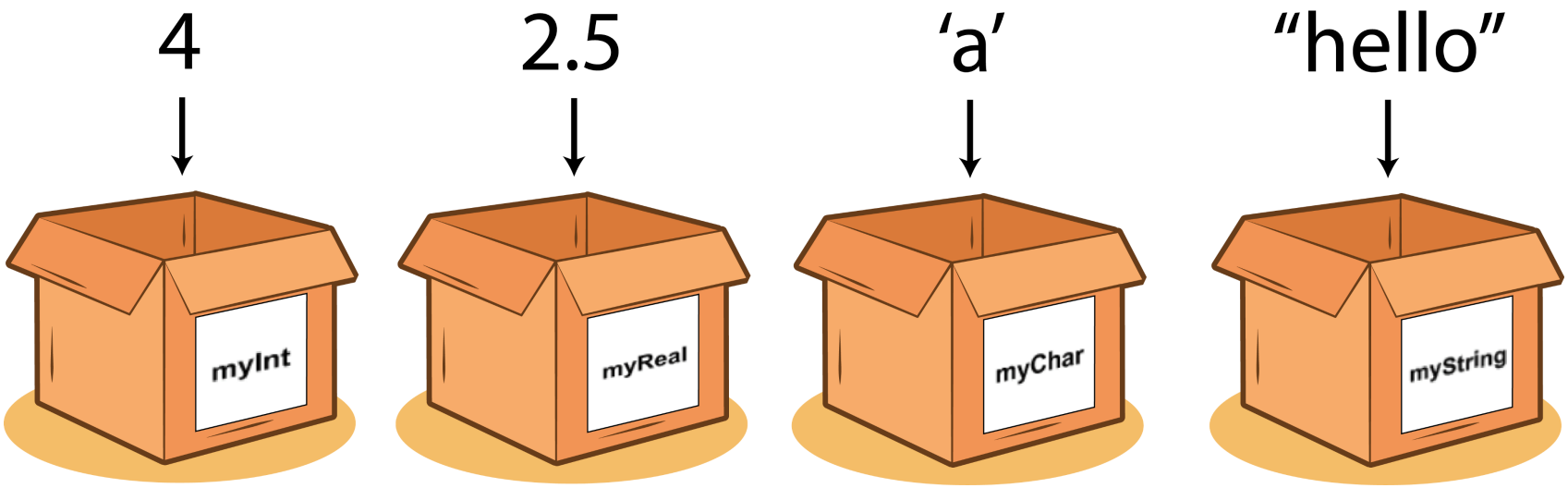
Introduction to Python

python™

Lucas MORLET
BSB : M1

VARIABLES



ASSIGN A VALUE TO A VARIABLE

```
# Create a number variable
number = 8.4
# Create a character variable
char = 'A'
# Create a text variable
string = "test"
# Print the content of the variables
print ( number, char, string )
```

Results

8.4 A test

INTEGERS (INT)

```
# Assign values
n = 3
m = 2
# Print the sum
print ( n + m )
# Print the type of the variable
print ( type(n) )
```

Results

```
5
< class 'int' >
```

DECIMAL NUMBER (FLOAT)

```
# Create a float
x = 3.14
# Print it and its type
print ( x, type(x) )
# Mix integers to get a float
n = 3
m = 2
print ( n/m, type(n/m) )
```

Results

3.14 < class 'float' >
1.5 < class 'float' >

BOOLEANS (BOOL)

```
# Create two booleans
a = True
b = False
# Print a bool and its type
print ( a, type(a) )
# Some operations on booleans
print ( not a, a and b, a or b )
```

Result

True <class 'bool'>
False False True

STRING (STR)

```
# Create two strings
s = ”Hello”
t = ”World !”
# Print it
print ( s, t )
print ( type(s) )
```

Result

```
Hello World !
<class ’str’>
```

DYNAMIC TYPE

```
# Create an integer
x = 4
print ( x, end = " " )
# Convert it to float
x = 8.6
print ( x, end = " " )
# Finally, its a string !
x = "I'm a string"
print ( x )
```

