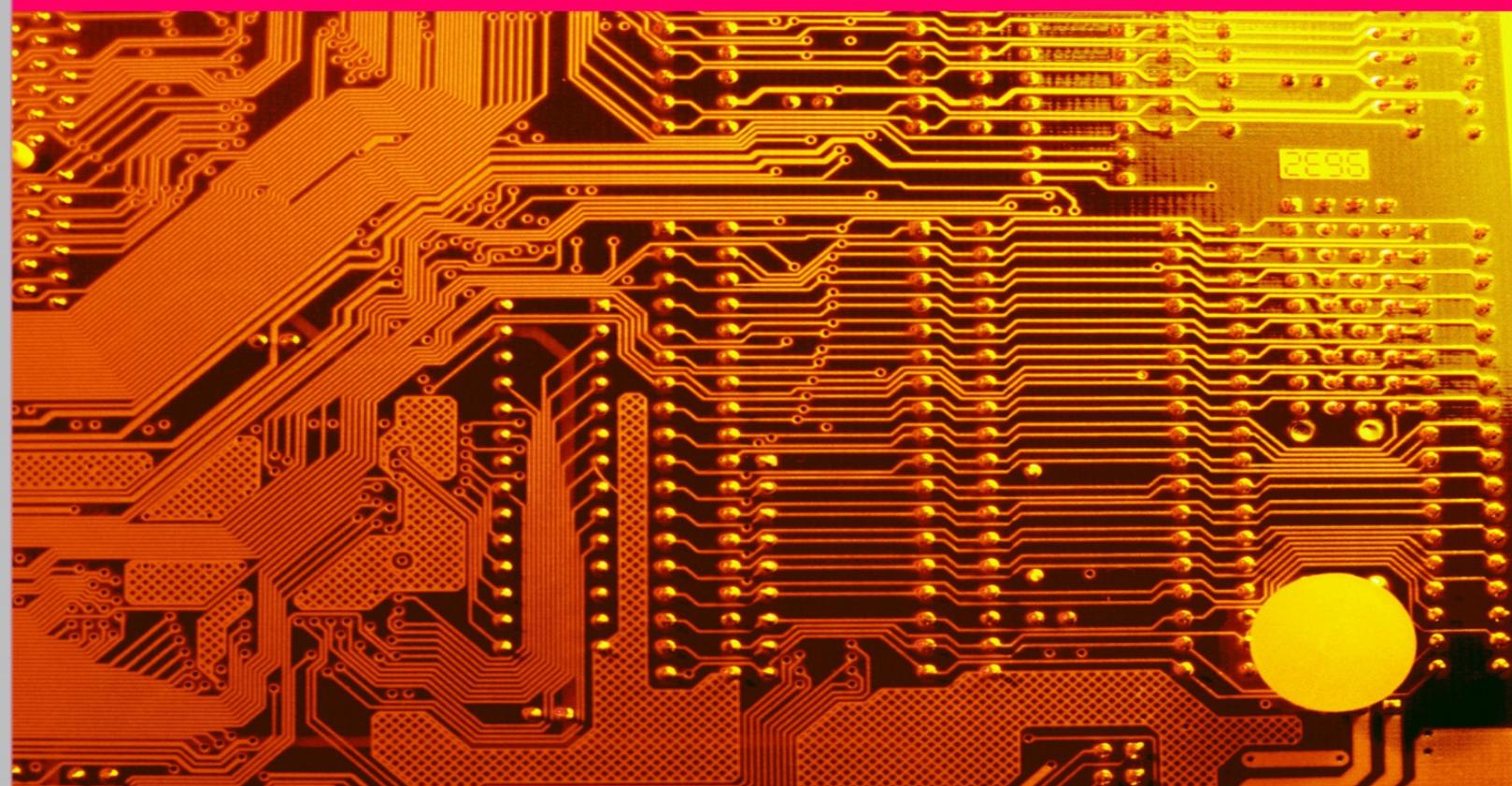


Python 3 : 400 exercises and solutions for beginners

ASSAD PATEL



Python : 400 Exercises for total beginner

Foreword

The main issue with all book dealing with Python is poorly-leveled.

Example :

Course: $1 + 1 = 2$

Exam : Calculate $\iiint x + y + z \, dx \, dy \, dz$

As in Python

Course

```
>>> 1 + 1
```

Exam :

Give the integral code by defining a function with n arguments and its block will contain three recursive functions with a list as return.

I've tried to give as much details as I could and a lot of schematics.

Actually, this book is for everyone.

Table des matières

- [**>>> 1. Addition operator `+`**](#)
- [**>>> 2. Minus operator `-`**](#)
 - [**>>> 2.1. Expressions with operator `+` and operator `-`**](#)
- [**>>> 3. Multiplication operator `*`**](#)
- [**>>> 4. Division operator `/` and floating numbers**](#)
 - [**>>> 4.1. Back to expression**](#)
 - [**>>> 4.1.1. A number is an expression**](#)
 - [**>>> 4.1.2. Expressions is a mix of operands and operators**](#)
- [**>>> 5. Modulo `%`, `//` and `**`**](#)
 - [**>>> 5.1. A deep dive into modulo `%`**](#)
- [**>>> 6. Operators priority and parentheses**](#)
- [**>>> 7. Value assignment**](#)
 - [**>>> 7.1. Print variable value with `idle`**](#)
 - [**>>> 7.2. `NameError`**](#)
- [**>>> 8. Variables assignments**](#)
 - [**>>> 8.1. Variables re-assignment**](#)
 - [**>>> 8.1.1. Swap variables values**](#)
- [**>>> 9. Boolean operator `not`**](#)
- [**>>> 10. Boolean operator `and`**](#)
 - [**>>> 10.1. `not` and `and`**](#)
- [**>>> 11. Boolean operator `or`**](#)
 - [**>>> 11.1. `not`, `and` and `or`**](#)
- [**>>> 12. Equality operators `==` and `!=`**](#)
- [**>>> 13. Operators `>` and `<`**](#)
 - [**>>> 13.1. Morgan laws**](#)
- [**>>> 14. Strings in Python**](#)
 - [**>>> 14.1. String Expressions**](#)
 - [**>>> 14.2. `" "` \(double quote\)**](#)
 - [**>>> 14.3. `+` concatenation**](#)

[>>> 14.3.1 `TypeError` : unmatch types](#)

[>>> 14.4. Operator `*`](#)

[>>> 14.5. Expressions with `+` and `*`](#)

[>>> 15. Functions in Python](#)

[>>> 15.1. built-in functions](#)

[>>> 15.1.1. `print\(\)`](#)

[>>> 15.1.2. `print\(expression\)`](#)

[>>> 15.1.3. `print\(variable\)`](#)

[>>> 15.1.4. built-in function `len\(\)`](#)

[>>> 15.1.5. Expressions with function call](#)

[>>> 15.1.6. Built-in function `type\(\)`](#)

[>>> 15.2. Defining our own functions](#)

[>>> 15.2.1. keyword `def` and indented block](#)

[>>> 15.2.2. Defining one-argument-function](#)

[>>> 16. If](#)

[>>> 16.1. `else`](#)

[>>> 16.2. Condition and range](#)

[>>> 17. Read with function `input\(\)`](#)

[>>> 17.1. Function `int\(\)`](#)

[>>> 17.2. Read with `input\(\)`](#)

[>>> 17.2.1. `input\(\)`, `int\(\)` et `if`](#)

[>>> 17.3. `elif`](#)

[>>> 17.3.1. `if \(expression\) :`](#)

[>>> 18. First module : `math`](#)

[>>> 18.1. Import with `import`](#)

[>>> 18.2. Inside a module](#)

[>>> 18.2.1. Function `dir\(\)`](#)

[>>> 18.2.2. Access to modules attributes with the dot `.` operator](#)

[>>> 18.2.3. Some maths](#)

[>>> 19. Looping with `while`](#)

[>>> 19.1. Sequences of numbers with `while\(\)`](#)

[>>> 19.2. I love maths and while\(\)](#)

[>>> 19.2.1.while and sequences](#)

[>>> 19.2.2. Boucle while et affichage de tables](#)

[>>> 19.2.3. Make choice with while\(\)](#)

[>>> 19.3. Recursives sequences](#)

[>>> 19.3.1. \$u_n + 1 = u_n\$](#)

[>>> 19.4. Nesting while and if](#)

[>>> 19.4.1. + / -](#)

[>>> 19.5. Inception : while nested in a while](#)

[>>> 19.5.1. Variables i and j are independant](#)

[>>> 19.5.2. Variables i and j are linked](#)

[>>> 19.5.3. Handle while in while](#)

[>>> 19.5.4.while inception](#)

[>>> 19.6. Maths with while\(\)](#)

[>>> 19.7. Break and continue](#)

[>>> 19.7.1. Break](#)

[>>> 19.7.3. continue](#)

[>>> 20. Lists](#)

[>>> 20.1.List elements](#)

[>>> 20.2. Manipulate list elements](#)

[>>> 20.3. while loop\(\) and list](#)

[>>> 20.3.1. List maximum](#)

[>>> 20.3.2. List Minimum](#)

[>>> 20.3.3. Some stats with lists](#)

[>>> 20.3.4. Find an element in a list](#)

[>>> 20.3.5. Find the list index](#)

[>>> 21. Back to String](#)

[>>> 21.1. Concatenation operator +](#)

[>>> 21.2. while\(\) and string](#)

[>>> 21.3. Copier une chaîne à l'aide de l'opérateur de concaténation](#)

[>>> 22. Lists and Strings](#)

[>>>22.1. Copying lists with method `append\(\)`](#)
[>>>22.2. Addition lists with method `append\(\)`](#)
[>>>22.3. Manipulates lists with `append\(\)`](#)
[>>> 22.4. Some lists methods](#)
 [>>> 22.4.1. `list.clear\(\)`](#)
 [>>> 22.4.2. `list.copy\(\)`](#)
 [>>> 22.4.3. IN PLACE METHODS](#)
[>>>23. Sort Algorithm](#)
 [>>> 23.1. Swap lists elements](#)
 [>>> 23.2. Sorting list ascending order](#)
[>>>24. Functions in Python : Part 2](#)
 [>>> 24.1. Defining functions with a return](#)
 [>>> 24.1.1. `module.fonction\(\)`](#)
 [>>> 24.1.2. Function with several arguments](#)
 [>>> 24.1.2. Fonction without any arguments](#)
 [>>> 24.1.3. Fonctions avec du if / else](#)
 [>>> 24.2. Define a function without return](#)
 [>>> 24.2.1. `pass` and `None`](#)
[>>> 25. Loop with `for ... in`](#)
 [>>> 25.1. `For ... in` with lists](#)
 [>>> 25.2. `For... in` on strings](#)
[>>> 26. Nested lists](#)
 [>>> 26.1. Skills on nested lists](#)
 [>>> 26.1. Some maths](#)
 [>>> 26.1.2. Rows](#)
 [>>> 26.2. Skills on nested lists : part 2](#)
[>>> 27. Methods : Part 1](#)
 [>>> 27.1. String Methods](#)
 [>>> 27.1.1. `split\(\)`](#)
 [>>> 21.1.2. `str.format\(\)`](#)
 [>>> 21.1.3. `f-string`](#)

[>>> 27.2.Methos on list object](#)

[>>> 27.2.1. sort \(\)](#)

[>>> 27.2.2. copy \(\)](#)

[>>> 27.2.3. join \(\)](#)

[>>> 28.Introduction to files](#)

[>>> 28.1.Files 101](#)

[>>> 28.1.1.Open\(\)](#)

[>>> 28.1.2. write\(\) and close \(\)](#)

[>>> 28.2.read\(\) and 'r'](#)

[>>> 28.2.1. Argument 'a'](#)

[>>> 28.3.Files and for](#)

[>>> 28.3.1. Copy a file](#)

[>>> 29.Tuples](#)

[>>> 29.1.Tuples 101](#)

[>>> 29.2.keyword in](#)

[>>> 29.3.Slicing \[: \]](#)

[>>> 29.3.1.Slicing again \[: \]](#)

[>>> 29.3.2.Slicing and negative indexes](#)

[>>> 29.3.3.Slicing and step \[: : \]](#)

[>>> 29.4. Operations on sequences](#)

[>>> 29.5.Tuple unpacking](#)

[>>> 29.5.1. Tuple unpacking : notation *](#)

[>>> 29.6.Back to for](#)

[>>> 29.6.1. range \(\)](#)

[>>> 29.6.2.list compréhension and for](#)

[>>> 29.6.3.break](#)

[>>> 29.6.4.for and list indexes](#)

[>>> 30.Dictionaries and Sets](#)

[>>> 30.1.Dictionnaires](#)

[>>> 30.1.1.Couple \(key,value\)](#)

[>>> 30.1.2. Skills on dictionaries](#)

[>>> 30.2.Sets](#)
[>>> 30.2.1.Sets 101](#)
[>>>30.2.2.built-in fuction set\(\)](#)
[>>> 30.2.3.Methods add \(\) and update\(\)](#)
[>>> 30.2.4.Set Theory 101](#)

[>>>31.Exceptions](#)
[>>> 31.1.There's a glitch](#)
[>>> 31.1.1.Syntax Errors](#)
[>>> 31.1.2.Semantic Errors](#)
[>>> 31.2.try ... except](#)

[>>> 32.Namespaces and Objects space](#)
[>>> 32.1. Namespaces and Objects space for numerical type](#)
[>>> 32.2.List object dans indexes](#)
[>>> 32.2.1.A list is a mutable objecy](#)
[>>> 32.2.3.Lists Operations](#)
[>>> 32.2.4.Nested list](#)

[>>> 33.Classes](#)
[>>> 33.1.Classes 101](#)
[>>> 33.1.1.How to create a class object](#)
[>>> 33.1.2.Operations on classes](#)
[>>> 33.1.3.Instantiation](#)
[>>> 33.2.Methods again](#)
[>>> 33.2.1.The instance is the argument](#)
[>>> 33.2.2.Method __init__](#)

[>>> 34.Fonctions:partie 2](#)
[>>> 34.1.Function arguments](#)
[>>> 34.1.1.Non mutable arguments](#)
[>>> 34.1.2.Mutables arguments](#)
[>>> 34.2.Default arguments and keyword arguments](#)
[>>> 34.2.1.Default arguments](#)
[>>> 34.2.2.keyword arguments](#)

[>>> 35.Variables scopes](#)

[>>> 35.1.Local variable](#)

[>>> 35.1.1.local variable and global variable](#)

[>>> 35.2.Function within a function](#)

[>>> 36.Modules : part 2](#)

[>>> 36.1.Module namespace](#)

[>>> 36.1.1.Fonctions defined within a module](#)

>>> 1. Addition operator +

« *please add* » a math teacher

>>> Exercise 1

For each code line

1° give the *expression*

2° what are *the operands* and operators ?

3° Give the value after being evaluated

```
>>> 1 + 2
```

```
>>> 1 + 2 + 3
```

```
>>> 1 + 2 + 3 + 4
```

Solution

Expression is : 1+2

Addition operator : +

Operands (or value) are : 1 et 2

```
>>> 1 + 2
```

```
3
```

Expression : 1 + 2 + 3

Addition operator : +

Operands : 1, 2 et 3

```
>>> 1 + 2 + 3
```

```
6
```

Expression : 1 + 2 + 3 + 4

Addition operator : +

Operands: 1, 2, 3 et 4

```
>>> 1+2+3+4
```

```
10
```

>>> Be Careful

One expression in Python and it gives you a unique value

```
>>> Expression
```

```
value
```

>>> Exercise 2

Evaluate the following expressions

```
>>> 2+3
```

```
>>> 5+4+9
```

```
>>> 1+10+11+12
```

>>> Exercise 3

Evaluate the following expressions

```
>>> 100 + 1259
```

```
>>> 3 + 99 + 402
```

```
>>> 1 + 5555 +66666
```

Solutions

>>> Exercise 2

```
>>> 2 + 3
```

```
5
```

```
>>> 5 + 4 + 9
```

```
18
```

```
>>> 1 + 10 + 11 + 12
```

```
34
```

>>> Exercise 3

```
>>> 100 + 1259
```

```
1359
```

```
>>> 3 + 99 + 402  
504  
>>> 1 + 5555 +66666  
72222
```

>>> 2. Minus operator -

« Sa vie était une constance soustraction d'elle-même » Clarice Lispector

>>> Exercice 4

Evaluate the code

```
>>> 1 - 1  
>>> 1 - 2 - 3  
>>> 50 - 44
```

Corrigé

L'expression is : $1 - 1$, operands 1 and 1, operator: -

```
>>> 1-1  
0
```

L'expression is $1-2-3$, operands are : 1,2 and 3 and operator : -

```
>>> 1-2-3  
-4
```

L'expression is $50-44$, operands are 50 and 44 and operator : -

```
>>> 50 - 44  
6
```

>>> Exercice 5

Evaluate the code

```
>>> 4-5  
>>> 1 - 18 -7  
>>> 100 - 78
```

>>> Exercice 6

Evaluate the code

```
>>> 1 - 5 -6  
>>> 2 - 97 -102  
>>> 500-400
```

Corrigés

>>> Exercice 5

```
>>> 4-5  
-1  
>>> 1 - 18 -7  
-24  
>>> 100 - 78  
22
```

>>> Exercice 6

```
>>> 1 - 5 - 6  
-10  
>>> 2 - 97 -102  
-197  
>>> 500 - 400  
100
```

>>> 2.1. Expressions with operator **+** and operator **-**

>>> **Exercice 7**

Evaluate the code

```
>>> 1 + 2 - 3  
>>> 1 + 1 - 2
```

Corrigé

Expression is $1 + 2 - 3$

It combines two operators + and – and it gives you a unique value as the output

```
>>> 1 + 2 - 3
```

```
0
```

Expression is $1 + 1 - 2$

```
>>> 1 + 2 - 2
```

```
0
```

>>> Exercice 8

Evaluate the code

```
>>> 4 - 6 + 1
```

```
>>> 1 - 2 - 10
```

Corrigé

Expression is $4 - 6 + 1$

```
>>> 4 - 6 + 1
```

```
-1
```

```
>>> 1 - 2 - 10
```

```
-11
```

>>> 3. Multiplication operator *

« Multiplions les pains » Anonyme

>>> Exercice 9

Evaluate the following expressions

```
>>> 3 * 1
```

```
>>> 1 * 2 * 3
```

```
>>> 2 * 4 * 16 * 256
```

Corrigé

Expression is : 3*1 operands : 3 and 1 and operator : *

```
>>> 3 * 1
```

3

Expression is : 1*2*3 operands :1, 2 et 3 and operator : *

```
>>> 1 * 2 * 3
```

6

Expression is : 2*4*16*256 operands: 2,4,16 et 256 and operator : *

```
>>> 2* 4 * 16 * 256
```

32768

>>> Exercice 10

Evaluate the following expressions

```
>>> 1 * 1
```

```
>>> 8 * 7 *6
```

```
>>> 10 * 9 * 8 * 7
```

>>> Exercice 11

Evaluate the following expressions

```
>>> 4 * 3
```

```
>>> 7 * 89 * 750
```

```
>>> 1 * 14 * 157 * 7456
```

Solutions

>>> Exercise 10

```
>>> 1 * 1
```

1

```
>>> 8 * 7 * 6
```

336

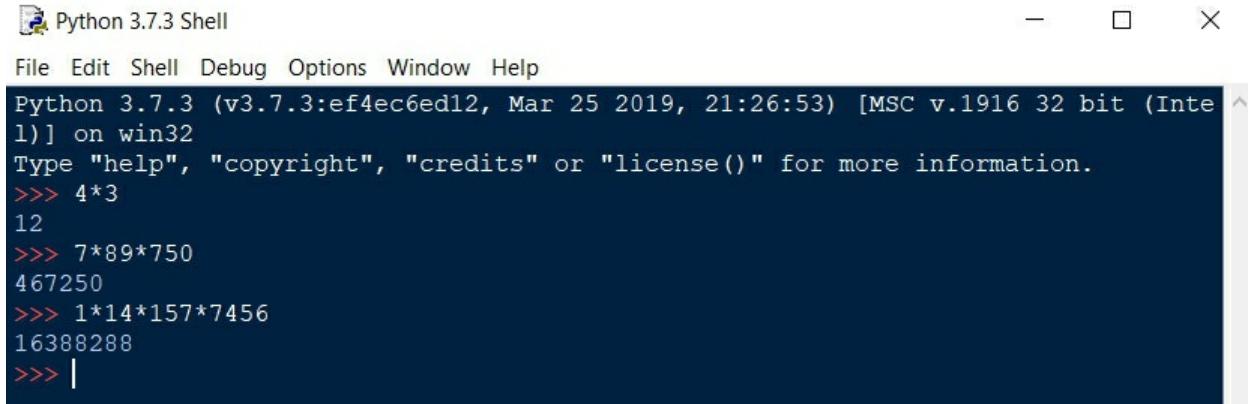
```
>>> 10 * 9 * 8 * 7
```

5040

>>> Exercice 11

```
>>> 4 * 3
```

```
12
>>> 7 * 89 * 750
467250
>>> 1 * 14 * 157 * 7456
16388288
```

A screenshot of the Python 3.7.3 Shell window. The title bar says "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window shows the Python interpreter's prompt (>>>) followed by the code and its output. The code is:
>>> 4*3
12
>>> 7*89*750
467250
>>> 1*14*157*7456
16388288
>>> |

>>> 4. Division operator / and floating numbers

« Divisons pour mieux régner » Anonyme

>>> Exercice 12

Please evaluate :

```
>>> 4 / 2
>>> 10 / 2
>>> 1 / 3
```

Expression is : 4 / 2 values are the integers 4 et 2 and the operator : / .

The output value would be 2.0 which is a floating number (float)

```
>>> 4 / 2
2.0
>>> 10 / 2
5.0
```

>>> 1 / 3

A retenir

integers numbers are type int

Floating numbers are type float

the operator / always returns a float

```
>>> int / int
```

float

>>> Exercise 13

Evaluate

```
>>> 2 / 3
```

>>> 4 / 5

>>> 6 / 8

>>> Exercise 14

Evaluate

```
>>> 57 / 49
```

>>> 126 / 789

```
>>> 1 / 10000
```

Solutions

Exercice 11

>>> 2 / 3

0.6666666666666666

>>> 4 / 5

0.8

>>> 6 / 8

0.75

Exercice 12

```
>>> 57 / 49  
1.163265306122449  
>>> 126 / 789  
0.1596958174904943  
>>> 1 / 10000  
0.0001
```

>>> 4.1. Back to expression

>>> 4.1.1. A number is an expression

Exercise 13

What happens ?

```
>>> 1  
>>> 2.5  
>>> 0.0
```

Solutions

1 is an expression (a number is an expression) his evaluation through `idle` would give the value 1 (green).

```
>>> 1  
1
```

As same as 1 is an expression, 2.5 is also an expression and his evaluation is 2.5 (green)

```
>>> 2.5  
2.5
```

Same (notice that 0 is an integer and 0.0 a float)

```
>>> 0.0  
0.0
```

Tips

```
>>> Expression
```

```
Value
```

>>> 4.1.2. Expressions is a mix of operands and operators

>>> **Exercise 14**

Evaluate

```
>>> 1 + 2 + 3
```

```
>>> 1 - 3 +2
```

```
>> 4 / 2
```

```
>>> 5 /
```

Solution exercise 14

Operands : 1 2 3

Operators : + +

Mix (expression) : 1 + 2 + 3

```
>>> 1 + 2 + 3
```

```
6
```

Operands : 1 3 2

Operators : - +

Mix (expression) : 1 -3 + 2

```
>>> 1 - 3 + 2
```

```
0
```

Operands : 4 2

Operators : /

Mix (expression) : 4 / 2

```
>> 4 / 2
```

```
2.0
```

Operands : 5

Operators : /

Mix (expression) : 5 /

```
>>> 5 /
```

```
SyntaxError : invalid syntax
```

There are valid expression which gives a unique value, and invalid expression that gives an Error (SyntaxError here).

Tips :

A valid expression gives a value

```
>>> Expression
```

```
Value
```

But an invalid expression gives an error

```
>>> invalid_expression
```

```
SyntaxError : invalid syntax
```

>>> **Exercise 15**

evaluate

```
>>> 1 +
```

```
>>> 2 + 2 -
```

```
>>> 1 * 4 * 3
```

Solutions exercise 15

```
>>> 1 +
```

```
SyntaxError : invalid syntax
```

```
>>> 2 + 2 -
```

```
SyntaxError : invalid syntax
```

```
>>> 1 * 4 * 3
```

```
12
```

Invalid expression : 1 +

Invalid expression : 2 + 2 -

>>> 5. Modulo %, // and **

« Modulons les fréquences et augmentons la puissance » moi

>>> Exercice 16

1°a) $\frac{17}{4}$?

b) give r and q (from the euclidian division) ?

c) evaluate

```
>>> 17 % 4
```

```
>>> 17 // 4
```

2° Evaluate

```
>>> 2 * 2
```

```
>>> 2 ** 2
```

```
>>> 4 ** 3
```

Corrigé

1° a) $17 = 4 * 4 + 1$

b) $q = 4$ and $r = 1$

c) operands : 17 4

operator : %

valid expression : 17 % 4

```
>>> 17 % 4
```

```
1
```

operands : 17 4

operator : //

valid expression : 17 // 4

```
>>> 17 // 4
```

```
4
```

2° On a

operands : 2 2

operator : *

valid expression : 2 * 2

```
>>> 2 * 2
```

4

operands : 2 2

operator : **

valid expression : 2 ** 2

```
>>> 2 ** 2
```

4

operands : 4 3

operator : **

valid expression : 4 ** 3

```
>>> 4 ** 3
```

64

>>> **Exercise 17**

Evaluate

```
>>> 15 % 4
```

```
>>> 7 ** 3
```

```
>>> 15 // 4
```

>>> **Exercice 18**

$1^\circ \frac{78}{33} ?$

2° Evaluate

```
>>> 78 % 33
```

3° Evaluate ?

```
>>> 78 // 33
```

Solutions

>>> Exercice 17

```
>>> 15 % 4
```

3

```
>>> 7**3
```

343

```
>>> 15 // 4
```

3

>>> Exercice 18

1° $78 = 33*2 + 12$ with $0 < 12 < 33$ q is 2 and r is 12

2° r

```
>>> 78 % 33
```

12

3° q

```
>>> 78 // 33
```

>>> 5.1. A deep dive into modulo %

>>> **Exercise 19**

1° a) $\frac{8}{2}$

b) Evaluate

```
>>> 8 % 2
```

2° a) $\frac{9065}{1332}$?

b) $1332 \mid 9065$?

Solution exercise 19

1°a) $8 = 4 \times 2 + 0$ a $0 \leq 2$ q = 4 and r = 0

b)

```
>>> 8 % 2
```

0

2 | 8

2° a)

```
>>> 9065 // 1332
```

6

```
>>> 9065 % 1332
```

1073

On a $9065 = 1332 * 6 + 1073$

b) $r = 1073$ and $q = 1332$. The r is not equals to zero so 1332 does not divide 9065.

Tips

```
>>> a % b
```

0

We got : $b | a$

Also for *even* number.

```
>>> a % 2
```

0

>>> Exercise 20

1° $100 | 77$?

2° $2 | 4$?

3° $47 | 34$?

Solutions exercice 20

1°

```
>>> 77 % 100
```

77

100 does not divide 77

2°

```
>>> 4 % 2
```

0

0 so 2 divides 4

3°

```
>>> 34 % 47
```

```
34
```

47 does not divide 34

>>> 6. Operators priority and parentheses

« des opérateurs sont prioritaires mais les parenthèses changent tout »
anonymes

>>> Exercise 21

Evaluate the following expressions

```
>>> 4 + 3 * 12
```

```
>>> 4 - 3 / 12
```

```
>>> 2 ** 3 * 3
```

```
>>> 1 + 3 - 4 + 5
```

Solutions exercise 21

Expression : $4 + 3 * 12$

Operators : + *

Priority operator : *

Sub-expression evaluated first: $3 * 12$

Sub-expression evaluated : $4 + 36$

```
>>> 4 + 3 * 12
```

```
# 4 + 36
```

```
40
```

Expression : $4 - 3 / 12$

Operators : - /

Priority operator : /

Sub-expression evaluated first: $3 / 12$

Sub-value : 0.25

Sub-expression evaluated : $4 - 0.25$

```
>>> 4 - 3 / 12
```

```
# 4 - 0.25
```

```
3.75
```

Operators : `**` *

Priority operator : `**`

Sub-expression evaluated first: $2^{**}3$

Sub-value : 8

Sub-expression evaluated : $8 * 3$

```
>>> 2 ** 3 * 3
```

```
# 8 * 3
```

```
24
```

Operators : + - +

Priority operator : same precedence for + and - (evaluation left to right)

Sub-expression evaluated first: $1 + 3$

Sub-value : 4

Sub-expression evaluated : $4 + 5$

Sub-value : 9

Final sub-expression : $4 - 9$

```
>>> 1 + 3 - 4 + 5
```

```
# 4 - 4 + 5
```

```
# 4 - 9
```

```
- 5
```

Tips

Operators precedence

1. Exponentiation `**`
2. Négation `-`
3. Multiplication `*` Division `/` Modulo `%` et Division entière `//`

4. Addition + et Soustraction -

If an expression has two operators with same precedence, sub-expressions will be evaluated from left to right

Si dans une expression les deux opérateurs ont la même priorité alors l'évaluation se fait de gauche à droite

```
>>> sub_expression1 sub_expression2  
# value1 sub_expression2  
# value1 value2  
value
```

is for comment

>>> Exercice 22

Evaluate and give the sub-expressions

```
>>> 4 * 5 + 3 - 4  
>>> 2 + 4 - 5 / 8  
>>> 3 * 4 ** 3
```

>>> Exercice 23

Evaluate and give the sub-expressions

```
>>> 4 + 5 + 1 - 9  
>>> 2 * 3 + 9 / 5  
>>> 5 // 4 + 5 % 4
```

Solutions

```
>>> 4 * 5 + 3 - 4  
# 4 * 5 + 3 - 5  
# 20 + 3 - 5  
# 20 - 1  
19
```

and

```
>>> 2 + 4 - 5 / 8
```

```
# 2 + 4 - 5 / 8
```

```
# 2 + 4 - 0.625
```

```
# 6 - 0.625
```

```
5.375
```

finally

```
>>> 3 * 4 ** 3
```

```
# 3 * 4 ** 3
```

```
# 3 * 64
```

```
192
```

exercise 23

```
>>> 4 + 5 + 1 - 9
```

```
# 4 + 5 + 1 - 9
```

```
# 9 + 1 - 9
```

```
# 10 - 9
```

```
# 1
```

```
1
```

and

```
>>> 2 * 3 + 9 / 5
```

```
# 2 * 3 + 9 / 5
```

```
# 6 + 9 / 5
```

```
# 6 + 1.8
```

```
# 7.8
```

```
7.8
```

finally

```
>>> 5 // 4 + 5 % 4
```

```
# 5 // 4 + 5 % 4
```

```
# 1 + 5 % 4
```

```
# 1 + 1
```

```
2
```

>>> Exercice 24

Evaluez les expressions suivantes

```
>>> (4 + 3) * 12
```

```
>>> (4 - 3) / 12
```

```
>>> 2 ** (3 * 3)
```

```
>>> 1 + (3 - 4) + 5
```

Corrigé exercice 24

Parentheses give the precedence

```
>>> (4+3) * 12
```

```
# (4+3) * 12
```

```
# 7 * 12
```

```
84
```

On a

```
>>> (4-3) / 12
```

```
# (4-3) / 12
```

```
# 1 / 12
```

```
0.083333333333333
```

also

```
>>> 2 ** (3*3)
```

```
# 2 ** (3*3)
```

```
# 2 ** 9
```

```
512
```

finally

```
>>> 1 + (3-4) + 5
```

```
# 1 + (3-4) + 5
```

```
# 1 - 1 + 5
```

```
# 0 + 5
```

```
5
```

>>> Exercise 25

Evaluate the following expressions

```
>>> 4 * (5 + 3) - 4
```

```
>>> 2 + (4 - 5) / 8
```

```
>>> (3 * 4) ** 3
```

>>> Exercise 26

Evaluate the following expressions

```
>>> 4 + 5 + (1 - 9)
```

```
>>> 2 * (3 + 9) / 5
```

```
>>> 5 // (4 + 5) % 4
```

Solutions

```
>>> 4 * (5 + 3) - 4
```

Parentheses subexpression first : (5+3)

Sub-value : 8

Sub-expression : $4 * 8 - 4$

Sub-value : 32

Sub-expression : $32 - 4$

```
>>> 4 * (5 + 3) - 4
```

```
# 4 * (5 +3) - 4
```

```
# 4 * 8 - 4
```

```
# 32 - 4
```

```
28
```

also

```
>>> 2 + (4 - 5) / 8
```

```
# 2 + (4 - 5) / 8
```

```
# 2 - 1 / 8
```

```
# 2 - 0.125
```

```
1.875
```

finally

```
>>> (3 * 4) ** 3
```

```
# (3 * 4) ** 3
```

```
# 12 ** 3
```

```
1728
```

Solutions

```
>>> 4 + 5 + (1 - 9)
```

```
# 4 + 5 + (1 - 9)
```

```
# 4 + 5 - 8
```

```
# 9 - 8
```

```
1
```

also

```
>>> 2 * (3 + 9) / 5
```

```
# 2 * (3 + 9) / 5
```

```
# 2 * 12 / 5
```

```
# 24 / 5
```

```
4.8
```

finally

```
>>> 5 // (4 + 5) % 4
```

```
# 5 // (4 + 5) % 4
```

```
# 5 // 9 % 4
```

```
# 0 % 4
```

```
0
```

>>> 7. Value assignment

>>> Exercice 27

Evaluate and explain this code

```
>>> nombre = 12.0
>>> nombre
>>> calcul = 20 * 4.0
>>> calcul
```

Solutions exercise 27

1.Variable name : nombre

2.Variable value : 12.0 (float)

```
>>> nombre = 12.0
>>> nombre
12.0
```

1.Variable name : calcul

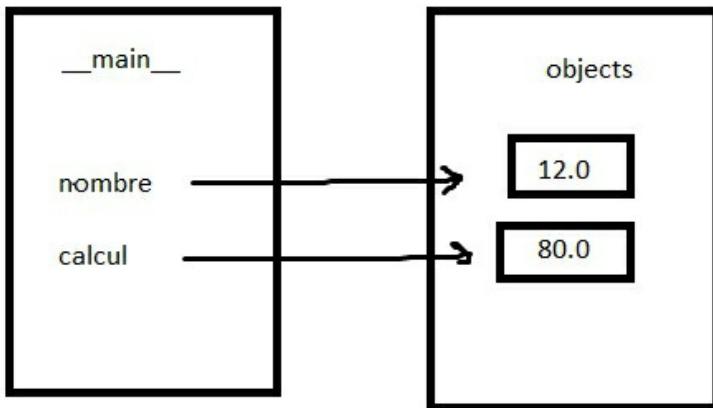
2.Variable value :

a) expression : $20 * 4.0$

b) value : 80.0

variable value : 80.0 (float)

```
>>> calcul = 20 * 4.0
# calcul = 20 * 40.0
# calcul = 80.0
>>> calcul
80.0
```



Tips

A variable has :

- A name
- A value

To give(or to affect) a value to a variable :

```
>>> variable = value
```

We also can affect an expression (which will be evaluated)

```
>>> variable = expression
```

>>> 7.1. Print variable value with idle

>>> Exercise 28

What will be the output ?

>> nombre = 1 + 1'."/>

```

*Python 3.7.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> nombre = 1 + 1
>>> nombre

```

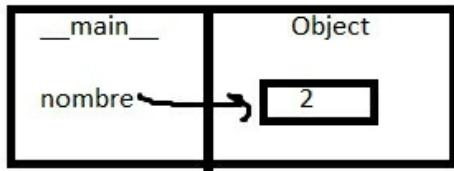
solutions exercise 28

```
>>> nombre = 1 + 1
# nombre = 1 + 1
```

```
# nombre = 2
```

```
>>> nombre
```

```
2
```



Tips

For seeing the variable's value with IDLE

```
>>> variable
```

```
Value
```

>>> Exercise 29

Explain this code

```
>>> va = 42
```

```
>>> vb = 12 / 5
```

```
>>> va
```

```
>>> vb
```

Solution exercise 29

Variables va and vb has values

We use idle to output these values

```
>>> va = 42
```

```
>>> vb = 12 / 5
```

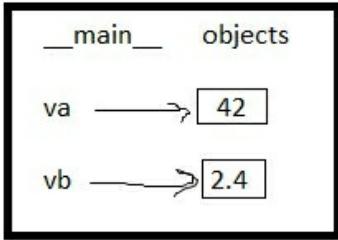
```
# vb = 2.4
```

```
>>> va
```

```
42
```

```
>>> vb
```

```
2.4
```



>>> 7.2. NameError

>>> Exercise 30

What's happen here ?

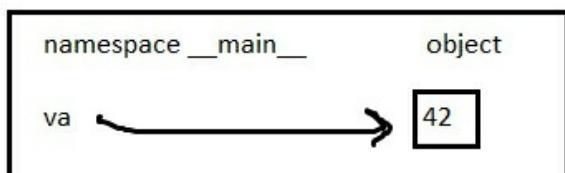
```
>>> va = 42
>>> vb
```

Solutions

On a

```
>>> va = 42
```

The namespace has only one variable name : va (value 42)



Then we ask for the value of the variable named `vb`. Or `vb` doesn't exist, we have an error message called `NameError`.

```
>>> vb
NameError : name 'vb' is not defined
```

>>> 8. Variables assignments

>>> Exercise 31

What does the following code do ?

```
>>> a = 1 + 2 * 4
```

```
>>> b = 1 + 1  
>>> c = a * b  
>>> c
```

Corrigé exercice 31

Expression : $1 + 2 * 4$

Assignment to : variable a

Value produced : 9

```
>>> a = 1 + 2 * 4  
# a = 1 + 2 * 4  
# a = 1 + 8  
# a = 9
```

Same for variable b

```
>>> b = 1 + 1  
# b = 2
```

Expression : $a * b$

Assignment to : variable c

Value produced : 18

```
>>> c = a * b  
# c = 9 * 2  
# c = 18  
>>> c  
18
```

A retenir

In expression operands can be variables.

When the expression will be evaluated, theses variables will be replaced by their values.

>>> **Exercise 32**

Describe the following code

```
>>> a = 2
>>> b = a * 2
>>> c = a + 1
>>> d = a + b + c + 1
```

Solutions exercise 32

Variable a

Value : 2

```
>>> a = 2
```

Variable b

Expression : a * 2

Value variable a : 2

Value b : 4

```
>>> b = a * 2
# b = 2 * 2
# b = 4
```

Variable c

Expression : a + 1

Value variable a : 2

Value c : 3

```
>>> c = a + 1
# c = 2 + 1
# c = 3
```

Variable d

Expression : a + b + c + 1

Value variable a : 2

Value variable b : 4

Value variable c : 3

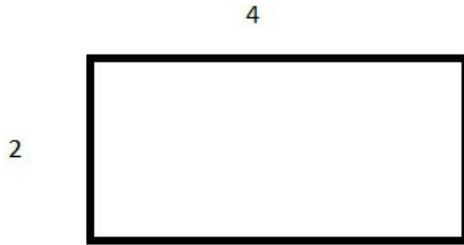
Value d : 10

```
>>> d = a + b + c + 1
```

```
# d = 2 + 4 + 3 + 1
```

```
# d = 10
```

>>> **Exercise 33**



1°

what's the area of this rectangle ?

