n) : T[i].\text{append}(x)$$

Définition de procédure et fonction en Python 3

E.1 Procédure

def *NomProcédure*(*liste d'arguments éventuels séparés par des virgules*) **:**
    # type : procédure
    # arguments : *noms* et *types*
    # rôle : (...)
    # variables : *noms* et *types*
    suite d'instructions
Fin de procédure

E.2 Fonction

def *NomFonction*(*liste d'arguments éventuels séparés par des virgules*) **:**
    # type : fonction
    # arguments : *noms* et *types*
    # rôle : (...)
    # valeur renvoyée : *type*
    # variables : *noms* et *types*
    suite d'instructions
    return *NomDeVariable* ou *ValeurCalculée*
Fin de fonction

Codes Ascii de 32 à 127

Décimal	Hexadécimal	Binaire	Caractère	Décimal	Hexadécimal	Binaire	Caractère
32	20	00100000	espace	80	50	01010000	P
33	21	00100001	!	81	51	01010001	Q
34	22	00100010	"	82	52	01010010	R
35	23	00100011	#	83	53	01010011	S
36	24	00100100	$	84	54	01010100	T
37	25	00100101	%	85	55	01010101	U
38	26	00100110	&	86	56	01010110	V
39	27	00100111	’	87	57	01010111	W
40	28	00101000	(	88	58	01011000	X
41	29	00101001	)	89	59	01011001	Y
42	2A	00101010	*	90	5A	01011010	Z
43	2B	00101011	+	91	5B	01011011	[
44	2C	00101100	,	92	5C	01011100	\
45	2D	00101101	-	93	5D	01011101	]
46	2E	00101110	.	94	5E	01011110	^
47	2F	00101111	/	95	5F	01011111	_
48	30	00110000	0	96	60	01100000	‘
49	31	00110001	1	97	61	01100001	a
50	32	00110010	2	98	62	01100010	b
51	33	00110011	3	99	63	01100011	c
52	34	00110100	4	100	64	01100100	d
53	35	00110101	5	101	65	01100101	e
54	36	00110110	6	102	66	01100110	f
55	37	00110111	7	103	67	01100111	g
56	38	00111000	8	104	68	01101000	h
57	39	00111001	9	105	69	01101001	i
58	3A	00111010	:	106	6A	01101010	j
59	3B	00111011	;	107	6B	01101011	k
60	3C	00111100	<	108	6C	01101100	l
61	3D	00111101	=	109	6D	01101101	m
62	3E	00111110	>	110	6E	01101110	n
63	3F	00111111	?	111	6F	01101111	o
64	40	01000000	@	112	70	01110000	p
65	41	01000001	A	113	71	01110001	q
66	42	01000010	B	114	72	01110010	r
67	43	01000011	C	115	73	01110011	s
68	44	01000100	D	116	74	01110100	t
69	45	01000101	E	117	75	01110101	u
70	46	01000110	F	118	76	01110110	v
71	47	01000111	G	119	77	01110111	w
72	48	01001000	H	120	78	01111000	x
73	49	01001001	I	121	79	01111001	y
74	4A	01001010	J	122	7A	01111010	z
75	4B	01001011	K	123	7B	01111011	{
76	4C	01001100	L	124	7C	01111100	|
77	4D	01001101	M	125	7D	01111101	}
78	4E	01001110	N	126	7E	01111110	~
79	4F	01001111	O	127	7F	01111111	delete