Result

4 8.6 I'm string

STANDARD STRUCTURE

```
x = 2
if ( x == 3 ) :
    print ( "That's true" )
else :
    print ( "That's false" )
```

Results

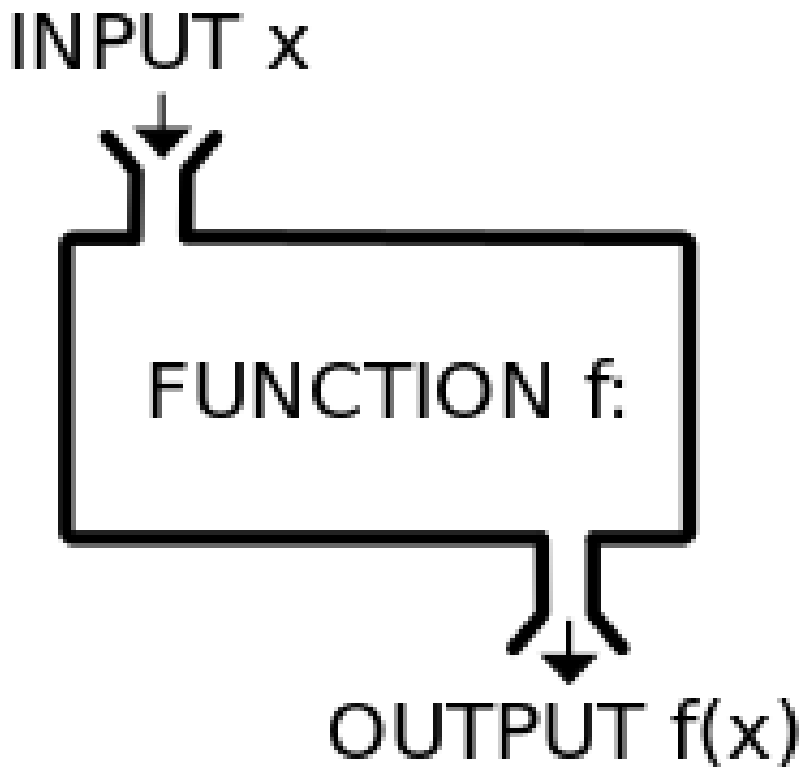
That's false

MULTIPLE ENDS STRUCTURE

```
x = 2
if ( x == 3 ) :
    print ( "That's true" )
elif ( x == 2 ) :
    print ( "That's false then true" )
else :
    print ( "That's false" )
```

Results

That's false then true


INPUT x
FUNCTION f:
OUTPUT f(x)

SOME BASIC FUNCTION

```
# Function definition
def my_function ( ) :
    print ( "I'm inside the function"  )

# Call this function
print ( "I will run some function" )
my_function ( )
```

Results

I will run some function
I'm inside the function

PASS A PARAMETER TO A FUNCTION

```
# Define the function
def two_times ( n ) :
    print ( "2 x ", n , "=", 2*n )

# Test it with different parameters
two_times ( 3 )
two_times ( 4 )
```

Results

```
2 x 3 = 6
2 x 4 = 8
```

PASS SEVERAL PARAMETERS TO A FUNCTION

```
# Define the function
def sum ( a, b ) :
    print ( a, "+", b, "=", a+b )

# Test it with different parameters
sum ( 2, 3 )
sum ( 1.4, 2.1 )
```

Results

2 + 3 = 5
1.4 + 2.1 = 3.5

RETURN

```
# Define the function
def square ( x ) :
    return x*x

# Test it with different parameters
y = square ( 2 )
print ( y )
y = square ( 2.5 )
print ( y )
```