2° a) Assign the value 4 to a variable called longueur

b) Assign the value 2 to a variable called largeur

3° Evaluate

```
>>> longueur * largeur
```

```
>>> aire = longueur * largeur
```

```
>>> aire
```

Soltuions exercise 33

1° Area = $4 * 2 = 8$ units

2° a)

```
>>> longueur = 4
```

b)

```
>>> largeur = 2
```

3°

```
>>> longueur * largeur
```

```
# 2 * 4
```

```
# 8
```

```
8
```

```
>>> aire = longueur * largeur
# aire = 2 * 4
# aire = 8
>>> aire
```

```
8
```

>>> Exercise 34 Tax

« i love tax »

- 1° a) Assign 100 to the variable `prixHT`
- b) assign the expression $20 / 100$ to the variable `taux_tva` (astuce : $\frac{20}{100}$)

2° Evaluate the following code

```
>>> tva = prixHT * taux_tva
>>> prixTTC = prixHT + tva
>>> prixTTC
```

Solutions exercise 34

1° a)

```
>>> prixHT = 100
```

b)

```
>>> taux_tva = 20 / 100
```

2°

```
>>> tva = prixHT * taux_tva
# tva = 100 * 0.2
# tva = 20.0
```

Then

```
>>> prixTTC = prixHT + tva
# prixTTC = 100 + 20.0
```

```
# prixTTC = 120.0
```

>>> 8.1. Variables re-assignment

>>> Exercise 35

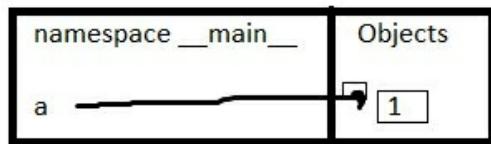
Explain the following code

```
>>> a = 1  
>>> a = 2  
>>> a
```

Solutions exercise 35

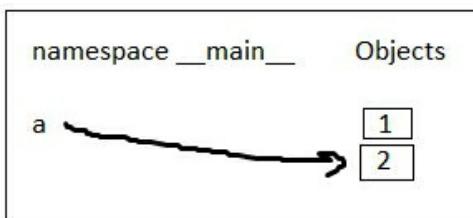
Value 1 is assigned to variable called a

```
>>> a = 1
```



```
>>> a = 2
```

Value 2 is assigned to variable a (or a refers to object 2)



Hint :

A variable has a value

Every variable's name is located in an area called : **namespace** (of the main program)

The value are located in another and separated area called : **Objects**

>>> Exercise 36

Explain this code

```
>>> a = 2
>>> b = 1 / 5
>>> a = a + b
>>> a
```

Solution exercise 36

On a

```
>>> a = 2
>>> b = 1 / 5
# b = 0.2
>>> a = a + b
# a= 2 + 0.2
# a = 2.2
>>> a
```

2.2

>>> 8.1.1. Swap variables values

>>> Exercise 37

1° Explain and evaluate the following code

```
>>> a = 1
>>> b = 2
>>> a = b
>>> b = a
>>> a
>>> b
```

2° Explain and evaluate the following code

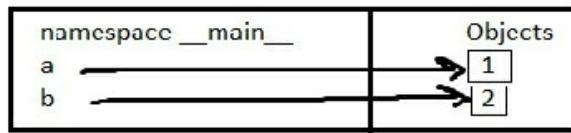
```
>>> a = 1
>>> b = 2
```

```
>>> c = a  
>>> a = b  
>>> b = c
```

Solutions exercise 37

1° First

```
>>> a = 1  
>>> b = 2
```



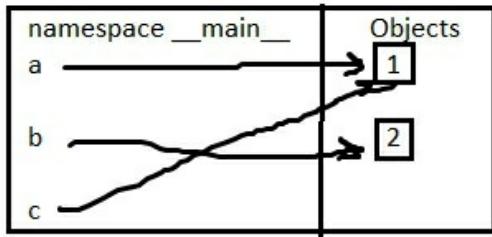
```
>>> a = b  
# a = 2  
>>> b = a  
# b = 2  
>>> a  
2  
>>> b  
2
```



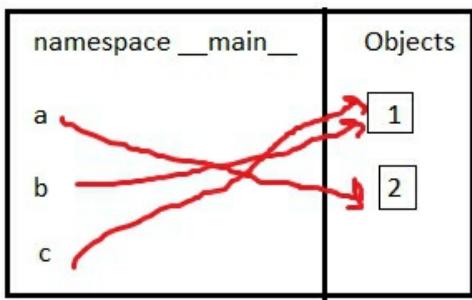
2° On a

```
>>> a = 1
```

```
>>> b = 2
>>> c = a
# c = 1
```



```
>>> a = b
# a = 2
>>> b = c
# b = 1
```



>>> Exercise 38

Explain the following code

```
>>> a = 1
>>> b = 2
>>> a, b = b, a
>>> a
>>> b
```

Solutions exercise 38

```
>>> a = 1
>>> b = 2
>>> a, b = b, a
# a = b
# a = 2
# b = a
# b = 1
>>> a
2
>>> b
1
```

Hint :

To swap variables values without using a temp variable

```
>>> var1 = value1
>>> var2 = value2
>>> var1, var2 = var2, var1
>>> var1
value2
>>> var2
value1
```

>>> 9. Boolean operator not

In memory of *Georges Boole* (1815-1864)

A boolean variable has its value in {True, False}

>>> Exercise 39

- 1° a) Assign variable boolean1 to value False
- b) Assign variable boolean1 to value False

2° a) Evaluate

```
>>> not booleen1
```

b) Evaluatez

```
>>> not booleen2
```

Solutions exercise 39

1° a)

```
>>> booleen1 = False
```

```
>>> booleen1
```

```
False
```

b) On a

```
>>> booleen2 = True
```

```
>>> booleen2
```

```
True
```

2° a)

```
>>> not booleen1
```

```
# not False
```

```
# True
```

```
True
```

b) On a

```
>>> not booleen2
```

```
# not True
```

```
# False
```

```
False
```

Tips

Boolean operator not behaves as the math negative operator

```
>>> not True
```

```
False
```

```
>>> not False
```

```
True
```

>>> Exercise 40

Explain and evaluate the following code

```
>>> boolean1 = True
```

```
>>> not True
```

```
>>> not False
```

```
>>> not boolean1
```

Corrigé exercice 40

```
>>> boolean1 = True
```

```
>>> not True
```

```
False
```

```
>>> not False
```

```
True
```

```
>>> not boolean1
```

```
# not True
```

```
# False
```

```
False
```

>>> 10. Boolean operator and

>>> Exercise 41

1° a) Assign to variable boolean1 the value False

b) Assign to variable boolean1 the value True

2° Evaluate the following expression

```
>>> boolean1 and boolean2
```

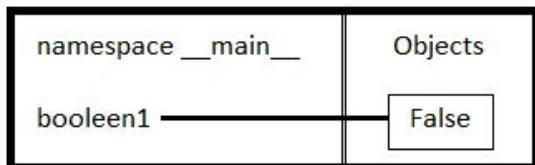
Solutions exercise 41

1° a)

```
>>> boolean1 = False
```

```
>>> boolean1
```

```
False
```

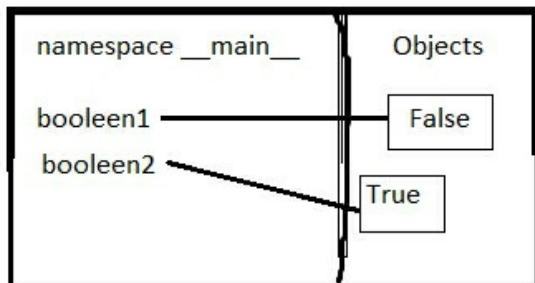


b) On a

```
>>> boolean2 = True
```

```
>>> boolean2
```

```
True
```



2° It's a valid expression

```
>>> boolean1 and boolean1
```

```
# False and True
```

```
# False
```

```
False
```

Tips

and	True	False
True	True	False
False	False	False

>>> **Exercise 42**

Evaluate the following expressions

```
>>> True and True  
>>> False and True  
>>> boolean1 = True  
>>> boolean2 = False  
>>> boolean1 and boolean2
```

Solutions

```
>>> True and True  
True  
>>> False and True  
False  
>>> boolean1 = True  
>>> boolean2 = False  
# boolean1 = True  
# boolean2 = False  
>>> boolean1 and boolean2  
# True and False  
# False  
False
```

>>> 10.1.not and and

>>> Exercice 43

Evaluate the following code

```
>>> not True and False  
>>> True and not False  
>>> not True and not False
```

Solutions

Boolean operators : not and

Operands : True False

Operator not has precedence over operator and

```
>>> not True and False  
# not True and False  
# False and False  
# False  
False
```

Operator not has precedence over operator and

```
>>> True and not False  
# True and not False  
# True and True  
#True  
True
```

Operators : not not

Would be evaluated from left to right

```
>>> not True and not False  
# not True and not False  
# False and not False  
# False and True  
#False  
False
```

Tips :

Boolean operators :

1. not
2. and

not has precedence over and

>>> Exercise 44

Evaluate the following code

```
>>> booleen1 = True  
>>> booleen2 = False  
>>> not booleen1 and booleen2  
>>> not not True  
>>> booleen1 and not booleen2
```

Solutions

```
>>> booleen1 = True  
>>> booleen2 = False  
# booleen1 = True  
# booleen2 = False  
>>> not booleen1 and booleen2  
# not booleen1 and booleen2  
# not True and booleen2  
# False and booleen2  
# False and False  
# False  
False  
>>> not not True  
# not not True  
# not False  
# True  
True  
>>> booleen1 and not booleen2  
# booleen1 and not booleen2
```

```
# booleen1 and not False  
# booleen1 and True  
# True and True  
# True  
True
```

>>> 11. Boolean operator or

>>> Exercise 45

Evaluate the following code

```
>>> True or False  
>>> False or False
```

Solutions

Boolean operator : or

Operands : True False

```
>>> True or False  
True
```

Boolean operator : or

Operands : False False

```
>>> False or False  
False
```

Tips:

Boolean operator or

Or	True	False
True	True	True
False	True	False

>>> Exercise 46

1° a) Assign boolean value True to booleen1

b) Assign boolean value False to boolean2

2° Evaluate

```
>>> boolean1 or boolean2
```

3°

```
>>> val1 or val2
```

```
False
```

What are the values of val1 and val2

Solutions

1° a)

```
>>> boolean1 = True
```

b)

```
>>> boolean2 = False
```

2°

```
>>> boolean1 or boolean2
```

```
# boolean1 or boolean2
```

```
# True or boolean2
```

```
# True or False
```

```
# True
```

```
True
```

3° on a

```
>>> val1 or val2
```

```
False
```

Back to the or table :

Or	True	False
True	True	True
False	True	False

The only way is :

```
>>> val1 = False
```

```
>>> val2 = False
```

```
>>> val2 or val2
```

```
False
```

```
>>> 11.1.not, and and or
```

>>> Exercise 47

Evaluate the following code

```
>>> not True or not False
```

```
>>> not (True and False) or (not True and False)
```

Solutions

```
>>> not True or not False
```

```
# not True or not False
```

```
# False or not False
```

```
# False or True
```

```
# True
```

```
True
```

The subexpressions will be evaluated in parentheses first

```
>>> not (True and False) or (not True and False)
```

```
# not (True and False) or (not True and False)
```

```
# not False or (False and False)
```

```
# True or False
```

```
# True
```

```
True
```

>>> Exercise 48

Evaluate the following code

```
>>> not True and (not False or True)
```

```
>>> not (False or True) and (True and not False)
```

Solutions

```
>>> not True and (not False or True)
```

```
# not True and (not False or True)
```

```
not True and (not False or True)
```

```
# not True and (True or True)
```

```
# not True and True
```

```
# False and True
```

```
# False
```

```
False
```

then

```
>>> not (False or True) and (True and not False)
```

```
# not (False or True) and (True and not False)
```

```
# not (True) and (True and not False)
```

```
# not True and (True and not False)
```

```
# not True and (True and True)
```

```
# not True and True
```

```
# False and True
```

```
# False
```

```
False
```

>>> 12. Equality operators `==` and `!=`

« Egaux dans l'adversité et différents dans la joie » anon234

>>> Exercise 49

Evaluate the following code

```
>>> 0 = 1
```

```
>>> 0 == 1
```

```
>>> 0 == 0  
>>> 0 != 1
```

Solutions

```
>>> 0 = 1
```

SyntaxError : can't assign to literal

>>> a = b the same a <- b

```
>>> 0 == 1
```

False

0 is not equal to 1, it's False

```
>>> 0 == 0
```

True

We're asserting that 0 is different to 1, it's True

```
>>> 0 != 1
```

True

Tips

- `==` equality
- `!=` difference

>>> **Exercise 50**

Evaluate the following code

```
>>> 1 == 1  
>>> 0 != 4  
>>> True == False  
>>> True == True
```

Solutions exercise 50

```
>>> 1 == 1
```

True

```
>>> 0 != 4  
True  
>>> True == False  
False  
>>> True == True  
True
```

>>> 13. Operators > and <

« la borne inférieure est le plus grand des minorants » Analyse 101

>>> Exercise 51

Evaluate the following code

```
>>> 0 > 1  
>>> 0 < 1  
>>> 45 + 3 > 45 - 3
```

Solution exercise 51

Operator : >

Operands : 0 and 1

```
>>> 0 > 1  
False
```

Operator : <

Operands : 0 and 1

```
>>> 0 < 1  
True
```

Operator : >

Operators : subexpression1 and subexpression2

Subexpression1 : 45 + 3

Subexpression2 : 45 - 3

```
>>> 45 + 3 > 45 - 3
```

```
# 45 + 3 > 45 - 3
```

```
# 48 > 45 - 3
```

```
# 48 > 42
```

```
# True
```

```
True
```

>>> Exercise 52

Evaluate the following code

```
>>> 1 > 2
```

```
>>> 2 > 3 + 2
```

```
>>> 10 + 2 > 17 - 1
```

Solutions exercise 52

```
>>> 1 > 2
```

```
False
```

```
>>> 2 > 3 + 2
```

```
# 2 > 3 + 2
```

```
# 2 > 5
```

```
# False
```

```
False
```

```
>>> 10 + 2 > 17 - 1
```

```
# 10 + 2 > 17 - 1
```

```
# 12 > 17 - 1
```

```
# 12 > 16
```

```
# False
```

```
False
```

>>> 13.1. Morgan laws

>>> Exercice 53

1° a) A and B are two sets, what is $\overline{A \cup B}$ according to morgan laws ?

b) $\overline{A \cap B}$?

2° a) Evaluate

```
>>> A = True
```

```
>>> B = False
```

b) Evaluate

```
>>> not (A or B)
```

```
>>> not A and not B
```

```
>>> not (A or B) == not A and not B
```

3° Evaluate

```
>>> not (A and B)
```

```
>>> not A or not B
```

```
>>> not (A and B) == not A or not B
```

Corrigé exercice 53

1° a) $\overline{A \cup B} = \overline{A} \cap \overline{B}$

b) $\overline{A \cap B} = \overline{A} \cup \overline{B}$

2° a)

```
>>> A
```

True

```
>>> B
```

False

b)

```
>>> not (A or B)
```

not (True or False)

not (True)

```
# False  
False  
>>> not A and not B  
# not True and not False  
# False and not False  
# False and True  
# False  
False  
>>> not (A or B) == not A and not B  
# False == False  
# True  
True
```

3°

```
>>> not (A and B)  
# not (True and False)  
# not (False)  
# True  
True  
>>> not A or not B  
# not True or not False  
# False or not False  
# False or True  
# True  
True  
>>> not (A and B) == not A or not B  
# True == True
```

```
# True
```

```
True
```

>>> 14. Strings in Python

« brisons les chaînes » Dans l'Antiquité

>>> 14.1. String Expressions

>>> Exercise 54

Evaluate the following code

```
>>> 'a'  
>>> 'spam'  
>>> 'egg'  
>>> Python
```

Solutions exercise 54

```
>>> 'a'  
'a'  
>>> 'spam'  
'spam'  
>>> 'egg'  
'egg'
```

If we don't put single quotes, there will be an error called : NameError.

```
>>> Python
```

```
NameError: name 'Python' is not defined
```

A retenir

- We use '' for string expressions

Shell

```
>>> 'string_expression'
```

```
[string_expressions]
```

>>> **Exercise 55**

Evaluate the following

```
>>> 1 * 2  
>>> True and False  
>>> 'hello'
```

1° Give the expressions

2° Give the evaluations of those expressions ?

Corrigé exercice 55

1°

Expression 1 : 1 * 2

Expression 2 : True and False

Expression 3 : 'hello'

2°

```
>>> 1 * 2  
# 1 * 2  
# 1  
2  
>>> True and False  
# False  
False  
>>> 'hello'  
'hello'
```

>>> 14.2. " " (double quote)

>>> **Exercise 56**

Evaluate the following code

```
>>> Python
```

```
>>> 'Python'  
>>> ' L'apostrophe '  
>>> "L'apostrophe"
```

Solution

NameError, Python is not a variable name

```
>>> Python  
Traceback (most recent call last):  
  File "<pyshell#13>", line 1, in <module>  
    Python  
NameError: name 'Python' is not defined  
>>> 'Python'  
'Python'  
>>> 'L'apostrophe'  
# 'L'apostrophe'  
SyntaxError : invalid syntax
```

SyntaexError : there's 3 quotes instead of 2

```
>>> "L'apostrophe"  
"L'apostrophe"
```

We use " " to fix it

A retenir

- to output a string expression we can use single ''
- if there a single quote within the expression we use " "

>>> **Exercise 57**

1° display in the shell the following expression

```
'Mathématiques'
```

2° display in the shell the following expression

```
"L'argent"
```

Solutions exercise 57

1°

```
>>> 'Mathématiques'
```

```
'Mathématiques'
```

2°

```
>>> "L'argent"
```

```
"L'argent"
```

>>> 14.3. + concatenation

>>> Exercise 58

```
>>> 'a' + 'b'
```

```
>>> "a" + "b"
```

```
>>> 'a' + 'b' + '1'
```

1° Give the expressions ?

2° Evaluate this code

Solutions exercise 58

1° expression 1 : 'a' + 'b'

Operands 'a' 'b'

Type of operands : string (str)

Operator + : concatenation

expression 2 : 'a' + 'b' + '1'

Operands 'a' 'b' '1'

Type of operands : string (str)

Operator + : concatenation

expression 3 : "a" + "b"

Operands 'a' 'b'

Type of operands : string (str)

Operator + : concatenation

2°

```
>>> 'a' + 'b'
```

```
'ab'
```

```
>>> "a" + "b"
```

```
'ab'
```

```
>>> 'a' + 'b' + '1'
```

```
'ab1'
```

>>> Exercice 59

Evaluate the following code

```
>>> '1' + '2'
```

```
>>> 1 + 2
```

```
>>> 'py' + 'thon'
```

Solutions exercise 59

Expression : '1' + '2'

Type : string (str)

Operator : + (concatenation)

```
>>> '1' + '2'
```

```
'12'
```

Expression : 1 + 2

Type : integer (int)

Operator : + (addition)

```
>>> 1 + 2
```

```
3
```

Expression : 'py' + 'thon'

Type : string (str)

Operator : + (concatenation)

```
>>> 'py' + 'thon'
```

```
'python'
```

```
>>> 14.3.1 TypeError : unmatch types
```

>>> Exercise 60

```
>>> 1 + 1
```

```
>>> '1' + 1
```

Solutions exercise 60

Expression : $1 + 1$

Types : integer (int)

```
>>> 1 + 1
```

```
2
```

Expression : $'1' + 1$

Types : string(str) and integer(int)

Operator : ??

```
>>> '1' + 1
```

```
TypeError: can only concatenate str (not "int") to str
```

>>> Exercise 61

Evaluate the following code

```
>>> 'hello' + 'world'
```

```
>>> 'hello ' + 'world'
```

```
>>> 'hello' + 1
```

Solutions exercise 61

```
>>> 'hello' + 'world'
```

```
'helloworld'
```

```
>>> 'hello ' + 'world'
```

```
'hello world'
```

```
>>> 'hello' + 1
```

```
TypeError : can only concatenate str (not int) to str
```

>>> Exercise 62

Evaluate the following code

```
>>> 1 + 1  
>>> 1 + '1'
```

Solutions exercise 62

```
>>> 1 + 1  
2  
>>> 1 + '1'  
# 1 + '1'  
# 1 + 1
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Addition operator takes only numeric operands

>>> 14.4. Operator *

>>> Exercise 63

Evaluate the following code

```
>>> 2 * 2  
>>> '2' + 2  
>>> '2' * 2
```

Solutions

expression: 2 * 2.

Operands types : int

Operator : *

```
>>> 2 * 2  
4
```

expression: '2' + 2

Operands types : str int

Operator : ??

```
>>> '2' + 2
```

TypeError: can only concatenate str (not "int") to str

expression: '2' * 2

Operands types : str int

Operator : *

```
>>> '2' * 2
```

```
'22'
```

>>> Exercise 64

Evaluate the following code

```
>>> 'a' * 10
```

```
>>> '1122'* 3
```

Solutions

```
>>> 'a' * 10
```

```
'aaaaaaaaaa'
```

```
>>> '01'* 4
```

```
'01010101'
```

>>> Exercise 65

Evaluate the following code

```
>>> 'a' * 10.0
```

```
>>> 'a'*'b'
```

Solutions

Expression : 'a' * 10.0

Operands : str float

Operator : *

```
>>> 'a' * 10.0
```

TypeError: can't multiply sequence by non-int of type 'float'

Expression : 'a'*'b'

Operands : str str

Operator : *

```
>>> 'a'*'b'
```

TypeError: can't multiply sequence by non-int of type 'str'

>>> 14.5. Expressions with + and *

>>> **Exercise 66**

Evaluate the following code

```
>>> 'a' + 'b' * 2
```

```
>>> 2 * 'a' + 'b'
```

```
>>> 2*('a' + 'b')
```

Corrigé exercice 66

Expression : 'a' + 'b' * 2

Operands : 'a' 'b' 2

Operators : + *

Operator precedence : *

```
>>> 'a' + 'b' * 2
```

```
# 'a' + 'b' * 2
```

```
# 'a' + 'bb'
```

```
# 'abb'
```

```
'abb'
```

Expression : 2 * 'a' + 'b'

Operands : 'a' 'b' 2

Operators : + *

Operator precedence : *

```
>>> 2 * 'a' + 'b'
```

```
# 2 * 'a' + 'b'
```

```
# 'aa' + 'b'
```

```
# 'aab'
```

```
'aab'
```

Parentheses first

```
>>> 2*('a' + 'b')  
# 2 * ('a' + 'b')  
# 2 * 'ab'  
# 'abab'  
'abab'
```

Tips :

- Operator * has precedence over operator +
- Expression in parentheses are evaluated first
- In case of all operators have the same precedence evaluation will be make left to right

>>> Exercise 67

Evaluate the following code

```
>>> 2*3*'a'  
>>> 'a'+'b'*2+4*'a'
```

Solutions

```
>>> 2*3*'a'  
# 2*3*'a'  
# 6*'a'  
# 'aaaaaa'  
'aaaaaa'  
>>> 'a'+'b'*2+4*'a'  
# 'a'+'b'*2+4*'a'  
# 'a'+'bb'+4*'a'  
# 'a'+'bb'+'aaaa'  
# 'abbaaaa'  
'abbaaaa'
```

>>> 15. Functions in Python

>>> 15.1. built-in functions

>>> 15.1.1. `print()`

>>> **Exercise 68**

Evaluate the following code

```
>>> 1
>>> "a"
>>> print(1)
>>> print("a")
```

Solutions

```
>>> 1
1
>>> "a"
'a'
```

Function call

Function name : `print()`

Parameter : 1 (int)

Output : 1

```
>>> print(1)
1
```

Function call

Function name : `print()`

Parameter : "a" str

Output : a

```
>>> print("a")
```

```
a
```

Tips :

Print() is our first built-in function

```
>>> print(expression)
```

```
value
```

>>> **Exercise 69**

```
>>> print(4)
```

```
>>> print("c")
```

1° a) how do we call these two code lines ?

b) Give the parameters ?

2° Evaluate the following code?

Solutions

1° a) function call

b) parameters are expressions : 4 and "c"

2°

```
>>> print(4)
```

```
4
```

```
>>> print("c")
```

```
c
```

>>> 15.1.2. **print(expression)**

>>> **Exercise 70**

```
>>> print(1 + 1)
```

```
>>> print("a"+ "b")
```

```
>>> print(True and False)
```

1° a) How do we call these 3 instructions ?

b) print() arguments ?

2° Evaluate the code ?

Solutions

1° a) We call print() 3 times

b) Arguments are expressions : `1 + 1 "a"+"b" True and False`

2°

```
>>> print(1 + 1)
# print (1+1)
# print (2)
2
>>> print("a"+"b")
# print("a"+"b")
# print("ab")
ab
>>> print(True and False)
# print(True and False)
# print(False)
False
```

A retenir :

`print()` can take an expression as argument.

Step 1 : expression is evaluated, output a value

Step 2 : print will display this value

```
>>> print(expression)
# print(value)
value
```

>>> Exercise 71

Evaluate the following code

```
>>> print(1+1+1)
>>> print("hello"+"world")
```

Solutions

```
>>> print(1+1+1)
# print(1+1+1)
# print(3)
3
>>> print("hello"+"world")
#print("hello"+"world")
# print("helloworld")
helloworld
```

>>> 15.1.3. print(variable)

>>> **Exercise 72**

Explain the following code

```
>>> a = 42
>>> a
>>> b = 1 + 1
>>> b
>>> print(a)
>>> print(b)
```

Solutions exercise 72

Variable a is assigned to value 42 (int)

```
>>> a = 42
>>> a
42
```

Variable b is assigned to the expression $1 + 1$

```
>>> b = 1 + 1
# b = 1 + 1
```

```
# b = 2
```

```
2
```

Fonction called : print()

Argument : variable a (refers to value 42)

```
>>> print(a)
```

```
# print(a)
```

```
# print(42)
```

```
42
```

Fonction called : print()

Argument : variable (refers to value 2)

```
>>> print(b)
```

```
# print(b)
```

```
# print(2)
```

```
2
```

A retenir :

Function print() can take as arguments a variable.

>>> Exercise 73

Evaluate the following code

```
>>> a = True and False
```

```
>>> b = "hello world"
```

```
>>> print(a)
```

```
>>> print(b)
```

Solutions exercise 73

```
>>> a = True and False
```

```
# a = True and False
```

```
# a = False
```

```
>>> b = "hello world"
```

```
>>> print(a)
```

```
# print(a)
```

```
# print(False)
```

```
False
```

```
>>> print(b)
```

```
# print("hello world")
```

```
# hello world
```

```
Hello world
```

>>> 15.1.3.1. **NameError** : variable unknown

>>> **Exercise 74**

Evaluate the following code

```
>>> a = 42.0
```

```
>>> print(a)
```

```
>>> print(b)
```

```
>>> b
```

Solutions

```
>>> a = 42.0
```

```
>>> print(a)
```

```
# print(42.0)
```

```
# 42.0
```

```
42.0
```

Fonction called : `print()`

Argument : variable `b` (refers to ??)

```
>>> print(b)
```

```
# print(b)
```

```
# NameError : name 'b' is not defined
```

```
NameError : name 'b' is not defined
```

```
>>> b
```

```
NameError : name 'b' is not defined
```

```
>>> 15.1.4. built-in function len()
```

```
« len for length »
```

```
>>> Exercise 75
```

Explain the following code

```
>>> chaine = "hello world"
```

```
>>> len("hello")
```

```
>>> len(1)
```

```
>>> len('1')
```

Corrigé exercice 75

Variable named chaine who refers to a string

Function called name : len

Argument : "hello" (str)

Output : hello length

```
>>> len("hello")
```

```
5
```

Function called name : len

Argument : 1 (int)

```
>>> len(1)
```

```
TypeError: object of type 'int' has no len()
```

TypeError : len doesn't take int (a fortiori a number) as argument

Function called name : len

Argument : '1' (str)

Output : '1' length

```
>>> len('1')
```

```
1
```

>>> Exercice 76

Evaluate the following code

```
>>> len("")  
>>> " * 2  
>>> len("spam")  
>>> len("egg")
```

Solutions exercise 76

```
>>> len("")
```

```
0
```

'' called empty string

Expression : '' * 2

Operands : '' 2

Operator : *

```
>>> " * 2
```

```
"
```

```
>>> len("spam")
```

```
4
```

```
>>> len("egg")
```

```
3
```

>>> 15.1.5. Expressions with function call

>>> Exercise 77

Evaluate the following code

```
>>> 1 + print(1)  
>>> 1 + len('1')
```

Solutions

Expression : 1 + print(1)

Operands : 1 print(1)

Operator : +

```
>>> 1 + print(1)
```

```
# 1 + NoneType
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

Expression : 1 + len('1')

Operands : 1 len('1')

Operator : +

```
>>> 1 +len('1')
```

```
# 1 + len('1')
```

```
# 1 + 1
```

```
# 2
```

```
2
```

>>> Exercise 78

Explain the following code

```
>>> print("hello") + 2
```

```
>>> 1 + len('11') + 3
```

Solutions exercise 78

```
>>> print("hello") + 2
```

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

```
>>> 1 + len('11') + 3
```

```
# 1 + len('11') + 3
```

```
# 1 + 2 + 3
```

```
# 6
```

```
6
```

>>> 15.1.6.Built-in function type()

« Rencontre du troisième type » Un film Python

>>> Exercise 79

Evaluate the following code

```
>>> type(15)
>>> type(10.2)
>>> type(True)
>>> type("true")
```

Solutions

Function call : type()

Argument : 15 (int)

```
>>> type(15)
# type(15)
<class 'int'>
```

Output tells 15 is from class int

```
>>> type(10.2)
# type(10.2)
<class 'float'>
```

Function call : type()

Argument : 10.2 (float)

```
>>> type(True)
# type(True)
<class 'bool'>
```

Function call : type()

Argument : True (bool)

```
>>> type("true")
# type("true")
<class 'str'>
```

Function call : type()

Argument : "true" (str)

Tips :

Type(argument) gives the type of the argument

- Int
- Float
- Bool
- str

On a

```
>>> type(argument)
```

```
<class type_argument>
```

>>> Exercise 80

1° Evaluate the following code

```
>>> type(4)
```

```
>>> type(3.14)
```

```
>>> type('python')
```

2° Evaluate the following code

```
>>> var = 45
```

```
>>> type (var)
```

Solutions exercise 80

1°

```
>>> type(4)
```

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

```
>>> type (3.14)
```

```
<class 'float'>
```

```
>>> type ('python')
```

```
<class 'str'>
```

2°

```
>>> var = 45
```

Function call : type()

Argument : variable var

```
>>> type (var)  
# type(45)  
# <class 'int'>  
<class 'int'>
```

>>> 15.2. Defining our own functions

>>> 15.2.1. keyword def and indented block

>>> **Exercise 81**

Explain the following code

```
>>> def f(x) :  
    return x + 1
```

1° give the name of the function ?

2° Evaluate the following code

```
>>> def
```

3° Explain the following code

```
>>> def f(x) :  
    return x
```

Solutions

1° The function name is : **f**

2°

```
>>> def
```

SyntaxError : invalid syntax

def is a keyword it's not a name (we don't have a NameError)

3° On a

```
>>> def f(x) :
```

```
return x
```

SyntaxError: expected an indented block

Best practice : 4 spaces for the indented block

Tips :

```
>>> def function_name(argument) :  
    block
```

>>> Exercise 82

```
>>> def vide() :  
    pass
```

1° Give the function name ?

2° Give its argument(s) ?

3° Is there an indentation ?

4° give the block ?

Solutions

1° The function name is : vide

2° There is no argument

3° It has an indentation (4 spaces)

4° The block has one instruction : `pass` (`pass` is key-word)

```
>>> pass  
>>>
```

>>> 15.2.2.Defining one-argument-function

>>> 15.2.2.1.Function call

>>> Exercise 83

```
>>> def f(x) :  
    return x + 1
```

1° Give the function name ?

2° Give the block ?

3° Evaluate the following code

```
>>> return
```

4° Evaluate the following code

```
>>> f(1)
```

Solutions

1° function name : f

2° block has only one instruction : return x + 1

3°

```
>>> return
```

SyntaxError: 'return' outside function

4° function call : f

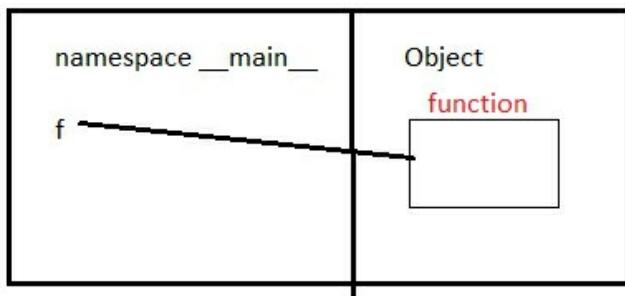
Argument : 1

```
>>> f(1)
```

```
# return 1 + 1
```

```
2
```

f(1) is an expression because the value returned.



Tips

Function call

```
>>> nom_fonction(argument)
```

>>> Exercise 84

```
>>> print(42)
```

```
>>> def f(x) :
```

```
    return x  
>>> f(1)
```

- 1° Give both function names ?
 - 2° where are the functions call ?
 - 3° Evaluate the code
- Corrigé exercice 84
- 1° name : print(built-in function)
Name : f (own function)
- 2° print(42) and f(1) are functions call
- 3°

```
>>> print(42)
```

```
42
```

```
>>> def f(x) :
```

```
    return x
```

```
>>> f(1)
```

```
# return 1
```

```
1
```

>>> 15.2.2.2. Defining one argument function and a return

>>> **Exercise 85**

- 1° Evaluate the following code

```
>>> 0 + 1
```

```
>>> -1 + 1
```

- 2° evaluate the following code

```
>>> def fonction (x) :
```

```
    return x+1
```

```
>>> fonction(0)
```

```
>>> fonction(-1)
```

Solutions

1°

```
>>> 0 + 1
```

```
1
```

```
>>> -1 + 1
```

```
0
```

2°

Function name : fonction

Argument : x

Return expression : x + 1

```
>>> def fonction(x):  
    return x + 1
```

variables du __main__

objets

fonction

function

```
>>> fonction(0)
```

```
# fonction(0):
```

```
# return 0 + 1
```

```
# return 1
```

```
1
```

```
>>> fonction(-1)
```

```
# fonction(-1):
```

```
# return -1 + 1  
# return 0  
0
```

A retenir :

```
>> def function_name (argument) :  
    block  
    return expression  
>>> function_name(argument)  
value
```

the function_name(argument) is an expression because it returned a value

>>> Exercise 86

Evaluate the following code

```
>>> def carre(x) :  
    return x*x  
>>> carre(0)  
>>> carre(-3)
```

Solutions

```
>>> carre(0)  
# carre(0)  
# return 0*0  
# return 0  
0  
>>> carre(-3)  
# carre(-3)  
# return -3 * -3  
# return 9  
9
```

>>> **Exercise 87 TypeError**

Explain the following code

```
>>> def f(x) :  
    return x + 1  
>> f()
```

Corrigé exercice 87

```
>>> f()  
TypeError: f() missing 1 required positional argument: 'x'
```

>>> 16. If

« Fais ton choix » Morpheus

>>> **Exercise 89**

1° Evaluate the following code

```
>>> if 4 > 0 :  
    print('positif')  
>>> if 4 < 0 :  
    print('positif')
```

2° Evaluate the following code

```
>>> if
```

Solutions

Condition : $4 > 0$ (True)

Block : `print('positif')`

Indentation : 4 spaces

```
>>> if 4 > 0 :  
    print('positif')
```

Condition : $4 > 0$ (True)

Block : `print('positif')`

Indentation : 0

```
>>> if 4 > 0 :  
    print('positif')  
SyntaxError : expected and indented block
```

Condition : $4 < 0$ (False)

```
>>> if 4 < 0 :  
    print('positif')  
  
>>>
```

2°

```
>>> if  
SyntaxError : invalid syntax
```

If is a python keyword

Tips

```
>>> if condition :  
    Block
```

>>> Exercise 89

1° a)

```
>>> if (1 > 0) :  
    print(1+1)
```

Give the block and the condition ?

Evaluate the code ?

2° Evaluate the following code ?

```
>>> if (0 == 1):  
    print(0)
```

Solutions exercise 89

1° a) Condition : $1 > 0$ (True)

Indentation : ok

Block : `print(1+1)`

```
>>> if (1 > 0):  
    print(1+1)  
# if True :  
# print(1+1)  
# print(2)  
# 2
```

2

2°

```
>>> 0 == 1  
False  
>>> if (0 == 1):  
# if False :  
>>>
```

>>> **Exercise 90**

Evaluate the following code

```
>>> a = True  
>>> b = False  
>>> if (a and b):  
    print('Hello')  
>>> if (a or b):  
    print('world')  
>>> if (not a):  
    print('python')  
>>> if (0):  
    print('0')  
>>> if (1):
```

```
print('le nombre 1')
>>> if (43046721) :
    print('6561')
```

Solutions

```
>>> if (a and b) :
# if (True and False) :
# if False
>>> if (a or b) :
# if (True or False) :
# if True :
# print('world')
world
>>> if (not a) :
# if (not True) :
# if False :
>>> if (0) :
# if False :
>>> if (1) :
# if True :
# print('le nombre 1')
le nombre 1
>>> if (43046721) :
# if True
# print('6561')
6561
```

Tips

```
>>> if 0 :  
# if False :  
>>> if 1 :  
# if True
```

>>> 16.1. **else**

« il faut avoir le choix » Anons

>>> **Exercise 91**

```
>>> pilule_bleu = False  
>>> if pilule_bleu :  
    print('Réalité')  
else :  
    print ('illusion')
```

Evaluate the following code

Solutions

```
>>> if pilule_bleu :  
# if False  
else :  
    print ('illusion')
```

illusion

Tips :

Condition could be

- expression with boolean operators
- a boolean variable

>>> **Exercise 92**

1° Soit le code suivant

```
>>> if 4 > 0 :
```

```
    print('4 est un nombre positif')
else :
    print ('ERROR 401')
```

- a) give the condition ?
b) what will be the output ?

2°

```
>>> variable1 = False
>>> if variable_1 :
    print('0 existe')
else :
    print ('Minkowski')
```

what will be the output ?

Solutions

1° a)

Condition : $4 > 0$ (True)

The if block will be evaluated

Block : `print('4 est un nombre positif')`

b)

```
>>> if 4 > 0 :
    # if True :
    # print('4 est un nombre positif')
    # 4 est un nombre positif
    4 est un nombre positif
```

2°

```
>>> variable1 = False
>>> if variable_1 :
    # if False :
```

```
# else :  
# print ('Minkowski')  
# Minkowski  
Minkowski
```

>>> Exercise 93

1° a) Evaluate the following code

```
>>> a = 4  
>>> a % 2
```

b) if **variable** refers to a positive integer, what can we say about this integer ?

```
>>> variable % 2  
0
```

c) What this condition is doing ?

```
>>> valeur % 2 == 0
```

2°

```
>>> variable_p = 57  
>>> if (variable_p % 2 == 0) :  
    print('pair')  
else :  
    print('impair')
```

- a) Evaluate **(variable_p % 2 == 0)** ?
- b) Give the euclidian division 57 / 2 ?
- c) What will be the output ?

Solution

1° a) variable a refers to value 4

```
>>> a = 4  
>>> a % 2  
# 4 % 2
```

```
# 0
```

```
0
```

4 is an even number

b) If

```
>>> number % 2
```

```
0
```

So number is an even number

```
>>> 4 % 2
```

```
0
```

c)

```
>>> valeur % 2 == 0
```

```
True
```

Valeur refers to an even number

```
>>> valeur % 2 == 0
```

```
False
```

Valeur refers to an odd number

```
>>> 4 % 2 == 0
```

```
True
```

```
>>> 5 % 2 == 0
```

```
False
```

2°

a)

```
>>> variable_p % 2 == 0
```

```
# 57 % 2 == 0
```

```
# False
```

```
False
```

b) $57 = 2 * 28 + 1$ r= 1 and q = 28

c)

```
>>> variable_p = 57
>>> if (variable_p % 2 == 0) :
# if (variable_p % 2 == 0) :
# if (57 % 2 == 0) :
# if (False) :
# else :
# print('impair')
impair
```

>>> **Exercise 94**

1° Evaluate the following code

```
>>> variable1 = 42
>>> if variable1 > 0 :
    print('positif')
else :
    print('négatif')
```

2° Evaluate the following code

```
>>> variable1 = -42
>>> if variable1 > 0 :
    print('positif')
else :
    print('négatif')
```

Solutions

1°

```
>>> variable1 = 42
>>> if variable1 > 0 :
# if variable1 > 0 :
```

```
# if 42 > 0 :  
# if True :  
# print('positif')  
positif
```

2°

```
>>> variable1 = -42  
>>> if variable1 > 0 :  
# if -42 > 0 :  
# if False :  
# else :  
# print('négatif')  
négatif
```

>>> **Exercise 95**

1° Evaluate this code

```
>>> variable1 = 10  
>>> variable2 = -10
```

2°

```
>>> (variable1 > 0 and variable2 > 10) or (variable1 < 0 and variable2 < 0)
```

3° Evaluate this code

```
>>> variable1 = 10  
>>> variable2 = 20  
>>> if (variable1 > 0 and variable2 > 0) or (variable1 < 0 and variable2 < 0) :  
    print('le produit des deux nombres est positif')  
else :  
    print('le produit des deux nombres est négatif')
```

Solutions

1° variables names : variable1 variable2

Values : 10 -10

2°

```
>>>(variable1 > 0 and variable2 > 0) or (variable1<0 and variable2 <0)
#(10 > 0 and 20 > 0) or (variable1<0 and variable2 <0)
#(True and True)or (variable1<0 and variable2 <0)
#(True) or (variable1<0 and variable2 <0)
#(True)or (10 < 0 and 10 <0)
# (True) or (False and False)
# True or False
# True
```

True

2°

```
>>> if (variable1 > 0 and variable2 > 0) or (variable1<0 and variable2 <0):
# if True :
    print('le produit des deux nombres est positif')
le produit des deux nombres est positif
```

>>> 16.2. Condition and range

« le ballon dans l'intervalle » Un commentateur de football

>>> **Exercise 96**

1° a) write the mathematical condition x belongs to [a ;b]?

b) write the same in python code

2° a) evaluate the following code

```
>>> var = 4
>>> (0 <= var and var <= 10)
```

b) evaluate

```
>>> var = 4
>>> if (0 <= var and var<= 10) :
```

```
    print(" 4 appartient à l'intervalle [0;10] ")
else :
    print(" 4 n'appartient pas à cet intervalle")
```

Solutions

1° a) a real x belongs to $[a ;b]$ if and only if $a \leq x \leq b$

b) we must have

1. $a \leq x$
2. $x \leq b$

Avec python keywords $(a \leq x)$ and $(x \leq b)$

2° a)

```
>>> var = 4
>>> (0 <= var and var <= 10)
# (0 <= var and var <= 10)
# (0 <= 4 and var <= 10)
# (True and var <= 10)
# (True and 4 <= 10)
# (True and True)
# True
```

True

b)

```
>>> if (0 <= var and var <= 10) :
# if (0 <= var and var<= 10) :
# if True :
# print (" 4 appartient à l'intervalle [0;10] ")
4 appartient à l'intervalle [0;10]
```

>>> **Exercise 97**

1° Give to the variable named **var** the value 15

2° evaluate the following code

```
>>> if (20 <= var and var <= 30) :  
    print(" 15 appartient à l'intervalle [20;30] ")  
else :  
    print(" 15 n'appartient pas à l'intervalle [20;30] ")
```

Corrigé exercice 97

1°

```
>>> var = 15
```

2°

```
>>> if (20 <= var and var <= 30) :  
# if (20 <= var and var <= 30) :  
# if (20 <= 15 and 15 <= 30) :  
# if (False and 15 <= 30) :  
# if (False) :  
# else :  
# print(" 15 n'appartient pas à l'intervalle [20;30] ")  
15 n'appartient pas à l'intervalle [20;30]
```

>>> **Exercise 98**

Evaluate the following code

```
>>> i = 65  
>>> j = 6.5  
>>> x 30  
>>> k = i / j  
>>> if (k * 3 and x % 3 == 0) :  
    print("Blue")  
else :  
    print("Red")
```

Solutions

```
>>> i = 65
>>> j = 6.5
# i = 65
# j = 6.5
>>> x = 30
>>> k = i / j
# x = 30
# k = i / j
# k = 65 / 6.5
# k = 10.0
>>> if (k * 3 and x % 3 == 0 ) :
# if (k * 3 and x % 3 == 0 ) :
# if (10.0 * 3 and 30 % 3 == 0) :
# if (30.0 and True) :
# if (True and True) :
# if (True) :
# print("Blue")
Blue
```

>>> Exercise 99

1° Evaluate the following code

```
>>> if (0) :
    print('ok')
else :
    print('0 est évalué comme un booléen valant False')
```

2°

```

>>> if (-42) :
    print('tout nombre autre que 0 est sera évalué comme un booléen valant
True')
else :
    print('0 est évalué comme un booléen valant False')

```

Solutions

1°

```

>>> if (0) :
# if (False) :
# else :
# print('0 est évalué comme un booléen valant False')
0 est évalué comme un booléen valant False

```

2°

```

>>> if (-42) :
# if (True) :
# print("Tout nombre autre que 0 est évalué comme un booléen valant True")
Tout nombre autre que 0 est évalué comme un booléen valant True

```

>>> Exercice 100 (Leap year) [read the function chapter and input chapter before]

1° a) Give the first condition of a leap year

A year is a leap year if and and only it can be divided by 4 and not by 100

b) If the previous condition is false we must check if it also can be divided by 400

2° Evaluate the following code

```

>>> def annee_bi(n) :
    if (n%4 ==0 and (n % 100 )!=0) or (n%400 == 0) :
        return ('annee bissextile')
    else :

```

```
    return ('sorry bro')  
>>> annee_bi(1984)  
>>> annee_bi(2000)  
>>> annee_bi(1900)
```

Solutions

1° a)

```
>>> annee_bissextile = int(input())  
>>> ((annee_bissextile % 4 )== 0) and ((annee_bissextile % 100) != 0)
```

b) on a

```
>>> annee_bissextile = int(input())  
>>> ((annee_bissextile % 4 )== 0) and ((annee_bissextile % 100) != 0) or  
(annee_bissextile % 400 ==0)
```

2° the function has one argument and return a string

```
>>> annee_bi(1984)  
# if (1984 % 4 == 0) and (1984 % 100!=0) or (1984%400==0) :  
# if (True and True) or (1984%400 ==0)  
# if True :  
# return année bissextile  
annee_bissextile  
>>> annee_bi(2000)  
# if (2000% 4 == 0) and (2000% 100!=0) or (2000%400==0) :  
# if True and False or True  
# if False or True  
# if True  
# return aneee bissextile  
annee_bissextile  
>>> annee_bi(1900)
```

```
# if (1900% 4 == 0) and (1900% 100!=0) or (1900%400==0) :  
# if True and False or False  
# if False or False  
# if False  
# return 'sorry bro'  
sorry bro
```

>>> 17. Read with function `input()`

>>> 17.1. Function `int()`

« je transforme du texte en nombre » Anons

>>> **Exercise 101**

1° evaluate the following code

```
>>> int('42')
```

2° evaluate the following code

```
>>> chaine = '41'  
>>> type(chaine)
```

3°

```
>>> int(chaine)
```

4°

```
>>>help(int)
```

Solutions

1°

1. a function : `int()`
2. single argument : string '42'

```
>>> int('42')
```

```
# int('42')
```

```
42
```

The result is an integer

2°

```
>>> chaine = '41'  
>>> type(chaine)  
<class 'str'>
```

3°

```
>>> int(chaine)  
# int('41')  
# 41  
41
```

4°

```
>>> help(int)  
Help on class int in module builtins:  
class int(object)  
| int([x]) -> integer  
| Convert a number or string to an integer, or return 0 if no arguments  
| are given. If x is a number, return x.__int__(). For floating point  
| numbers, this truncates towards zero.
```

>>> Exercise 102

Evaluate the following code

```
>>> int('100')  
>>> var = '9'  
>>> int(var)
```

Corrigé exercice 102

```
>>> int('100')  
100
```

```
>>> var = '9'  
>>> int(var)  
# int('9')  
# 9  
9
```

>>> **Exercise 103**

```
>>> int(100.0)  
>>> var = 9.3  
>>> int(var)
```

Solutions

```
>>> int(100.0)  
100  
>>> var = 9.3  
>>> int(var)  
# int(9.3)  
# 9  
9
```

>>> 17.2. Read with `input()`

« Lecture et écriture » Algorithmique

```
>>> help(input)  
input(prompt=None, /)  
    Read a string from standard input.
```

>>> **Exercise 104**

1° a)

```
>>> print('hello')
```

b)

```
>>> input()
```

42

2° a)

```
>>> input('que vaut 1+1 ?')
```

b)

```
>>> var = input ('que vaut 1+1 ?')
```

Solutions

1°a)

```
>>> print('hello')
```

hello

b)

```
>>> input()
```

42

'42'

2° a)

```
>>> input('que vaut 1+1 ?')
```

que vaut 1+1 ?2

'2'

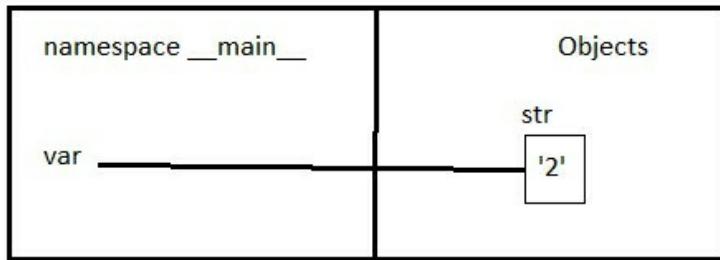
b)

```
>>> var = input('que vaut 1+1 ?')
```

que vaut 1+1 ?2

```
>>> var
```

'2'



>>> Exercise 105

Evaluate the following code

```
>>> input ( )
windows
>>> var = input ('Que vaut la racine carrée de 16 ?')
Que vaut la racine carrée de 16 ?4
>>> var
```

Corrigé exercice 105

```
>>> input ( )
windows
'windows'
>>> var = input ('Que vaut la racine carrée de 16 ?')
Que vaut la racine carrée de 16 ?4
>>> var
'4'
```

>>> 17.2.1. input (), int () et if

>>> Exercise 106

1°

```
>>> int(input ( ))
```

42

2°

```
>>> var = int(input())
```

```
27
```

```
>>> var
```

```
>>> type(var)
```

solutions

1°

```
>>> int(input())
```

```
# input()
```

```
42
```

```
#'42'
```

```
# int('42')
```

```
#42
```

```
42
```

2°

```
>>> var = int(input())
```

```
# var = int(input())
```

```
# 27
```

```
#'27'
```

```
# var = int('27')
```

```
# var = 27
```

```
>>> var
```

```
27
```

```
>>> type(var)
```

```
#type(27)
```

```
#<class 'int'>
```

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

Tips

```
>>> f(g())
```

Function name : f

Argument : function g

```
>>> f()  
# call g()  
# result  
# f(result)  
# result2  
result2
```

>>> Exercise 107

Evlauate the following code

1°

```
>>> int(input('que vaut 1+1 ? '))
```

2° Que vaut var à la fin de ce code ?

```
>>> var = int(input())  
34  
>>> var = var + 42  
>>> print(var)
```

Corrigé exercice 107

1°

```
>>> int(input('que vaut 1+1 ? '))  
que vaut 1+1 ? 2  
'2'
```

Function : int()

Argument : input()

First call : input()

```
>>> int(input('que vaut 1+1 ?'))
```

que vaut 1+1?2

```
# input()
```

```
# '2'
```

```
# int('2')
```

```
# 2
```

```
2
```

2°

```
>>> var = int(input( ))
```

```
# var = int(input( ))
```

```
# 34
```

```
# '34'
```

```
# var = int('34')
```

```
# var = 34
```

```
>>> var = var + 42
```

```
# var = var + 42
```

```
# var = 34 + 42
```

```
# var = 34 + 42
```

```
# var = 76
```

```
>>> print(var)
```

```
# print(76)
```

```
# 76
```

```
76
```

>>> Exercise 108

1° which function can transform a string into an integer ?