Results

```
4
6.25
```

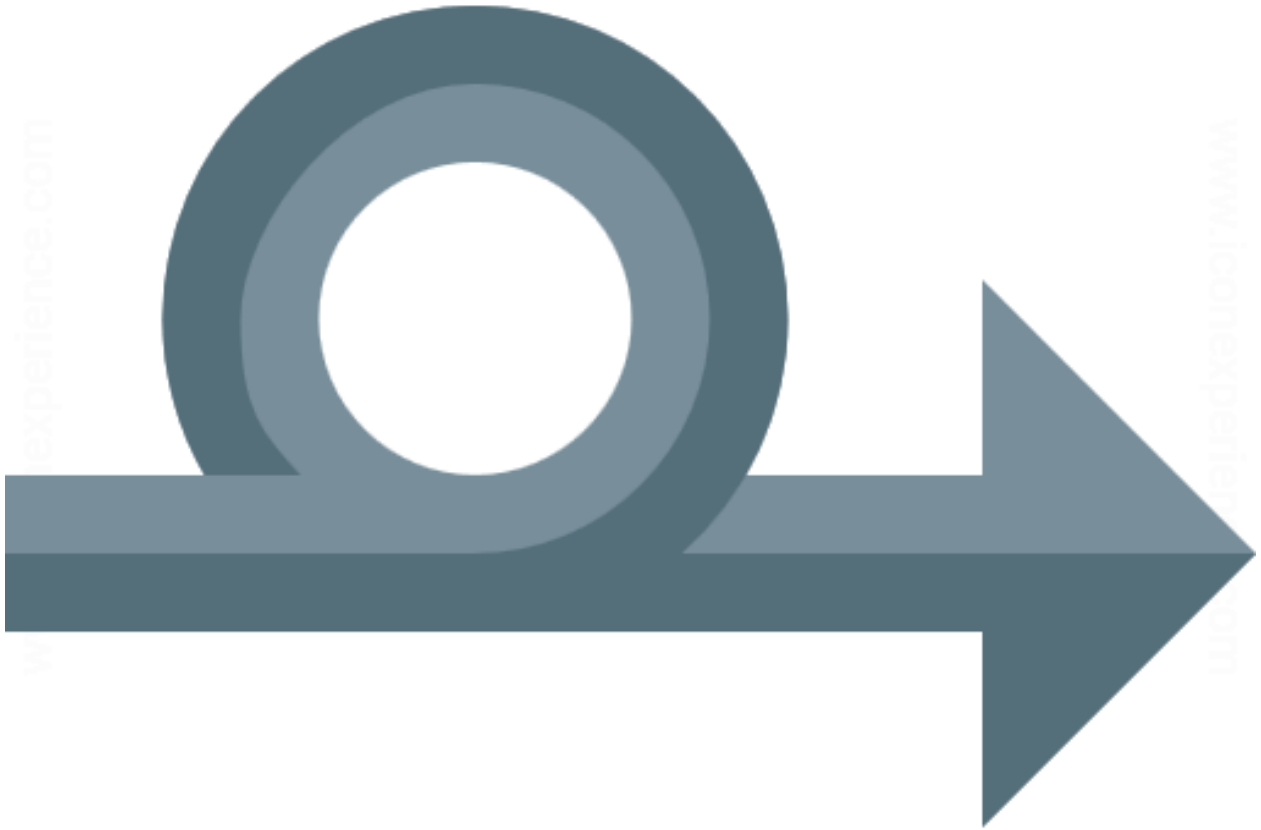
VARIABLE SCOPE

```
def some_function ( x ) :
    x = 2 * x
    print ( "During the function : ", x )

x = 3
print ( Before the function, x )
some_function ( x )
print ( After the function, x )
```

Results

Avant la fonction : 3
Pendant la fonction : 6
Apres la fonction : 3

Loops



For-loop through a set

```
# For a specif set of values
for i in [ 2, 3, 5, 7, 11 ] :
    print ( i, end = " " )
```

Results

2 3 5 7 11

For-loop through a range

```
# From 0 to n-1
for i in range ( 6 ) :
    print ( i, end = " " )
print ( " " )
```

Results

0 1 2 3 4 5

```
# From m to n-1
for i in range ( 2, 8 ) :
    print ( i, end = " " )
print ( " " )
```

Results

2 3 4 5 6 7

```
# With a specific step between numbers
for i in range ( 0, 8, 2 ) :
    print ( i, end = " " )
print ( " " )
```

Results

0 2 4 6

While loop

```
n = 18
while ( n > 1 ) :
    print ( n, end = " " )
    n = n / 2
print ( " " )
```

Results

18, 9.0, 4.5, 2.25, 1.125

Nested loops

```
for i in range ( 1, 3 ) :
    for j in [ 'a', 'b', 'c' ] :
        print ( i, j, end = ", " )
print ( " " )
```

Results

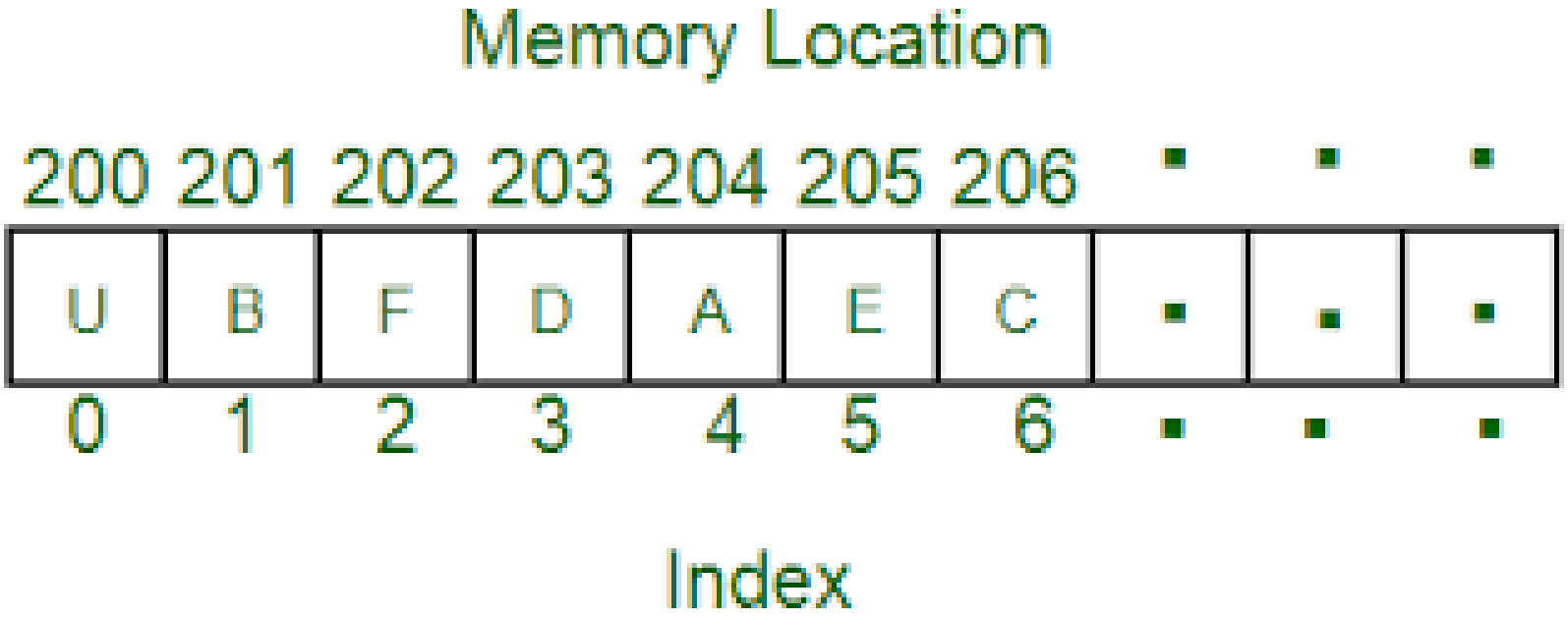
1 a, 1 b, 1 c, 2 a, 2 b, 2 c

Linked loops

```
for i in range ( 1, 5 ) :
    for j in range ( 2 * i ) :
        print ( ".", end = "" )
    print ( " ", end = "" )
```

Results

.. ..
(2 dots then 4 dots then 6 dots and to finish 8 dots)


Memory Location
200 201 202 203 204 205 206
U B F D A E C
0 1 2 3 4 5 6
Index

WHAT IS IT ?