2° Evaluate the following code

```
>>> print('Quel âge avez vous ?')
>>> var = int(input( ))
34
>>> print('vous avez', var , 'ans')
```

Solutions

1° we look into `help(int)`

```
int ([x]) → integer
```

2°

```
>>> print('Quel âge avez vous ?')
Quel âge avez-vous ?
>>> var = int(input())
34
# var = 34
>>> print('vous avez', var , 'ans')
# print('vous avez', 34, 'ans')
vous avez 34 ans
```

>>> Exercise 109

Evaluate the following code

```
>>> var = -42
>>> if (var > 0) :
    print('positive')
else
    print('negative')
```

Solutions

```
>>> var = -42
>>> if (var > 0) :
```

```
# if (-42 > 0) :  
# if False :  
else :  
    print('negative')  
negative
```

>>> 17.3. elif

>>> **Exercise 110**

Evaluate the following code

```
>>> var = int(input('Entrez un nombre entier '))  
Entrez un nombre entier 0  
>>> if (var >0) :  
    print('le nombre est positif')  
elif (var == 0) :  
    print('le nombre est nul')  
else :  
    print('le nombre est négatif')
```

Corrigé exercice 110

```
>>> var = int(input('Entrez un nombre entier'))  
Entrez un nombre entier 0  
  
>>> if (var >0) :  
# if (0 > 0) :  
#if False :  
# elif (var == 0 ) :  
# elif (0 == 0) :  
# elif True :
```

```
print('le nombre est nul')
```

```
le nombre est nul
```

>>> **Exercise 111 (Majeur, mineur)**

Evaluate the following code

```
>>> var = int(input('Quel âge avez vous ? '))
```

```
Quel âge avez vous ? -42
```

```
>>> if (var > 18) :
```

```
    print('majeur')
```

```
elif (var < 18) :
```

```
    print('mineur')
```

```
else :
```

```
    print('veuillez entrez un nombre positif')
```

Solutions

```
>>> var = int(input('Quel âge avez vous ? '))
```

```
Quel âge avez vous ? -42
```

```
# var = -42
```

```
>>> if (var > 18) :
```

```
# if (-42 > 18) :
```

```
# if False :
```

```
# elif (-42 < 18) :
```

```
# elif (True)
```

```
# print('mineur')
```

```
mineur
```

```
>>> 17.3.1. if (expression) :
```

>>> **Exercise 112**

Evaluate the following code

```
>>> if (4-4) :  
    print('ok')  
else :  
    print('pas ok')  
>>> if (15-19) :  
    print('ok')  
else :  
    print('pas ok')
```

Solutions

```
>>> if (4-4) :  
# if (4-4) :  
# if(0) :  
# if (False) :  
# else :  
# print('pas ok')  
pas ok
```

On a

```
>>> if (15-19) :  
# if (15-19) :  
# if (-4) :  
# if (True) :  
# print('ok')  
ok
```

Tips

```
>>> if (expression) :  
    block
```

>>> Exercise 113

1°) give the answer

```
>>> if(0) :  
    print('rien')  
else :  
    print('0 est évalué à False')
```

- i) rien ii) 0 est évalué à False

2° give the answer

```
>>> if (42) :  
    print('good number')  
else :  
    print('pas good number')
```

- i) good number ii) pas good number

Solutions

1° i)

```
>>> if(0) :  
# if(False) :  
#else :  
# print('0 est évalué à False')  
0 est évalué à False
```

2° ii) good number

```
>>> if (42) :  
# if (True) :  
# print('good number')  
good number
```

>>> Exercise 114

1° Evaluate following code

```
>>> if ([]):
    print('ok')
else :
    print('liste vide')
>>> if ({}):
    print('ok')
else :
    print('ensemble vide')
>>> if (''):
    print('ok')
else :
    print('chaîne vide')
```

2°

```
>>> if (None):
    print('ok')
else :
    print('None')
>>> if ({'livre' : 42}):
    print('ok')
else :
    print('pas ok')
```

Corrigé exercice 114

```
>>> if ([]):
# if(False):
# else :
    print('liste vide')
```

```
liste vide
>>> if ({}):
# if(False)
#
else :
    print('ensemble vide')
```

```
ensemble vide
>>> if ():
# if(False):
else :
    print('chaîne vide')
chaîne vide
```

2°

```
>>> if (None):
# if(False)
else :
    print('None')
None
>>> if ({'livre' : 42})
# if(True):
# print('ok')
ok
```

>>> 18. First module : math

>>> 18.1. Import with import

>>> **Exercise 115** $\sqrt{2}$

Evaluate the following code

```
>>> sqrt  
>>> import  
>>> import math  
>>> import module
```

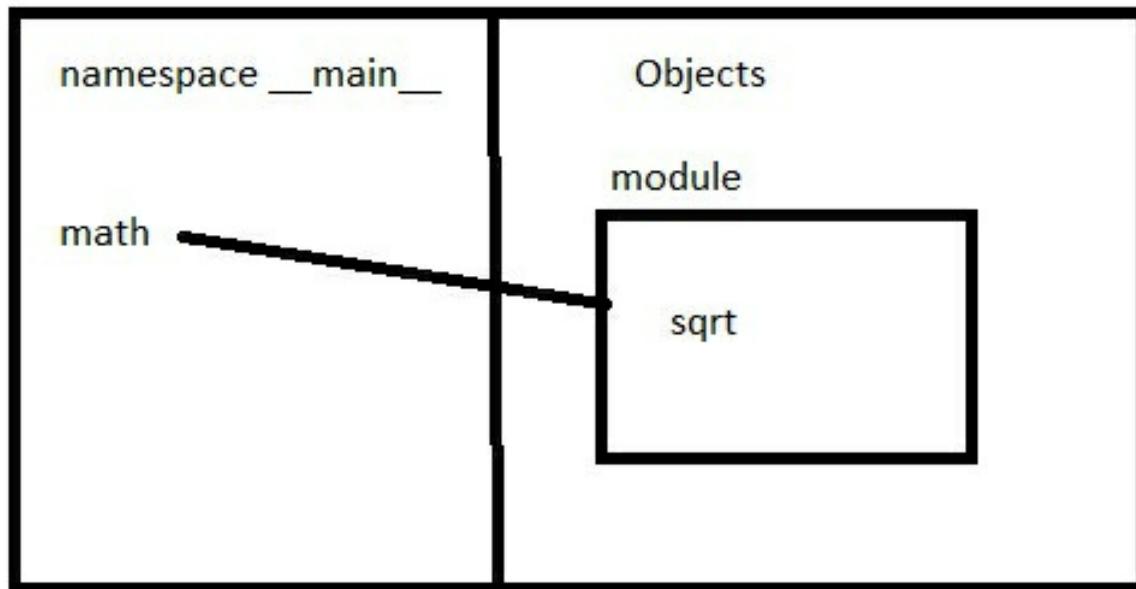
Solutions

On a

```
>>> sqrt  
NameError: name 'sqrt' is not defined  
>>> import  
SyntaxError : invalid Syntax
```

Import is a python keyword

```
>>> import math  
>>>
```



a

```
>>> import module
```

On

ModuleNotFoundError: No module named 'module'

>>> 18.2.Inside a module

>>> 18.2.1.Function dir()

>>> **Exercise 116**

Evaluate the following code

```
>>> help(dir)
>>> dir()
>>> import math
>>> dir(math)
```

Solutions

```
>>> help(dir)
```

Help on built-in function dir in module builtins:

dir(...)

dir([object]) -> list of strings

If called without an argument, return the names in the current scope.

Else, return an alphabetized list of names comprising (some of) the attributes

of the given object, and of attributes reachable from it.

If the object supplies a method named `__dir__`, it will be used; otherwise

the default `dir()` logic is used and returns:

for a module object: the module's attributes.

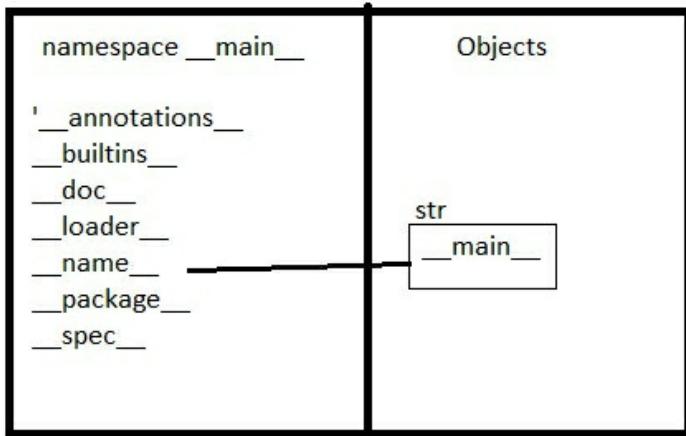
for a class object: its attributes, and recursively the attributes of its bases.

for any other object: its attributes, its class's attributes, and recursively the attributes of its class's base classes.

Let's see inside our namespace `__main__`

```
>>> dir()
```

```
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__',  
'__package__', '__spec__']
```

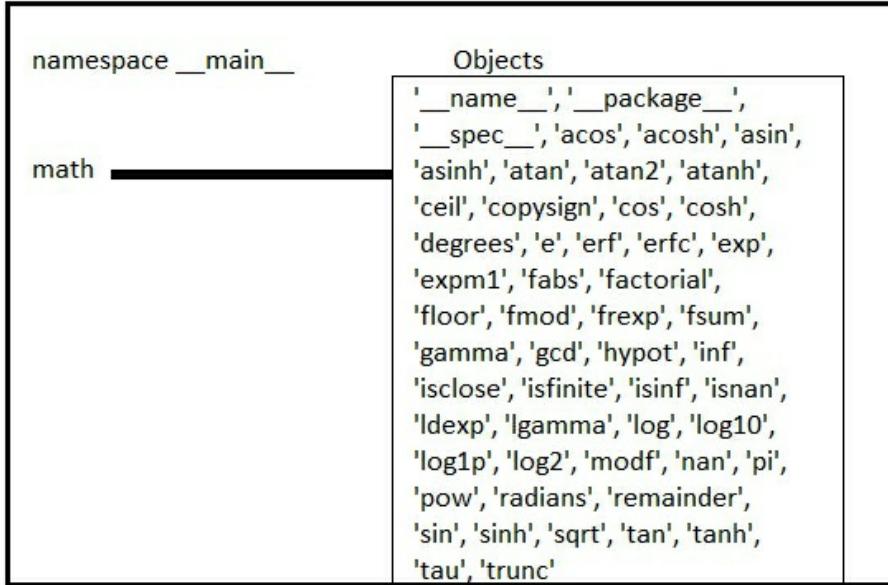


```
>>> import math
```

```
>>> dir(math)
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',  
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',  
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',  
'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan',  
'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow',  
'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

```
>>>
```



>>> Exercise 117 (module sys)

1° Import module sys

2° Display his attributes with dir()

Solutions

1°

```
>>> import sys
```

2°

```
>>> dir(sys)
```

```
['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__',
 '__interactivehook__', '__loader__', '__name__', '__package__', '__spec__',
 '__stderr__', '__stdin__', '__stdout__', '_clear_type_cache',
 '_current_frames', '_debugmallocstats', '_enablelegacywindowsfsencoding',
 '_framework', '_getframe', '_git', '_home', '_xoptions', 'api_version', 'argv',
 'base_exec_prefix', 'base_prefix', 'breakpointhook', 'builtin_module_names',
 'byteorder', 'call_tracing', 'callstats', 'copyright', 'displayhook', 'dllhandle',
 'dont_write_bytocode', 'exc_info', 'excepthook', 'exec_prefix', 'executable',
 'exit', 'flags', 'float_info', 'float_repr_style', 'get_asyncgen_hooks',
 'get_coroutine_origin_tracking_depth', 'get_coroutine_wrapper',
 'getallocatedblocks', 'getcheckinterval', 'getdefaultencoding',
 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile',
 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval', 'gettrace',
```

```
'getwindowsversion', 'hash_info', 'hexversion', 'implementation', 'int_info',
'intern', 'is_finalizing', 'last_traceback', 'last_type', 'last_value', 'maxsize',
'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks',
'path_importer_cache', 'platform', 'prefix', 'set_asasyncgen_hooks',
'set_coroutine_origin_tracking_depth', 'set_coroutine_wrapper',
'setcheckinterval', 'setprofile', 'setrecursionlimit', 'setswitchinterval',
'setrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'version', 'version_info',
'warnoptions', 'winver']
```

>>> 18.2.2. Access to modules attributes with the dot . operator

« *Tout est mathématiques* » Un présocratique

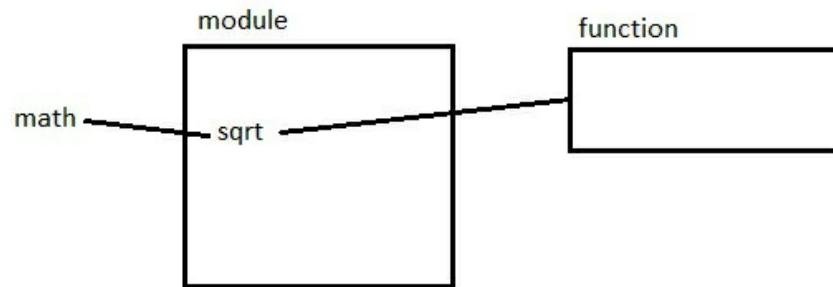
>>> **Exercise 118 (Function sqrt ())**

Evaluate the following code

```
>>> import math
>>> math.sqrt(7)
```

Solutions

```
>>> import math
>>> math.sqrt(7)
2.6457513110645907
```



Tips

```
>>> module.function(argument)
```

doc

```
>>> help(math)
```

Help on built-in module math:

NAME

math

DESCRIPTION

This module is **always available**.

FUNCTIONS

[...]

cos(x, /)

Return the cosine of x (measured in radians)

sqrt(x, /)

Return the square root of x.

>>> Exercise 119 (function cos ())

Evaluate the following code

```
>>> cos (0)
>>> import math
>>> math.cos(0)
>>> math.cos(3.1415)
```

Solutions

```
>>> cos (0)
NameError : name 'cos' is not defined
```

Semantic Error, there is no name **cos** in namespace **__main__**

```
>>> import math
>>> math.cos(0)
1.0
>>> math.cos(3.1415)
-0.999999957076562
```

>>> 18.2.3. Some maths

« toujours tout prendre au premier degré »

>>> Exercise 120

Part 1

$$-4x^2 - 16x + 84 = 0 \text{ with } x \text{ a real number}$$

Evaluate the following code

```
>>> import math
>>> a,b,c= -4,-16,84
>>> delta = b**2 - 4 * a * c
>>> delta > 0
>>> x1 = (-b + math.sqrt(delta) ) / (2*a)
>>> x2 = (-b - math.sqrt(delta) ) / (2*a)
>>> x1
>>> x2
```

Part 2

Evaluate the following code

```
>>> import math
>>> print('***Equation du second degré dans R***')
>>> print('ax**2 + b*x + c')
>>> a = int(input('entrez le coefficient a ?'))
>>> b = int(input('entrez le coefficient b ?'))
>>> c = int(input('entrez le coefficient c ?'))
>>> delta = b**2 - 4 * a * c
>>> if (delta > 0) :
    print('Two solutions')
elif (delta == 0) :
    print('one solution')
else :
```

```
print('no solutions in R')  
>>> x1 = (-b + math.sqrt(delta)) / (2*a)  
>>> x2 = (-b - math.sqrt(delta)) / (2*a)  
>>> print('x1 = ', x1, ' and x2 = ', x2)
```

solutions

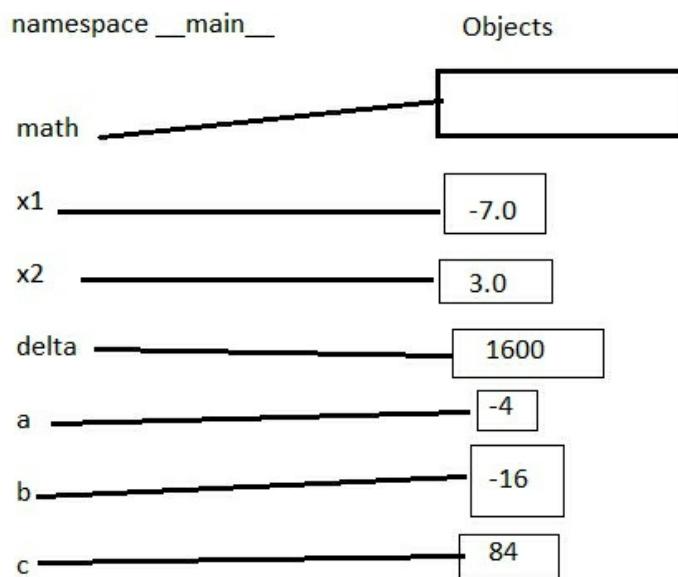
part 1

```
>>> import math  
>>> a,b,c= -4,-16,84  
#a = -4  
# b = -16  
# c = 84  
>>> delta = b**2 - 4 * a * c  
# delta = (-16) ** 2 - (4 *(-4) *84)  
# delta = 256 - (-1344)  
# delta = 256 + 1344  
# delta = 1600  
>>> delta > 0  
# 1600 > 0  
# True  
True  
>>> x1 = (-b + math.sqrt(delta)) / (2*a)  
# x1 = (-(-16)+ math.sqrt(1600)) / (2 * (-4))  
# x1 = ( 16 + 40) / (-8)  
# x1 = 56 / -8  
# x1 = -7.0  
>>> x2 = (-b - math.sqrt(delta)) / (2*a)
```

```

# x2 = (-(-16) - math.sqrt(delta)) / (2*(-4))
# x2 = (16 - 40) / -8
# x2 = -24 / -8
# x2 = 3.0
>>> x1
-7.0
>>> x2
3.0

```



Partie 2

```

>>> import math
>>> print('***Equation du second degré dans R***')
*** Equation du second degré dans R***
>>> print('ax**2 + b*x + c')
ax**2 + b*x + c
>>> a = int(input('entrez le coefficient a ?'))
entrez le coefficient a ?-4
>>> b = int(input('entrez le coefficient b ?'))

```

```
entrez le coefficient b ?-16
```

```
>>> c = int(input('entrez le coefficient c ?'))
```

```
entrez le coefficient c ?84
```

```
>>> delta = b**2 - 4 * a * c
```

```
# detla = 1600
```

```
>>> if (delta > 0) :
```

```
# if (1600 > 0) :
```

```
# if (True) :
```

```
# print('two solutions')
```

```
two solutions
```

```
>>> x1 = (-b + math.sqrt(delta) ) / (2*a)
```

```
# x1 = -7.0
```

```
>>> x2 = (-b - math.sqrt(delta) ) / (2*a)
```

```
# x2 = 3.0
```

```
>>> print('x1 = ', x1, ' and x2 = ', x2)
```

```
# print('x1 vaut ', -7.0, ' and x2 = ', 3.0)
```

```
# x1 = -7.0 and x2 = 3.0
```

```
x1 = -7.0 and x2 = 3.0
```

>>> Exercise 121

1° a) give the function name ?

b) give its arguments ?

c) give else return expression ?

```
>>> def equation2(a,b,c) :
```

```
    delta = b**2-4*a*c
```

```
    if (delta >0) :
```

```
        x = (-b -math.sqrt(delta)) /( 2*a)
```

```
        y = (-b+ math.sqrt(delta)) / (2*a)
```

```
    return print('les solutions sont', x , 'et', y)
elif(delta ==0) :
    x = -b / (2*a)
    return print('une solution double', x )
else :
    return 'pas de solution'
```

2° Evaluate the following code

```
>>> equation2(-4,-16,84)
```

Solutions

- 1° a) function name : `equation2`
- b) three arguments
- c) the expression is `'pas de solution'`

2°

```
>>> equation2(-4,-16,84)
# delta = (-4)**2-4*(-16)*(84)
# delta = 1600
# if (1600>0) :
# if True :
# x = (-(-16)+ math.sqrt(1600) ) / (2 * (-4))
# x = -7.0
# y = (-(-16) – math.sqrt(delta))/ (2*(-4))
# y = 3
# return print('les solutions sont', x , 'et', y)
# return les solutions sont -7 et 3
```

les solutions sont -7 et 3

>>> 19. Looping with `while`

« La boucle est bouclée »

>>> **Exercise 122**

1° Evaluate

```
>>> print('hello world', 'hello world', 'hello world')
```

- a) give print arguments ?
- b) give arguments type ?

2°

```
>>> a= 0
>>> while (a < 3 ) :
    print('hello world')
    a = a + 1
```

Resolve $a < 3$ (a a positive integer) and evaluate this code

Solutions

- 1° a) print takes 3 arguments
- b) 'hello world' is string type
- c)

```
>>> print('hello world', 'hello world', 'hello world')
```

```
hello world hello world hello world hello world
```

2° $a < 3$ has solutions the set $\{0,1,2\}$

```
>>> a = 0
```

```
>>> a < 3
```

```
True
```

```
>>> a = 1
```

```
>>> a < 3
```

```
True
```

```
>>> a = 2
```

```
>>> a < 3
```

```
True
```

```
>>> a = 3
```

```
>>> a < 3
```

```
False
```

```
iteration a < 3  print('hello world')          a = a +1
1      True    hello world                      a = 0 +1
2      True    hello world hello world          a = 1 +1
3      True    hello world hello world hello world  a= 2 +1
4      False
```

```
>>> a= 0
```

```
>>> while (a < 3) :
```

```
    print('hello world')
```

```
    a = a + 1
```

```
hello world
```

```
hello world
```

```
hello world
```

Tips for n iterations

```
>>> variable = 0
```

```
>>> while (variable < n) :
```

```
    block
```

```
    variable = variable + 1
```

>>> Exercise 123 Infinite loop

```
>>> var = 0
```

```
>>> while (var < 3) :
```

```
    print('infinity')
```

1° give the missing instruction in while block ?

2° evaluate this code ?

Corrigé exercice 124

1° It missed the var variable

2°

```
iterations  var < 3  print('infinity')  var
1          True      'infinity'      0
2          True      'infinity'      0
3          True      'infinity'      0
4          True      'infinity'      0
```

....

```
43046721 True      'infinity'      0
+ infini  True      'infinity'      0
```

>>> Exercise 124

1° evaluate

```
>>> var = 5
>>> while ( var > 0 ):
    print('Hello')
    var = var - 1
```

2° evaluate

```
>>> var = 5
```

```
>>> while (var > 0 ):  
    print('Hello')  
    var = var - 1
```

solutions

1° Semantic error

```
>>> var = 5  
>>> while ( var > 0 ):  
    print('Hello')  
    var = var - 1
```

SyntaxeError : expected an indentend block

tour	var	var >0	affichage
1	5	True	Hello
2	4	True	Hello
3	3	True	Hello
4	2	True	Hello
5	1	True	Hello
	0	False	

2°

```
>>> var = 5  
>>> while (var > 0 ):  
    print('Hello')  
    var = var - 1
```

```
hello  
hello  
hello  
hello  
hello
```

```
>>> var
```

```
0
```

>>> 19.1.Sequences of numbers with while()

« tant qu'il y aura des pizzas »

>>> **Exercise 125 (0 to 5)**

1° evaluate

```
>>> print(0)
>>> print(1)
>>> print(2)
>>> print(3)
>>> print(4)
>>> print(5)
```

2° evaluate

```
>>> print(0,1,2,3,4,5)
```

3° evaluate

```
>>> a = 0
>>> while (a < 6) :
    print(a)
    a = a + 1
```

Solutions

1°

```
>>> print(0)
0
>>> print(1)
1
>>> print(2)
```

```
2
>>> print(3)
3
>>> print(4)
4
>>> print(5)
5
```

We called print() six times

2°

```
>>> print(0,1,2,3,4,5)
0 1 2 3 4 5
```

We called print() once but with 6 int arguments

3°

```
iteration a <6 print(a)  a = a + 1
0      True  0      a = 1
1      True  1      a = 2
2      True  2      a = 3
3      True  3      a = 4
4      True  4      a = 5
5      True  5      a = 6
6      False
```

```
0
1
2
3
4
5
>>>
```

Tips

```
>>> a = 0
>>> while ( a < n ) :
    print(a)
    a = a +1
```

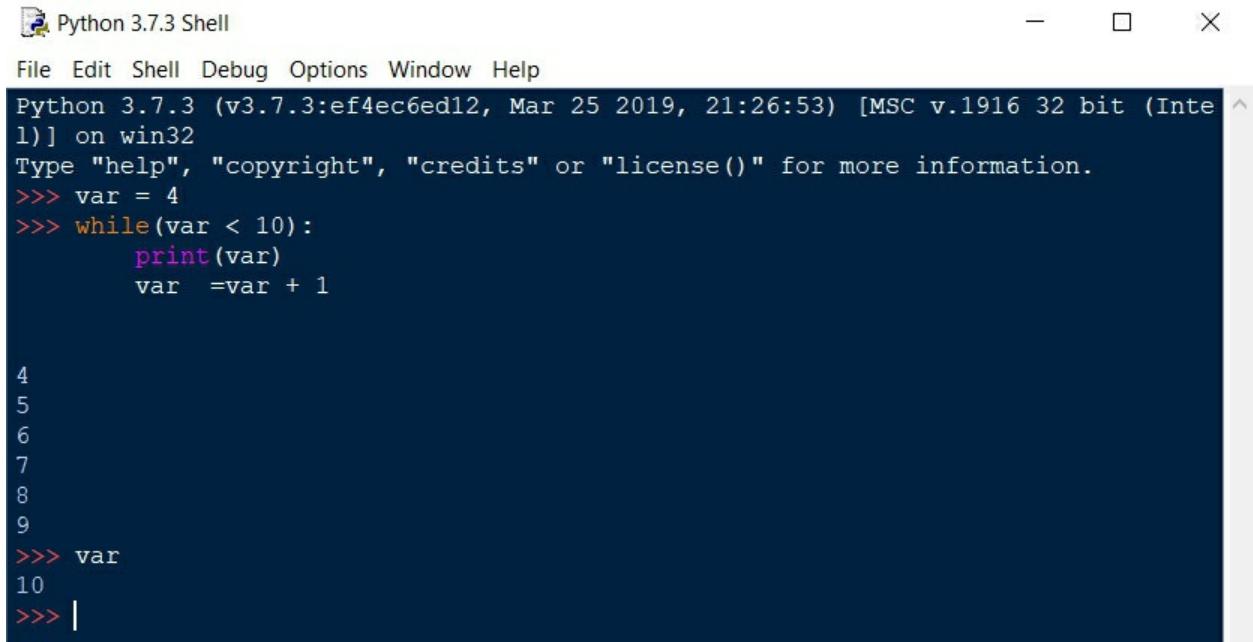
```
0
1
2
3
4
[...]
n-2
n-1
```

>>> Exercise 127

Evaluate

```
>>> var = 4
>>> while (var < 10) :
    print(var)
    var = var + 1
>>> var
```

Solutions



Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> var = 4
>>> while(var < 10):
    print(var)
    var  =var + 1

4
5
6
7
8
9
>>> var
10
>>> |
```

var	var < 10	tour
4	True	1
5	True	2
6	True	3
7	True	4
8	True	5
9	True	6
10	False	

>>> 19.2. I love maths and while()

Il existe une affinité certaine entre une suite numérique et les boucles

>>> **Exercise 128**

$$u_n = n + 2 \quad (n \text{ a positive integer})$$

evaluate

```
>>> i = 0
>>> u_i = 0
>>> while (i < 10):
    u_i = i + 2
    print(u_i)
    i = i+1
```

Solutions

While block has 3 instructions

```
u_i = i + 2
```

```
print(u_i)
```

```
i = i+1
```

```
>>> i = 0
```

```
>>> u_i = 0
```

```
>>> while (i < 10) :
```

```
    u_i = i + 2
```

```
    print(u_i)
```

```
    i = i + 1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
11
```

tour	i	u_i	affichage	i < 10
1	0	2	2	True
2	1	3	3	True
3	2	4	4	True
4	3	5	5	True
5	4	6	6	True
6	5	7	7	True
7	6	8	8	True
8	7	9	9	True
9	8	10	10	True
10	9	11	11	True

>>> Exercise 128

$$u_n = \cos(n)$$

```
>>> import math
```

1° Evaluate the following code

```
>>> i= 0
>>> u_i = 0
>>> u_i = i + 2
>>> while (i < 10) :
    print(u_i)
    i = i+1
```

2° Evaluate

```
>>> i = 0
>>> u_i = 0
>>> u_i = i + 2
>>> while (i < 10) :
    print(u_i)
    i = i+1
```

Solutions

```
>>> import math
>>> i = 0
>>> while (i<10) :
```

```
print(math.cos(i))
```

```
i=i+1
```

```
1.0
0.5403023058681398
-0.4161468365471424
-0.9899924966004454
-0.6536436208636119
0.28366218546322625
0.960170286650366
0.7539022543433046
-0.14550003380861354
-0.9111302618846769
```

2° variable u_i is out the loop, incorrect !

```
>>> i = 0
>>> u_i = 0
>>> u_i = i + 2
# u_i = 0 + 2
# u_i = 2
>>> while (i < 10) :
    print(u_i)
    i = i+1
```

```
2
2
2
2
2
```

```
2
2
2
2
2
```

Tips $u_n = \text{expression}$

```
>>> i= 0
>>> while (i < n) :
    u_i = expression
    print(u_i)
    i= i +1
```

>>> 19.2.1.while and sequences

>>> **Exercise 130**

$u_n = \exp(n)$

Evaluate the following code

```
>>> import math
>>> i = 0
>>> u_i = math.exp(i)
>>> while (u_i < 1000) :
    i = i + 1
    u_i = math.exp(i)
>>> i
```

Solutions

```
>>> import math
>>> i = 0
```

```
>>> while (i < 10) :  
    u_i = math.exp(i)  
    print(u_i)  
    i=i+1
```

```
1.0  
2.718281828459045  
7.38905609893065  
20.085536923187668  
54.598150033144236  
148.4131591025766  
403.4287934927351  
1096.6331584284585  
2980.9579870417283  
8103.083927575384
```

$$u_n < 1000 \Leftrightarrow \exp(n) < 1000 \Leftrightarrow n < \ln(1000)$$

Let's see

```
>>> import math  
>>> i = 0  
>>> u_i = math.exp(i)  
>>> while (u_i < 1000) :  
    i = i + 1  
    u_i = math.exp(i)  
>>> i
```

```
7
```

tour	i	u_i
1	1	2.718281828459045
2	2	7.38905609893065
3	3	20.085536923187668
4	4	54.598150033144236
5	5	148.4131591025766
6	6	403.4287934927351
7	7	1096.6331584284585

>>> Exercise 130

$$u_n = \ln(n)$$

1° Fill the blanks do display the 10 first terms

```
>>> import
>>> help (math)
#
log(...)

log(x, [base=math.e])
```

Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.

```
>>> i =
>>> u_i
>>> while ( i < ) :
    u_i = math.log(i)
    print( )
    i =
```

2° evaluate

```
>>> import math
>>> i = 1
>>> u_i = math.log(i)
>>> while (u_i < 10 ) :
    i = i + 1
```

```
u_i = math.log(i)
```

```
>>> i
```

```
22027
```

Solutions

1°

$$u_n > 10 \Rightarrow \ln(n) > 10 \Rightarrow n > e^{10} \Rightarrow n > 22026.46$$

If $n > 22027$, we have $\ln(n) > 10$

Let's check it

```
>>> import math  
>>> i = 22027  
>>> u_i = math.log(i)  
>>> u_i
```

```
10.000024252584158
```

```
>>> import math  
>>> help (math)  
#  
log(...)  
    log(x, [base=math.e])
```

Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.

```
>>> i = 1  
>>> u_i = math.log(i)  
>>> while ( i < 11 ) :  
    u_i = math.log(i)  
    print( u_i )  
    i = i + 1
```

```
0.0
0.6931471805599453
1.0986122886681098
1.3862943611198906
1.6094379124341003
1.791759469228055
1.9459101490553132
2.0794415416798357
2.1972245773362196
2.302585092994046
```

2°

```
>>> import math
>>> i = 1
>>> u_i = math.log(i)
>>> while (u_i < 10) :
    i = i + 1
    u_i = math.log(i)
>>> i
22027
```

>>> **Exercise 132**

$$u_n = \sqrt{(n + 2)}$$

1° write with quantifiers $\lim u_n = + \infty$

2° evaluate the following code

```
>>> import math
>>> i = 0
>>> while (i < 5) :
```

```
u_i = math.sqrt(i+2)  
i = i+1
```

3° Evaluate

```
>>> import math  
>>> i = 0  
>>> u_i = math.sqrt(i+2)  
>>> while (u_i < 100) :  
    i = i+1  
    u_i = math.sqrt(i+2)  
>>> i  
9998
```

Solutions

1° $\forall M > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 u_n \geq M$

$M > 0, n \geq n_0 \geq M^2 - 2$ soit

$n \geq M^2 - 2 \Leftrightarrow n + 2 \geq M^2 \Leftrightarrow \sqrt{n + 2} \geq M \Leftrightarrow u_n \geq M$

2°

```
>>> import math  
>>> i = 0  
>>> while (i < 5) :  
    u_i = math.sqrt(i+2)  
    i = i+1
```

```
1.4142135623730951  
1.7320508075688772  
2.0  
2.23606797749979
```

```
2.449489742783178
```

$$u_n \geq 100 \Leftrightarrow \sqrt{n+2} \geq 100 \Leftrightarrow n+2 \geq 100^2 \Leftrightarrow n+2 \geq 10000 \Leftrightarrow n \geq 9998$$

If $n \geq 9998$ we have $u_n \geq 100$

Let's check it

```
>>> import math  
>>> i = 998  
>>> u_i = math.sqrt(i+2)  
>>> u_i  
100.09995004993759
```

3°

```
>>> import math  
>>> i = 0  
>>> u_i = math.sqrt(i+2)  
>>> while (u_i < 100) :  
    i = i+1  
    u_i = math.sqrt(i+2)  
>>> i  
9998
```

>>> 19.2.2. Boucle while et affichage de tables

>>> **Exercice 132**

1° Evaluate

```
>>> print(' 2 fois 0 = ', 2*0)  
>>> print ('2 fois 1 = ', 2*1)  
>>> print (' 2 fois 2 = ', 2*2)
```

2° Evaluate

```
>>> i = 0
```

```
>>> while (i < 10) :  
    print(' 2 fois ' , i , ' = ' , 2*i)  
    i = i+1
```

Solutions

1°

```
>>> print(' 2 fois 0 = ' , 2*0)
```

Print() takes two arguments, a string and an expression

```
>>> print(' 2 fois 0 = ' , 2*0)  
# print('2 fois 0 = ' , 0)
```

```
2 fois 0 = 0
```

```
>>> print ('2 fois 1 = ' , 2*1)
```

```
2 fois 1 = 2
```

```
>>> print (' 2 fois 2 = ' , 2*2)
```

```
2 fois 2 = 4
```

2°

```
# print ('2 fois' , i , ' = ' , 2*i)  
>>> i = 0  
>>> while (i < 10) :  
    print(' 2 fois ' , i , ' = ' , 2*i)  
    i = i+1
```

```
2 fois 0 = 0
```

```
2 fois 1 = 2
```

```
2 fois 2 = 4
```

```
2 fois 3 = 6
```

```
2 fois 4 = 8
```

```
2 fois 5 = 10
```

```
2 fois 6 = 12
```

```
2 fois 7 = 14
```

```
2 fois 8 = 16
```

```
2 fois 9 = 18
```

>>> Exercise 133

Evaluate the following code

```
>>> i = 0
>>> while (i<11):
    print(' 2 plus ', i , ' = ', 2+i)
    i=i+1
```

Solutions

```
>>> i = 0
>>> while (i<11):
    print(' 2 plus ', i , ' = ', 2+i)
    i=i+1
```

```
2 plus 0 = 2
```

```
2 plus 1 = 3
```

```
2 plus 2 = 4
```

```
2 plus 3 = 5
```

```
2 plus 4 = 6
```

```
2 plus 5 = 7
```

```
2 plus 6 = 8
```

```
2 plus 7 = 9
```

```
2 plus 8 = 10
```

```
2 plus 9 = 11
```

```
2 plus 10 = 12
```

>>> Exercise 134

Evaluate the following code

```
>>> i = 0
>>> while(i<11):
    print(' 7 x ', i, ' = ', 7*i)
    i=i+1
```

Solutions

```
>>> i = 0
>>> while(i<11):
    print(' 7 x ', i, ' = ', 7*i)
    i=i+1
```

```
7 x  0 = 0
7 x  1 = 7
7 x  2 = 14
7 x  3 = 21
7 x  4 = 28
7 x  5 = 35
7 x  6 = 42
7 x  7 = 49
7 x  8 = 56
7 x  9 = 63
7 x 10 = 70
```

>>> 19.2.3. Make choice with while()

>>> Exercise 136

Evaluate the following code

```
>>> var = 7
```

```
>>> i = 0
>>> while (i < 11) :
    print(var, ' fois ' , i , ' = ' , i *var)
    i = i + 1
```

2° Evaluate the following code

```
>>> var = int(input('please give a value ? '))
>>> i = 0
>>> while (i < var) :
    print(var, ' fois ' , i , ' = ' , i * var)
    i = i + 1
```

Corrigé exercice 136

print () takes five arguments

1 : var (variable)

2 : string

3 : i (variable)