- An array is a data structure
- It is composed of a fixed number of same type elements
- Each element contains a value
- Each element should be accessed by its index

Be careful !

In Python, there is no **array**, only **list**
Lists are similar to array, without the fixed number constraint

In a first part, you will use Python's list as array
Later, you will learn how to exploit list functionalities

CREATE AN ARRAY

```
# Define an array composed of ten zeros
tab = [0] * 10
# Display the array infos
print ( "The array is :" , tab )
print ( "Its length is :" , len(tab) )
```

Results

The array is : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Its length is : 10

READ THE VALUE OF AN ELEMENT

```
# Define the array [ a, b, c, d, e ]
tab = [ 'a', 'b', 'c', 'd', 'e' ]
# Display the array infos
print ( "The array is :" , tab )
print ( "The element number 1 is :" , tab[1] )
```

Results

```
The array is : ['a', 'b', 'c', 'd', 'e']
The element number 1 is : b
```

STORE A VALUE INSIDE AN ELEMENT

```
# Define an array composed of five zeros
tab = [0] * 5
print ( "Before modification :" , tab )
# Store a 2 inside element number 2
tab[2] = 2
print ( "After modification :" , tab )
```

Results

Before modification : [0, 0, 0, 0, 0]
After modification : [0, 0, 2, 0, 0]

MODIFY ELEMENTS OF AN ARRAY

```
# Define an array
tab = [ 2, 3, 5, 7, 11 ]
print ( "Original array :" , tab )
# Multiply element number 3 by 10
tab[3] = 10 * tab[3]
print ( "After multiplication :" , tab )
# Sum two elements of the array
tab[2] = tab[1] + tab[2]
print ( "After sum :" , tab )
```

Results

Original array : [2, 3, 5, 7, 11]
After multiplication : [2, 3, 5, 70, 11]
After sum : [2, 3, 8, 70, 11]

ITERATE THROUGH AN ARRAY

```
# Define an array composed of ten zeros
tab = [0] * 10
# Define every element by iterating through the array
for i in range ( len ( tab ) ) :
    tab [ i ] = i

print ( "After iteration :" , tab )
```

Results

After iteration : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

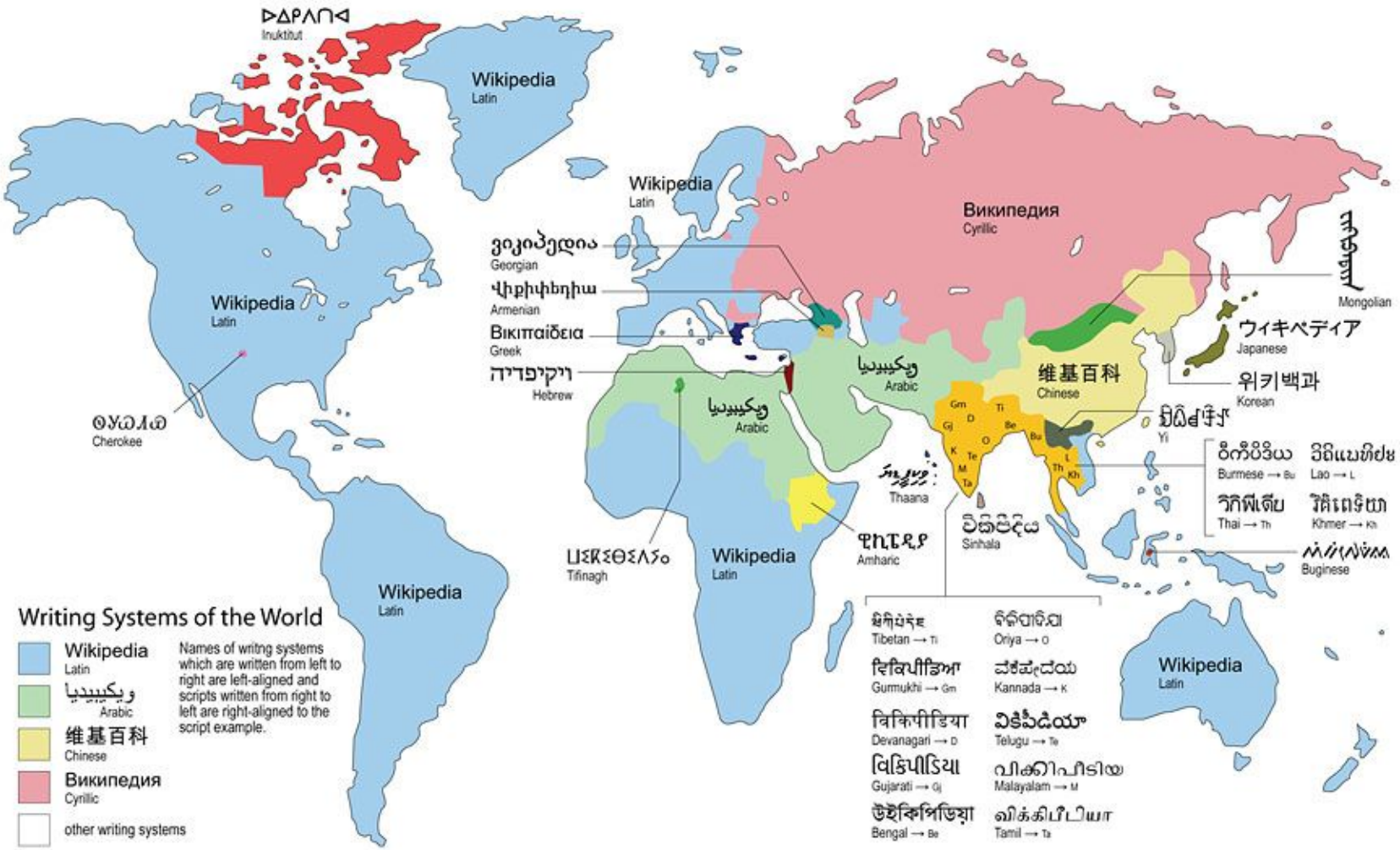
MODIFY AN ARRAY BY ITERATION

```
# Previous slide code
tab = [0] * 10
for i in range ( len ( tab ) ) :
    tab [ i ] = i

# Square every element of the array
for i in range ( len ( tab ) ) :
    tab [ i ] = tab [ i ] * tab [ i ]

print ( "Finally :" , tab )
```

Results

Finally : [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

TEXT - CHARACTERS



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[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	[NEGATIVE 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	[ENG OF 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	[SUBSTITUTE]	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	[DEL]

American Standard Code for Information Interchange

Keep in mind that every character is a number

TAKE A CLOSER LOOK AT ASCII TABLE

Characters groups

- [48 ; 57] : digits (0...9)
- [65 ; 90] : Upper case letters (A...Z)
- [97 ; 122] : Lower case letters (a...z)
- [0 ; 31] : Control characters (*e.g.* new line)

COMPARISON BETWEEN CHARACTERS

```
# Comparison between lower cases
e = 'e'
t = 't'
print ( e < t )

# Comparison between lower and upper cases
a = 'a'
Z = 'Z'
print ( a < Z )
```

Results

True
False

EXAMPLE

```
# Function that tests if a character is an upper case letter
def is_upper_case ( char ) :
    if ( 'A' <= char and char <= 'Z' ) :
        return True
    else :
        return False

# Test this function
print ( is_upper_case ('A'), is_upper_case ('n'), is_upper_case ('9') )
```

Results

```
True False False
```

CONVERSION BETWEEN INTEGER AND CHARACTER

```
# Convert a character to an integer
char = 'N'
number = ord ( char )
print ( char, "->", number )

# Convert an integer to a character
number = 93
char = chr ( number )
print ( number, "->", char )
```

Results

```
N -> 78
93 -> ]
```

CHARACTER SHIFTING

```
# Pick a character and a shift
char = 'A'
shift = 4
# Apply the shift
number = ord ( char )
res = chr ( number + shift )
# Display the result
print ( char, "+", shift, "=", res )
```