4 : string

5 : expression

```
print(var, ' fois ' , i , ' = ' , i * var)
```

1°

7 fois 0 = 0

7 fois 1 = 7

7 fois 2 = 14

7 fois 3 = 21

7 fois 4 = 28

7 fois 5 = 35

7 fois 6 = 42

7 fois 7 = 49

7 fois 8 = 56

7 fois 9 = 63

7 fois 10 = 70

tour	i	i < 11
1	0	True
2	1	True
3	2	True
4	3	True
5	4	True
6	5	True
7	6	True
8	7	True
9	8	True
10	9	True
11	10	True
12	11	False

2° On a

```
>>> var = int(input('please give a value ? '))
```

```
Please give a value ? 11
```

```
>>> i = 0
```

```
>>> while (i < var) :
```

```
    print(var, ' fois ', i, ' = ', i * var)
```

```
    i = i + 1
```

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte ^  
1)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> var = int(input('please give a value ? '))  
please give a value ? 11  
>>> i=0  
>>> while (i < var) :  
    print(var, ' fois ' , i , ' = ' , i * var)  
    i = i + 1  
  
11  fois  0  =  0  
11  fois  1  =  11  
11  fois  2  =  22  
11  fois  3  =  33  
11  fois  4  =  44  
11  fois  5  =  55  
11  fois  6  =  66  
11  
    fois  7  =  77  
11  fois  8  =  88  
11  fois  9  =  99  
11  fois  10  =  110  
>>> |
```

>>> **Exercice 136 (à faire après avoir lu le passage sur les fonctions)**

Fill the blanks

```
>>> def table(nombre, longueur) :  
    i=0  
    while(i< ) :  
        print(    , 'fois', i, ' = ', i *    )  
        i=  
>>> table(4,5)
```

```
4 fois 0 = 0
4 fois 1 = 4
4 fois 2 = 8
4 fois 3 = 12
4 fois 4 = 16
4 fois 5 = 20
```

Solutions

```
>>> def table(nombre, longueur) :
    i=0
    while(i< longueur) :
        print(nombre, 'fois', i, '=', i * nombre )
        i= i+1
>>> table(4,5)
4 fois 0 = 0
4 fois 1 = 4
4 fois 2 = 8
4 fois 3 = 12
4 fois 4 = 16
4 fois 5 = 20
```

>>> 19.3. Recursives sequences

>>> 19.3.1. $u_{n+1} = u_n$

>>> Exercise 137

$$u_{n+1} = 2u_n + 1 \text{ and } u_0 = 4$$

1° Evaluate the following code

```
>>> u0 = 4
```

```
>>> u1 = 2*u0 + 1
>>> u2 = 2*u1 + 1
>>> u3 = 2*u2 + 1
```

2° a) Evaluate the following code

```
>>> i = 0
>>> a = 4
>>> while(i<10) :
    s= 2*a + 1
    a = s
    i = i +1
    print(a, end =' ')
>>> a
```

Solutions

1°

```
>>> u0 = 4
>>> u1 = 2*u0 + 1
# u1 = 2 * 4 + 1
# u1 = 9
>>> u2 = 2*u1 + 1
# u2 = 2 * 9 + 1
# u2 = 19
>>> u3 = 2*u2 + 1
# u3 = 2 * 19 + 1
# u3 = 39
```

2°

```
>>> i = 0
```

```

>>> a =4
>>> while(i<10) :
    s= 2*a + 1
    a = s
    i = i +1
    print(a, end = ' ')
9 19 39 79 159 319 639 1279 2559 5119
>>> a
#u10
5119

```

Tips

```

>>> u_0 = value
>>> while (i < n ) :
    s = expression
    a= s
    i = i+1
>>> a
# u_n

```

>>> 19.3.1.1. $u_{n+2} = u_{n+1} + u_n$

>>> Exercise 139

Fibonacci : $u_0 = 0, u_1 = 1, u_{n+2} = u_{n+1} + u_n$

1° evaluate the following code

```

>>> u0, u1 = 0, 1
>>> u2 = u1 + u0
>>> u3 = u2 + u1

```

```
>>> u4 = u3 + u2
>>> u5 = u4 + u3
>>> u6 = u5 + u4
```

2° evaluate the following code

```
>>> a, b = 0, 1
>>> s, i = 0, 0
>>> while(i< 10) :
    s = a + b
    a = b
    b = s
    i = i + 1
>>> s, a, b
```

3° Evaluate the following code

```
>>> a, b = 0, 1
>>> s, i = 0, 0
>>> while(i< 10) :
    a, b = b, a+b
    i = i + 1
    print(a,b)
```

Solutions

1°

```
>>> u0, u1 = 0, 1
>>> u2 = u1 + u0
# u2 = 1 + 0
# u2 = 1
>>> u3 = u2 + u1
# u3 = 1 + 1
```

```
# u3 = 2
>>> u4 = u3 + u2
# u4 = 2 + 1
# u4 = 3
>>> u5 = u4 + u3
# u5 = 3 + 2
# u5 = 5
>>> u6 = u5 + u4
# u6 = 5 + 3
# u6 = 8
```

2°

```
>>> a, b = 0, 1
>>> s, i = 0, 0
>>> while(i< 10) :
    s = a + b
    a = b
    b = s
    i = i + 1
>>> s, a, b
(89,55,89)
```

tour	s	a	b
1	1	1	1
2	2	1	2
3	3	2	3
4	5	3	5
5	8	5	8
6	13	8	13
7	21	13	21
8	34	21	34
9	55	34	55
10	89	55	89

We can notice that

```
>>> s = a + b
>>> a = b
>>> b = a + b
```

Is same as :

```
>>> a, b = b , a + b
```

3°

```
>>> a, b = 0, 1
>>> s, i = 0, 0
>>> while(i< 10) :
    a, b = b, a+b
    i = i + 1
    print(a, b)
```

```
1 1
1 2
2 3
3 5
5 8
8 13
```

```
13 21  
21 34  
34 55
```

A retenir $u_{n+2} = \text{expression}$ with $u_0 = \text{value1}$ and $u_1 = \text{value2}$

```
>>> a = value1  
>>> b = value2  
>>> while (i < n) :  
    s = expression  
    a = b  
    b = s
```

>>> 19.4. Nesting **while** and **if**

>>> **Exercise 139**

Evaluate the following code with

Var input : - 3

Var input : 3

```
>>> bool1 = True  
>>> while (bool1) :  
    print('ok')  
    var = int(input())  
    if var > 0 :  
        bool1= False  
    else  
        bool1 = True
```

Solutions

bool1 is a boolean variable

```
>>> type(bool)
```

```
<class 'bool'>
```

Loop : while

Block : print(), var, if else

```
>>> while (bool1) :  
    print('ok')  
    var = int(input())  
    if var > 0 :  
        bool1= False  
    else  
        bool1 = True
```

we give 3 as value

```
>>> bool1 = True  
>>> while (bool1) :  
    # while (True)  
    print('ok')  
    #ok  
    var = int(input())  
    -3  
    # var = - 3  
    if var > 0 :  
        #if -3 > 0 :  
        #if False  
        else :  
            bool1 = True  
    #end of the loop, back to while  
    # while(True)
```

ok

We give 3 as value

```
var = int(input())
3
if var > 0 :
#if 3 > 0 :
#if True :
bool1= False
back to the loop
#while(False):we get out
```

>>> Exercise 140

1° Evaluate the following code

```
>>> var = 4
>>>(0 <= var) and (var <=5)
>>> var2 = 8
>>> var3 = -1
>>> (0 <= var2) and (var2 <=5)
>>> (0 <= var3) and (var3 <=5)
```

2°

a) we give 42 as value

b) we give 4

```
>>> n = int(input('give a number between 0 and 5'))
>>> if (0 <= n) and (n <= 5) :
    print("good in the range")
else :
```

```
print("pas good")
```

3° Evaluate the following code

```
>>> bool1 = True
>>> while bool1 :
    var = int(input('give a number between 0 and 5 to get out the loop'))
    if (0 <= var) and (var <= 5) :
        print('bye')
        bool1 = False
```

Solutions

Partie 1

```
0 <= x and x <= 5
```

```
>>> var = 4
>>>(0 <= var) and (var <=5)
# (0 <= 4 ) and (4 <=5)
# True and True
# True
>>> var2 = 8
>>> var3 = -1
>>> (0 <= var2) and (var2 <=5)
# (0 <= 8) and (8 <=5)
# True and False
# False
>>> (0 <= var3) and (var3 <=5)
# (0 <= -1 ) and (-1 <= 5)
# False and True
# False
```

Part

2° a)

```
>>> n = int(input('give a number between 0 and 5'))  
give a number between 0 and 5 42  
>>> if (0 <= n) and (n <= 5) :  
# if ( 0 <= 42 ) and (n <= 5) :  
# if True and (42 <= 5) :  
# if True and False :  
# if False :  
else :  
#print("pas good")  
pas good
```

b)

```
>>> n = int(input('give a number between 0 and 5'))  
give a number between 0 and 5 4  
>>> if (0 <= n) and (n <= 5) :  
# if (0 <= 4) and (n <= 5) :  
# if True and (4 <= 5) :  
# if True and True :  
# if True :  
# print("good in the range")  
good in the range
```

3°

code

```
>>> bool1 = True  
>>> while bool1 :  
# while True :
```

```
var = int(input('give a number between 0 and 5 to get out of the loop '))

# 42

# if (0 <= var) and (var <= 5) :

# if (0 <= 42) and (var <= 5) :

# if True and 42 <= 5 :

# if True and False :

# if False :

# back to the loop

# while (bool1) :

# while (True) :
```

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1) on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> bool1=True
>>> while bool1:
    var = int(input('give a number between 0 and 5 to get out of the loop'))
    if (0<= var) and (var<=5):
        print('bye')
        bool1=False

give a number between 0 and 5 to get out of the loop42
give a number between 0 and 5 to get out of the loop4
bye
>>>
```

```
# while True :
```

```
var = int(input('entrez un nombre entre 0 et 5 pour sortir de
la boucle '))
# 4

# if (0 <= var) and (var <= 5) :
# if (0 <= 4) and (var <= 5) :
# if True and 4 <= 5 :
# if True and True:
# if True :
# print('ok')
# bool1 = False

# while (bool1) :
# while (False) :
# we get out
bye
```

Tips

```
bool1 = True
while (True) :
    Instructions
    if( ) :
        bool1 = False
```

>>> Exercise 141

Evaluate the following code

```
>>> bool = True
>>> while(bool) :
    n = int(input('give a number between 0 and 1'))
    if (0 <= n) and (n <= 1) :
        print("number is in the range")
```

```
bool = False
```

Corrigé

```
>>> bool = True
>>> while(bool) :
    n = int(input('give a number between 0 and 1 '))
    if (0 <= n) and (n <= 1) :
        print("number is in the range")
        bool = False
give a number between 0 and 1 15
give a number between 0 and 1 15
give a number between 0 and 1 1
```

```
>>> 19.4.1. + / -
```

« c'est ça ... plus ou moins »

>>> Exercise 142

module **random**

```
>>> import random
>>> help(random)
Help on module random: [...]
NAME random - Random variable generators.
randint(self, a, b)
Return random integer in range [a, b], including both end points.
```

1° Evaluate the following code

```
>>> random.randint(0,10)
>>> random.randint(0,100)
>>> random.randint(0,1000)
```

2° guess a number bewteen 0 and 10

Fill the blank

```
>>> import
>>> nombre_mystere =
>>> bool1 =
>>> print('*** MYSTERY NUMBER ***')
>>> while (bool1) :
    = int(input('enter a number?'))
    if      :
        print('too big')
    elif    :
        print('too small')
    else :
        print('Bravo')
    bool1 =
```

Solutions

1° I've got these numbers

```
>>> random.randint(0,10)
6
>>> random.randint(0,100)
87
>>> random.randint(0,1000)
457
```

2°

First we declared two variables : mystery_number and input_number

```
>>> import random
>>> mystery_number = random.randint(0,10)
```

```
>>> input_number = int(input('give a number between 0 and 10 '))
```

```
>>> bool = True
>>> mystery_number = random.randint(0,10)
>>> while(bool) :
    input_number = int(input('give a number between 0 and 10? '))
    if (input_number > mystery_number) :
        print('too big')
    elif(input_number < mystery_number) :
        print('too small')
    else :
        print('bravo')
    bool = False
```

>>> **Exercise 143**

1° Evaluate the following code

```
>>> print('*****')
>>> print('*****')
>>> print('*****')
>>> print('*****')
```

2 ° Evaluate the following code

```
>>> i = 0
>>> while (i < 4) :
    print('*****')
    i = i+1
```

3° Fill the blank

```
>>>     = (input('how many lines ??'))
>>> i = 0
```

```
>>> while < number_line :  
    print('****')  
    i=
```

Solutions

1°

```
>>> print('****')  
****  
>>> print('****')  
****  
>>> print('****')  
****  
>>> print('****')  
****
```

2°

```
>>> i = 0  
>>> while (i < 4) :  
    print('****')  
    i = i+1  
****  
****  
****  
****
```

3°

```
>>> number_ligne = int(input('combien de lignes voulez vous ?'))  
>>> while i < number_line :  
    print('****')  
    i=i+1
```

>>> Exercise 144

Fill the blanks

```
>>> def plusMoins( ) :  
    nombre_mystere =  
    ne = int(input('give a number between 0 and 100'))  
    bool =  
    while (bool) :  
        if(      ) :  
            print('trop grand')  
        elif (   ) :  
            print('trop petit')  
        else :  
            print('good, le nombre mystere etait bien', nombre_mystere)  
        bool =  
>>> plusMoins()
```

Corrigé

```
>>> def plusMoins( ) :  
    nombre_mystere = random.randint(0,100)  
    ne = int(input('give a number between 0 and 100'))  
    bool = True  
    while (bool) :  
        if(ne > nombre_mystere) :  
            print(' trop grand ')  
        elif (ne < nombre_mystere) :  
            print(' trop petit ')  
        else :
```

```
print(' good, le nombre mystere etait bien ', nombre_mystere)
bool= False
```

>>> 19.5. Inception : while nested in a while

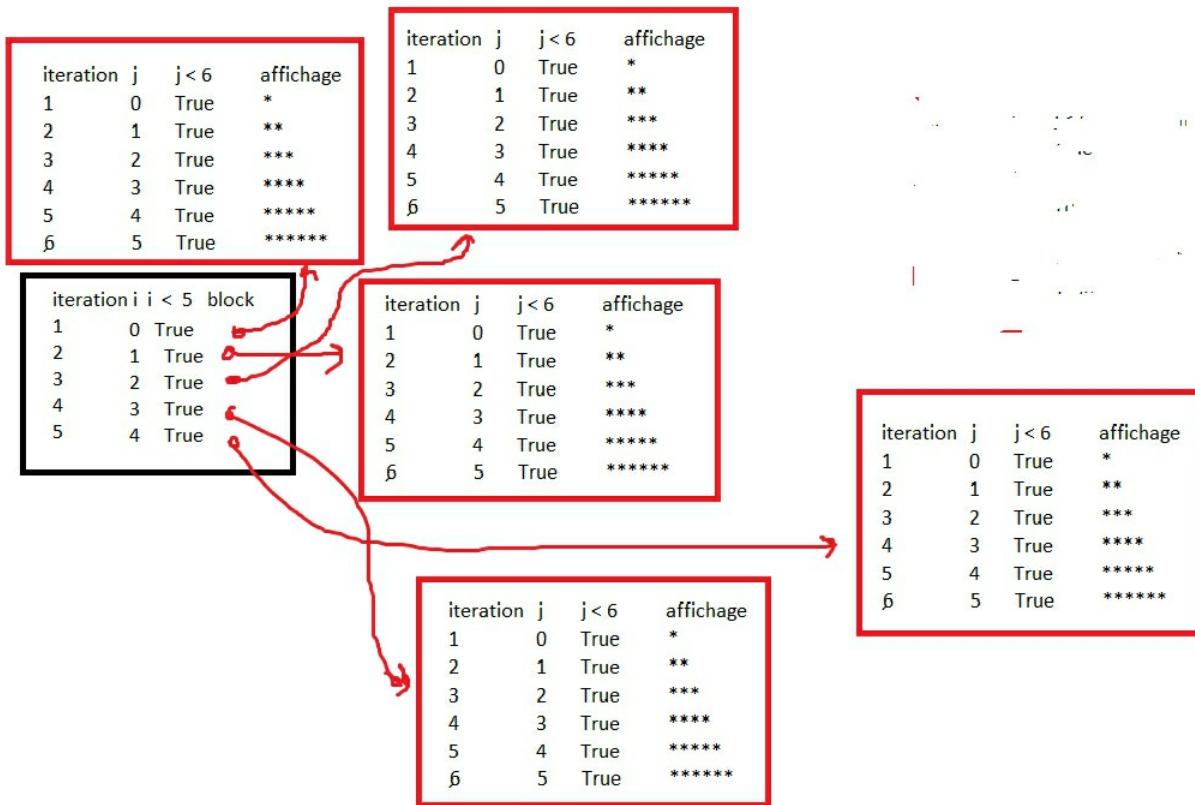
>>> 19.5.1. Variables i and j are independant

>>> **Exercise 145**

Evaluate the following code

```
>>> i =0
>>> while (i < 5) :
    j=0
    while (j < 6) :
        print('*', end = "")
        j=j+1
    print()
    i= i+1
```

Corrigé exercice 146



screenshot

```

>>> i=0
>>> while(i<5):
    j=0
    while(j<6):
        print('*',end= ' ')
        j=j+1
    print()
    i=i+1

*****
*****
*****
*****
*****
>>> |

```

Tips

```

>>> i = 0
>>> while (i < number_1):

```

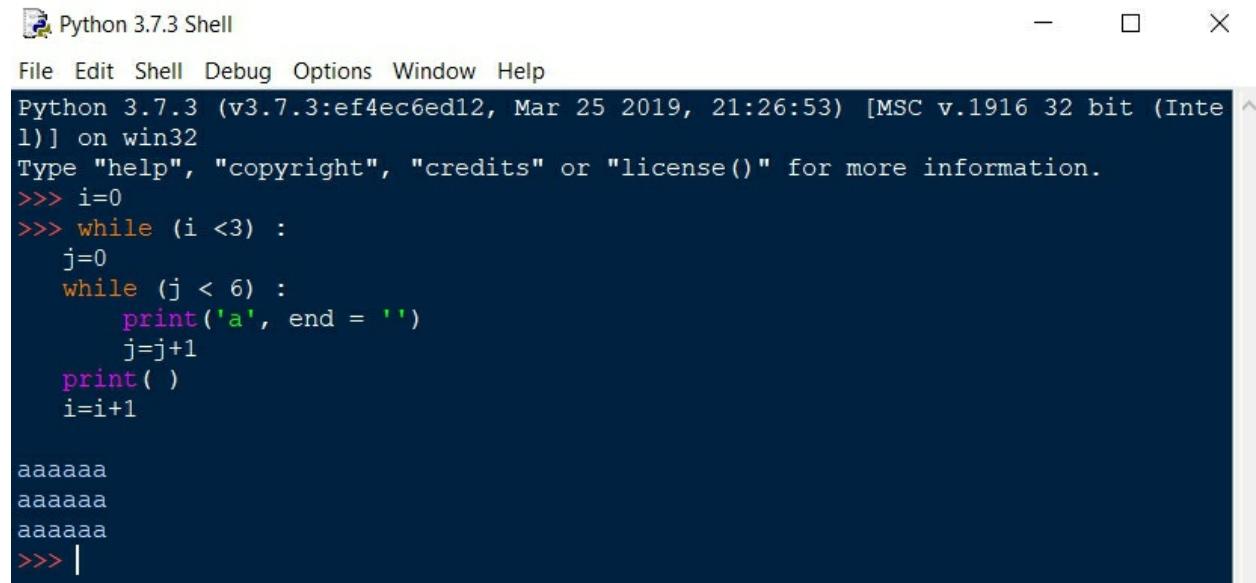
```
j=0
while (j < number_2) :
    block
    j=j+1
    i=i+1
```

>>> Exercise 146

Evaluate the following code

```
>>> i = 0
>>> while (i <3) :
    j=0
    while (j < 6) :
        print('a', end = '')
        j=j+1
    print( )
    i=i+1
```

Solutions



Python 3.7.3 Shell

```
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Inte
1)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> i=0
>>> while (i <3) :
    j=0
    while (j < 6) :
        print('a', end = '')
        j=j+1
    print( )
    i=i+1

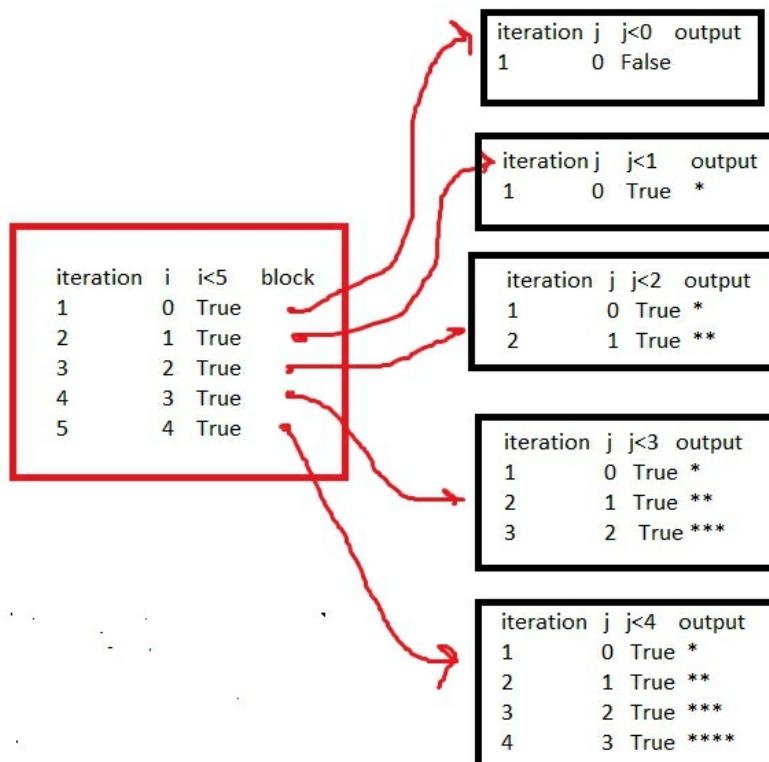
aaaaaa
aaaaaa
aaaaaa
>>> |
```

>>> 19.5.2. Variables i and j are linked

>>> exercice 147

```
>>> i = 0
>>> while (i < 5) :
    j=0
    while (j < i) :
        print('*', end = "")
        j=j+1
    print()
    i= i+1
```

Solutions



```
*  
**  
***
```

>>> Exercise 148

Evaluate the following code

```
>>> i =0
>>> while (i < 10) :
    j = 0
    while (j < i ) :
        print('a', end = '')
        j=j+1
    print( )
    i=i+1
```

solutions

```
>>> i=0
>>> while (i < 10) :
    j = 0
    while (j < i ) :
        print('a', end = '')
        j=j+1
    print( )
    i=i+1

a
aa
aaa
aaaa
aaaaa
aaaaaa
aaaaaaa
aaaaaaaaa
aaaaaaaaaa
>>> |
```

>>> 19.5.3. Handle while in while

>>> Exercise 149

We want to display this

```
aaaaab
```

1° Evaluate the following code

```
>>> i =0  
>>> while (i <5) :  
    print('a', end =")  
    i= i+1
```

2° Fill the blanks

```
>>> nombre_a = int(input(      ))  
>>> i = 0  
>>> while ( i <    ) :  
    print('a',end="")  
    if (i ==  ) :  
        print('b',end="")  
    i=i+1
```

aaaaaaab

Solutions

1°

```
>>> i =0  
>>> while (i <7) :  
    print('a', end = "")  
    i=i+1
```

aaaaaaa

2°

```
>>> nombre_a = int(input('entrez le nombre de a '))  
entrez le nombre de a7  
>>> i = 0  
>>> while ( i < nombre_a) :  
    print('a',end="")
```

```
if (i == nombre_a - 1) :  
    print('b',end="")  
    i=i+1
```

aaaaaaab

>>> 19.5.4.while inception

>>> **Exercise 150**

Evaluate the following code

```
>>> i=0  
>>> while (i< 5) :  
    j=0  
    while (j< 4) :  
        print('a', end = "")  
        j=j+1  
    k=0  
    while (k < 3) :  
        print('b', end = "")  
        k=k+1  
    print()  
    i=i+1
```

Solutions

aaaabbb
aaaabbb
aaaabbb
aaaabbb
aaaabbb

>>> **Exercise 151**

Evaluate the following code

```

>>> i =0
>>> while (i < 5) :
    j =0
    while (j< 3) :
        print('*' , end = '')
        j = j+1
    k=0
    while (k<2) :
        print('x', end = '')
        k = k+1
    print( )
    i= i+1

```

Solutions

```

>>> while (i < 5) :
    j =0
    while (j< 3) :
        print('*' , end = '')
        j = j+1
    k=0
    while (k<2) :
        print('x', end = '')
        k = k+1
    print( )
    i= i+1

***xx
***xx
***xx
***xx
***xx
>>> |

```

>>> 19.5.4.1. Update conditions in nested loops

>>> **Exercise 152**

1° Evaluate the following code

```

>>> i =0
>>> condition1 = 1

```

```
>>> while (i < 3) :  
    j=0  
    while (j < condition1) :  
        print('*', end =")  
        j=j+1  
    k=0  
    while (k < 4) :  
        print('x', end =")  
        k=k+1  
    print()  
    condition1 = condition1 + 1  
    i=i+1
```

2° Evaluate the following code

```
>>> i =0  
>>> condition1 = 1  
>>> condition2 = 5  
>>> while (i < 3) :  
    j=0  
    while (j < condition1) :  
        print('*', end =")  
        j=j+1  
    k=0  
    while (k < condition2) :  
        print('x', end =")  
        k=k+1  
    print()
```

```
condition1 = condition1 + 1  
condition2 = condition2 - 1  
i=i+1
```

Solutions

1°

```
>>> i=0
>>> condition1=1
>>> while (i < 3) :
    j=0
    while (j < condition1) :
        print('*', end = '')
        j=j+1
    k=0
    while (k < 4) :
        print('x', end = '')
        k=k+1
    print()
    condition1 = condition1 + 1
    i=i+1
```

SyntaxError: invalid syntax

```
>>> while (i < 3) :
    j=0
    while (j < condition1) :
        print('*', end = '')
        j=j+1
    k=0
    while (k < 4) :
        print('x', end = '')
        k=k+1
    print()
    condition1 = condition1 + 1
    i=i+1
```

```
*XXXX
**XXXX
***XXXX
>>> |
```

2°

```
>>> while (i < 3) :
    j=0
    while (j < condition1) :
        print('*', end ='')
        j=j+1
    k=0
    while (k < condition2) :
        print('x', end ='')
        k=k+1
    print()
    condition1 = condition1 + 1
    condition2 = condition2 - 1
    i=i+1
```

```
*xxxxx
**xxxx
***xxx
>>> |
```

Ln: 155 Col: 4

>>> Exercise 153

```
aaaaab
aaaabb
aaabbb
aabbba
abbbbb
bbbbbb
```

1° Fill this tab

iteration	nombre_a	nombre_b
1		
2		
3		
4		
5		
6		

2° Evaluate the following code

```
>>> i=0
>>> nombre_a = 5
>>> nombre_b = 1
```

```

>>> while(i< ) :
    j = 0
    while (j <      ) :
        print(  , end =")
        j=
    k=0
    while (k <      ) :
        print(  , end="")
        k=
    print()
    nombre_a = nombre_a -1
    nombre_b =
    i=

```

Solutions

1°

iteration	nombre_a	nombre_b
1	5	1
2	4	2
3	3	3
4	2	4
5	1	5
6	0	6

2°

```

>>> i=0
>>> nombre_a = 5
>>> nombre_b = 1
>>> while(i< 6 ) :
    j = 0
    while (j < nombre_a ) :

```

```
print( 'a' , end =")  
j= j+1  
k=0  
while (k < nombre_b ) :  
    print('b' , end="")  
    k= k + 1  
print()  
nombre_a = nombre_a -1  
nombre_b = nombre_b + 1  
i= i + 1
```

```
aaaaab  
aaaabb  
aaabbb  
aabbbb  
abbbbb  
bbbbbb
```

>>> Exercise 154

We want to display this

```
*
```



```
***
```



```
*****
```

Evaluate the following code

```
>>> i = 0  
>>> nbespaces =  
>>> nbetoiles = 1  
>>> while (i < 3) :
```

```
j=0
while (j <      ) :
    print(", end= ")
    j=
k=0
while( k < nbetoiles) :
    print(    , end=' ')
    k=k+1
print()
nbespaces =
nbetoiles = nbetoiles +2
i=
```

Solutions

```
>>> i = 0
>>> nbespaces = 3
>>> nbetoiles = 1
>>> while (i < 3) :
    j=0
    while (j < nbespaces) :
        print(", end= ")
        j=j+1
    k=0
    while( k < nbetoiles) :
        print('*', end=" ")
        k=k+1
    print()
```

```
nbespaces = nbespaces -1  
nbetoiles = nbetoiles +2  
i=i+1
```

>>> Exercice 155

We want to display this

```
***  
***  
***  
***
```

Solutions

```
>>> i =0  
>>> while (i< 4) :  
    j=0  
    while(j<2) :  
        print(, end = "")  
        j=j+1  
    k=0  
    while(k<3) :  
        print(*, end = "")  
        k=k+1  
    print()  
    i=i+1
```

```
***  
***  
***  
***
```

>>>19.6. Maths with while()

>>> Exercise 156

1° Evaluate the following code

```
>>> 1+2+3+4+5+6+7+8+9
```

2° Evaluate the following code

```
>>> sum = 1  
>>> sum = sum +2  
>>> sum = sum + 3  
>>> sum = sum + 4  
>>> sum = sum +5  
>>> sum
```

3° Fill the blanks

```
>>> i = 0  
>>> sum = 0  
>>> while (i <  ) :  
    sum = sum +  
    i=  
>>> print('1+2+3+4+5+6+7+8+9 = ', )
```

Solutions

1°

```
>>> 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
```

```
45
```

$\sum_{i=1}^9 i = 45$

2°

```
>>> sum = 1  
>>> sum = sum +2
```

```
# sum = 1 + 2
# sum = 3
>>> sum = sum + 3
# sum = 3 + 3
# sum = 6
>>> sum = sum + 4
# sum = 6 + 4
# sum = 10
>>> sum = sum +5
# sum = 10 + 5
# sum 15
>>> sum
15
```

3°

```
>>> i = 0
>>>sum =0
>>> while (i < 10) :
    sum = sum + i
    i= i + 1
>>> print('1+2+3+4+5+6+7+8+9 = ', sum )
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45
```

Tips :

```
>>> i = 0
>>> s = 0
>>> while (i<n) :
    s = s + i
```

```
i = i+1
```

>>> Exercice 157

1° evaluate the following code

```
>>> (101) * ((0+100) / 2)
```

2° Evaluate the following code

```
>>> i = 0
>>> sum = 0
>>> while( i < 101) :
    sum = sum + i
    i = i+ 1
>>> print(sum)
```

Solutions

1°

```
>>> (101) * ((0+100) / 2)
5050.0
```

2°

```
>>> i = 0
>>> sum = 0
>>> while( i < 101) :
    sum = sum + i
    i = i+ 1
>>> print(sum)
# print(5050)
```

```
5050
```

>>> Exercice 158

1° evaluate the following code

```
>>> 1 * 2 * 3 * 4 *5 * 6 * 7 * 8 * 9
```

2° evaluate the following code

```
>>> i = 1
>>> prod = 1
>>> while (i<10) :
    prod = prod * i
    i = i+1
>>> prod
```

Solutions

1° On a

```
>>> 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9
362880
```

$$\prod_{i=1}^9 i$$

2°

```
>>> i = 1
>>> prod = 1
>>> while (i<10) :
    prod = prod * i
    i = i+1
>>> prod
```

362880

>>> **Exercise 159**

Evaluate the following code to get 20 !

```
>>> i = 1
>>> p = 1
>>> while (i<21) :
```

```
p = p * i  
i = i + 1  
>>> print(p, 'vaut 20 ! ')
```

Solutions

```
>>> i = 1  
>>> p = 1  
>>> while (i<21):  
    p = p * i  
    i = i + 1  
>>> print(p, 'vaut 20 ! ')  
# print(2432902008176640000 , 'vaut 20 ! ')  
2432902008176640000 vaut 20 !
```

>>> 19.7. Break and continue

>>> 19.7.1. Break

>>> **exercise 160**

Evaluate the following code

```
>>> i=0  
>>> while(i<10):  
    i=i+1  
    if i==5:  
        break  
    print(i)
```

Solutions

iteration	i	i<10	output	i== 5
1	1	True	1	False
2	2	True	2	False
3	3	True	3	False
4	4	True	4	False
5	5	True	5	True

```
>>> i=0
```

```
>>> while(i<10):
```

```
    i=i+1
```

```
    if i==5:
```

```
        break
```

```
    print(i)
```

```
1
```

```
2
```

```
3
```

```
4
```

>>> Exercise 161

Evaluate the following code

```
>>> i = 0
```

```
>>> while(i<3) :
```

```
    if i== 1 :
```

```
        break
```

```
    print(i)
```

```
    i=i+1
```

solutions

```
>>> i = 0
```

```
>>> while(i<3) :
```

```
    if i== 1 :
```

```
        break
```

```
print(i)  
i=i+1
```

```
0
```

>>> 19.7.3. continue

>>> Exercise 162

Evaluate the following code

```
>>> i=0  
>>> while(i<10):  
    i=i+1  
    if i==5:  
        continue  
    print(i)
```

Solutions

```
>>> i=0  
>>> while(i<10):  
    i=i+1  
    if i==5:  
        continue  
    print(i)
```

```
1  
2  
3  
4  
6  
7  
8
```

```
9  
10
```

>>> Exercise 163

Evaluate the following code

```
>>> i = 0  
>>> while(i<3):  
    i = i + 1  
    if i== 1 :  
        continue  
    print(i)
```

Solutions

iteration	i	i < 3	i == 1	output
1	0	True	True	
2	1	True	False	2
3	2	True	False	3
4	3	False		

```
>>> i = 0  
>>> while(i<3):  
    i = i + 1  
    if i== 1 :  
        continue  
    print(i)
```

```
2  
3
```

>>> 20. Lists

« La liste des courses »

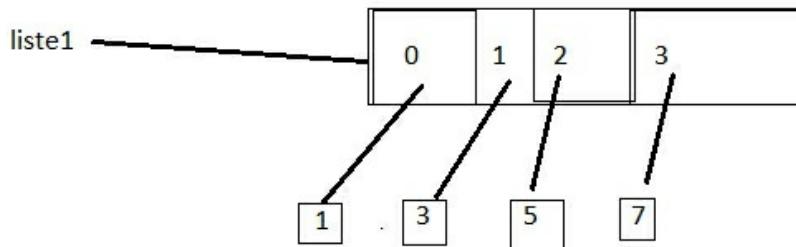
>>> Exercise 164

Evaluate the following code

```
>>> [0,1,2,3,4]  
>>> liste1=[1,3,5,7]  
>>> liste1
```

Solutions

```
>>> [0,1,2,3,4]  
[0,1,2,3,4]  
>>> liste1  
[1, 3, 5, 7]
```



>>> **Exercise 165**

Evaluate the following code

```
>>> []  
>>> liste1 =[]  
>>> liste2 = [2.0,3,0.3]  
>>> type(liste2)
```

Solutions

```
>>> []  
[]
```

Empty list

```
>>> liste1 = []  
>>> liste1  
[]
```

```
>>> liste2 = [2.0,3,3.0]
```

```
>>> liste2
```

```
[2.0, 3, 3.0]
```

```
>>> type(liste2)
```

```
<type 'list'>
```

>>> **Exercise 166**

Evaluate the following code

```
>>> []
```

```
>>> liste1=[5,4.0,3]
```

```
>>> liste1
```

```
>>> type(liste1)
```

Solutions

```
>>> []
```

```
[]
```

```
>>> liste1 = [5, 4.0, 3]
```

```
>>> type(liste1)
```

```
<class 'list'>
```

>>> 20.1.List elements

« numérotions ... »

>>> **Exercise 167**

Evaluate the following code

```
>>> liste1 = [1,2,3]
```

```
>>> liste1[0]
```

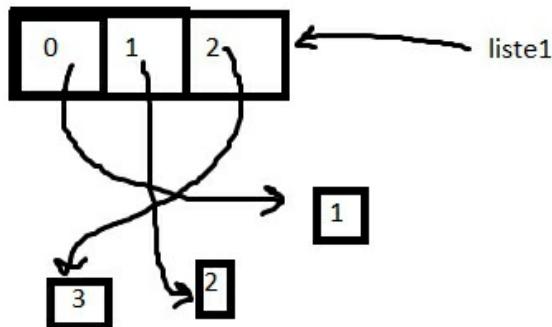
```
>>> liste1[1]
```

```
>>> liste1[2]
```

```
>>> liste1[3]
```

```
>>> liste1[4]
```

Solutions



```
>>> liste1[0]
```

```
1
```

```
>>> liste1[1]
```

```
2
```

```
>>> liste1[2]
```

```
3
```

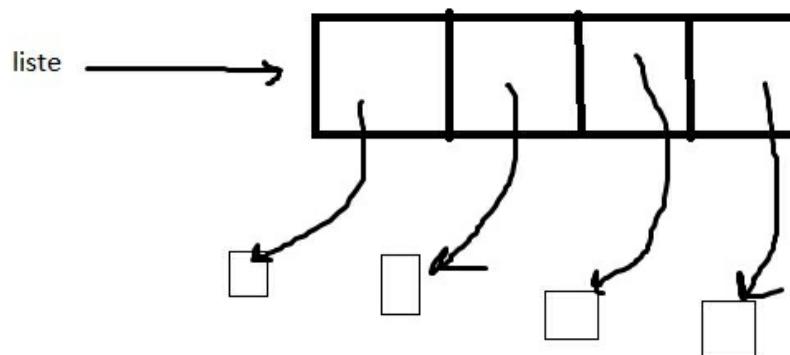
```
>>> liste1[4]
```

IndexError : list index out of range

>>> Exercise 168

```
>>> liste = [1,3,5,7]
```

1° Complete this picture



2° Evaluate the following code

```
>>> liste[9]
```

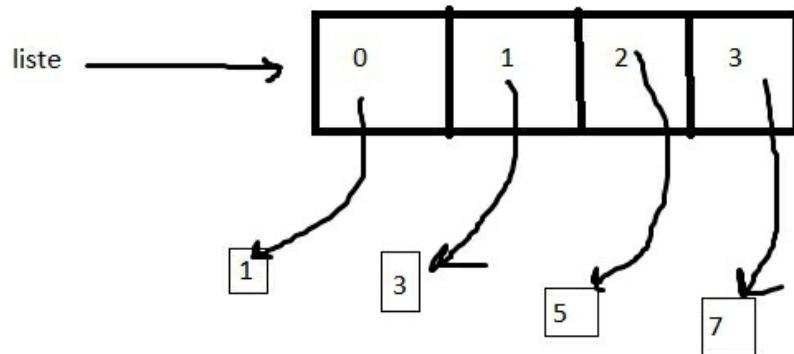
3° Evaluate the following code

```
>>> a = liste[0]
```

```
>>> a
```

Solutions

1°



2° We got a semantic error

```
>>> liste[9]
```

IndexError : index out of range

3°

```
>>> a = liste[0]
```

```
# a = 1
```

```
>>> a
```

```
1
```

>>> 20.2. Manipulate list elements

>>> **Exercise 169**

Evaluate the following code

```
>>> liste = [1, 2.0 ,3]
```

```
>>> liste[0]
```

```
>>> print(liste[0], liste[1], liste[2])
```

```
>>> liste[0] = 4
```

```
>>> liste
```

Solutions

```
>>> liste=[1,2.0,3]
```

```
>>> liste[0]
```

```
1
```

Function print() takes three arguments :

Arguments : lists elements

```
>>> print(liste[0], liste[1], liste[2])
```

```
1 2.0 3
```

```
>>> liste[0] = 4
```

```
>>> liste[0]
```

```
4
```

```
>>> liste
```

```
[4, 2.0, 3]
```

Tips

```
>>> liste = ['a','b','c']
```

```
>>> liste[0] = 'd'
```

```
>>> liste
```

```
['d','b','c']
```

Object list is **mutable**

>>> **Exercise 170**

Evaluate the following code

```
>>> var = [4,2,4,5]
```

```
>>> var[2]
```

```
>>> print(var)
```

```
>>> var[0] = 4
```

```
>>> var[1]=4
```

```
>>> var
```

Solutions

```
>>> var = [4, 2, 4, 5]
```

```
>>> var[2]
```

```
4
```

```
>>> print(var)
```

```
[4, 2, 4, 5]
```

```
>>> var[0]= 4
```

```
# [0, 2, 4, 5]
```

```
>>> var[1] = 4
```

```
# [0,1,4,5]
```

```
>>> var
```

```
[0, 1, 4, 5]
```

>>> Exercise 171

Evaluate the following code

```
>>> liste = [0,1,2,3,4,5]
```

```
>>> i = 0
```

```
>>> while(i<6) :
```

```
    liste[i]= liste[i]+1
```

```
    i = i+1
```

```
>>> liste
```

Solutions

```
iteration liste[i] = liste[i] + 1
1      liste[0] = liste[0] + 1 = 0 + 1 = 1
2      liste[1] = liste[1] + 1 = 1 + 1 = 2
3      liste[2] = liste[2] + 1 = 2 + 1 = 3
4      liste[3] = liste[3] + 1 = 3 + 1 = 4
5      liste[4] = liste[4] + 1 = 4 + 1 = 5
6      liste[5] = liste[5] + 1 = 5 + 1 = 6
```

```
>>> liste
```

```
[1, 2, 3, 4, 5, 6]
```

>>> 20.3. while loop() and list

>>> **Exercise 172**

function len()

```
>>> help(len)
```

```
len (obj, /)
```

Return the number of items in a container

A list is a container

1° Evaluate the following code

```
>>> liste = [0,1,2,3,4,5]
```

```
>>> len(liste)
```

2° Evaluate the following code

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]
```

```
>>> len(liste)
```

```
>>> i = 0
```

```
>>> while (i<len(liste)) :
```

```
    print(liste[i], end = "")
```

```
    i=i+1
```

Solutions

1°

```
>>> liste = [0,1,2,3,4,5]
```

```
>>> len(liste)
```

```
6
```

2°

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]
```

```
>>> len (liste)
```

```
11
```

Function name : len

Arguments : a list

```
>>> i =0
```

```
>>> while (i < len(liste)) :
```

```
    print(liste[i])
```

```
    i= i+1
```

iteration	i < len(liste) = 11	output
1	True	0
2	True	1
3	True	2
4	True	3
5	True	4
6	True	5
7	True	6
8	True	7
9	True	8
10	True	9
11	True	10

>>> Exercise 173

Evaluate the following code

```
>>> l = [5,3,2,4,2,3]
```

```
>>> i =0
```

```
>>> while (i < len(l) ) :
```

```
    print(l[i], end = ' ')
```

```
i=i+1
```