Results

A + 4 = E

```
>>> print("Hello World!")
Hello World!
>>>
```

COMMON FUNCTIONS ON STRINGS

```
string = "Hello World !"
print ( "The string length is :", len ( string ) )
print ( "There is :", string.count('o'), " 'o' in this string" )
print ( "The first 'o' is in position number :", string.find('o') )
print ( "There is :", string.count(' ')+1, " words in this string" )
print ( string.upper() )
print ( string.lower() )
print ( string.replace( 'o', 'a' ) )
```

Results

length : 13 There is 2 'o' First position : 4 3 words
HELLO WORLD ! hello world ! Hella Warld !

CONCATENATION AND MERGE

```
# Concatenate two strings
str_A = "Hello "
str_B = "World !"
string = str_A + str_B
print ( string )

# Split strings
substrings = string.split ( 'o' )
print ( substrings )
```

Results

```
Hello World !
[ 'Hell', ' W', 'rld !' ]
```

EXAMPLE

```
# Function that converts date from DD/MM/YYYY to YYYY-MM-DD
def date_from_french_to_ISO ( date ) :
    substrings = date.split('/')
    res = substrings[2]
    res += "-"
    res += substrings[1]
    res += "-"
    res += substrings[0]
    return res

# Test the function
date = date_from_french_to_ISO ( "15/10/2025" )
print ( date ) # 2025-10-15
```

CONVERSION BETWEEN STRING AND LIST

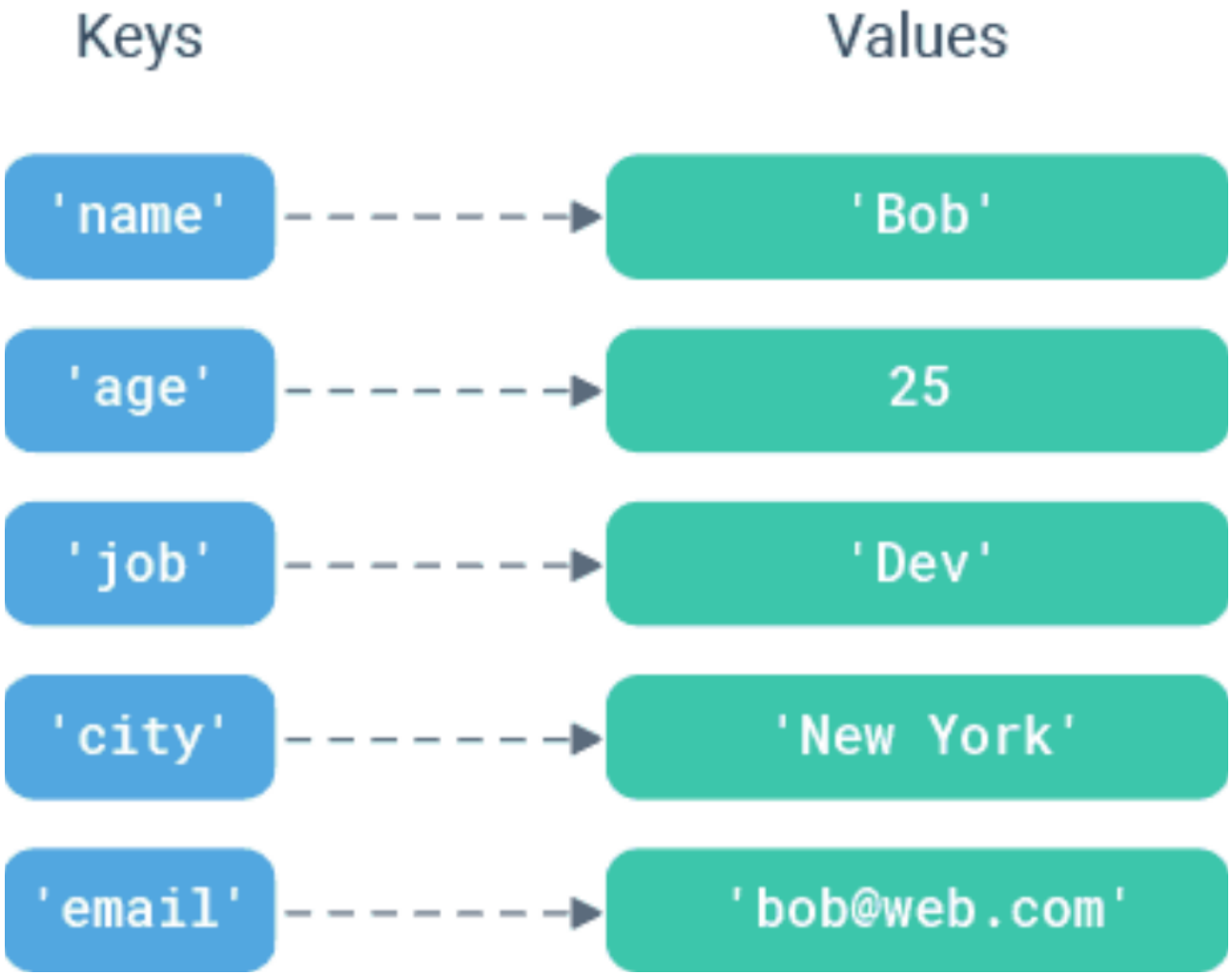
```
# String to list
string = ”test”
array = list(string)
print ( array )

# List to string
new_string = ””.join(array)
print ( new_string )
```