Solutions

```
>>> l = [5,3,2,4,2,3]
>>> i =0
>>> while (i < len(l) ) :
    print(l[i], end = ' ')
    i=i+1
```

```
5 3 2 4 2 3
```

>>> Exercise 174

Evaluate the following code

```
>>> ls = [0,0,0,0]
>>> i =0
>>> while (i < len(ls)) :
    ls[i]=1
    i=i+1
```

Solutions

```
>>> i =0
>>> while (i < len(ls)) :
    ls[i]=1
    i=i+1
>>> ls
[1, 1, 1, 1]
```

Tips :

```
>>> i =0
>>> while (i < len(liste) ) :
    block
```

>>> Exercise 175

1° Evaluate the following code

```
>>> notes =[0,0,0,0]
>>> notes[0] = int(input())
12
>>> notes[0]
```

2° Evaluate the following code

```
>>> i =0
>>> while (i < len(notes)) :
    print('note ', i+1, '=', notes[i])
    i=i+1
```

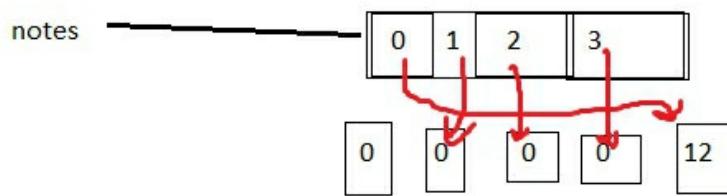
3° Evaluate the following code

```
>>> notes = [0,0,0,0]
>>> i = 0
>>> while (i < len(notes)) :
    notes[i] = int(input('give a value : '))
    i = i+1
```

Corrigé

1°

```
>>> notes =[0,0,0,0]
>>> notes[0] = int(input())
12
>>> notes[0]
12
>>> notes
[12, 0, 0, 0]
```



2°

```
>>> i = 0
>>> while (i < len(notes)) :
    print('note ', i+1, '=', notes[i])
    i = i + 1
```

```
note 1 = 12
note 2 = 0
note 3 = 0
note 4 = 0
```

3°

```
>>> notes = [0,0,0,0]
>>> i = 0
>>> while (i < len(notes)) :
    notes[i] = int(input('give a value '))
    i = i + 1
```

```
give a value 10
give a value 11
give a value 12
give a value 13
```

```
>>> notes
```

```
[10, 11, 12, 13]
```

>>> **Exercice 176**

Evaluate the following code by giving five values

```
>>> l = [0,1,2,3,4]
>>> i=0
>>> while (i<len(l)) :
    l[i] = int(input('give a value'))
    i=i+1
>>> l
```

Corrigé exercice 176

```
>>> l = [0,1,2,3,4]
>>> i=0
>>> while (i<len(l)) :
    l[i] = int(input('give a value '))
    i=i+1
give a value 10
give a value 20
give a value 30
give a value 40
give a value 50
>>> l
```

[10, 20, 30, 40, 50]

>>> 20.3.1. List maximum

>>> **Exercise 177**

1° Evaluate the following code

```
>>> liste = [1,2,3]
>>> if (liste[0] > liste[1] and liste[0] > liste[2]) :
    print(liste[0], 'is the maximum')
elif (liste[1] > liste[0] and liste[1] > liste[2]) :
```

```
    print(liste[1], 'is the maximum')
else :
    print(liste[2], 'is the maximum')
```

2° Evaluate the following code

```
>>> max = liste[0]
>>> i=0
>>> while (i<len(liste)) :
    if (liste[i] > max ) :
        max = liste[i]
    i=i+1
>>> print(max, ' is the maximum')
```

Solutions

1°

```
>>> if (liste[0] > liste[1] and liste[0] > liste[2]) :
    # if (1 > 2 and 1 > 3) :
    # if (False and False) :
    # if (False) :
    elif (liste[1] > liste[0] and liste[1] > liste[2]) :
        # elif (2 > 1 and 2 > 3) :
        # elif (True and False) :
        # elif (False) :
    else :
        print(liste[2], ' is the maximum')
```

3 is the maximum

2°

```
>>> liste = [1,2,3]
```

```
>>> max = liste[0]
# max = 1
>>> i=0
>>> while (i<len(liste)) :
# while (i < 3) :
# while (True) :
#         if (liste[i] > max ) :
#             if (liste[0] > max ) :
#                 if (1 > 1) :
#                     if (False)
# i=0+1
#             i = 1
# while (1 < 3 ) :
# if (liste[1] > max) :
# if (2 > 1) :
# if True :
#             max = liste[1]
#             max = 2
# i=2
# while (2 <3) :
# if (liste[2] > max ) :
# if (3 > 2) :
# max = liste[2]
# max  = 3
# i = 3
# while (3 < 3) :
```

```
# Sortie de la boucle
>>> print(max, ' is the maximum')
3 is the maximum
```

```
iteration i < len(liste) if (liste[i] > max) max
1      True      False      1
2      True      True       2
3      True      True       3
```

Tips

- `max = liste[0]`
- `if liste [i] > max`
- `max = liste[i]`

>>> Exercise 178

Evaluate the following code

```
>>> a = [0,1,2,3,4,5,6,7,8,9]
```

```
>>> max = a[0]
>>> i=0
>>> while (i < len(a)) :
    if (a[i] > max) :
        max = a[i]
    i=i+1
>>> max
```

Solutions

iteration	<code>a[i] > max</code>	<code>max = a[i]</code>
1	False	
2	True	1
3	True	2
4	True	3
5	True	4
6	True	5
7	True	6
8	True	7
9	True	8
10	True	9

```
>>> max
```

```
9
```

>>> 20.3.2. List Minimum

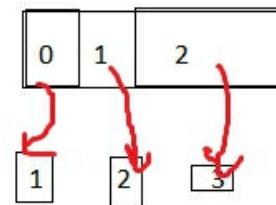
>>> **Exercise 179**

Evaluate the following code

```
>>> liste = [1,2,3]
>>> min = liste[0]
>>> i= 0
>>> while (i < len(liste)) :
    if (liste[i] < min ) :
        min = liste[i]
    i=i+1
>>> print(min, 'is the minimum of the list')
```

Solutions

iteration	<code>i < len(liste)</code>	<code>min</code>	<code>liste[i] < min</code>
1	True	1	False
2	True	1	False
3	True	1	False



```
>>> print(min, 'is the minimum of the list')
```

```
1 is the minimum of the list
```

>>> **Exercise 180**

Evaluate the following code

```
>>> a = [7,3,3,3,4,1,6,7,8,9]
>>> min = a[0]
>>> i=0
```

```
>>> while(i< len(a)) :  
    if(a[i] < min) :  
        min = a[i]  
    i=i+1
```

Solutions

```
>>> a = [7,3,3,3,4,1,6,7,8,9]  
>>> min = a[0]  
>>> i=0  
>>> while(i< len(a)) :  
    if(a[i] < min) :  
        min = a[i]  
    i=i+1  
>>> min  
1
```

>>> 20.3.3. Some stats with lists

>>> **Exercise 181 (Sum)**

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]
```

Soit la liste suivante

```
>>> s = 0  
>>> i = 0  
>>> while (i <11) :  
    s=s+i  
    i=i+1
```

2° Evaluate the following code

```
>>> s = 0  
>>> i = 0  
>>> while (i < len(liste)) :
```

```
s= s+liste[i]  
i=i+1
```

Solutions

1°

```
>>> s = 0  
>>> i = 0  
>>> while (i <11) :  
    s=s+i  
    i=i+1  
>>> s
```

55

2°

```
>>> s = 0  
>>> i = 0  
>>> while (i < len(liste)) :  
    s= s + liste[i]  
    i=i+1  
>>> s
```

55

```

iteration i < len(liste) s = s + liste[i]
1      True      s = 0 + 0 = 0
2      True      s = 0 + 1 = 1
3      True      s = 1 + 2 = 3
4      True      s = 3 + 3 = 6
5      True      s = 6 + 4 = 10
6      True      s = 10 + 5 = 15
7      True      s = 15 + 6 = 21
8      True      s = 21 + 7 = 28
9      True      s = 28 + 8 = 36
10     True      s = 36 + 9 = 45
11     True      s = 45 + 10 = 55

```

Tips :

```

>>> liste =[1,3,4,17,20]
>>> i=0
>>> s=0
>>> while (i < len(liste)) :
    s=s+ liste[i]
    i=i+1
>>> s
45

```

>>> Exercise 182

1° Sum the éléments of this list

```
>>> liste1 =[73,27974,97979]
```

2° Multiply all éléments of this list

```
>>> liste2 = [470,8408,80808]
```

Solutions

1°

```

>>> liste1 =[73,27974,97979]
>>> i= 0
>>>s =0

```

```
>>> while (i< len(liste1)) :  
    s = s + liste1[i]  
    i=i+1  
>>> s
```

```
126026
```

2°

```
>>> liste2 = [470,8408,80808]  
>>> i = 0  
>>> p =1  
>>> while (i< len(liste2)) :  
    p = p * liste2[i]  
    i=i+1  
>>> p
```

```
319333822080
```

>>> Exercise 183 (average)

1° Evaluate the following code

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]  
>>> somme,i=0,0  
>>> while (i <(len(liste)) :  
    somme = somme + liste[i]  
    i=i+1  
>>> somme
```

2° Evaluate the following code

```
>>> average = somme / (len(liste))
```

Solutions

1°

```
>>> somme,i=0,0
>>> while (i <(len(liste)) :
    somme = somme + liste[i]
    i=i+1
>>> somme
55
```

2°

```
>>> average = somme / (len(liste))
# average = somme / (11)
# average = 55 / 11
# average = 5.0
5.0
```

>>> **Exercise 184**

Find the average of this list

```
>>> liste = [26,4748,26537,198]
```

Solutions

```
>>> liste = [26,4748,26537,198]
>>> i 0
>>> s = 0
>>> while(i<len(liste)) :
    s= s +liste[i]
    i=i+1
>>> moyenne = s / (len(liste))
# moyenne = 31509 / 4
# moyenne = 7877.25
7877.25
```

>>> 20.3.4. Find an element in a list

>>> **Exercise 185**

1°

```
>>> liste = [0,1,2,3]
```

1° Is 3 an element of this list

2° Evaluate the following code

```
>>> if (liste[0] == 3) :  
    print("3 est dans la liste")  
elif(liste[1]==3) :  
    print("3 est dans la liste")  
elif(liste[2]==3) :  
    print("3 est dans la liste")  
elif(liste[3] == 3) :  
    print("3 est dans la liste")  
else :  
    print("3 n'est pas dans la liste")
```

3° why this code is incorrect ?

```
>>> i = 0  
>>> while (i < len(liste)) :  
    if (liste[i]==3) :  
        print("3 est dans la liste")  
    i=i+1  
else :  
    print(" 3 n'est pas dans la liste" )
```

4° Evaluate the following code

```
>>> i= 0  
>>> b = False
```

```
>>> while (i< len(liste)) :  
    if (liste[i] == 3) :  
        b = True  
    i=i+1
```

5° Evaluate the following code

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]  
>>> i = 0  
>>> while (i<len(liste)) :  
    print(liste[i], end=' ')  
    i=i+1  
>>> nombre = int(input('entrez un nombre entier'))  
>>> i=0  
>>> b = False  
>>> while (i <len(liste)) :  
    if (liste[i] ==  nombre) :  
        b= True  
    i=i+1  
>>> if b :  
    print(nombre, ' est dans la liste ')  
else :  
    print(nombre, " n'est pas dans la liste")
```

Solutions

1° 3 is an element of this list

2°

```
>>> liste = [0,1,2,3]  
>>> if (liste[0] == 3) :
```

```
# if (1 == 3) :  
# if (False) :  
# elif(liste[1]==3) :  
# elif (1 == 3) :  
# elif(False) :  
# elif(liste[2]==3) :  
# elif (2 == 3) :  
# elif(False) :  
# elif(liste[3] == 3) :  
# elif (3 == 3) :  
# elif(True) :  
# print("3 est dans la liste")
```

3 est dans la liste

3°

```
>>> i = 0  
>>> while (i < len(liste)) :  
    if (liste[i]==3) :  
        print("3 est dans la liste")  
        i=i+1  
    else :  
        print(" 3 n'est pas dans la liste")
```

4°

```
>>> i= 0  
>>> b = False  
>>> while (i< len(liste)) :  
    if (liste[i] == 3) :
```

b = True

i=i+1

iteration	i < len(liste)	liste[i] == 3
1	True	False
2	True	False
3	True	False
4	True	True

5°

```
>>> liste = [0,1,2,3,4,5,6,7,8,9,10]
```

```
>>> i = 0
```

```
>>> while (i<len(liste)) :
```

```
    print(liste[i], end=' ')
```

```
    i=i+1
```

```
0 1 2 3 4 5 6 7 8 9 10
```

```
>>> nombre = int(input('entrez un nombre entier'))
```

5

```
>>> i=0
```

```
>>> b = False
```

```
>>> while (i <len(liste)) :
```

```
    if (liste[i] == nombre) :
```

```
# if (liste[5] == 5 ) :
```

```
# if (True) :
```

```
# b= True
```

```
    i= i +1
```

```
>>> if b :
```

```
# if True
```

```
# print(nombre, ' est dans la liste ')
```

5 est dans la liste

Tips

```
bool1= False
while ( i < len(list) )
    if (list[i] == element) :
        bool1= True
```

>>> Exercice 186

Soit la liste suivante :

```
>>> liste = [-1,2,3,4]
```

1° Is -1 and 10 belong to this list ?

2° evaluate the following code

```
>>> liste = [-1,2,3,4]
>>> i= 0
>>> test = False
>>> while (i < len(liste)) :
    if (liste[i] == -1) :
        test = True
        i=i+1
    if (test) :
        print('-1 est dans la liste')
    else :
        print('pas dans la liste')
```

Solutions

1° -1 and 10 are not in the list

2°

```
>>> liste = [-1,2,3,4]
>>> i= 0
```

```
>>> test = False
>>> while (i < len(liste)) :
    if (liste[i] == -1) :
        test = True
        i=i+1
>>> if (test) :
    print('-1 est dans la liste')
else :
    print('pas dans la liste')
-1 est dans la liste
```

And

```
>>> liste = [-1,2,3,4]
>>> i= 0
>>> test = False
>>> while (i < len(liste)) :
    if (liste[i] == 10) :
        test = True
        i=i+1
>>> if (test) :
# if (False) :
else :
    print('pas dans la liste')
pas dans la liste
```

>>> 20.3.5. Find the list index

>>> **Exercise 185**

Soit la liste suivante

```
>>> liste = [8,3,5,2,9]
```

1° Give the index of element 5 ?

2° Evaluate the following code

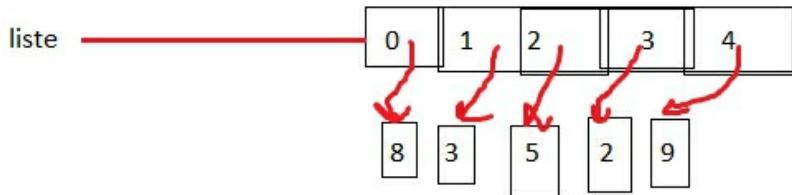
```
>>> i = 0
>>> indice = -1
>>> while (i<len(liste)) :
    if (liste[i]==5) :
        indice = i
    i=i+1
>>> if (indice != -1)
    print("5 appartient à la liste en ",i," ième position")
```

3° Evaluate the following code

```
>>> liste = [8,3,5,2,9]
>>> indice = - 1
>>> i = 0
>>> nombre_entre = int(input("Entrez un nombre entier"))
>>> while (i< len(liste)) :
    if (liste[i] == nombre_entre) :
        indice = i
    i=i+1
>>> if (indice!= -1) :
    print(nombre_entre, "est dans la liste en", indice, " ième position")
else :
    print(nombre_entre, " n'est pas dans la liste")
```

Solutions

1° 5 belongs to the list, his index is 2



2°

```
>>> liste = [8,3,5,2,9]
>>> i = 0
>>> indice = -1
>>> while (i<len(liste)) :
    if (liste[i]==5) :
        indice = i
    i=i+1
# if (liste[2] == 5 ) :
# if (5 == 5) :
# if (True) :
# indice = 2
>>> if (indice != -1) :
# if (2 != -1) :
# if (True)
# print("5 appartient à la liste en ",i,"ième position")
5 appartient à la liste en 2 ième position
```

3°

```
>>> liste = [8,3,5,2,9]
>>> indice = -1
>>> i = 0
>>> nombre_entre = int(input("Entrez un nombre entier"))
```

Entrez un nombre entier 2

```
>>> while (i< len(liste)) :  
    if (liste[i] == nombre_entre) :  
        indice = i  
        i=i+1  
    # if (liste[3] == nombre_entre) :  
    # if (2 == 2) :  
    # if (True) :  
    # indice = 3  
>>> if (indice!= -1) :  
    # if ( 3!= -1) :  
    # if (True) :  
    # print(nombre_entre, "est dans la liste en", indice, "ieme position)
```

2 est dans la liste en 3 ieme position

Tips

```
indice = -1  
while( i < len(liste)) :  
    If(liste[i] == element) :  
        indice = i
```

>>> Exercise 186

1° Evaluate the following code

```
>>> liste = [1,4,3,8,9]  
>>> Test = False  
>>> i = 0  
>>> while( i < len(liste)) :  
    if (liste[i] == 4) :  
        Test = True
```

```
i=i+1  
>>> if (Test) :  
    print('4 is in the list')  
else :  
    print('nope')
```

2° Evaluate the following code

```
>>> liste = [1,4,3,8,9]  
>>> indice = - 1  
>>> i = 0  
>>> while( i < len(liste)) :  
    if (liste[i] == 4) :  
        indice = i  
    i=i+1  
>>> print('4 index is', indice)
```

Solutions

```
>>> liste = [1,4,3,8,9]  
>>> Test = False  
>>> i = 0  
>>> while( i < len(liste)) :  
    if (liste[i] == 4) :  
        Test = True  
    i=i+1  
>>> if (Test) :  
    print('4 is in the list')  
else :  
    print('nope')
```

'4 is in the list'

2°

```
>>> liste = [1,4,3,8,9]
>>> indice = - 1
>>> i = 0
>>> while( i < len(liste)) :
    if (liste[i] == 4) :
        indice = i
    i=i+1
>>> print('4 index is', indice)
```

4 index is 1

>>> 21. Back to String

« hello world ! » un codeur

>>> **Exercise 187**

Evaluate the following code

```
>>> mot = 'lol'
>>> mot[0]
>>> mot[1]
>>> mot[2]
>>> mot[3]
>>> i= 0
>>> while (i<len(mot)) :
    print(mot[i])
    i=i+1
```

Solutions

```
>>> type(mot)
```

```
<class 'str'>
```



```
>>> mot[0]
```

```
'l'
```

```
>>> mot[1]
```

```
'o'
```

```
>>> mot[2]
```

```
'l'
```

```
>>> mot[3]
```

```
IndexError : string index out of range
```

```
>>> while (i<len(mot)) :
```

```
    print(mot[i])
```

```
    i=i+1
```

```
l
```

```
o
```

```
l
```

iteration	<code>i < len(mot)</code>	<code>print(mot[i])</code>
1	True	l
2	True	o
3	True	l

```
>>> Exercise 188
```

Evaluate the following code

```
>>> var = 'hello'
```

```
>>> var[0]
```

```
>>> var[1]
```

```
>>> i = 0
```

```
>>>while (i<len(var)) :  
    print(var[i],end=' ')  
    i=i+1
```

Solutions

```
>>> var[0]  
'h'  
>>> var[1]  
'e'  
>>> while (i<len(var)) :  
    print(var[i],end=' ')  
    i=i+1  
h e l l o
```

>>> 21.1. Concatenation operator +
>>> **Exercise 189**

Evaluate the following code

```
>>> chaine1 = 'Hello'  
>>> chaine2 = ' world'  
>>> chaine3 = chaine1 + chaine2
```

Solutions

Expression : chaine1 + chaine 2

Operands : chaine1(str) chaine2(str)

Operator : + (concatenation)

```
>>> chaine3 = chaine1 + chaine2  
# chaine3 = chaine1 + chaine2  
# chaine3 = 'Hello' + chaine2  
# chaine3 = 'Hello' + ' world'  
# chaine3 = 'Hello world'
```

```
>>> chaine3
```

```
'Hello world'
```

>>> **Exercise 190**

Evaluate the following code

```
>>> chaine1 = 'aaaaaa'
```

```
>>> chaine2 = 'bbbbbb'
```

```
>>> chaine3 = chaine1 + chaine2
```

```
>>> 'aaaaa ' + 'bbbbbb'
```

Solutions

```
>>> chaine3 = chaine1 + chaine2
```

```
# chaine3 = 'aaaaaa' + 'bbbbbb'
```

```
# chaine3 = 'aaaaaabbffff'
```

```
>>> 'aaaaa ' + 'bbbbbb'
```

```
'aaaaa bbbbb'
```

>>> 21.2. while() and string

>>> **Exercise 191**

Soit le code suivant

```
>>> chaine1 = 'aaabaaa'
```

1° Does character **b** belongs to this string ?

2° Evaluate the following code

```
>>> len(chaine1)
```

```
>>> chaine1[2] == 'b'
```

```
>>> chaine1[3] == 'b'
```

```
>>> chaine1[4] == 'b'
```

3° Evaluate the following code

```
>>> appartient = False
```

```
>>> i = 0
>>> while (i< len(chaine1)) :
    if (chaine1[i]) == 'b':
        appartient = True
    i=i+1
>>> if (appartient) :
    print('b is in')
else :
    print("b is not")
```

Solutions

1° 'aaabaaa'

2°

Function name : len

Argument : chaine1 (str)

Return : string length

```
>>> len(chaine1)
7
>>> chaine1[2] == 'b'
# 'a'== 'b'
# False
>>> chaine1[3]== 'b'
# 'b' == 'b'
# True
>>> chaine1[4]== 'b'
# 'a' == 'b'
# False
```

3°

```
>>> appartient = False
>>> i = 0
>>> while (i< len(chaine1)) :
    if (chaine1[i]) == 'b' :
        appartient = True
    i=i+1
>>> if (appartient) :
    print('b is in')
else :
    print("b is not")
```

b is in

>>> **Exercise 192**

1° evaluate the following code

```
>>> chaine1= 'manga'
>>> caractereR = input('what are you looking for ? ')
```

2° Fill the blanks

```
>>> chaine1= 'manga'
>>> caractereR = input('what are you looking for?')
>>> appartient
>>> i=
>>> while (i<      ) :
    if (chaine[i] ==      ) :
        appartient =
    i=i+1
>>> if (appartient) :
    print(caractereR, ' belongs to')
```

```
else :  
    print(caractereR, " doesn't belong")
```

Solutions

1°

Help on built-in function input in module builtins:

```
input(prompt=None, /)
```

Read a string from standard input. The trailing newline is stripped.

caractereR will be a string type

```
>>> caractereR = input('what are you looking for ?')
```

What are you looking for ?**a**

```
>>> caractereR
```

```
'a'
```

2°

```
>>> chaine1= 'manga'  
>>> caractereR = input('what are you looking for ?')
```

What are you looking for?**a**

```
>>> appartient = False
```

```
>>>i=0
```

```
>>> while (i< len(chaine1) ) :
```

```
    if (chaine[i] == caractereR) :
```

```
        appartient = True
```

```
    i=i+1
```

```
>>> if (appartient) :
```

```
    print(caractereR, ' belongs to')
```

```
else :
```

```
    print(caractereR, " doesn't belong to ")
```

```
a belongs to
```

>>> Exercise 193

```
>>> chaine1 = 'manga'
```

1° how many a in chaine1 ?

2° Evaluate the following code

```
>>> chaine1 = 'manga'
```

```
>>> compteur = 0
```

```
>>> i = 0
```

```
>>> while( i< len(chaine1)) :
```

```
    if (chaine1[i] == 'a') :
```

```
        compteur = compteur +1
```

```
    i=i+1
```

```
>>> print("character 'a' appears ", compteur, ' times in ', chaine1)
```

Solutions

1° manga

2°

iteration	i < len(chaine1)	chaine1[i] == 'a'	compteur
1	True	False	0
2	True	True	1
3	True	False	1
4	True	False	1
5	True	True	2

```
>>> print("character 'a' appears ", compteur, ' times in ', chaine1)
```

```
character 'a' appears 2 times in manga
```

>>> Exercise 194

```
>>> chaine1 = 'probabilites'
```

1° How many i and b ?

2° Evaluate the following code

```
>>> chaine1 = 'probabilites'
```

```
>>> i=0
```

```
>>>compteur_i = 0
>>> while (i < len(chaine1) ) :
    if(chaine1[i] == 'i' ) :
        compteur_i = compteur_i + 1
        i = i +1
>>> compteur_i
>>> i = 0
>>> compteur_b = 0
>>> while (i < len(chaine1) ) :
    if(chaine1[i] == 'b' ) :
        compteur_b = compteur_b + 1
        i = i +1
>>> compteur_b
```

Solutions

1° probabilités

2°

```
>>> chaine1 = 'probabilites'
>>> i=0
>>>compteur_i = 0
>>> while (i < len(chaine1) ) :
    if(chaine1[i] == 'i' ) :
        compteur_i = compteur_i + 1
        i = i +1
>>> compteur_i
2
>>> i = 0
```

```
>>> compteur_b = 0
>>> while (i < len(chaine1) ) :
    if(chaine1[i] == 'b') :
        compteur_b = compteur_b + 1
    i = i +1
>>> compteur_b
```

2

>>> Exercise 195

Fill the blanks

```
>>> chaine_entre = input('give a sentence ')
>>> caractere_recherche = input('give a char ? ')
>>> compteur =0
>>> i = 0
>>> while (i<len(chaine_entre)) :
    if (chaine_entre[i] == caractere_recherche) :
        compteur = compteur + 1
    i = i+1
>>> print('char', caractere_recherche, 'appears ', compteur, 'times')
```

Solutions

```
>>> chaine_entre = input('give a sentence ')
entrez une phrase ou un mot : maths and physics
>>> caractere_recherche = input('give a char ? ')
Give a char ? h
>>> print('char', caractere_recherche, 'appears ', compteur, 'times')
# print('char', 'h', 'appears', 2, 'times')
char h appears 2 times
```

iteration	if(chaine_entre[i] == 'h')	compteur
1	False	
2	False	
3	False	
4	True	1
5	False	
6	False	
7	False	
8	False	
9	False	
10	False	
11	False	
12	True	2
13	False	
14	False	
15	False	
16	False	
17	False	

>>> Exercise 196

```
vowels = {'a','e','i','o'}
```

Evaluate the following code

```
>>> chaine='anticonstitutionnellement'
>>> vowels=0
>>> consonne=0
>>> i=0
>>> while(i< len(chaine)) :
    if(chaine[i]== 'a' or chaine[i] == 'e' or chaine[i] =='i' or chaine[i]=='o') :
        vowels = vowels +1
    else :
        consonne = consonne +1
    i=i+1
>>> print(" there is ", vowels, 'vowels and ', consonne, 'consonnes.')
```

Solutions

```
A N T I C O N S T I T U T I O N N E L L E M E N T  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
>>> vowels
```

```
9
```

```
>>> consonne
```

```
16
```

```
>>> print(" there is ", vowels, 'vowels and ', consonne, 'consonnes.')
```

```
There is 9 vowels and 16 consonnes.
```

```
A N T I C O N S T I T U T I O N N E L L E M E N T  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

```
>>> 21.3. Copier une chaîne à l'aide de l'opérateur de concaténation
```

```
>>> Exercise 197
```

```
1° Evaluate the following code
```

```
>>> chaine = ''  
>>> chaine1= 'ab'  
>>> chaine = chaine + 'a'  
>>> chaine = chaine + 'b'  
>>> chaine == chaine1
```

```
2° Evaluate the following code
```

```
>>> chaine1 = 'anticonstitutionnellement'  
>>> ch_copie = ''  
>>> i = 0  
>>> while(i< len(chaine1)) :  
    ch_copie = ch_copie + chaine1[i]
```

```
i=i+1  
>>> ch_copie
```

Solutions

```
>>> chaine = ''  
>>> chaine1 = 'ab'  
>>> chaine = chaine + 'a'  
# chaine = ' ' + 'a'  
# chaine = 'a'  
>>> chaine = chaine + 'b'  
# chaine = 'a' + 'b'  
# chaine = 'ab'  
>>> chaine == chaine1  
# 'ab' == 'ab'  
# True  
True
```

2°

```

iteration  i < len(chaine1)    ch_copie = ch_copie + chaine1[i]
1          True                  '' + 'a'
2          True                  'a' + 'n'
3          True                  'an' + 't'
4          True                  'ant' + 'i'
5          True                  'anti' + 'c'
6          True                  'antic' + 'o'
7          True                  'antico' + 'n'
8          True                  'anticon' + 's'
9          True                  'antincons' + 't'
10         True                  'anticonst' + 'i'
11         True                  'anticonsti' + 't'
12         True                  'anticonstit' + 'u'
13         True                  'anticonstitu' + 't'
14         True                  'anticonstitut' + 'i'
15         True                  'anticonstituti' + 'o'
16         True                  'anticonstitutio' + 'n'
17         True                  'anticonstitution' + 'n'
18         True                  'anticonstitutionn' + 'e'
19         True                  'anticonstitutionne' + 'l'
20         True                  'anticonstitutionnel' + 'l'
21         True                  'anticonstitutionnell' + 'e'
22         True                  'anticonstitutionnelle' + 'm'
23         True                  'anticonstitutionnellem' + 'e'
24         True                  'anticonstitutionnelleme' + 'n'
25         True                  'anticonstitutionnellemen' + 't'

```

```

>>> ch_copie
anticonstitutionnellement

```

Tips : copy a string

```

string_copy = ''
while (i < len(string_original)) :
    string_copy = string_copy + string_original
    i = i+1

```

>>> **Exercise 198**

Evaluate the following code

```

>>> ch = 'raikage'

```

```

>>> chc = ''
>>> while (i< len(ch)) :
    chc = ch[i] + chc
    i=i+1
>>> chc

```

2° write the right to code to get :

```

>>> chc == ch
True

```

Solutions

1°

iteration	i < len(ch)	chc = ch[i] + chc
1	True	'r' + ''
2	True	'a' + 'r'
3	True	'i' + 'ar'
4	True	'k' + 'iar'
5	True	'a' + 'kiar'
6	True	'g' + 'akiar'
7	True	'e' + 'gakiar'

```

>>> chc
egakiar

```

2°

```

>>> ch = 'raikage'
>>> chc = ''
>>> while (i< len(ch)) :
    chc = chc + ch[i]
    i=i+1
>>> chc
raikage

```

>>> 22. Lists and Strings

>>> Exercise 199

1° Evaluate the following code

```
>>> ch1 = 'hello'  
>>> liste1 = ['h','e','l','l','o']  
>>> len(ch1) == len(liste1)  
>>> print(ch1)  
>>> print(liste1)  
>>> type(ch1)  
>>> type(liste1)
```

2° Evaluate the following code

```
>>> i = 0  
>>> n = 0  
>>> while(i< len(ch1)) :  
    if(ch1[i] == liste1[i]) :  
        n=n+1  
    i=i+1
```

Solutions

1°

```
>>> len(ch1) == len(liste1)  
# 5 == 5  
# True  
True  
>>> print(ch1)  
hello  
>>> print(liste1)
```

```
[h', 'e', 'T', 'T', 'o']
```

```
>>> type(ch1)
```

```
<class 'str'>
```

```
>>> type(liste1)
```

```
<class 'list'>
```

2°

```
>>> i = 0
```

```
>>> n = 0
```

```
>>> while(i < len(ch1)) :
```

```
    if(ch1[i] == liste1[i]) :
```

```
        n=n+1
```

```
    i=i+1
```

iteration	i < len(ch1)	ch1[i] == liste1[i]	n = n +1
1	True	'h' == 'h'	1
2	True	'e' == 'e'	2
3	True	'T' == 'l'	3
4	True	'T' == 'l'	4
5	True	'o' == 'o'	5

>>> 22.1. Copying lists with method `append()`

>>> **Exercise 200**

Evaluate the following code

```
>>> liste1 = ['a','b']
```

```
>>> liste = []
```

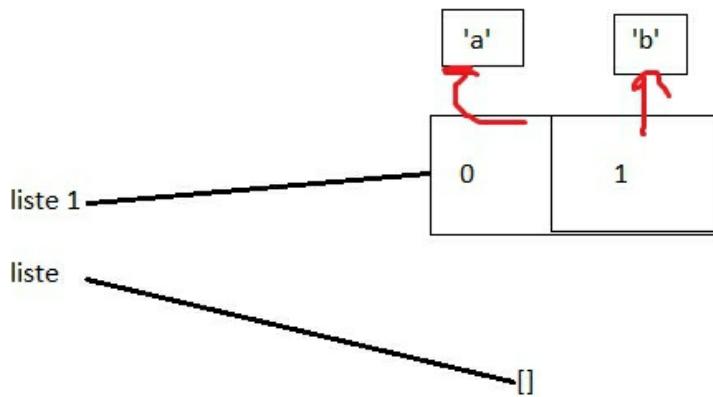
```
>>> liste1[0]
```

```
>>> liste[0] = 1
```

```
>>> liste.append(1)
```

```
>>> liste
```

Solutions



```
>>> liste1[0]
'a'
>>> liste[0] = 1
```

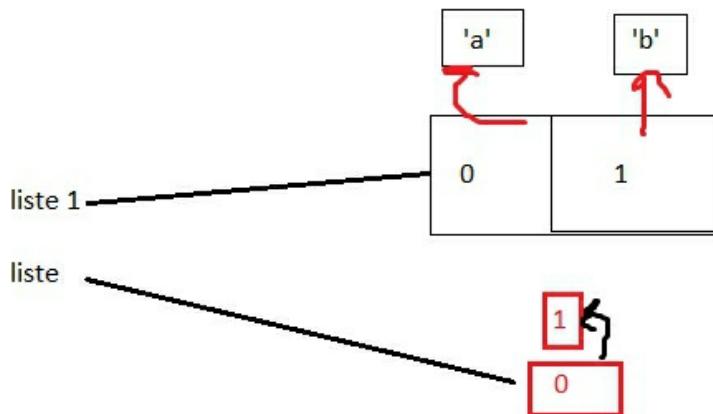
IndexError : list assignment index out of range

```
>>> liste.append(1)
```

```
# append(liste,1)
```

```
>>> liste
```

```
[1]
```



Tips :

```
>>> list.append(argument)
```

Argument type : int, float, boolean, list, str

>>> Exercise 201

Evaluate the following code

```
>>> l = []
>>> l.append(12)
>>> l.append(True)
>>> l.append('hello')
>>> l.append([1,2])
>>> l.append(4<3)
>>> l
```

Solutions

```
>>> l = [ ]
>>> l.append(12)
# l = [12]
>>> l.append(True)
# l = [12, True]
>>> l.append('hello')
# l = [12, True, 'hello']
>>> l.append([1,2])
# l = [12, True, 'hello', [1,2] ]
>>> l.append(4<3)
# l.append (4 < 3)
# l.append (False)
# l = [12, True, 'hello', [1,2], False]
>>> l
[12, True, 'hello', [1,2], False]
```

>>> Exercise 202

1° Evaluate the following code

```
>>> l1 = [1,'a']  
>>> lv =[]  
>>> lv.append(l1[0])  
>>> lv
```

2° Evaluate the following code

```
>>> liste = [1,'a',2,'b',3,'c',4,'d']  
>>> listec = [ ]  
>>> i = 0  
>>> while(i<len(liste)) :  
    listec.append(liste[i])  
    i=i+1  
>>>listec
```

Solutions

1°

```
>>> lv.append(l1[0])  
# lv.append(1)  
>>> lv  
[1]
```

2°

iteration	i < len(liste)	listec.append(liste[i])
1	True	[1]
2	True	[1,'a']
3	True	[1,'a',2]
4	True	[1,'a',2,'b']
5	True	[1,'a',2,'b',3]
6	True	[1,'a',2,'b',3,'c']
7	True	[1,'a',2,'b',3,'c',4]
8	True	[1,'a',2,'b',3,'c',4,'d']

```
>>>listec
```

```
[1,'a',2,'b',3,'c',4,'d']
```

Tips : copy a list with while

```
>>> liste = [1,'a',2,'b',3,'c',4,'d']  
>>> listec = []  
>>> i = 0  
>>> while(i<len(liste)) :  
    listec.append(liste[i])  
    i=i+1
```

>>> Exercise 203

```
>>> l1 = [0,1,2,3,4,'a','b','c','d']  
>>> l2 = [ ]
```

Copy l1 in l2

Solutions

```
>>> l1 = [0,1,2,3,4,'a','b','c','d']  
>>> l2 = [ ]  
>>> i=0  
>>> while(i<len(l1)) :  
    l2.append(l1[i])  
    i=i+1  
>>>l2
```

```
[0,1,2,3,4,'a','b','c','d']
```

>>>22.2. Addition lists with method append ()

>>> Exercise 204

1° Evaluate the following code

```
>>> l1 = [0,1,2]
```

```
>>> l2 = [1,1,1]
>>> l1 + l2
>>> l3 = l1 +l2
```

2° Evaluate the following code

```
>>> l1 = [0,1,2]
>>> l2 = [1,1,1]
>>> i=0
>>> l3 = []
>>> while(i<len(l1)) :
    l3.append(l1[i]+l2[i])
    i=i+1
```

Solutions

1° Expression : [0,1,2] + [1,1,1]

Operator : + (list concatenation)

Operands : [0,1,2] [1,1,1]

```
>>> [0,1,2] + [1,1,1]
[0, 1, 2, 1, 1, 1]
>>> l1+l2
[0, 1, 2, 1, 1, 1]
>>> l3 = l1 + l2
# l3 = l1 + [1,1,1]
# l3 = [0,1,2] + [1,1,1]
# l3 = [0,1,2,1,1,1]
[0, 1, 2, 1, 1, 1]
```

2°

iteration	l3.append(l1[i] + l2[i])	l3
1	l3.append(0+0)	[0]
2	l3.append(1+1)	[0,2]
3	l3.append(2+1)	[0,2,3]

```
>>> l3
```

```
[1,2,3]
```

Tips

- Operator + concatenates lists
- To add list elements we can use `append(liste1[i]+liste2[i])`

>>> Exercise 205

Evaluate the following code

```
>>> l1 = ['a', 'b', 'c', 'd']
>>> l2 = ['e', 'f', 'g', 'h']
>>> l3 = l1 +l2
>>> l3
```

Solutions

```
>>> l1 = ['a', 'b', 'c', 'd']
>>> l2 = ['e', 'f', 'g', 'h']
>>> l3 = l1 +l2
# l3 = ['a', 'b', 'c', 'd']+ ['e', 'f', 'g', 'h']
# l3 = ['a', 'b', 'c', 'd','e', 'f', 'g', 'h']
>>> l3
['a', 'b', 'c', 'd','e', 'f', 'g', 'h']
```

>>> Exercise 206

```
>>> l1 = [0,0,0,0,0,0,0]
>>> l2 = [1,1,1,1,1,1,1]
```

1º Evaluate the following code

```
>>> len(l1) == len (l2)
```

2° Evaluate the following code

```
>>> l1 = [0,0,0,0,0,0,0]
>>> l2 = [1,1,1,1,1,1,1]
>>> l3 = []
>>> i=0
>>> while(i<len(l1)) :
    l3.append(l1[i]+l2[i])
    i=i+1
```

Solutions

1° the two lists must be the same length

```
>>> len(l1) == len (l2)
True
```

2°

```
>>> l1 = [0,0,0,0,0,0,0]
>>> l2 = [1,1,1,1,1,1,1]
>>> l3 = []
>>> i=0
>>> while(i<len(l1)) :
    l3.append(l1[i]+l2[i])
    i=i+1
>>> l3
```

```
[1,1,1,1,1,1,1]
```

>>> 22.3. Manipulates lists with `append ()`

>>> **Exercise 207**

Evaluate the following code

```
>>> l1 = [-1,2,3,-6,8]
```

```
>>> l2 = []
>>> l3 = []
>>> i=0
>>> while (i< len(l1)) :
    if(l1[i]>0) :
        l2.append(l1[i])
    else :
        l3.append(l1[i])
    i=i+1
```

Solutions

```
>>> i=0
>>> l1 = [-1,2,3,-6,8]
>>> l2 = []
>>> l3 = []
>>> while (i< len(l1)) :
    if(l1[i]>0) :
        l2.append(l1[i])
    else :
        l3.append(l1[i])
    i=i+1
>>> l2
[2,3,8]
>>> l3
[-1,-6]
```

>>> **Exercise 207**

```
>>> l1 = [-7,3,-5,-7,8,9,-3]
```

Loop over this list to split positive and negative numbers in two lists

Solutions

```
>>> l1 = [-7,3,-5,-7,8,9,-3]
```

```
>>> i=0
```

```
>>> l2 =[]
```

```
>>> l3=[]
```

```
>>> while(i<len(l1)) :
```

```
    if(l1[i]> 0) :
```

```
        l2.append(l1[i])
```

```
    else :
```

```
        l3.append(l1[i])
```

```
    i=i+1
```

```
>>> l2
```

```
[3,8,9]
```

```
>>> l3
```

```
[-7,-5,-7,-3]
```

iteration	if (l1[i]>0)	l2	l3
1	False	[]	[-7]
2	True	[3]	[-7]
3	False	[3]	[-7,-5]
4	False	[3]	[-7,-5,-7]
5	True	[3,8]	[-7,-5,-7]
6	True	[3,8,9]	[-7,-5,-7,-3]

>>> **Exercise 208**

Fill the blanks to get the two lists l2 and l3

```
>>> l1 = [0,1,2,3,4,5,6,7,8,9]
```

```
>>> l2,l3= ,[]
```

```
>>> i=0
```

```
>>> while (i<  ):  
    if(  ):  
        l2.append( )  
    else :  
        .append(l1[i])
```

```
    i=
```

```
>>> l2
```

```
[0,1,2,3,4]
```

```
>>> l3
```

```
[5,6,7,8,9]
```

Solutions

```
>>>l1 = [0,1,2,3,4,5,6,7,8,9]
```

```
>>> l2 = []
```

```
>>> l3= []
```

```
>>> i=0
```

```
>>> while (i< len(l1) ):
```

```
    if( l1[i] <= 5 ):
```

```
        l2.append(l1[i])
```

```
    else :
```

```
        l3.append(l1[i])
```

```
    i= i + 1
```

```
>>> l2
```

```
[0,1,2,3,4]
```

```
>>> l3
```

```
[5,6,7,8,9]
```

>>> Exercise 209

Evaluate the following code

```
>>> l1 = [1,4,6,8,-5,3,0,-16,17]
>>> l2 = []
>>> l3=[]
>>> i=0
>>> while (i<len(l1)) :
    if(l1[i]> 1) :
        l2.append(l1[i])
    else :
        l3.append(l1[i])
    i=i+1
```

corrigé

```
>>> l2
[4, 6, 8, 3, 17]
>>> l3
[1, -5, 0, -16]
```

>>> 22.4. Some lists methods

« listons toutes les méthodes » me

>>> 22.4.1. `list.clear()`

>>> **Exercise 210**

Evaluate the following code

```
>>> l = [1,2,3,4]
>>> l.clear()
>>> l
```

Solutions

```
>>>help(list)
```

```
clear(self, /)
|   Remove all items from list.
>>> l = [1,2,3,4]
>>> l.clear( )
# clear(l)
# clear([1,2,3,4])
>>> l
```

```
[]
```

>>> **exercise 211**

Evaluate the following code

```
>>> l = [1,2,3,4]
>>> del l[:]
>>> l
>>> l1 =[1,2,3,4]
>>> l1.clear()
```

Solutions

```
>>> l = [1,2,3,4]
>>> del l[:]
# del l[:]
# del [1,2,3,4]
# l = []
>>> l
[]
>>> l.clear()
# [1,2,3,4].clear()
# []
```

```
>>> l1
```

```
[]
```

```
>>> 22.4.2.list.copy()
```

>>> Exercise 212

Evaluate the following code

```
>>> l = [1,2,3,4]
```

```
>>> l.copy()
```

```
>>> a = l.copy()
```

```
>>> a
```

Solutions

```
>>> help(list.copy)
```

Help on method descriptor.

```
copy(self, /)
```

Return a shallow copy of the list

```
>>> l = [1,2,3,4]
```

```
>>> l.copy()
```

```
[1, 2, 3, 4]
```

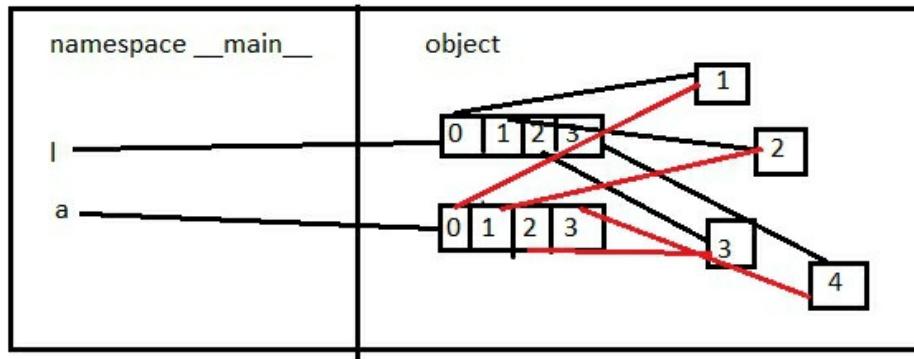
```
>>> a = l.copy()
```

```
# a = [1, 2, 3, 4].copy()
```

```
# a= [1, 2, 3, 4]
```

```
>>> a
```

```
[1, 2, 3, 4]
```



>>> 22.4.3. IN PLACE METHODS

>>> Exercise 213

Evaluate the following code

```
>>> help(list.sort)
>>> l = [3, 2, 1]
>>> l.sort()
```

Solutions

```
>>> help(list.sort)
Help on method_descriptor:
sort(self, /, *, key=None, reverse=False)
    Stable sort *IN PLACE*.
```

```
>>> l = [3,2,1]
```

```
>>> l.sort()
```

```
# sort([3,2,1])
```

```
# [1,2,3]
```

```
>>> l
```

```
[1,2,3]
```

>>> Exercise 214

Evaluate the following code

```
>>> l1 = [8,0,1,3,5]
```

```
>>> l2= [1,0,24,5]
>>> l1.sort()
>>> l2.sort()
```

Solutions

```
>>> l1.sort()
# sort(l1)
>>> l1
[0, 1, 3, 5, 8]
>>> l2.sort()
# sort(l2)
l2= [1,0,24,5]
```

>>> **Exercise 215**

```
>>> help(list.reverse)
Help on method_descriptor:
reverse(self, /)
    Reverse *IN PLACE*.

>>> l = [-8, 1, 2, 11, 13]
>>> l.reverse( )
>>> l
```

Solutions

```
>>> l = [-8, 1, 2, 11, 13]
>>> l.reverse( )
# [-8, 1, 2, 11, 13].reverse()
# [13, 11, 2, 1, -8]
>>> l
[13, 11, 2, 1, -8]
```

>>> Exercise 216

Evaluate the following code

```
>>> help(list.pop)
```

Help on method_descriptor:

```
pop(self, index=-1, /)
```

Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

```
>>> l = [-8, 1, 2, 11, 13]
```

```
>>> l.pop()
```

```
>>> l.pop(1)
```

```
>>> a = []
```

```
>>> a.pop()
```

Solutions

```
>>> l.pop()
```

```
# pop([-8, 1, 2, 11, 13], index =-1)
```

```
# return 13
```

13

```
>>> l.pop(3)
```

```
# pop(l, index = 3)
```

```
# pop([-8, 1, 2, 11], index = 3)
```

```
# return 11
```

11

```
>>> a = []
```

```
>>> a.pop()
```

IndexError: pop from empty list

>>>23.Sort Algorithm

>>> 23.1. Swap lists elements

>>> **Exercise 217**

1° Evaluate the following code

```
>>> a=2
>>> b=1
>>> a,b = b,a
>>> a
>>> b
```

2° Evaluate the following code

```
>>> l1=[5,4,3,2,1]
>>> l1[0],l1[4]=l1[4],l1[0]
>>> l1[1],l1[3]=l1[3],l1[1]
>>> l1
```

Solutions

1° We could use a third variable called temp

```
>>> temp = a
>>> a = b
>>> b= temp
```

But in Python there's an easy way

```
>>> a,b = b,a
>>> a
1
>>> b
2
```

2°

```
>>> l1=[5,4,3,2,1]
>>> l1[0],l1[4]=l1[4],l1[0]
# l1[0]= 1 et l1[4]=5
# l1 = [1,4,3,2,5]
>>> l1[1],l1[3]=l1[3],l1[1]
# l1[1] = 2 et l1[3] = 4
# l1 = [1,2,3,4,5]
>>> l1
[1,2,3,4,5]
```

>>> **Exercise 220**

1° Evaluate the following code

```
>>> l1 = [5,2,9,1,10]
>>> l1[0], l1[3] = l1[3],l1[1]
>>> l1[3], l1[2] = l1[2], l1[3]
>>> l1
```

2° Evaluate the following code

```
>>> l1 = [1,2,3,4,5]
>>> l1[4], l1[0] = l1[0], l1[4]
>>> l1[3], l1[1] = l1[1], l1[3]
>>> l1
```

Solutions

1°

```
>>> l1[0], l1[3] = l1[3],l1[1]
# l1 = [1,2,9,5,10]
>>> l1[3], l1[2] = l1[2], l1[3]
# l1 = [1,2,5,9, 10]
```

```
>>> l1
```

```
[1, 2, 5, 9, 10]
```

2°

```
>>> l1 = [1,2,3,4,5]  
>>> l1[4], l1[0] = l1[0], l1[4]  
# l1 = [5,2,3,4,1]  
>>> l1[3], l1[1] = l1[1], l1[3]  
# l1 = [5,4,3,2,1]  
>>> l1
```

```
[5, 4, 3, 2, 1]
```

>>> 23.2. Sorting list ascending order

>>> **Exercise 221**

1° a) Evaluate the following code

```
>>> l = [4,3,2,1]  
>>> i = 0  
>>> indiceminimum = 0  
>>> while (i< len(l)) :  
    if(l[i]< l[indiceminimum]) :  
        indiceminimum = i  
    i= i +1  
>>> indiceminimum
```

b) evaluate the following code

```
>>> l[0],l[indiceminimum] = l[indiceminimum],l[0]
```

2° a) evaluate the following code

```
>>> i = 1  
>>> indiceminimum = 1
```

```
>>> while (i< len(l)) :  
    if(l[i]< l[indiceminimum]) :  
        indiceminimum = i  
    i=+1  
>>> indiceminimum
```

b)

```
>>>l[1],l[indiceminimum]=l[indiceminimum],l[1]
```

3° evaluate the following code

```
>>> l = [4,3,2,1]  
>>> i=0  
>>> while (i < len(l)) :  
    indiceminimum = i  
    j=i  
    while (j < len(l)) :  
        if (l[j] <l[indiceminimum]) :  
            indiceminimum = j  
        j=j+1  
    l[i],l[indiceminimum] = l[indiceminimum],l[i]  
    i=i+1
```

Solutions

1° a)

```
>>> i = 0  
>>> indiceminimum = 0  
>>> while (i< len(l)) :  
    if(l[i]< l[indiceminimum]) :  
        indiceminimum = i
```

```
i=+1
>>> indiceminimum
3
```

iteration	if(l[i] < l[indiceminimum])	indiceminimum
1	l[0] < l[0] False	0
2	l[1] < l[0] True	1
3	l[2] < l[1] True	2
4	l[3] < l[2] True	3

b)

```
>>> l[0],l[indiceminimum] = l[indiceminimum],l[0]
# l[0],l[3] = l[3],l[0]
# l = [1,3,2,4]
```

2° a)

```
# l = [1,3,2,4]
>>> i = 1
>>> indiceminimum = 1
>>> while (i < len(l)) :
    if(l[i] < l[indiceminimum]) :
        indiceminimum = i
    i = i + 1
>>> indiceminimum
```

2

iteration	if (l[i] < l[indiceminimum])	indiceminimum
1	l[1] < l[1] False	1
2	l[2] < l[1] True	2
3	l[3] < l[2] False	2

b) On a

```
>>> l[1],l[indiceminimum] = l[indiceminimum],l[1]
# 3,2 = 2,3
```

```
# l = [1,2,3,4]
```

3°

```
>>> l = [4,3,2,1]
>>> i=0
>>> while (i < len(l)) :
```

 indiceminimum = i

 j=i

 while (j < len(l)) :

 if (l[j] <l[indiceminimum]) :

 indiceminimum = j

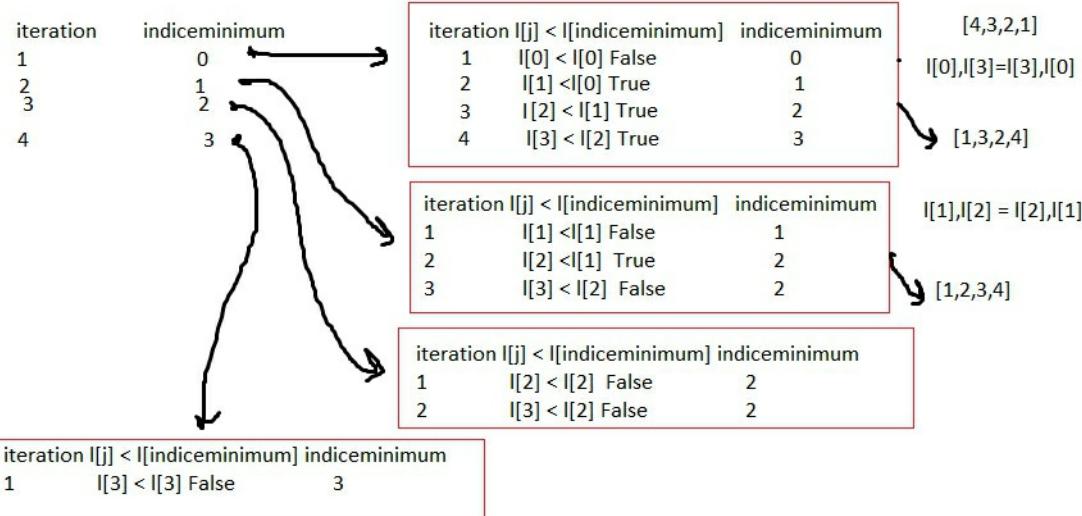
 j=j+1

 l[i],l[indiceminimum]=l[indiceminimum],l[i]

 i=i+1

```
>>> l
```

```
[1,2,3,4]
```



>>> **Exercise 222**

Fill the blanks to sort this list by ascending order

```
>>> l= [11, 16, 2, 4, -1, 0]
```

```
>>> i =0
>>> while ( ) :
    indiceminimum =
    j=i
    while(j <len(l)) :
        if (l[j] < ) :
            indiceminimum = j
        j=
        l[i],l[ ]=l[indiceminimum],l[i]
        i=i+1
>>> l
```

Solutions

```
>>> i =0
>>> while ( i < len(l) -1 ) :
    indiceminimum = i
    j=i
    while(j <len(l)) :
        if (l[j] < l[indiceminimum] ) :
            indiceminimum = j
        j= j +1
        l[i],l[indiceminimum]=l[indiceminimum],l[i]
        i=i+1
>>> l
[-1, 0, 2, 4, 11, 16]
```

>>>24.Functions in Python : Part 2

>>> 24.1. Defining functions with a return

>>> Exercise 223

1°a) evaluate the following code

```
>>> 0 +1
```

```
>>> -1 +1
```

b) evaluate the following code

```
>>> def fonction (x) :
```

```
    return x+1
```

```
>>> fonction(0)
```

```
>>> fonction(-1)
```

Solutions

1° a)

```
>>> 0+1
```

```
1
```

```
>>> -1+1
```

```
0
```

b)

```
>>> def fonction(x) :
```

```
    return x +1
```

```
>>> fonction(0)
```

```
# fonction(0) :
```

```
#             return 0 +1
```

```
#             return 1
```

```
1
```

```
>>> fonction(-1)
```

```
# fonction(-1) :
```

```
#           return -1 + 1
#
#           return 0
0
```

Tips

```
def function_name (argument) :
    block
    return expression
```

>>> Exercice 224

Evaluate the following code

```
>>> def carre(x) :
    return x*x
>>> carre(0)
>>> carre(-3)
```

Solutions

```
>>> def carre(x) :
    return x*x
>>> carre(0)
# carre(0)
# return 0*0
# return 0
0
>>> carre(-3)
# carre(-3)
# return -3 * -3
# return 9
9
```

```
>>> 24.1.1. module.fonction()
```

>>> **exercise 225**

1° evaluate the following code

```
>>> sqrt(4)
```

```
>>> import math
```

```
>>> math.sqrt(4)
```

```
>>> math.sqrt(5)
```

2° $f(x) = \sqrt{x + 1}$

a) fill the blanks

```
>>> def fonction(x) :
```

```
    return
```

b) evaluate the following code

```
>>> fonction(4)
```

```
>>> fonction(5)
```

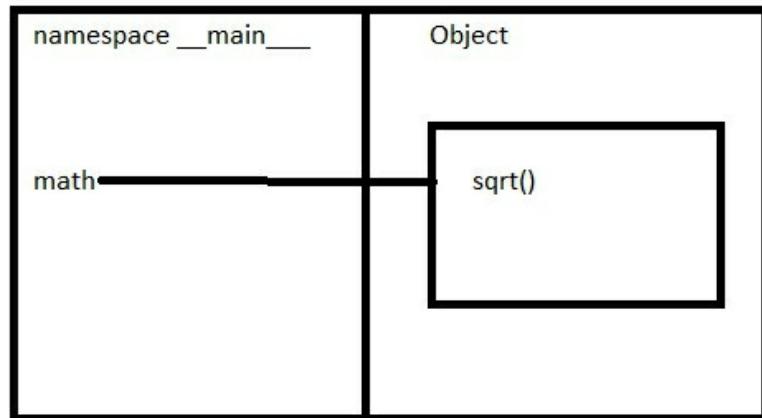
```
>>> fonction(-5)
```

Solutions

1°

```
>>> sqrt(4)
```

```
NameError: name 'sqrt' is not defined
```



```
>>> help(math.sqrt)
```

Help on built-in function sqrt in module math :

```
sqrt(x, /)
```

Return the square root of x.

```
>>> import math
```

```
>>> math.sqrt(4)
```

```
2.0
```

```
>>> math.sqrt(5)
```

```
2.23606797749979
```

2°

```
>>> def fonction(x) :
```

```
    return math.sqrt(x+1)
```

`math.sqrt(x+1)` is an expression

b)

```
>>> fonction(4)
```

```
# fonction(4)
```

```
# return math.sqrt(4+1)
```

```
# return math.sqrt(5)
```

```
# 2.23606797749979
```

```
2.23606797749979
```

```
>>> fonction(5)
```

```
# fonction(5)
```

```
# return math.sqrt(5+1)
```

```
# return math.sqrt(6)
```

```
# 2.449489742783178
```

```
2.449489742783178
```

```
>>> fonction(-5)
# fonction(-5)
# return math.sqrt(-5+1)
# return math.sqrt(-4)
# ValueError : math domain error
ValueError : math domain error
```

>>> 24.1.2. Function with several arguments

>>> **Exercise 226**

1° moyenne4 should return the mean of a,b,c and d

```
>>> def moyenne4(a,b,c,d) :
    return
```

2° Evaluate the following code

```
>>> moyenne(10,11,14,0)
```

Solutions

1°

```
>>> def moyenne4(a,b,c,d) :
    return (a+b+c+d) / 4
```

2°

```
>>> moyenne(10, 11, 14, 0)
# moyenne(10, 11, 14, 0)
# return (10+11+14+0)/4
# return 8.75
```

8.75

Tips

A function can takes n arguments ($n \geq 2$)

```
def nom_fonction(arg1, arg2, ..., argn) :
    block
```

return expression

>>> Exercise 227

Evaluate the following code

```
>>> def affiche(a,b,c) :  
    return print(a,b,c)  
>>> affiche(1,2,3)
```

Solutions

Function name : affiche

Arguments : 1 2 3 (int)

Return : print(1,2,3) (expression)

```
>>> affiche (1,2,3)  
# affiche (1,2,3)  
# return print(1,2,3)  
1 2 3
```

>>> 24.1.2.Fonction without any arguments

>>> Exercise 228

1° Evaluate the following code

```
>>> def affiche() :  
    x = 1  
    print(x)  
    print('****')  
>>> affiche()
```

2° a) Evaluate the following code

```
>>> x
```

b) how do we call variable x defined in affiche ?

Solutions

1° Function name : affiche

Arguments : None

Block : three instructions

Return : None

```
>>> affiche()
```

```
# affiche( ) :
```

```
# x = 1
```

```
# print(x)
```

```
# print (1)
```

```
1
```

```
# print('****')
```

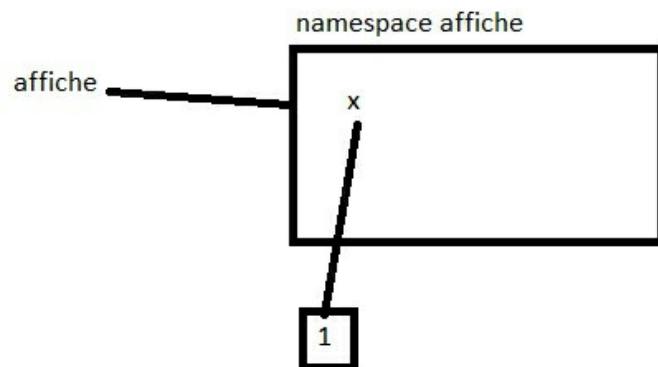
```
****
```

2° a)

```
>>> x
```

NameError : name 'x' is not defined

b) variable x is a local variable, he's defined in affiche namespace



Tips

```
def function_name( ) :
```

```
    block
```

defined a function without argument

```
>>> fonction( )
```

Is a function call

>>> Exercise 229

Evaluate the following code

```
>>> def boucle1( ) :  
    for i in range(10) :  
        print(i)  
>>> boucle1()
```

Solutions

Function name : boucle1

Arguments : None

Block : for loop

Return : None

```
>>> boucle1  
0 1 2 3 4 5 6 7 8 9
```

>>> 24.1.3. Fonctions avec du if / else

>>> Exercise 230 (Syracuse function)

1° Evaluate the following code

```
>>> x = 2  
>>> if (x % 2 == 0) :  
    print(x/2)  
else :  
    print(3*x+1)
```

2° Evaluate the following code

```
>>> def f(x) :  
    if ( x % 2 == 0 ) :  
        return x / 2  
    else :  
        return 3 * x + 1
```

```
>>> for i in range(10):  
    print(f(i),end="")
```

Solutions

1°

```
>>> x = 2  
>>> if (x % 2 == 0):  
# if (2%2 ==0 ):  
# if True :  
# print (x/2)  
# print(2/2)  
# print(1.0)  
1.0
```

2°

```
>>> for i in range(10):  
    print(f(i),end = ' ')
```

```
0.0 5 1.0 11 2.0 17 3.0 23 4.0 29
```

Tips

```
def function_name(arguments) :  
    if condition :  
        return expression  
    else :  
        return expression2
```

>>> Exercise 231

1° Evaluate the following code

```
>>> m = 2  
>>> n = 3
```

```
>>> res = 0
>>> if (m<n) :
    res = -1
elif (m>n) :
    res = 1
else :
    res
>>> res
```

2° Evaluate the following code

```
>>> def g(a,b) :
    if (a<b) :
        return -1
    elif (a>b) :
        return 1
    else :
        return 0
>>> g(1,0)
>>> g(0,1)
```

Solutions

1°

```
>>> m = 2
>>> n = 3
>>> res = 0
>>> if (m<n) :
    # if (2 <3 ) :
    # if True :
```

```
res = -1
```

```
>>> res
```

```
-1
```

2°

```
>>> def g(a,b):
```

```
    if (a<b):
```

```
        return -1
```

```
    elif (a>b):
```

```
        return 1
```

```
    else :
```

```
        return 0
```

```
>>> g(1,0)
```

```
# if (1 <0):
```

```
# if False :
```

```
# elif (1 > 0):
```

```
# return 1
```

```
1
```

```
>>> g(0,1)
```

```
# if (0 < 1):
```

```
# if True:
```

```
# return - 1
```

```
-1
```

>>> 24.1.3.1.Function with a while : Fibonacci

>>> **Exercise 232**

1° $F_0 = 0, F_1 = 1$ $F_0 = 0$ and $F_{n+2} = F_{n+1} + F_n$

Calculate F_2 and F_3

2° a) Evaluate the following code

```
>>> f0, f1 = 0,1  
>>> f2 = f1 + f0  
>>> f3 = f2 + f1  
>>> f4 = f3 + f2
```

b) Fill this table with the expression $k = i + j$ until the 10th iteration

iteration	k	i	j

c) evaluate the following code

```
>>> z = 0  
>>> i = 1  
>>> j = 0  
>>> while (z < 10) :  
    k = i + j  
    print(k, end = "")  
    i, j = k, i  
    z+=1
```

3° evaluate the following code

```
>>> def fib(n) :  
    z=0  
    i=1  
    j = 0  
    while (z < n) :  
        k = i+j
```

```
print(k, end = ' ')
i, j = k , i
z = z+1
>>> fib(10)
```

Solutions

$$1^{\circ} \quad F_2 = F_1 + F_0 = 1 + 0 = 1 \text{ and}$$

$$F_3 = F_2 + F_1 = 1 + 1 = 2$$

2^o a)

```
>>> f0 = 0
>>> f1 = 1
>>> f2 = f1 + f0
# f2 = 1 + 0
# f2 = 1
>>> f3 = f2 + f1
# f3 = 1 + 1
# f3 = 2
>>> f4 = f3 + f2
# f4 = 2 + 1
# f4 = 3
```

b) note : 10th iteration : $89 = 55 + 34$

iteration	k	i	j
1	1	1	0
2	2	1	1
3	3	2	1
4	5	3	2
5	8	5	3
6	13	8	5
7	21	13	8
8	34	21	13
9	55	34	21
10	89	55	34

iteration	k	i	j
iteration + 1	B	a	b

Iteration + 1 : $B = a + b$

- $a = k$
- $b = i$

c)

```
>>> z = 0
>>> i = 1
>>> j = 0
>>> while (z < 10) :
    k = i + j
    print(k, end = "")
    i, j = k, i
    z = z + 1
```

1 2 3 5 8 13 21 34 55 89

3° fonction_name : fib

Arguments : n (int)

Block : instructions + while loop

```
>>> fib (10)
```

```
1 2 3 5 8 13 21 34 55 89
```

>>> 24.1.3.2.Function who returns a list

>>> **Exercise 233**

Evaluate the following code

```
>>> def f(x) :
```

```
    l = []
```

```
    for i in range(3) :
```

```
        x = x + 2
```

```
        l.append(x)
```

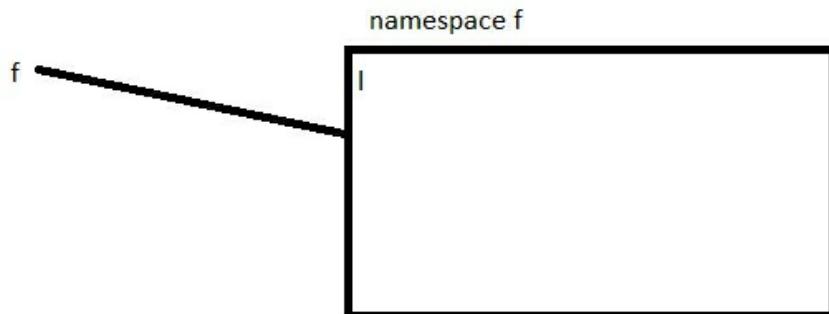
```
    return l
```

```
>>> f(2)
```

Solutions

```
>>> f(2)
```

```
[4, 6, 8]
```



>>> **Exercise 234 (doc python)**

Evaluate the following code

```
>>> def fib2(n) :
```

```
    result = []
```

```
    a, b = 0, 1
```

```
    while a < n :
```

```
result.append(a)
a, b = b, a+ b
return result
>>> fib2(10)
```

Solutions

```
>>> fib2(10)
```

```
[0, 1, 1, 2, 3, 5, 8]
```

>>> 24.2. Define a function without return

>>> **Exercise 235**

Evaluate the following code

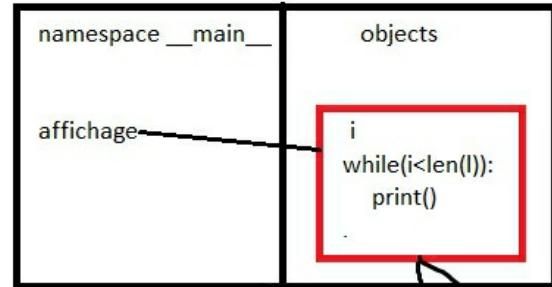
```
>>> def affichage (l) :
    i=0
    while(i<len(l)) :
        print(l[i],end = ' ')
        i=i+1
>>> liste = [1,2,3,4,5]
>>> st = 'python'
>>> affichage(liste)
>>> affichage(st)
```

Solutions

```

iteration i < len(l)    print(l[i], end = "")
1      True          1
2      True          1 2
3      True          1 2 3
4      True          1 2 3 4
5      True          1 2 3 4 5

```



```

iteration i < len(st)    print(l[i], end = "")
1      True          p
2      True          p y
3      True          p y t
4      True          p y t h
5      True          p y t h o
6      True          p y t h o n

```

>>> Exercise 236

1° Fill the blanks

```

>>> liste = [0, 'a', 1, 'b', 2, 'c']

>>> def affichage (l) :
    i=0
    while(i<    ) :
        print( [i], end = ' ')
        i=

```

2° Evaluate the following code

```
>>> affichage(liste)
```

Solutions

```

>>> liste = [0, 'a', 1, 'b', 2, 'c']

>>> def affichage (l) :
    i=0
    while(i<len(l)) :
        print(l[i], end = ' ')
        i=i+1

```

2°

```
>>> affichage(liste)
```

```
0 a 1 b 2 c
```

>>> **Exercise 237**

$u_{n+1} = u_n + 1$ and $u_0 = 1$

1° Evaluate the following code

```
>>> i = 0
>>> j = 1
>>> l = []
>>> while (i < 10) :
    k = j + 1
    l.append(k)
    j = k
    i = i + 1
>>> l
```

2° Evaluate the following code

```
>>> def rec(n) :
    i = 0
    j = 1
    l = []
    while(i < n) :
        k = j + 1
        l.append(k)
        j = k
        i = i + 1
    print(l)
```

```
>>> rec(10)
```

Solutions

1°

	iteration	k = j + 1	l.append(k)	j = k
1	2	[2]	2	
2	3	[2,3]	3	
3	4	[2,3,4]	4	
4	5	[2,3,4,5]	5	
5	6	[2,3,4,5,6]	6	
6	7	[2,3,4,5,6,7]	7	
7	8	[2,3,4,5,6,7,8]	8	
8	9	[2,3,4,5,6,7,8,9]	9	
9	10	[2,3,4,5,6,7,8,9,10]	10	
10	11	[2,3,4,5,6,7,8,9,10,11]	11	

```
>>> l
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

2°

```
>>> rec(10)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

>>> 24.2.1. pass and None

>>> **Exercise 238**

Evaluate the following code

```
>>> def f(x) :
```

```
    x = 4
```

```
>>> x
```

```
>>> f
```

```
>>> f()
```

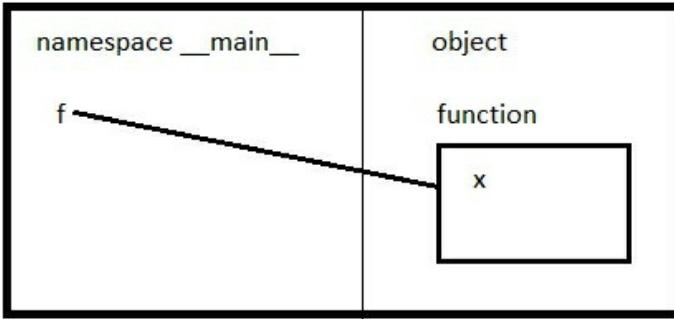
```
>>> f(1)
```

```
>>> print(f(1))
```

Solution

```
>>> x
```

```
NameError: name 'x' is not defined
```



```
>>> f
```

```
<function f at 0x035E7C48>
```

```
>>> f()
```

```
TypeError: f() missing 1 required positional argument: 'x'
```

```
>>> f(1)
```

```
# x = 4
```

```
>>> print(f(1))
```

```
None
```

When a function has no return, automatically it will get a None return

>>> Exercise 239

Evaluate the following code

```
>>> def f(x) :
```

```
    pass
```

```
>>> f
```

```
>>> f(1)
```

```
>>> print(f())
```

Solutions

Function name : f

Block : one instruction (pass)

Return : None

```
>>> f
<function f at 0x03A80C90>
>>> f(1)
>>>
>>> print(f(1))
None
```

>>> 25.Loop with for ... in

>>> 25.1.For ... in with lists

>>> **Exercise 240**

1° Evaluate the following code

```
>>> liste = ['python', 'java', 'c']
>>> i = 0
>>> while (i<len(liste)) :
    print(l[i])
    i=i+1
```

2° Evaluate the following code

```
>>> for i in liste :
    print(i)
```

Solutions

1° On a une variable liste qui contient des éléments de type String.

```
>>> while (i<len(liste)) :
    print(l[i])
    i=i+1
```

```
python
java
```

c

2°

```
>>> i
```

NameError

```
>>> for i in liste :
```

```
    print(i)
```

python

java

c

```
>>> i
```

'c'

Tips

```
for variable in list :
```

```
    block
```

>>> Exercise 241

Evaluate the following code

1° a)

```
>>> liste = [1,2,3,4,5]
```

```
>>> for i in liste :
```

```
    print(i)
```

```
>>> i
```

b)

```
>>> liste = [1,2,3,4,5]
```

```
>>> for i in liste :
```

```
    i
```

2°

```
>>> l=['a','b','c','d']
```

```
>>> for i in l :  
    print(i, end=' ')
```

Solutions

1° a)

```
>>> liste = [1,2,3,4,5]  
>>> for i in liste :  
    print(i)
```

```
1  
2  
3  
4  
5
```

```
>>> i
```

```
5
```

b)

```
>>> liste = [1,2,3,4,5]  
>>> for i in liste :  
    i
```

```
1  
2  
3  
4  
5
```

2°

```
>>> l=['a','b','c','d']  
>>> for i in l :  
    print(i, end=' ')
```

```
a b c d
```

>>> 25.2.For... in on strings

>>> **Exercise 242**

Evaluate the following code

```
>>> for i in 'antinconstitutionnellement' :  
    print(i, end = ' ')
```

Solution

```
>>> for i in 'antinconstitutionnellement' :  
    print(i, end = ' ')
```

```
antinconstitutionnellement
```

>>> **Exercise 243**

Evaluate the following code

```
>>> a = 24  
>>> for i in a :  
    print(i)  
>>> a = '24'  
>>> for i in a :  
    print(i)
```

Solutions

```
>>> a = 24  
>>> for i in a :  
    print(i)  
  
TypeError: 'int' object is not iterable  
  
>>> a = '24'  
>>> for i in a :
```

```
print(i)
```

```
2
```

```
4
```

Tips :

We can use for loops on iterable objects like Strings and Lists.

>>> 26.Nested lists

>>> Exercise 244

Evaluate the following code

```
>>> ld = [1,[2,3],4]
>>> ld[0]
>>> ld[1]
>>> ld[1][0]
>>> ld[1][1]
>>> liste = ld[1]
>>> def affichage (l) :
    i = 0
    while(i<len(l)) :
        print(l[i])
        i = i + 1
>>> affichage(ld)
>>> affichage(liste)
>>> for k in ld :
    print(k, end =")")
```

Solutions

```
>>> ld = [1,[2,3],4]
>>> ld[0]
```

```
1  
>>> ld[1]  
[2,3]  
>>>ld[1][0]  
2
```

```
>>>ld[1][1]  
3  
>>> liste = ld[1]
```

```
# liste = [2,3]  
>>> liste
```

```
[2,3]  
>>> affichage(ld)
```

```
2  
[2,3]  
4  
>>> affichage(liste) :  
2  
3
```

```
iteration  for k in ld  print(k, end = "")  
1          k = 1          1  
2          k = [2,3]      1 [2,3]  
3          k = 4          1 [2,3] 4
```

```
>>> for k in ld :  
    print(k, end =")  
1 [2,3] 4
```

>>> **Exercise 245**

Evaluate the following code

```
>>> liste = [[0,0], [1,1], ['b','c'], 'd']
>>> for k in liste :
    k
>>> liste[0][1]
>>> liste[2][1]
>>> liste[3]
```

Solutions

```
>>> for k in liste :
```

```
    k
```

```
[0, 0]
```

```
[1, 1]
```

```
['b','c']
```

```
'd'
```

```
>>> liste[0][1]
```

```
# liste[0,0]
```

```
# liste[0, 0]
```

```
# 0
```

```
0
```

```
>>> liste[2][1]
```

```
# liste ['b','c']
```

```
# liste ['b','c']
```

```
# 'c'
```

```
'c'
```

```
>>> liste[3]
```

```
'd'
```

>>> 26.1. Skills on nested lists

>>> 26.1. Some maths

>>> Exercise 246

note	5	7	8	10	11	12	13	15	16	17	18	19
nombre élèves	1	1	1	6	10	4	4	4	2	1	1	1

```
>>> s = [[5,1],[7,1],[8,1],[10,6],[11,10],[12,4],[13,4],[15,4],[16,2],[17,1],[18,1],[19,1]]
```

Evaluate the following code

```
>>> def sommeEleves (l) :  
    somme = 0  
    i=0  
    while (i<len(l)) :  
        somme = somme + l[i][1]  
        i=i+1  
    return somme  
>>> sommeEleves(s)
```

Solutions

$$\sum_{i=0}^{11} s_{i1}$$

```
>>> someEleves(s)
```

36

iteration	$i < \text{len}(l)$	$\text{somme} = \text{somme} + l[i][1]$
1	True	$\text{somme} = 0 + l[0][1] = 0 + 1 = 1$
2	True	$\text{somme} = 1 + l[1][1] = 1 + 1 = 2$
3	True	$\text{somme} = 2 + l[2][1] = 2 + 1 = 3$
4	True	$\text{somme} = 3 + l[3][1] = 3 + 6 = 9$
5	True	$\text{somme} = 9 + l[4][1] = 9 + 10 = 19$
6	True	$\text{somme} = 19 + l[5][1] = 19 + 4 = 23$
7	True	$\text{somme} = 23 + l[6][1] = 23 + 4 = 27$
8	True	$\text{somme} = 27 + l[7][1] = 27 + 4 = 31$
9	True	$\text{somme} = 31 + l[8][1] = 31 + 2 = 33$
10	True	$\text{somme} = 33 + l[9][1] = 33 + 1 = 34$
11	True	$\text{somme} = 34 + l[10][1] = 34 + 1 = 35$
12	True	$\text{somme} = 35 + l[11][1] = 35 + 1 = 36$

>>> 26.1.1.1.Sum

>>> Exercise 247

```
>>> l = [[1,1],[2,2],[3,3]]
```

1° Evaluate the following code

```
>>> l[0]
>>> l[1]
>>> l[2]
```

2° Evaluate the following code

```
>>> l[0][0] + l[1][0] + l[2][0]
>>> l[1][0] + l[1][1] + l[1][2]
```

3° Evaluate the following code

```
>>> i = 0
>>> s = 0
>>> while(i<3) :
    s = s + l[i][0]
    i=i+1
```

b) Evaluate the following code

```
>>> i =0
```

```
>>> s = 0
>>> while(i<3) :
    s = s + l[i][1]
    i=i+1
```

c) Evaluate the following code

```
>>> i =0
>>> s = 0
>>> while (i< 2):
    j= 0
    while (j < 3) :
        s = l[j][i]
        j=j+1
    i=i+1
```

Solutions

1°

Expressions : l[0] l[1] and l[2]

```
>>> l[0]
[1,1]
>>> l[1]
[2,2]
>>> l[2]
[3,3]
```

2°

```
>>> l[0][0] + l[1][0] + l[2][0]
# 1 + 2 + 3
6
>>> l[0][1] + l[1][1] + l[2][1]
```

```
# 1 + 2 + 3
```

```
6
```

3º a)

```
>>> i =0
>>> s = 0
>>> while(i<3) :
    s = s + l[i][0]
    i=i+1
>>> s
```

```
6
```

b)

```
>>> i =0
>>> s = 0
>>> while(i<3) :
    s = s + l[i][1]
    i=i+1
>>> s
```

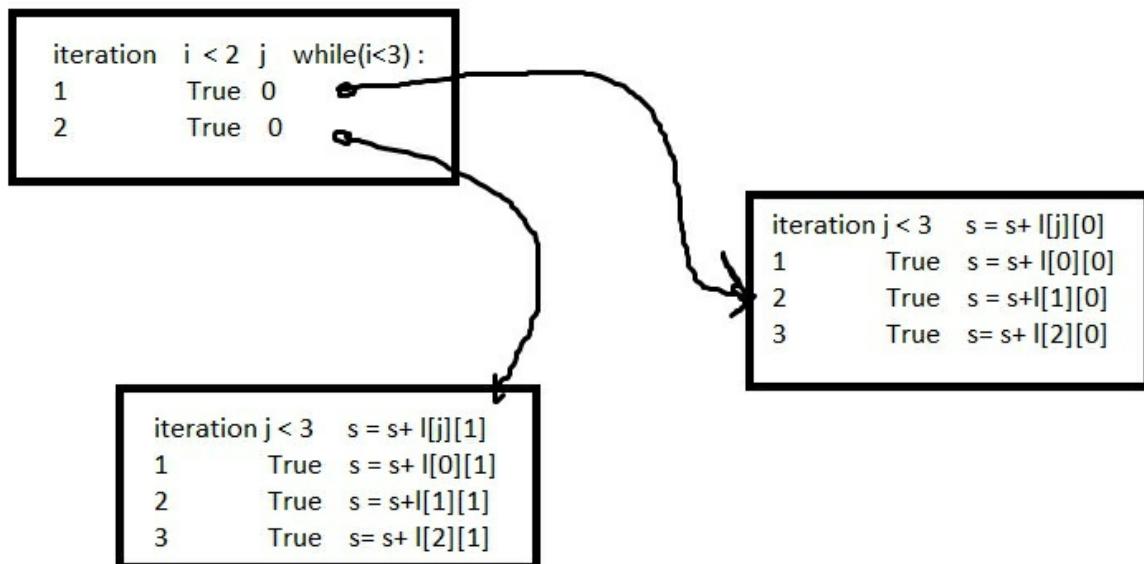
```
6
```

3º

```
>>> i =0
>>> s = 0
>>> while (i< 2):
    j= 0
    while (j < 3) :
        s =s + l[j][i]
```

```
j=j+1  
i=i+1  
>>> s
```

```
12
```



Tips : To sum an n-element list who has k-nested lists

```
>>> i = 0  
>>> s = 0  
>>> while (i < n):  
    j= 0  
    while (j < k):  
        s = s + l[j][i]  
        j=j+1  
    i=i+1
```

>>> 26.1.2.Rows

>>> **Exercise 248**

1	2
3	4
5	6

1° Evaluate the following code

```
>>> i = 0
>>> s = 0
>>> while (i<2) :
    s = s + l[0][i]
    i = i +1
>>> i =0
>>> s =0
>>> while (i<2) :
    s = s + l[1][i]
    i = i +1
>>> i =0
>>> s =0
>>> while (i<2) :
    s = s + l[2][i]
    i = i +1
```

2° Evaluate the following code

```
>>> l = [[1,2], [3,4], [5,6] ]
>>> i = 0
>>> s = 0
>>> while (i<3) :
    j = 0
    while(j<2) :
        s = s + l[i][j]
```

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

Solutions

1°

```
>>> i = 0  
>>> s = 0  
>>> while (i<2) :  
    s = s + l[0][i]  
    i = i +1  
>>> s
```

3

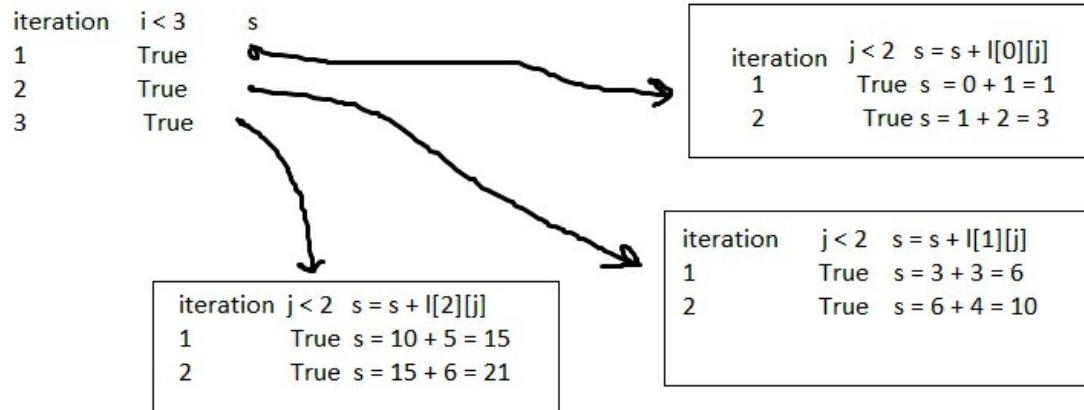
```
>>> i =0  
>>> s =0  
>>> while (i<2) :  
    s = s + l[1][i]  
    i = i +1  
>>> s
```

7

```
>>> i =0  
>>> s =0  
>>> while (i<2) :  
    s = s + l[2][i]  
    i = i +1  
>>> s
```

11

2°



```

>>> s
21

```

>>> Exercise 249

```

>>> s = [[5,1],[7,1],[8,1],[10,6],[11,10],[12,4],[13,4],[15,4],[16,2],[17,1],
[18,1],[19,1]]

```

$$1^{\circ} \sum_{i=0}^{11} s_{i1}$$

Evaluate the following code

```

>>> def nombreEleves (l):
    i= 0
    somme = 0
    while(i<len(l)):
        somme = somme + l[i][1]
        i=i+1
    return somme
>>> nombreEleves (s)

```

$$2^{\circ} \sum_{i=0}^{11} s_{i0} * s_{i1}$$

Evaluate the following code

```
>>> def mulSomme(l) :  
    i = 0  
    s = 0  
    while(i < len(l)) :  
        s = s + l[i][0]*l[i][1]  
        i = i + 1  
    return s  
>>> mulSomme(s)
```

3° Evaluate the following code

```
>>> moyenne = mulSomme(s) / nombreEleves(s)
```

Solutions

```
>>> s = [[5,1],[7,1],[8,1],[10,6],[11,10],[12,4],[13,4],[15,4],[16,2],[17,1],  
[18,1],[19,1]]
```

$$1^{\circ} \sum_{i=0}^{11} s_{i1} = 36$$

```
>>> def nombreEleves (l):  
    i = 0  
    somme = 0  
    while(i < len(l)):  
        somme = somme + l[i][1]  
        i = i + 1  
    return somme  
>>> nombreEleves (s)
```

36

2°)

$$s_{00} * s_{01} + s_{10} * s_{11} + s_{20} * s_{21} + s_{30} * s_{31} + s_{40} * s_{41} + s_{50} * s_{51} + s_{60} * s_{61} + s_{70} * s_{71} + s_{80} * s_{81} + s_{90} * s_{91} + s_{100} * s_{101} + s_{110} * s_{111} = 436$$

```
>>> def mulSomme(l) :  
    i = 0  
    s = 0  
    while(i < len(l)) :  
        s = s + l[i][0]*l[i][1]  
        i = i + 1  
    return s  
  
>>> mulSomme(s)
```

3°

>>> Exercise 250

1° Evaluate the following code

```
>>> donnees = [[0,4],[1,4],[6,2],[9,10],[11,12],[14,1],[16,1],[19,2]]  
>>> i = 0  
>>> s = 0  
>>> while(i<len(donnees)):  
    s=s+s[i][1]  
    i=i+1  
>>> s
```

2° Evaluate the following code

```
>>> j =0
>>> k= 0
>>> while(j<len(donnees)) :
    k = k + donnees[j][0]*donnees[j][1]
    j=j+1
>>> moyenne = k / s
```

Solutions

1°

```
>>> i =0
>>> s =0
>>> while(i<len(donnees)):
    s=s+s[i][1]
    i=i+1
>>> s
36
```

2°

```
>>> j =0
>>> k= 0
>>> while(j<len(donnees)) :
    k = k + donnees[j][0]*donnees[j][1]
    j=j+1
>>> moyenne = k / s
# 306 / 36
8.5
```

>>> 26.2. Skills on nested lists : part 2

A retenir

Pour parcourir une liste de longueur n, qui contient des sous-listes égaux à m

```
while (i < n)
```

```
    block
```

```
        while(j < m) :
```

```
            block
```

>>> Exercice 252

Evaluate the following code

```
>>> tab = [[1,2,3], [4,5,6], [7,8,9], [10,11,12]]
```

```
>>> i = 0
```

```
>>> while (i < len(tab) ) :
```

```
    j=0
```

```
    while(j<3) :
```

```
        print(tab[i][j], end = ' ')
```

```
        j=j+1
```

```
    i=i+1
```

Solutions

```
>>> i = 0
```

```
>>> while (i < len(tab) ) :
```

```
    j=0
```

```
    while(j<3) :
```

```
        print(tab[i][j], end = ' ')
```

```
        j=j+1
```

```
    i=i+1
```

```
1 2 3 4 5 6 7 8 9 10 11 12
```

>>> Exercice 253

Evaluate the following code

```
>>> l1 =[0,1,2]  
>>> l2 = [3,4,5]  
>>> l3 = [6,7,8]
```

Sum all the éléments of the lists

Solutions

```
>>> l = [l1, l2, l3]  
>>> l  
[ [0, 1, 2], [3, 4, 5], [6, 7, 8] ]  
>>> i =0  
>>> s = 0  
>>> while( i < 3) :  
    j = 0  
    while (j < 3) :  
        s = s + l[i][j]  
        j=j+1  
    i=i+1  
>>> s  
36
```

>>> **Exercise 254**

Sudoku grid

4	1	5	6	3	8	9	7	2
3	6	2	4	7	9	1	8	5
7	8	9	2	1	5	3	6	4
9	2	6	3	4	1	7	5	8
1	3	8	7	5	6	4	2	9
5	7	4	9	8	2	6	3	1
2	5	7	1	6	4	8	9	3
8	4	3	5	9	7	2	1	6
6	9	1	8	2	3	5	4	7

Evaluate the following code

1°

```
>>> i=0
>>> s = 0
>>> while (i< 9) :
    s = s + l[0][i]
    i = i+1
>>> s
```

2°

```
>>> i =0
>>> s = 0
>>> while (i < 9) :
    j = 0
    while(j< 9) :
        s = s + l[i][j]
        j = j +1
    i = i +1
>>> s
```

Solutions

```
>>> l=[[4,1,5,6,3,8,9,7,2], [3,6,2,4,7,9,1,8,5], [7,8,9,2,1,5,3,6,4],  
[9,2,6,3,4,1,7,5,8],[1,3,8,7,5,6,4,2,9],[5,7,4,9,8,2,6,3,1],[2,5,7,1,6,4,8,9,3],  
[8,4,3,5,9,7,2,1,6],[6,9,1,8,2,3,5,4,7]]
```

```
>>> i=0  
>>> s = 0  
>>> while (i< 9) :  
    s = s + l[0][i]  
    i = i+1  
>>> s  
45
```

2° Evaluate the following code

```
>>> i =0  
>>> s = 0  
>>> while (i < 9) :  
    j = 0  
    while(j< 9) :  
        s = s + l[i][j]  
        j = j +1  
    i = i +1  
>>> s  
405
```

>>> 27.Methods : Part 1

« faisons les choses avec méthode » Descartes

>>> 27.1.String Methods

Dès qu'on parle d'objet il existe des méthodes qui vont avec.