Results

```
[ ’t’, ’e’, ’s’, ’t’ ]
test
```

Dictionaries

CONCEPT


Keys
Values
'name'
'Bob'
'age'
25
'job'
'Dev'
'city'
'New York'
'email'
'bob@web.com'

CREATE A DICTIONARY

```
alice = {
    first_name : "Alice",
    last_name : "Liddell",
    birth_year : 2003,
    group : "B1",
    grades : [ 10.5, 12, 9.5 ]
}
print ( "Dictionary :", alice )
print ( "Alice's last name :", alice["last_name"] )
```

Results

Dictionary : {'first_name' : 'Alice', 'last_name' : 'Liddell', 'birth_year' : 2003, 'group' : 'B1', 'grades' : [10.5, 12, 9.5]}
Alice's last name : Liddell

MODIFY A DICTIONARY

```
# Modify an existing value
alice [ "group" ] = "B2"

# Add a new key/value pair
alice [ "major" ] = "Data Science"

# Remove an existing key/value pair
alice.pop("birth_year") print( "Dictionary :" alice)
```

Results

Dictionary : {'first_name' : 'Alice', 'last_name' : 'Liddell', ~~'birth_year' : 2003,~~

'group' : 'B2', 'grades' : [10.5, 12, 9.5], 'major' : 'Data Science'}

KEYS AND VALUES

```
# Read keys and values of a dictionary
keys   = alice.keys()
values = alice.values()
print ( keys )
print ( values )
```

Results

dict_keys(['first_name', 'last_name', 'group', 'grades', 'major'])

dict_values(['Alice', 'Liddell', 'B2', [10.5, 12, 9.5],
'Data Science'])

ITERATE THROUGH A DICTIONARY

```
items = alice.items()
print ( "**********" )
for key, value in items :
    print ( key, "->", value )
print ( "**********" )
```

Results

```
**********
first_name -> Alice
last_name -> Liddell
group -> B2
grades -> [10.5, 12, 9.5]
major -> Data Science
**********
```

FLASH FORWARD : DATAFRAME

	A	B	C	D
1	first_name	last_name	group	major
2	Alice	Liddell	B2	Data Science
3	Bob	Bobson	B1	Web
4	Carol	Luis	A1	Data Science
5	Dave	Davidson	A1	Embedded systems
6	Eve	Gatherer	A2	Web

```
students = {
    first_name : [ "Alice", "Bob", "Carol", "Dave", "Eve" ],
    last_name : [ "Liddell", "Bobson", "Luis", "Davidson", "Gatherer" ],
    group : [ "B2", "B1", "A1", "A1", "A2" ],
    major : ["Data Science","Web","Data Science","Embedded systems","Web"]
}
```