>>> **Exercise 255**

1° Evaluate the following code

```
>>> ch = 'hello'  
>>> i = 0  
>>> compteur = 0  
>>> while(i < len(ch)) :  
    if (ch[i] == 'l') :  
        compteur = compteur + 1  
    i = i + 1  
>>> print('l = ', compteur )
```

2° Evaluate the following code

```
>>> ch.count('l')  
>>> print('la lettre l apparaît', ch.count('l'), 'fois')
```

Solutions

1°

iteration	if (ch[i] == 'l')	compteur
1	False	0
2	False	0
3	True	1
4	True	2
5	False	0

```
>>> print('l = ', compteur )
```

```
# print('l =', 2)
```

```
l = 2
```

2°

```
>>> ch.count('l')  
# 'hello'.count('l')  
# count('hello', 'l')  
# 2
```

Tips

first method : `count`

help on method_descriptor:

`count(...)`

`S.count(sub[, start[, end]]) -> int`

Return the number of non-overlapping occurrences of substring sub in string S[start:end]. Optional arguments start and end are interpreted as in slice notation.

`>>> String.count(sub)`

>>> Exercise 256

Evaluate the following code

```
>>> ch = 'anticonstitutionnellement'
>>> var = ch.count('n')
>>> var2 = ch.count('k')
>>> ch * 2
>>> ch[1]= 'a'
```

Solutions



```
>>> var = ch.count('n')
# var = anticonstitutionnellement.count('n')
# var = anticonstitutionnellement.count('n')
# var = 5
```

```
>>> var
5
>>> var2 = ch.count('k')
>>> var2
0
>>> ch * 2
'anticonstitutionnellementanticonstitutionnellement'
>>> ch[1]='a'
TypeError: 'str' object does not support item assignment
```

>>> Exercice 257 Méthode find()

1° Evaluate the following code

```
>>> chaine = 'python'
>>> def recherche (s, c) :
    i = 0
    indice = -1
    while(i<len(s)) :
        if(s[i] == c) :
            indice = i
            i=i+1
    return indice
>>> recherche(chaine, 'p')
>>> recherche(chaine, 'k')
```

2° Evaluate the following code

```
>>> help(str.find)
Help on method_descriptor:
find(...)
```

S.find(sub[, start[, end]]) -> int

Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

```
>>> ch.find('p')  
>>> chaine.find('k')
```

Solutions

1°

```
>>> chaine = 'python'  
>>> def recherche (s, c) :  
    i = 0  
    indice = -1  
    while(i<len(s)) :  
        if(s[i] == c) :  
            indice = i  
        i=i+1  
    return indice  
>>> recherche(chaine, 'p')  
# if(s[0] == 'p') :  
#indice = 0  
# return 0  
0  
>>> recherche(chaine, 'k')  
-1
```

2°

Return the lowest index in **S** where substring sub is found

```
>>> chaine.find('p')
```

0

```
>>> chaine.find('k')
```

-1

>>> **Exercise 258**

Evaluate the following code

```
>>> ch = ' end of the line'
```

```
>>> ch.find('e')
```

```
>>> ch.find('o')
```

```
>>> ch.find(' ')
```

```
>>> ch.find(4)
```

Solutions

```
>>> ch.find('e')
```

1

```
>>> ch.find('o')
```

5

```
>>> ch.find(' ')
```

0

```
>>> ch.find(4)
```

TypeError : must be str, not int

```
>>> 27.1.1. split( )
```

« split » Titre d'un film

>>> **Exercise 259**

Evaluate the following code

```
>>> ch = 'bus'
```

```

>>> i=0
>>> a = []
>>> while(i<len(ch)) :
    a.append(ch[i])
    i=i+1
>>> b =[]
>>> b.append(ch)

```

2° Evaluate the following code

```

>>> help(str.split)
>>> ch = ' dans le bus'
>>> ch.split()
>>> liste = ch.split( )

```

Solutions

1°

iteration	i < len(ch)	a.append(ch[i])
1	True	['b']
2	True	['b','u']
3	True	['b','u','s']

```

>>> b =[]
>>> b.append(ch)
# b.append('bus')
# ['bus']
>>> b
['bus']

```

2°

```

>>> help(str.list)
Help on method_descriptor:

```

```
split(self, /, sep=None, maxsplit=-1)
```

Return a list of the words in the string, using sep as the delimiter string.

```
>>> ch = ' dans le bus'
```

```
>>> ch.split()
```

```
# ' dans le bus'.split()
```

```
# ['dans', 'le', 'bus']
```

```
['dans', 'le', 'bus']
```

```
>>> liste = ch.split( )
```

```
# liste = ['dans', 'le', 'bus']
```

>>> **Exercise 260**

1° Evaluate the following code

```
>>> ch = 'black rainbow'
```

```
>>> ch[5]
```

```
>>> def recherche(ch) :
```

```
    I = 0
```

```
    indice= -1
```

```
    while(i<len(ch)) :
```

```
        if(ch[i]== " ") :
```

```
            indice = i
```

```
        i=i+1
```

```
    return indice
```

```
>>> recherche(ch)
```

2° Evaluate the following code

```
>>> j=0
```

```
>>> a=[]
```

```
>>> while (j < recherche(ch)) :
```

```
a.append(ch[j])
j=j+1
>>> k = recherche +1
>>> b = []
>>> while(k < len(ch)) :
    b.append(ch[k])
    k=k+1
>>> b
>>> print(a,b)
>>> ch.split()
```

Solutions

1°

```
>>> ch = 'black rainbow'
>>> ch[5]
'
>>> def recherche(ch) :
    I = 0
    indice= -1
    while(i<len(ch)) :
        if(ch[i]== " ") :
            indice = i
        i=i+1
    return indice
>>> recherche(ch)
```

5

2°

```

iteration  j< recherche(5)    a.append(ch[j])
1         True                  ['b']
2         True                  ['b','l']
3         True                  ['b','l','a']
4         True                  ['b','l','a','c']
5         True                  ['b','l','a','c','k']

```

```

>>> a
['b', 'l', 'a', 'c', 'k']
>>> k = recherche +1
# k = 5 + 1
# k = 6
>>> b = []
>>> while(k < len(ch)) :
    b.append(ch[k])
    k=k+1
>>> b
['r', 'a', 'i', 'n', 'b', 'o', 'w']
>>> print(a,b)
['b', 'l', 'a', 'c', 'k'] ['r', 'a', 'i', 'n', 'b', 'o', 'w']
>>> ch.split()
['black', 'rainbow']

```

>>> **exercice 261**

1° Evaluate the following code

```

>>> ch = 'Aristote Platon Socrate'
>>> liste = [ ]
>>> i = 0
>>> while (i<len(ch)) :

```

```
liste.append(ch[i])  
i=i+1
```

2° Evaluate the following code

```
>>> liste2 = ch.split()  
>>> liste2
```

Solutions

1°

```
>>> ch = 'Aristote Platon Socrate'  
>>> liste = [ ]  
>>> i = 0  
>>> while (i<len(ch)) :  
    liste.append(ch[i])  
    i=i+1  
>>> liste  
['A', 'r', 'i', 's', 't', 'o', 't', 'e', ' ', 'P', 'l', 'a', 't', 'o', 'n', ' ', 'S', 'o', 'c', 'r', 'a', 't', 'e']
```

2°

```
>>> liste2 = ch.split()  
>>> liste2  
['Aristote', 'Platon', 'Socrate']
```

>>> 21.1.2. str.format ()

>>> **Exercise 262**

Evaluate the following code

```
>>> help(str.format)  
>>> '{}'.format('hello')  
>>> '{} {}'.format('hello', 'world')
```

Solutions

```
Help on method_descriptor:
```

```
format(...)
```

```
S.format(*args, **kwargs) -> str
```

Return a formatted version of `S`, using substitutions from `args` and `kwargs`.

The substitutions are identified by braces ('{' and '}').

```
>>> '{}'.format('hello')
```

```
# format('{}', 'hello')
```

```
# format('hello')
```

```
'hello'
```

```
>>> '{} {}'.format('hello', 'world')
```

```
# format('{} {}'.format('hello', 'world'))
```

```
# format('hello' 'world')
```

```
'hello world'
```

>>> **Exercise 263**

Evaluate the following code

```
>>> '{}'.format('salut')
```

```
>>> '{2} {1} {0}'.format('hello', 'salut', 'bonjour')
```

Solutions

```
>>> '{}'.format('salut')
```

```
# format('{}', 'salut')
```

```
# format('salut')
```

```
'salut'
```

```
>>> '{2} {1} {0}'.format('hello', 'salut', 'bonjour')
```

```
# format('{2} {1} {0}', 'hello', 'salut', 'bonjour')
```

```
# format('{bonjour} {salut} {hello}')
```

```
'bonjour salut hello '
```

Tips

```
>>> '{} {} {}'.format(str1, str1,str3)
' str1 str2 str3 '
>>> '{2} {0} {1}'.format(str1, str2,str3)
' str2 str1 str2 '
```

>>> 21.1.3.f-string

>>> **Exercise 263**

Evaluate the following code

```
>>> a = 2
>>> print('la valeur de a est', a)
>>> print(f'la valeur de a est {a}')
>>> print(f'{a+2}')
```

Solutions

```
>>> a = 2
>>> print('la valeur de a est', a)
# 'la valeur de a est', 2
la valeur de a est 2
>>> print(f'la valeur de a est {a}')
# print('la valeur de a est {2}')
# print('la valeur de a est 2'
la valeur de a est 2
>>> print(f'{a+2}')
# print(f'{2+2}')
# print(f'{4}')
# print('{4}')
# print(4)
```

Tips

```
>>> print(f"{{expression}}")
```

value

>>> 27.2. Methos on list object

« méthode, méthode »

```
>>> 27.2.1. sort ( )
```

>>> **Exercise 264**

1° Evaluate the following code

```
>>> l = [5,4,3,2,1]
>>> def tri (l) :
    i=0
    while(i<len(l)) :
        j=i
        while(j<len(l)) :
            if (l[j] < l[i]) :
                l[i],l[j] = l[j], l[i]
            j=j+1
        i=i+1
>>> tri(l)
```

2° Evaluate the following code

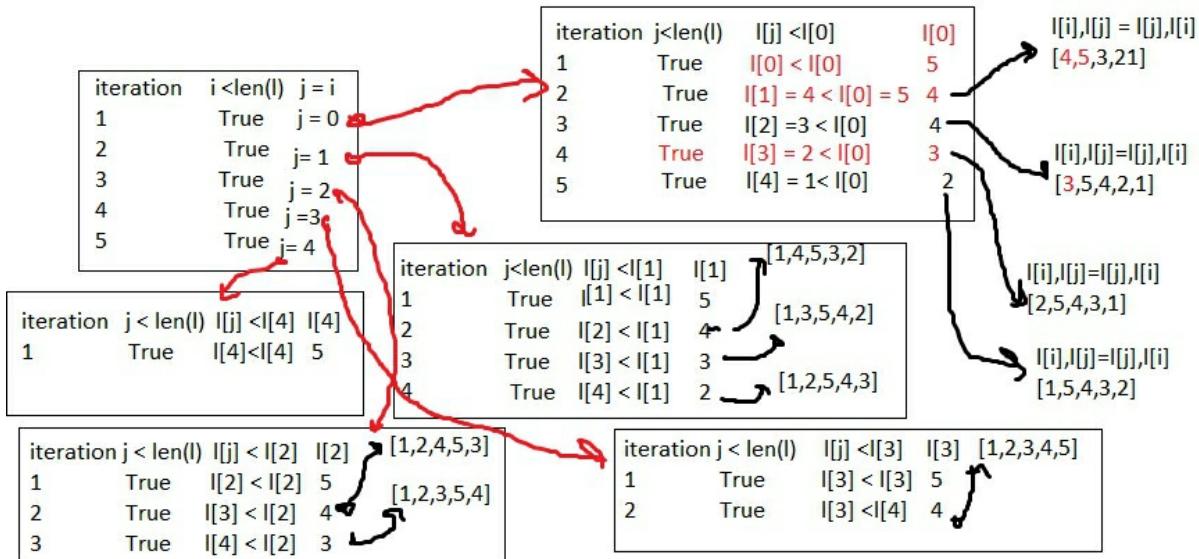
```
>>> l.sort()
```

Solutions

1°

```
>>> tri (l)
```

[1,2,3,4,5]



2°

```
>>> l.sort()
# [1,2,3,4,5].sort()
# sort([1,2,3,4,5])
[1, 2, 3, 4, 5]
```

>>> **Exercise 265**

Evaluate the following code

```
>>> l = [4,2,1,-1]
>>> l.sort()
>>> l2 = [2,2,11,4,-4,4]
>>> l2.sort()
```

Solutions

```
>>> l = [4,2,1,-1]
>>> l.sort()
>>> l
[-1, 1, 2, 4]
>>> l2 = [2,2,11,4,-4,4]
>>> l2.sort()
```

```
>>> l2
```

```
[-4, 2, 2, 4, 4, 11]
```

```
>>> 27.2.2. copy ( )
```

« copier, coller » clique droit

>>> Exercise 266

Evaluate the following code

```
>>> l1 = [1,2,[0,0,0],4,5]
```

```
>>> l2 = []
```

```
>>> i=0
```

```
>>> while(i<len(l1)) :
```

```
    l2.append(l1[i])
```

```
    i=i+1
```

```
>>> l2
```

```
>>> l3 = l1.copy( )
```

Solutions

```
>>> l2
```

```
[1,2,[0,0,0],4,5]
```

iteration	<code>i < len(l1)</code>	<code>l2.append(l1[i])</code>
1	True	[1]
2	True	[1,2]
3	True	[1,2,[0,0,0]]
4	True	[1,2,[0,0,0],4]
5	True	[1,2,[0,0,0],4,5]

```
>>> l3 = l1.copy()
```

```
>>> l3
```

```
[1, 2, [0,0,0], 4, 5]
```

>>> Exercise 267

Evaluate the following code

```
>>> l1 = [5,4,0,1,-1]
>>> l1.sort()
>>> l2 = l1.copy()
>>> if (l1 == l2) :
    print("same")
>>> l2.append(3)
>>> l1 == l2
```

Solutions

```
>>> l1 = [5,4,0,1,-1]
>>> l1.sort()
# l1 = [-1,0,1,4,5]
>>> l2 = l1.copy()
# l2 = l1.copy()
# l2 = [-1,0,1,4,5]
>>> if (l1 == l2) :
# if ([-1,0,1,4,5] == [-1,0,1,4,5]) :
# if (True) :
# print("same")
same
>>> l2.append(3)
# [-1,0,1,4,5].append(3)
# [-1,1,0,1,4,5,3]
>>> l1 == l2
False
```

```
>>> 27.2.3. join ( )
>>> Exercise 268
```

1° Evaluate the following code

```
>>> ch = 'python java C'  
>>> help(str.join)  
>>> '.join(['python', 'java', 'c'])
```

2° Evaluate the following code

```
>>> ';' .join(['python', 'java', 'c'])
```

Solutions

1°

```
>>> ch.split()  
# 'python java C'.split()  
# split('python java C', ' ')  
['python', 'java', 'C']  
>>> help(str.join)
```

Help on method_descriptor:

join(**self**, iterable, /)

Concatenate any number of strings.

The string whose method is called is inserted in between each given string.

The result is returned as a new string.

```
>>> '.join(['python', 'java', 'c'])  
# 'python' 'java' 'c'  
# python java c  
python java c
```

2°

```
>>> ';' .join(['python', 'java', 'c'])  
# 'python';'java';'c'  
# python java c
```

python;java;c

Tips

```
>>> string.join( iterable)
```

Iterable objects we have seen so far :

- Strings (str)
- Lists (list)

>>> Exercise 269

1° Evaluate the following code

```
>>> l = ['h','e','l','l','o']  
>>> ''.join(l)
```

2° Evaluate the following code

```
>>> l= ['hello', 'world']  
>>> ';' .join(l)
```

Solutions

1°

```
>>> l = ['h','e','l','l','o']  
>>> ''.join(l)  
hello
```

2°

```
>>> l= ['hello', 'world']  
>>> ';' .join(l)  
h;e;l;l;o
```

>>> Exercise 270

1° Evaluate the following code

```
>>> ",".join('hello')  
>>> 'hello'.split()
```

2° Evaluate the following code

```
>>> a = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'  
>>> a.split()  
>>> b= ';' .join(a)  
>>> b
```

Solutions

1°

```
>>> "," .join('hello')  
'h,e,l,l,o'  
>>> 'hello'.split()  
['hello']
```

2°

```
>>> a.split()  
['Lorem', 'ipsum', 'dolor', 'sit', 'amet,', 'consectetur', 'adipiscing', 'elit.'][br/>>>> b= ';' .join(a)  
'L;o;r;e;m; ;i;p;s;u;m; ;d;o;l;o;r; ;s;i;t; ;a;m;e;t;;; ;c;o;n;s;e;c;t;e;t;u;r;  
;a;d;i;p;i;s;c;i;n;g; ;e;l;i;t;.'
```

>>> 28. Introduction to files

« fichez moi tout le monde » observateur 1

>>> 28.1. Files 101

>>> 28.1.1. Open()

>>> **Exercise 271**

On a Linux shell

```
krang84@AWESOM-O:~$ pwd  
/home/utilisateur  
krang84@AWESOM-O:~$ ls  
film.mkv  python_basesfondamentales.pdf
```

```
krang84@AWESOM-O:~$ python3
```

```
>>>
```

Evaluate the following code

```
>>> f = open('fichier', 'w')
```

Solutions

Function name : open(builtin)

Arguments : str str

```
>>> f = open('fichier', 'w')
```

```
krang84@AWESOM-O:~$ ls
```

```
file.mkv fichier python_basesfondamentales.pdf
```

Tips

```
>>> help(open)
```

```
open(file, mode ='r', buffering =-1, encoding = None, errors = None,  
newline =None, closefd = True, opener = None)
```

>>> **Exercise 272**

1° Evaluate the following code

```
>>> objet_fichier1 = open('fichier1','w')
```

```
>>> fichier2 = open('fichier2','w')
```

2° on a windows system

```
>>> objet_fichier1 = open('fichier1','w')
```

```
PermissionError: [Errno 13] Permission denied: 'fichier'
```

Solutions

1°

Function name : open

Arguments : fichier1 w

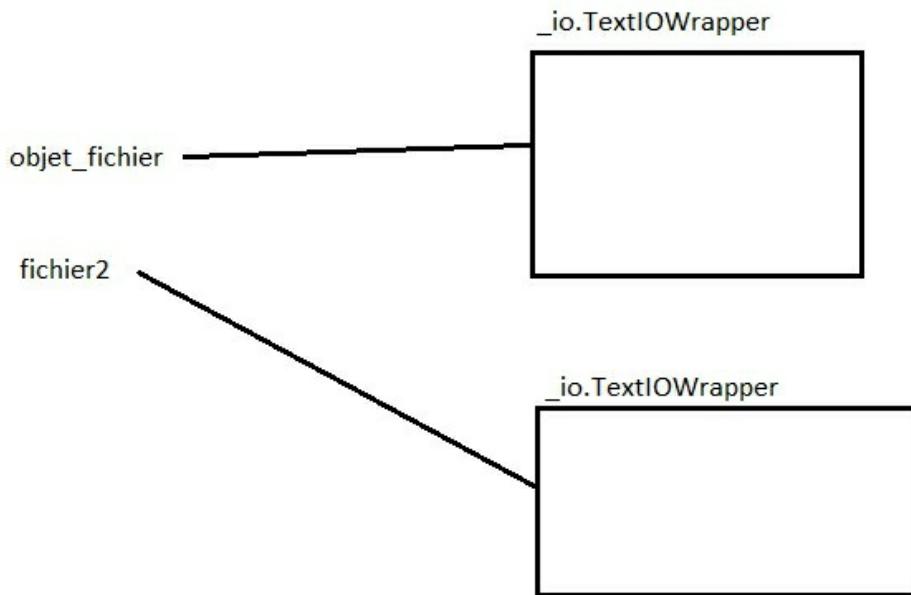
```
krang84@AWESOM-O:~$ ls
```

```
fichier1.txt
```

function name : open

arguments : fichier2 w

```
krang84@AWESOM-O:~$ ls
fichier1.txt fichier2.txt
```



2°

```
>>> objet_fichier1 = open('fichier1','w')
```

```
PermissionError: [Errno 13] Permission denied: 'fichier'
```

You have to give the absolute path for example

```
>>> objet_fichier1 = open(r"C:\Documents\code\fichier1", 'w')
```

>>> 28.1.2. write() and close ()

>>> **Exercise 273**

```
krang84@AWESOM-O:~$ ls
```

```
python_bases.pdf
```

Evaluate the following code

```
>>> f = open('fichier', 'w')
>>> f.write('Hello world ! ')
>>> f.close()
```

Solutions

```
>>> f = open('fichier', 'w')
```

On a Linux shell

```
krang84@AWESOM-O:~$ ls  
python_bases.pdf fichier
```

The method `write()` comes from class `io.TEXTIOBASE`

write(s)

Write the string `s` to the stream and return the number of characters written

```
>>> f.write('hello world')
```

```
14
```

```
>>> f.close()
```

The method `close` comes from class `io.IOBASE`

close()

Flush and close this stream. This method has no effect if the file is already closed. Once the file is closed, any operation on the file (e.g. reading or writing) will raise a `ValueError`.

As a convenience, it is allowed to call this method more than once; only the first call, however, will have an effect.

```
krang84@AWESOM-O:~$ ls  
python_bases.pdf fichier  
krang84@AWESOM-O:~$ cat fichier  
hello world
```

>>> Exercise 274

Evaluate the following code

```
>>> montexte = open('fichier','w')  
>>> montexte.write('spam egg')  
>>> montexte.close()
```

Solutions

```
>>> montexte.write('spam egg')
```

```
8
```

linux shell

```
krang84@AWESOM-O:~$ ls
```

```
fichier.txt
```

```
krang84@AWESOM-O:~$ cat fichier
```

```
spam egg
```

>>> **Exercise 275**

1° Evaluate the following code

```
>>> montexte = open('fichier','w')  
>>> montexte.write('spam egg')  
>>> montexte.close ()
```

2° Evaluate the following code

```
>>> montexte = open('fichier','w')  
>>> montexte.write('python')  
>>> montexte.write('3')
```

Solutions

1°

```
>>> montexte = open('fichier','w')  
>>> montexte.write('spam egg')  
8  
>>> montexte.close ()
```

```
spam egg
```

fichier

2°

```
>>> montexte = open('fichier','w')
>>> montexte.write('python')
6
>>> montexte.write('3')
1
```

```
python3
```

fichier

```
krang84@AWESOM-O:~$ ls
fichier.txt
krang84@AWESOM-O:~$ cat fichier
python3
```

>>> 28.2.read() and 'r'

>>> **Exercise 276**

Evaluate the following code

```
>>> f = open('fichier', 'w')
>>> f.write('hello python')
>>> f.close()
>>> f.read()
>>> a = open('fichier', 'r')
>>> a.read()
```

Solutions

```
>>> f = open('fichier', 'w')
>>> f.write('hello Python')
12
>>> f.close()
>>> f.read()
```

ValueError : I/O operation on closed file.

```
>>> a = open('fichier', 'r')
>>> a.read()
```

'Hello Python'

Tips :

Method **read()** comes from class **io.TEXTIOBASE**
read(size=-1)

Read and return at most size characters from the stream as a single str. If size is negative or None, reads until EOF

>>> **Exercise 277**

Evaluate the following code

```
>>> f = open('fichier', 'w')
>>> f.write('Hello World')
>>> f.read()
```

```
>>> f.close()  
>>> f = open('fichier', 'w')  
>>> f.write('Hello Python')  
>>> f.read()
```

Solutions

```
>>> f.write('Hello World')  
11  
>>> f.read()  
'Hello World'  
>>> f.close()  
>>> f = open('fichier', 'w')  
>>> f.write('Hello Python')  
12  
>>> f.read()  
io.UnsupportedOperation: not readable
```

>>> 28.2.1. Argument 'a'

```
>>> help(open)  
# [...] 'a'          open for writing, appending to the end of the file if it exists
```

>>> **Exercise 278**

Evaluate the following code

```
>>> f = open ('fichier ', 'a ')  
>>> f.write('hello python ')  
>>> f.close ()  
>>> f = open('fichier ', 'r ')  
>>> f.read ()  
>>> f.close ()
```

```
>>> f = open('fichier ', 'a ')
>>> f.write('spam egg ')
>>> f.close( )
>>> f = open('fichier ', 'r ')
>>> f.read ( )
```

Solutions

```
>>> f = open ('fichier ', 'a ')
>>> f.write('hello python ')
12
>>> f.close ( )
>>> f = open('fichier ', 'r')
>>> f.read ( )
'hello python '
>>> f.close()
>>> f = open('fichier ', 'a ')
>>> f.write('spam egg ')
8
>>> f.close()
>>> f = open('fichier ', 'r ')
>>> f.read()
'hello python8 '
```

>>> 28.3.Files and for
« file is an iterable object)

>>> **Exercise 279**

1° Evaluate the following code

```
>>> f = open(r'C:\program\python\livre\fichier','w')
```

```
>>> for i in range (100) :  
    f.write('hello world\n')  
>>> f.close()
```

2° Evaluate the following code

```
>>> f = open(r'C:\program\python\livre\fichier','w')  
>>> i=0  
>>> while(i<10) :  
    f.write(i)  
    i=i+1  
>>> f.close()  
>>> f = open(r'C:\program\python\livre\fichier','w')  
>>> for i in range (10) :  
    f.write('hello {i} fois\n')  
>>> f.close()
```

Solutions

1°

```
>>> f = open(r'C:\program\python\livre\fichier','w')  
>>> for i in range (100) :  
    f.write('hello world\n')  
12  
12  
12  
12  
...  
12  
>>> f.close()
```

```
C:\program\python\livre\fichier'
```

```
hello world
```

fichier

2°

```
>>> f = open(r'C:\program\python\livre\fichier','w')
>>> i=0
>>> while(i<10) :
    f.write(i)
    i=i+1
TypeError: write() argument must be str, not int
>>> f = open(r'C:\program\python\livre\fichier','w')
>>> for i in range (10) :
    f.write(f'hello {i} fois\n')
```

```
>>> f.close()
```

 fichier - Bloc-notes

Fichier Edition Format Affichage Aide

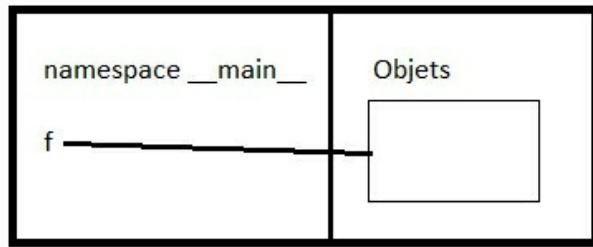
```
hello 0 fois
hello 1 fois
hello 2 fois
hello 3 fois
hello 4 fois
hello 5 fois
hello 6 fois
hello 7 fois
hello 8 fois
hello 9 fois
```

>>> **Exercise 280**

Evaluate the following code

```
>>> f = open(r'C:\program\python\livre\fichier','w')
>>> for i in range(10) :
    f.write(f'5 * {i} = {5*i}\n')
>>> f.close()
```

Solutions



```

iteration '5 * {i} = {5*i}\n'
1      5 * 0 = 0
2      5 * 1 = 5
3      5 * 2 = 10
4      5 * 3 =15
5      5 * 4 =20
6      5* 5 =25
7      5 * 6 =30
8      5 * 7 = 35
9      5 * 8 =40
10     5 * 9 = 45

```

>>> 28.3.1. Copy a file

>>> **Exercise 281**

fichier.txt

Hello world

Spam egg

Machine learning

1° Evaluate the following code

```

>>> f = open(r'C:\program\python\livre\fichier','r')
>>> for i in f :
    print(i)

```

2° Evaluate the following code

```

>>> f = open(r'C:\program\python\livre\fichier','r')
>>> f2= open(r'C:\program\python\livre\fichier2','w')
>>> for i in f :
    f2.write(i)

```

```
>>> f2.close()  
>>> f2= open(r'C:\program\python\livre\fichier2','r')
```

Solutions

1° f is an iterable object

```
>>> for i in f :
```

```
    print(i)
```

```
Hello world
```

```
Spam Egg
```

```
Machine Learning
```

```
iteration print(i)  
1     Hello world  
2     Spam egg  
3     Machine learning
```

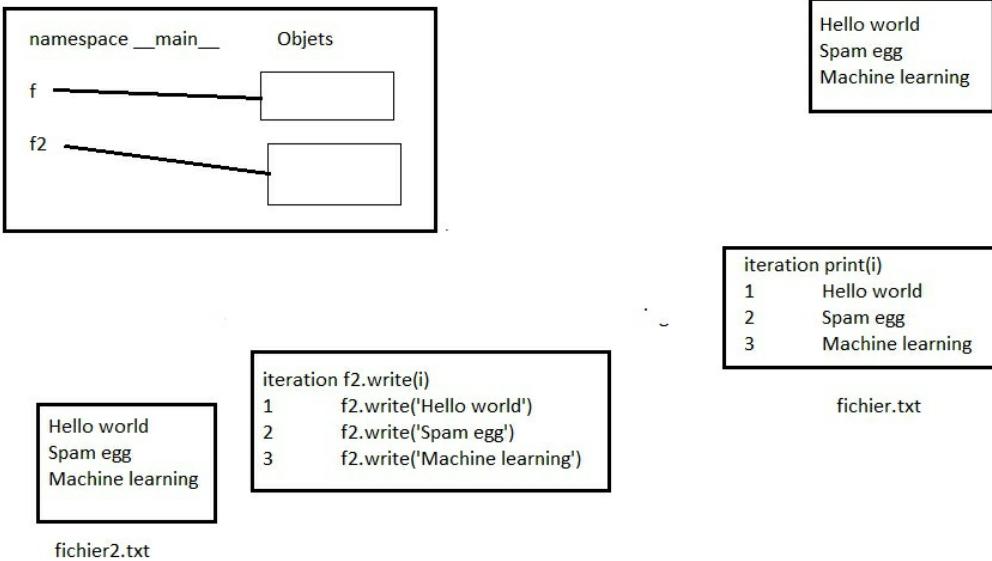
2°

```
>>> f = open(r'C:\program\python\livre\fichier','r')  
>>> f2= open(r'C:\program\python\livre\fichier2','w')  
>>> for i in f :  
    f2.write(i)  
>>> f2.close()  
>>> f2= open(r'C:\program\python\livre\fichier2','r')  
>>> for i in f2 :  
    print(i)
```

```
Hello world
```

```
Spam Egg
```

```
Machine Learning
```



>>> 29.Tuples

« n-uple, p-uple, t-uple » un cours de dénombrement

>>> 29.1.Tuples 101

>>> Exercise 282

Evaluate the following code

```
>>> t = (12345, 'python ')
>>> liste = [12345, 'python ']
>>> t[0]
>>> t[1]
>>> t[2]
>>> type(t)
>>> type(liste)
>>> liste[0] = 72
>>> liste
>>> t[0] = 72
```

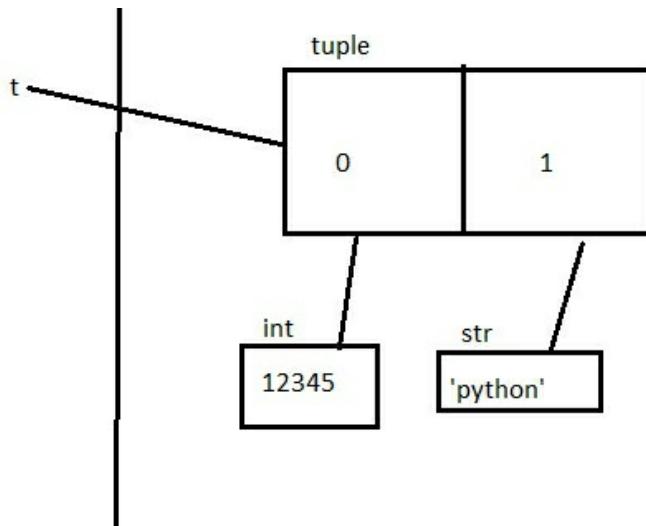
Solutions

```
>>> t = (12345, 'python')
```

```

>>> liste = [12345, 'python']
>>> t[0]
12345
>>> t[1]
'python'
>>> t[2]
IndexError : tuple out of range
>>> type(t)
<class 'tuple'>
>>> type(liste)
<class 'list'>
>>> liste[0]= 72
>>> liste
[72, 'python']
>>> t[0] = 72
TypeError : 'tuple' object doest not support item assignment

```



>>> **Exercise 283**

1° Create a tuple with three éléments, two numbers and a string

2° Evaluate the following code

```
>>> t = ()  
>>> t = (1)  
>>> t1 = (1,)  
>>> t2 = 1,  
>>> type (t)  
>>> type(t1)  
>>> type(t2)
```

Solutions

1° For example

```
>>> t = (1,2,'trois')
```

2°

```
>>> t = ()  
>>> t = (1)  
# t = (1)  
# t = 1  
>>> t1 = (1)  
>>> t2 = 1,  
>>> type (t)  
<class 'int'>  
>>> type(t1)  
<class 'tuple'>  
>>> type(t2)  
<class 'tuple'>
```

>>> 29.2.keyword in

>>> **Exercise 284**

1° Evaluate the following code

```
>>> t = (1, 'python', 2, 3, 5.8)
>>> len(t)
>>> t = (1, 'python', 2, 3, 5.8)
>>> i=0
>>> while (i<len(t)) :
    if (t[i] == 1) :
        print('1 est un élément du tuple')
    else :
        print("1 n'est pas un élément du tuple")
    i=i+1
```

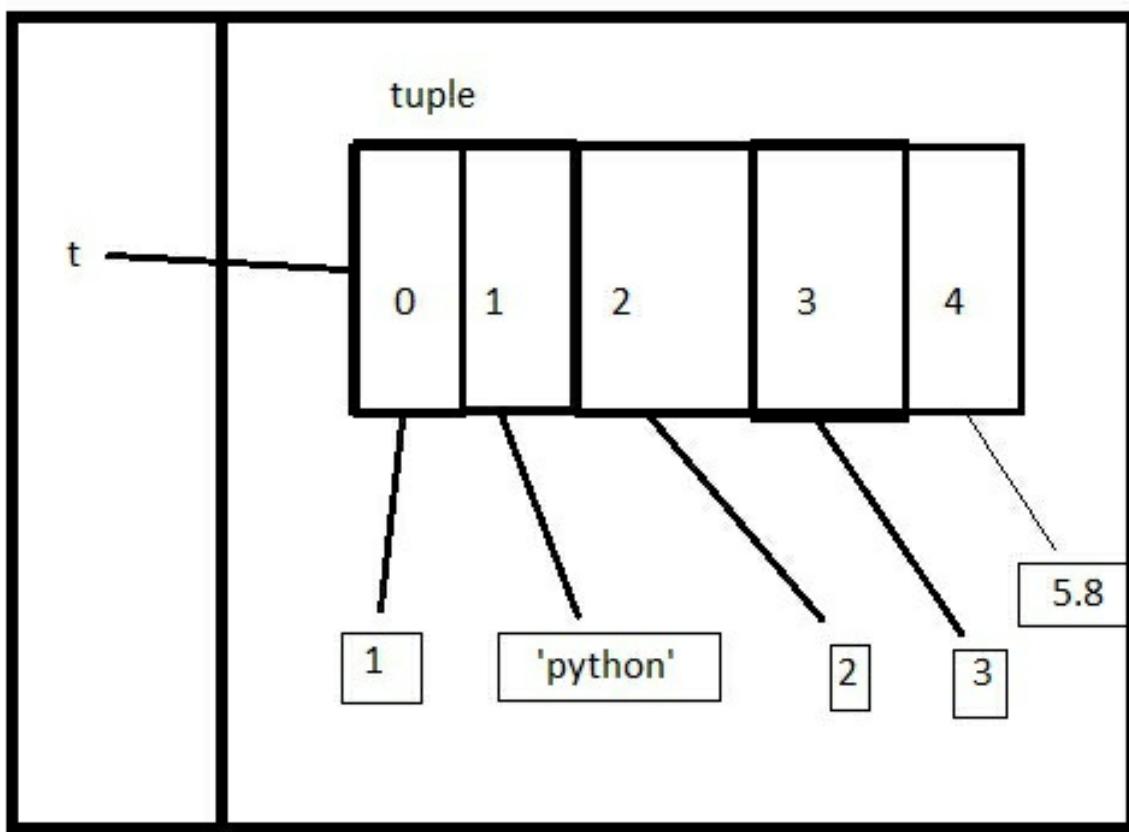
2° Evaluate the following code

```
>>> t = (1, 'python', 2, 3, 5.8)
>>> 1 in t
>>> var = 'python'
>>> var in t
>>> 9 in t
```

3° Evaluate the following code

```
>>> chaine = 'hello python 3'
>>> 'hello' in chaine
>>> liste = [[1,1],2,3, 'python']
>>> 1 in liste
>>> [1,1] in liste
>>> 'python' in liste
>>> var = 3
>>> var in liste
```

Solutions



1°

```
>>> t = (1, 'python', 2, 3, 5.8)
```

```
>>> len(t)
```

5

```
iteration  if (t[i] == 1):
1          True
2          False
3          False
4          False
5          False
```

2°

```
>>> t = (1, 'python', 2, 3, 5.8)
```

```
>>> 1 in t
```

True

```
>>> var = 'python'  
>>> var in t  
# 'python' in t  
# True  
True  
>>> 9 in t  
False
```

3° Voyons si cela marche avec les chaînes de caractère et la liste

```
>>> chaine = 'hello python '3'  
>>> 'hello' in chaine  
True  
>>> liste = [[1,1],2,3, 'python']  
>>> 1 in liste  
False  
>>> [1,1] in liste  
True  
>>> 'python' in liste  
True  
>>> var = 3  
>>> var in liste  
True
```

>>> Exercice 285

1° Evaluate the following code

```
>>> t = 1, 2, 'spam', 'egg'  
>>> i=0  
>>> while(i<len(t) ) :
```

```
if t[i] == 'spam' :  
    print('spam is in')  
else :  
    print("spam is not")
```

2° Use the keyword in

Solutions

1°

```
>>> i=0  
>>> while(i<len(t) ) :  
    if t[i] == 'spam' :  
        print('spam is in')  
    else :  
        print("spam is not")  
spam is in  
)
```

2°

```
>>> 'spam' in t :
```

```
True
```

>>> 29.3.Slicing [:]

« couper » un sabreur

>>> **Exercise 286**

Evaluate the following code

```
>>> t = (1, 'deux', 3, 'quatre', 5)  
>>> len(t)  
>>> t[0:3]  
>>> t[3:5]
```

```
>>> a = t[0:3]
>>> b = t[3:5]
```

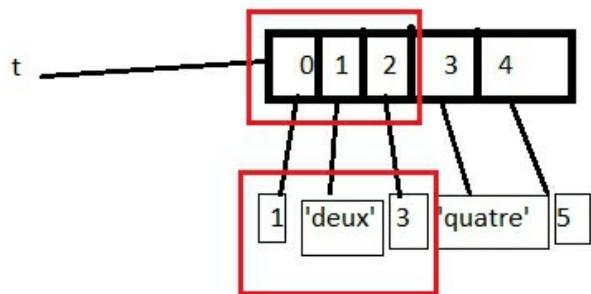
Solutions

```
>>> len(t)
```

```
5
```

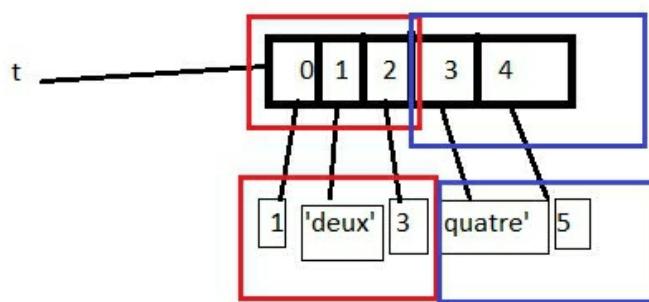
```
>>> t[0:3]
```

```
(1, 'deux', 3)
```



```
>>> t[3:5]
```

```
('quatre', 5)
```



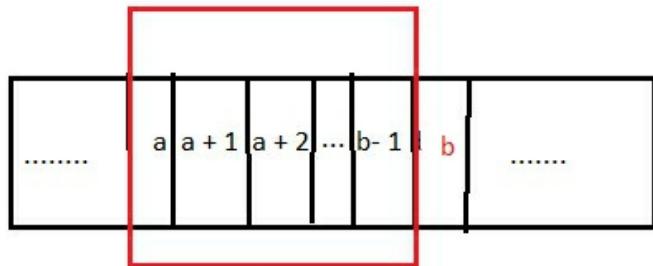
Tips

```
>>> object[a :b]
```

Gets element from a to $b-1$

Slicing are applied to :

- strings
- lists
- tuples



>>> Exercise 287

1° Evaluate the following code

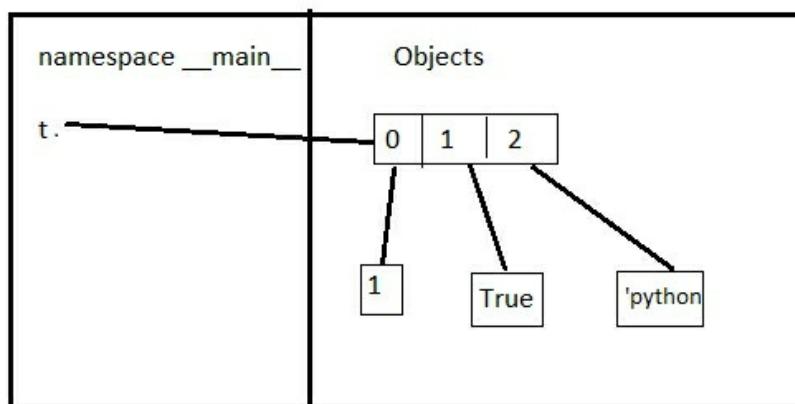
```
>>> t = (1, True, 'python')
>>> t[0:2]
>>> t[0:1]
```

2° Evaluate the following code

```
>>> ch = 'HELLO PYTHON'
>>> liste = [1,2,3,4, 'cinq']
>>> ch[0:4]
>>> ch[5:6]
>>> liste[1:4]
```

Solutions

1°



```
>>> t[0:2]
```

```
# (t[0],t[1])
```

```
# (1, True)
```

```
(1,True)
```

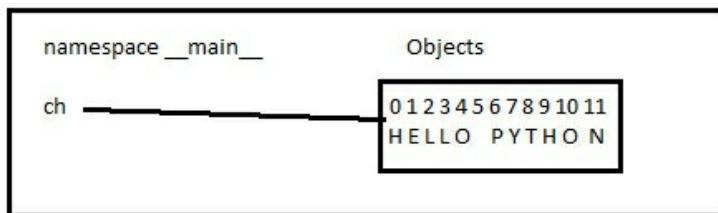
```
>>> t[0:1]
```

```
# (t[0])
```

```
# (1,)
```

```
(1,)
```

2°



```
>>> ch[0:4]
```

```
# 'ch[0]ch[1]ch[2]ch[3] '
```

```
# 'HELL'
```

```
'HELL'
```

```
>>> ch[5:6]
```

```
# ch[5]
```

```
# ''
```

```
''
```

```
>>> liste[1:4]
```

```
# [liste[1], liste[2], liste[3]]
```

```
# [1,2,3]
```

```
[1,2,3]
```

>>> 29.3.1.Slicing again [:]

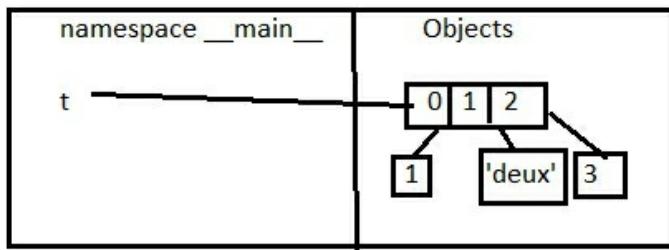
« slice, slice » un sliceur

>>> Exercise 288

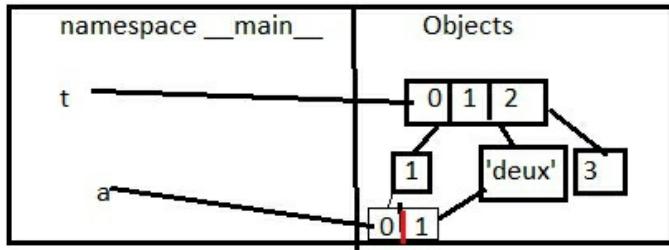
Evaluate the following code

```
>>> t = (1, 'deux',3)
>>> a = t[0:2]
>>> a
>>> type(a)
>>> t[:3]
>>> t[0:]
>>> t[:]
>>> b = t[:]
```

Solutions



```
>>> a = t[0:2]
# a = t[0:2]
# a = (t[0], t[1])
# a = (1, 'deux')
>>> a
(1, 'deux')
>>> type(a)
<class 'tuple'>
```



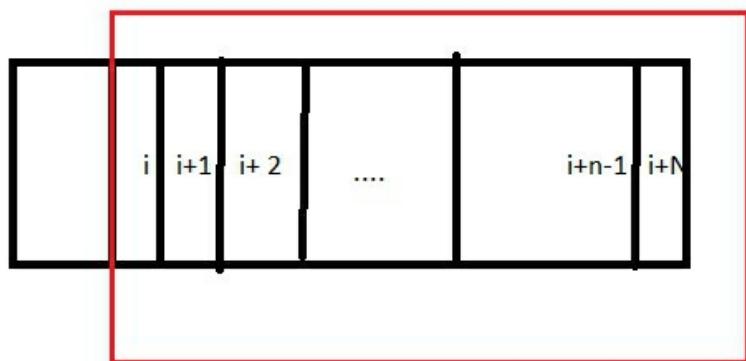
```

>>> t[:3]
# (t[0],t[1],t[2])
# (1, 'deux', 3)
(1, 'deux', 3)
>>> t[0:]
# (t[0],t[1],t[2])
# (1, 'deux', 3)
(1, 'deux', 3)
>>> t [ : ]
(1, 'deux', 3)
>>> b = t [ : ]
# b = (1, 'deux', 3)
>>> b
(1, 'deux', 3)
>>> b == t
True

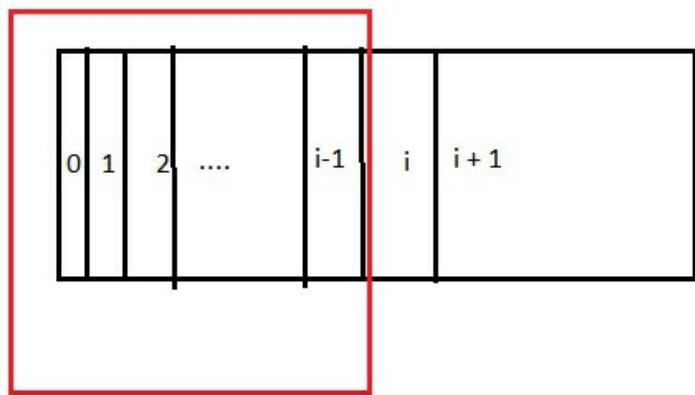
```

Tips :

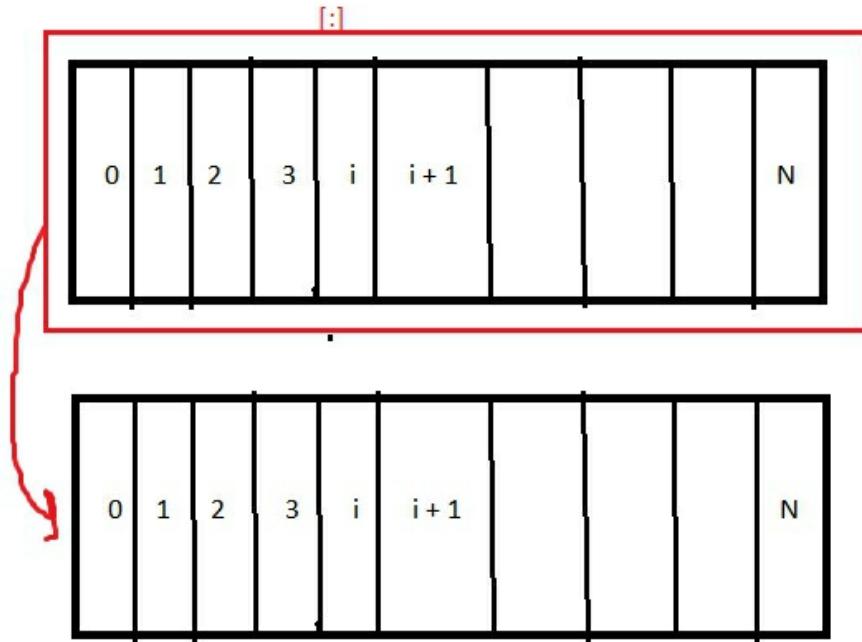
- [i :]



• $[: i]$



• $[:]$



>>> Exercice 289

1° Evaluate the following code

```
>>> chaine ='java, C et Python'
>>> liste = [2,3,5,7,11,13]
>>> tulipe = (1, 'deux', 3, True, False)
>>> chaine[1:4]
>>> liste[2:5]
>>> tulipe[1:3]
```

2° Copy the three sequences (chaine, liste and tuple) with [:]

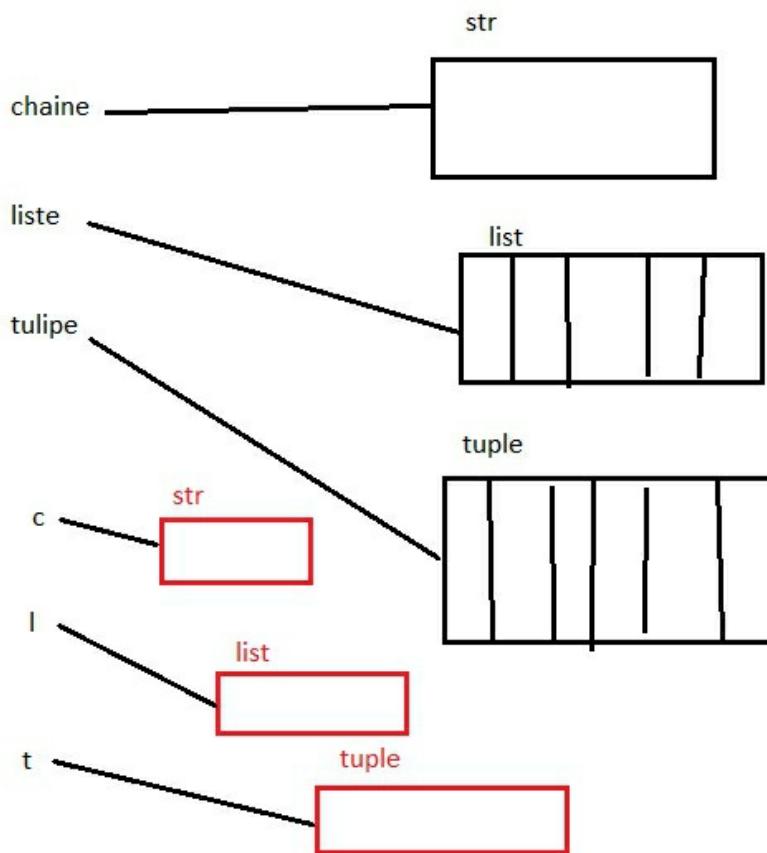
Solutions

```
>>> chaine ='java, C et Python'
>>> liste = [2,3,5,7,11,13]
>>> tulipe = (1, 'deux', 3, True, False)
>>> chaine[1:4]
# chaine[1]+chaine[2]+chaine[3]
# a+v+a
```

```
'ava'  
>>> liste[2:5]  
# liste[2] + liste[3] + liste[4]  
[5, 7, 11]  
>>> tulipe[1:3]  
# tulipe[1]+tulipe[2]  
# (deux,) + (3,)  
# (deux, 3)  
(deux, 3)
```

2°

```
>>> c = chaine[:]  
>>> l = liste[:]  
>>> t = tulipe[:]
```



>>> Exercise 290

Evaluate the following code

```

>>> p = (2,3,5,7,11,13,17)
>>> p[2:]
>>> p[:2]
>>> t = p[:]
  
```

Solutions

```

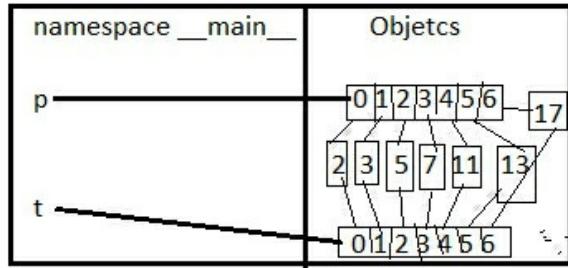
>>> p = (2,3,5,7,11,13,17)
>>> p[2:]
# p[2]+ p[3] + p[4] + p[5] + p[6]
# (5,7,11,13,17)
(5,7,11,13,17)
  
```

```

>>> p[:2]
# p[0]+p[1]
# (2,3)
(2,3)

>>> t = p[:]
# t = (2,3,5,7,11,13,17)
>>> t
(2,3,5,7,11,13,17)

```



>>> 29.3.2.Slicing and negative indexes

>>> **Exercise 291**

1° Evaluate the following code

```

>>> ch = 'kayak'
>>> len(ch)
5
>>> ch[0]
>>> ch[1]
>>> ch[2]
>>> ch[3]
>>> ch[4]

```

2° Evaluate the following code

```

>>> ch[-5]
>>> ch[-4]

```

```
>>> ch[-3]
>>> ch[-2]
>>>ch[-1]
```

3°Evaluate the following code

```
>>> ch[-5 :-1]
>>> ch[-4:]
>>> ch[-5:]
>>> ch[:-1]
>>> b= ch[:]
```

Solutions

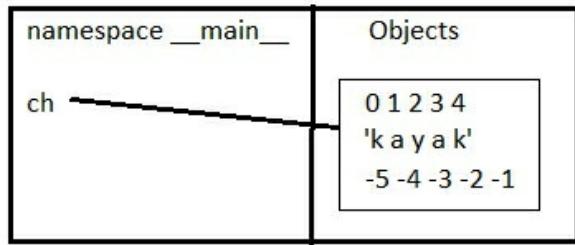
1°

```
>>> ch[0]
'k'
>>> ch[1]
'a'
>>> ch[2]
'y'
>>> ch[3]
'a'
>>> ch[4]
'k'
```

2°

```
>>> ch[-5]
'k'
>>> ch[-4]
'a'
>>>ch[-3]
```

```
'y'  
>>> ch[-2]  
'a'  
>>> ch[-1]  
'k'
```

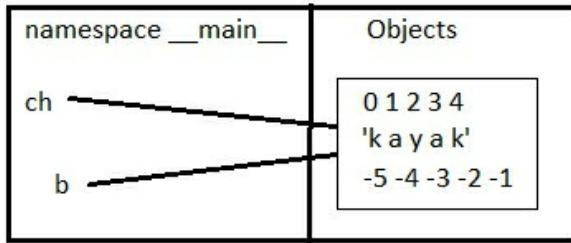


3°

```
>>> ch[-5:-1]  
# 'ch[-5]+ch[-4]+ch[-3]+ch[-2]'  
# 'kaya'  
'kaya'  
>>> ch[-4:]  
# 'ch[-4]+ch[-3]+ch[-2]+ch[-1]'  
# 'ayak'  
'ayak'  
>>> ch[-5:]  
# 'ch[-5]+ch[-4]+ch[-3]+ch[-2]+ch[-1]'  
# 'kayak'  
'kayak'  
>>> ch[: -1]  
# 'ch[-5]+ch[-4]+ch[-3]+ch[-2]'  
# 'kaya'  
>>> b = ch[:]
```

```
>>> b
```

```
'kayak'
```



>>> Exercise 292

Evaluate the following code

```
>>> ch = 'anticonstitutionnellement'  
>>> ch[-6 :-1]  
>>> len(ch)  
>>> ch[5:4]  
>>> chc= ch[:]  
>>> ch[0:]  
>>> ch[-100]
```

Solutions

```
>>> ch = 'anticonstitutionnellement'
```

```
>>> ch[-6 :-1]
```

```
'lemen'
```

```
>>> len(ch)
```

```
25
```

```
>>> ch[5:4]
```

```
"
```

```
>>> chc= ch[:]
```

```
# chc = 'anticonstitutionnellement'
```

```
>>> ch[0:]
```

```
# ch[0] + ch[1] + ... + ch[24]
```

```
# 'anticonstitutionnellement'
```

```
>>> ch[-100]
```

IndexError: string index out of range

```
>>> 29.3.3.Slicing and step [ :: ]
```

```
>>> Exercise 293
```

1° Evaluate the following code

```
>>> ch = 'anticonstitutionnellement'
```

```
>>> len(ch)
```

```
25
```

```
>>> ch[0:25:1]
```

```
>>> ch[0:25:2]
```

```
>>> ch[::-3]
```

2° Evaluate the following code

```
>>> ch[-25 :-11:2]
```

```
>>> ch [ : -1 : 1]
```

```
>>> ch [ :: -1]
```

```
>>> ch [ ::-2]
```

Solutions

1°

```
>>> ch [0:25:1]
```

```
# 'ch[0]+ch[1]+...+ch[24] '
```

```
# 'anticonstitutionnellement'
```

anticonstitutionnellement

```
>>> ch[0:25:2]
```

```
# 'ch[0]+ch[2]+ch[4]'+...+ch[24]'
```

```
# 'atcnnnttonleet'
```

```
'atcnttonleet'  
>>> ch[::-3]  
# 'ch[0] + ch[3] + ch[6] + ... + ch[24] '  
# 'ainitnlmt'  
'ainitnlmt'
```

2°

```
>>> ch[-25:-11:2]  
# 'ch[-25] + ch[-23] + ... + ch[-13] '  
# 'atnctt'  
'atnctt'  
>>> ch[ : -1 : 1]  
# 'ch[-25] + ch[-24] + ... + ch[-2] '  
# 'anticonstitutionnellemen'  
'anticonstitutionnellemen'  
>>> ch[ : :-1]  
'tnemellennoitutitsnocitna'  
>>> ch[ : :-2]  
''
```

Tips :

- `[: :-1]` reverses the sequence

>>> **Exercise 294**

Evaluate the following code

```
>>> ch = 'bibliotheque'  
>>> ch[0]  
>>> ch[-1]  
>>> ch[::-1]  
>>> ch[ ::-1]
```

```
>>> ch[-1000 :-1]
```

Solutions

```
>>> ch = 'bibliothèque'
```

```
>>> ch[0]
```

b

```
>>> ch[-1]
```

e

```
>>> ch[::-1]
```

'bibliothèque'

```
>>> ch[::-1]
```

eugehtoilbib

>>> 29.3.3.1.Palindrome and slicing

« Dans la langue française un palindrome est un mot qui s'écrit de façon identique dans les deux sens de lecture »

Kayak

Bob

kook

>>> **Exercise 295**

1° if way

```
>>> a = 'bob'
```

```
>>> len(a)
```

```
>>> if (a[-1] == a[0]) :
```

```
    print(a, 'est un palindrome')
```

```
>>> b = 'kayak'
```

```
>>> if (b[-1] == b[0] and b[-2] == b[1] )
```

```
    print(b, 'est un palindrome')
```

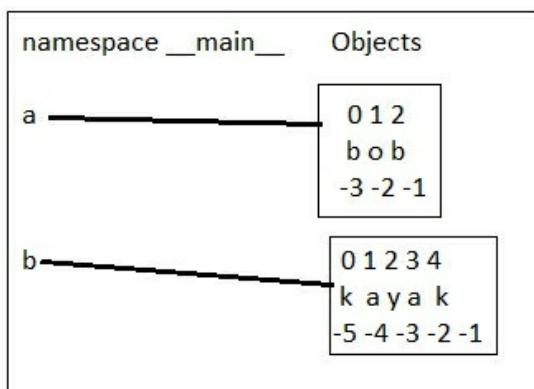
2° slice [: :-1]

```

>>> c = 'palindrome'
>>> d = c[::-1]
>>> if (d == c) :
    print(c, 'est un palindrome')
>>> b ='kayak'
>>> f = b[::-1]
>>> if (b == f) :
    print(f, 'est un palindrome')

```

Solutions



1°

```

>>> a= 'bob'
>>> len(a)
3
>>> if (a[-1] == a[0]) :
# if ('b' == 'b')
# if (True)
# print(a, 'est un palindrome')
# print(bob 'est un palindrome')
# bob est un palindrome
bob est un palindrome

```

```
>>> if (b[-1] == b[0] and b[-2] == b[1]) :  
# if ('k' == 'k' and 'a' == 'a') :  
# if (True and True) :  
# if True :  
# print(b, 'est un palindrome')  
# print('kayak', 'est un palindrome')  
# 'kayak est un palindrome'  
kayak est un palindrome
```

2°

```
>>> c = 'palindrome'  
>>> d = c[::-1]  
'emordnilap'  
>>> if (d == c) :  
# if ('palindrome' == 'emordnilap') :  
# if (False) :  
  
>>> b = 'kayak'  
>>> f = b[::-1]  
# f = b [::-1]  
# f = 'kayak'  
>>> if (b == f) :  
# if ('kayak' == 'kayak')  
# if (True) :  
# print (f, 'est un palindrome')  
# print ('kayak', 'est un palindrome')  
# print('kayak est un palindrome')  
kayak est un palindrome
```

>>> Exercise 296

```
>>> pal = 'esope reste ici et se repose'
```

Is pal a palindrome ?

Solutions

```
>>> pal[::-1]
```

```
esoper es te ici etser epose
```

```
>>> ch = pal[::-1]
```

```
>>> ch == pal
```

```
False
```

>>> 29.4. Operations on sequences

>>> Exercise 297

1° Evaluate the following code

```
>>> t = ('un', 2, 'trois')
```

```
>>> t1 = (4, 'cinq', True)
```

```
>>> t + t1
```

```
>>> otd = t + t1
```

```
>>> print(otd)
```

2° Evaluate the following code

```
>>> tulipe = (True, False)
```

```
>>> tulipe * 3
```

```
>>> trois_tulipe = tulipe*3
```

```
>>> print(trois_tulipe)
```

Solutions

1°

```
>>> t + t1
```

```
# ('un', 2, 'trois') + (4, 'cinq', True)
```

```
# ('un', 2, 'trois', 4, 'cinq', True)
>>> otd = t + t1
# otd = ('un', 2, 'trois', 4, 'cinq', True)
>>> print(otd)
('un', 2, 'trois', 4, 'cinq', True)
```

2°

```
>>> tulipe * 3
# (True, False) * 3
# (True, False, True, False, True False)
>>> trois_tulipe = tulipe * 3
# trois_tulipe = (True, False) * 3
# trois_tulipe = (True, False, True, False, True False)
>>> print(trois_tulipe)
(True, False, True, False, True False)
```

>>> **Exercise 298**

1° Evaluate the following code

```
>>> ch = 'ABABABABAB'
>>> ch1 = 'C'
>>> ch + ch1
>>> 'AAA' * 3
```

2° Evaluate the following code

```
>>> li = [1,2,3]
>>> li2 = [4,5]
>>> li + li2
>>> li * 3 + li2 * 4
```

3° Evaluate the following code

```
>>> tuple = ('rose', 'tulipe', False)
```

```
>>> tuple2 = ('Vrai', 'Java')
>>> tuple * 2 + tuple2 * 4
```

Solutions

1°

```
>>> ch = 'ABABABABABAB'
>>> ch1 = 'C'
>>> ch + ch1
# 'ABABABABABAB' + 'C'
# 'ABABABABABABC'
'ABABABABABABC'
>>> 'AAA' * 3
'AAAAAAA'
```

2°

```
>>> li = [1,2,3]
>>> li2 = [4,5]
>>> li + li2
# [1,2,3] + [4,5]
# [1,2,3,4,5]
[1,2,3,4,5]
>>> li * 3 + li2 * 4
# [1,2,3] * 3 + [4,5] * 4
# [1,2,3,1,2,3,1,2,3] + [4,5,4,5,4,5,4,5]
# [1,2,3,1,2,3,1,2,3,4,5,4,5,4,5,4,5]
[1, 2, 3, 1, 2, 3, 1, 2, 3, 4, 5, 4, 5, 4, 5, 4, 5]
```

3°

```
>>> tuple = ('rose', 'tulipe', False)
```

```
>>> tuple2 = ('Vrai', 'Java')
>>> tuple * 2 + tuple2 * 4
# ('rose', 'tulipe', False) * 2 + ('Vrai', 'Java') * 4
# ('rose', 'tulipe', False, 'rose', 'tulipe', False) + ('Vrai', 'Java','Vrai',
'Java','Vrai', 'Java','Vrai', 'Java')
# ('rose', 'tulipe', False, 'rose', 'tulipe', False, 'Vrai', 'Java','Vrai', 'Java','Vrai',
'Java','Vrai', 'Java')
('rose', 'tulipe', False, 'rose', 'tulipe', False, 'Vrai', 'Java','Vrai', 'Java','Vrai',
'Java','Vrai', 'Java')
```

>>> 29.5.Tuple unpacking

>>> **Exercise 299**

1° Evaluate the following code

```
>>> tuple = ('Hello World', 1, True)
>>> a,b,c = tuple
>>> a
>>> b
>>> c
```

2° Evaluate the following code

```
>>> li = [1,2]
>>> a,b = li
>>> print(a,b)
```

Solutions

1°

```
>>> a,b,c = tuple
# a,b,c = ('Hello World', 1, True)
# a = 'Hello World'
# b = 1
```

```
# c = True
```

```
>>> a
```

```
'Hello World'
```

```
>>> b
```

```
1
```

```
>>> c
```

```
True
```

2°

```
>>> a, b = li
```

```
# a, b = [1, 2]
```

```
# a = 1
```

```
# b = 2
```

```
>>> print(a, b)
```

```
# print(1,2)
```

```
1 2
```

Tips

Tuple unpacking

```
>>> var1, var2, ..., varN = (el1, el2, ...., elN)
```

>>> Exercise 300

Evaluate the following code

```
>>> a,b,c,d = (1,2,3,4)
```

```
>>> print(a,b,c,d)
```

```
>>> e,f = (1,)
```

Solutions

```
>>> a,b,c,d = (1,2,3,4)
```

```
# a = 1
```

```
# b = 2
```

```
# c = 3
# d = 4
>>> print(a,b,c,d)
# print(1,2,3,4)
```

```
1 2 3 4
```

```
>>> e,f = (1,)
# e = 1
# f = ???
```

ValueError: not enough values to unpack (expected 2, got 1)

>>> 29.5.1. Tuple unpacking : notation *

>>> **Exercise 301**

Evaluate the following code

```
>>> tu = (1, 2, 'trois', 'quatre')
>>> x, *y = tu
>>> print(x,y)
>>> *c, d = tu
>>> print(c,d)
```

Solutions

```
>>> tu = (1, 2, 'trois', 'quatre')
>>> x, *y = tu
>>> x, *y = (1, 2, 'trois', 'quatre')
# x = 1
# y = [2, 'trois', 'quatre']
>>> print(x, y)
# print(1, [2, 'trois', 'quatre'])
```

```
1 [2, 'trois', 'quatre']
```

```
>>> *c, d = tu
>>> *c, d = (1,2,'trois', 'quatre')
# c = [1,2,'trois']
# d = 'quatre'
>>> print(c,d)
# print([1,2,'trois'],'quatre')
[1,2,'trois'] 'quatre'
```

>>> Exercise 302

Evaluate the following code

```
>>> tuple = ('a','b','c',1,2,3,4,True,False,('tuple', 'tulipe'))
>>> x,y,z = tuple
>>> print(x,y,z)
>>> *k,j = tuple
>>> print(k)
>>> print(j)
```

Solutions

```
>>> tuple = ('a','b','c',1,2,3,4,True,False,('tuple', 'tulipe'))
>>> x,y,z = tuple
ValueError: too many values to unpack (expected 3)
>>> print(x,y,z)
NameError: name 'x' is not defined
>>> *k,j = tuple
# k = ('a','b','c',1,2,3,4,True,False)
# j = ('tuple', 'tulipe')
>>> print(k)
('a','b','c',1,2,3,4,True,False)
```

```
>>> print(j)
('tuple', 'tulipe')
```

>>> 29.6. Back to for

>>> 29.6.1. range ()
« range () ta chambre » anon324

>>> **Exercise 303**

1° Evaluate the following code

```
>>> a = [0,1,2,3,4,5,6,7,8,9]
>>> for i in a :
    print(i+1, end = ' ')
```

2° Evaluate the following code

```
>>> for i in range(10) :
    b = i+1
    print(b, end = ' ')
```

Solutions

```
>>> a = [0,1,2,3,4,5,6,7,8,9]
>>> for i in a :
    print(i+1, end = ' ')
1 2 3 4 5 6 7 8 9 10
```

iteration	print(i+1, end="")
1	1
2	2
3	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11

2°

```
>>> for i in range(10) :  
# for i in 0,1,2,3,4,6,7,8,9 :  
1 2 3 4 5 6 7 8 9 10
```

Tips :

- for loop applies to iterable objects
- iterable objects are
 - string
 - liste
 - tuple
 - files

```
range -----> iterable
```

>>> Exercise 304

1° Evaluate the follow code

```
>>> for i in range(15) :  
    print(i*2, end=' ')  
>>> for i in range(10) :  
    print(i/10, end=' ')
```

2° Evaluate the following code

```
>>> b = 0  
>>> for i in range(101) :  
    b = b+i  
>>> print(b)
```

Solutions

1°

```
>>> for i in range(15) :  
    print(i*2, end=' ')
```

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
```

```
>>> for i in range(10) :
```

```
    print(i/10, end=' ')
```

```
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

2°

```
>>> b = 0
```

```
>>> for i in range(101) :
```

```
    b = b + i
```

```
# b = 0
```

```
# b = 1
```

```
# ...
```

```
# b = 5050
```

```
>>> print(b)
```

```
5050
```

>>> 29.6.1.1. range() against the machine

>>> **Exercise 305**

1° Evaluate the following code

```
>>> for i in range(10) :
```

```
    print(i, end = " ")
```

2° Evaluate the following code

```
>>> for i in range(2,8) :
```

```
    print(i, end = ")")
```

Solutions

1°

```
>>> for i in range(10):
```

```
    # for i in range 0,1,2,3,4,5,6,7,8,9
```

```
        print(i)
```

```
0 1 2 3 4 5 6 7 8 9
```

2°

```
>>> for i in range(2,8) :  
# for i in 2,3,4,5,6,7  
    print(i, end = ")
```

```
2 3 4 5 6 7
```

>>> **Exercise 306**

1° Evaluate the following code

```
>>> a= ['un', 'deux', 3]  
>>> i =0  
>>> while (i < len(a)) :  
    print(i, a[i])  
    i = i + 1
```

2° Evaluate the following code

```
>>> for i in a :  
    print(i, a[i])  
>>> for i in (range(len(a))) :  
    print(i, a[i])
```

Solutions

1°

```
>>> a= ['un', 'deux', 3]  
>>> i =0  
>>> while (i < len(a)) :  
    print(i, a[i])  
    i = i + 1
```

```
0 un
```

```
1 deus
```

2 3

2°

```
>>> for i in a :
```

```
    # i = 'un'
```

```
    # print('un',a['un'])
```

TypeError: list indices must be integers or slices, not str

```
>>> for i in (range(len(a)) :
```

```
    # for i in range(2) :
```

```
    # for i in 0,1,2 :
```

```
        print(i, a[i])
```

0 un

1 deux

2 3

Tips

```
>>> for i in range(len(sequence)) :
```

```
    block
```

>>> 29.6.2.list compréhension and for

>>> **Exercise 307**

1° Evaluate the following code

```
>>> for i in range(11) :
```

```
    print(i, "fois 7 = ", i*7)
```

2° Evaluate the following code

```
>>> a = [print(i, " fois 7 = ", i*7) for i in range(11)]
```

Solutions

1°

0 fois 7 = 0

1 fois 7 = 7

2 fois 7 = 14

3 fois 7 = 21

4 fois 7 = 28

5 fois 7 = 35

6 fois 7 = 42

7 fois 7 = 49

8 fois 7 = 56

9 fois 7 = 63

10 fois 7 = 70

>>>

```
iteration      print(i,"fois 7", i*7)
1              0 fois 7 = 0
2              1 fois 7 = 7
3              2 fois 7 = 14
4              3 fois 7 = 21
5              4 fois 7 = 28
6              5 fois 7 = 35
7              6 fois 7 = 42
8              7 fois 7 = 49
9              8 fois 7 = 56
10             9 fois 7 = 63
11             10 fois 7 = 70
```

2°

```
>>> a = [print(i, " fois 7 = ", i*7) for i in range(11)]
```

0 fois 7 = 0

1 fois 7 = 7

2 fois 7 = 14

3 fois 7 = 21

4 fois 7 = 28

5 fois 7 = 35

6 fois 7 = 42

```
7 fois 7 = 49  
8 fois 7 = 56  
9 fois 7 = 63  
10 fois 7 = 70
```

```
>>> a
```

```
[None, None, None, None, None, None, None, None, None, None]
```

Tips

List comprehension

```
>>> [ expression boucle_for ]
```

>>> Exercise 308 (stars)

4 rows and 5 stars

```
*****  
*****  
*****  
*****
```

1° Evaluate the following code

```
>>> nombre_lignes = int(input('rows ? '))  
rows ? 3  
>>> for i in range (nombre_lignes) :  
    print('*****')
```

2° Evaluate the following code

```
>>> def affichage_etoile5( ) :  
    for i in range() :  
        print('*****')  
>>> affichage_etoile5 (4)  
>>> affichage_etoile5 (7)
```

3° Evaluate the following code

```
>>> a = [print('*' * 5) for i in range(5)]
```

solutions

1°

```
>>> for i in range(3) :
```

```
    print('*****')
```

```
*****
```

```
*****
```

```
*****
```

```
>>> nombre_lignes = int(input('rows? '))
```

```
rows ? 3
```

```
>>> for i in range (3) :
```

```
    print('*****')
```

```
*****
```

```
*****
```

```
*****
```

2°

```
>>> def affichage_etoile5(n) :
```

```
    for i in range(n) :
```

```
        print('*****')
```

```
>>> affichage_etoile5(4)
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

```
>>> affichage_etoile5(7)
```

```
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

```
>>>
```

3°

```
>>> a = [print('*' * 5) for i in range(5)]  
# a = for i in range(5) :  
    print('*' * 5)
```

```
*****  
*****  
*****  
*****  
*****  
*****
```

>>> **Exercise 309**

1° Evaluate the following code

```
>>> print('*' * 5)
```

2° Evaluate the following code

```
>>> def affichage_etoile (lignes, etoiles) :  
    for i in range (lignes) :  
        print('*' * etoiles)  
>>> affichage_etoile(4,5)  
>>> affichage_etoile(1,10)
```

3° Evaluate the following code

```
>>> def affichage_etoile (lignes, etoiles) :  
    return [print('*' * etoiles) for i in range (lignes) ]  
>>> affichage_etoile(2,3)
```

Solutions

1°

```
>>> print('*' * 5)  
# print('*****')  
*****
```

2°

```
>>> affichage_etoile(4,5)
```

```
*****  
*****  
*****  
*****
```

```
>>> affichage_etoile(1,10)
```

```
*****
```

3°

```
>>> affichage_etoile(2,3)
```

```
***  
***
```

```
[None, None]
```

>>> 29.6.2.1.Compréhension de liste avec un if

>>> Exercice 310

1° Evaluate the following code

```
>>> for i in range(10) :  
    if (i % 2 == 0) :  
        print(i)
```

2° Evaluate the following code

```
>>> a = [print(i) for i in range(10) if i%2 == 0]
```

Solutions

1°

```
>>> for i in range(10) :
```

```
    if (i % 2 == 0) :
```

```
        print(i, end =")")
```

```
0 2 4 6 8
```

2°

```
>>> a = [print(i) for i in range(10) if i % 2 == 0]
```

```
0  
2  
4  
6  
8
```

Tips

```
[ expression for_loop if_test ]
```

>>> Exercise 311

1° Evaluate the following code

```
>>> ch = 'anticonstitutionnellement'  
>>> for i in ch :  
    if (i in ['a','e','i','o','u']) :  
        print(i)
```

2° Evaluate the following code

```
>>> ch = 'anticonstitutionnellement'  
>>> a = [print(i, end = ' ') for i in ch if (i in ['a','e','i','o','u'])]
```

Solutions

1°

```
>>> ch = 'anticonstitutionnellement'  
>>> for i in ch :  
    if (i in ['a','e','i','o','u']) :  
        print(i, end = ' ')\na i o i u i o e e e
```

2°

```
>>> ch = 'anticonstitutionnellement'  
>>> a = [print(i, end = ' ') for i in ch if (i in ['a','e','i','o','u'])]\na i o i u i o e e e
```

>>> Exercise 312

```
*  
**  
***  
****  
*****
```

1° Evaluate the following code

```
>>> nombre_ligne = int (input( 'nombre de lignes ?'))  
'nombre de lignes ?' 5  
>>> i = 0  
>>> while (i < nombre_ligne):  
    netoiles = 5  
    j=0  
    while (j < netoiles):  
        print('*', end="")  
        j= j + 1
```

```
print()  
i= i +1
```

2° Evaluate the following code

```
>>> i=0  
>>> while(i<5) :  
    print('*'*i)  
    i=i+1
```

3° Evaluate the following code

```
>>> for i in range(5) :  
    print('*'*i)
```

4° Evaluate the following code

```
>>> a = [print('*'*i) for i in range(5)]
```

5° Evaluate the following code

```
>>> def etoiles (n) :  
    return [print('*'*i) for i in range(n) ]  
>>> etoiles(5)
```

Solutions

1°

```
>>> nombre_ligne = int (input( 'nombre de lignes ?'))  
'nombre de lignes ?' 5  
>>> i =0  
>>> while (i< nombre_ligne):  
    netoiles = 5  
    j=0  
    while (j< netoiles):  
        print('*', end="")  
        j= j + 1
```

```
print()  
i= i +1
```

```
*****  
*****  
*****  
*****  
*****
```

2°

```
>>> i=0  
>>> while(i<5) :  
    print('*'*i)  
    i=i+1
```

```
*
```



```
**
```



```
***
```



```
****
```

3°

```
>>> for i in range(5) :  
    print('*'*i)
```

```
*
```



```
**
```



```
***
```



```
****
```

4°

```
>>> a = [print('*'*i) for i in range(5)]  
*
```

```
**  
***  
****
```

5°

```
>>> def etoiles (n) :  
    return [print('*'*i) for i in range(n) ]  
>>> etoiles(5)  
# return [print('*'*i) for i in range(5) ]
```

```
*  
**  
***  
****
```

[None, None, None, None, None]

>>> **Exercise 313**

1° Evaluate the following code

```
>>> i = 0  
>>> while(i<101) :  
    if (i % 11 == 0) :  
        print(i, end = ' ')  
    i=i+1
```

2° Evaluate the following code

```
>>> for i in range(1,10) :  
    print(i, end = ' ')
```

3° Evaluate the following code

```
>>> a = [print(i, end = ' ') for i in range(1,101) if (i % 11 == 0 )]
```

Solutions

1°

```
>>> while(i<101):  
    if(i%11==0):  
        print(i, end = ' ')  
    i=i+1
```

```
0 11 22 33 44 55 66 77 88 99
```

2°

```
>>> for i in range(1,10) :  
    print(i, end = ' ')
```

```
1 2 3 4 5 6 7 8 9
```

3°

```
>>> a = [print(i, end = ' ') for i in range(1,101) if (i % 11 == 0 )]
```

```
0 11 22 33 44 55 66 77 88 99
```

>>> 29.6.2.1.List compréhension + for

>>> **Exercise 314**

1° Evaluate the following code

```
>>> a = [[1,2,3],[4,5,6],[7,8,9]]  
>>> for i in a :  
    i  
>>> b = [i for i in a ]  
>>> def element_liste (l) :  
    return [i for i in l]
```

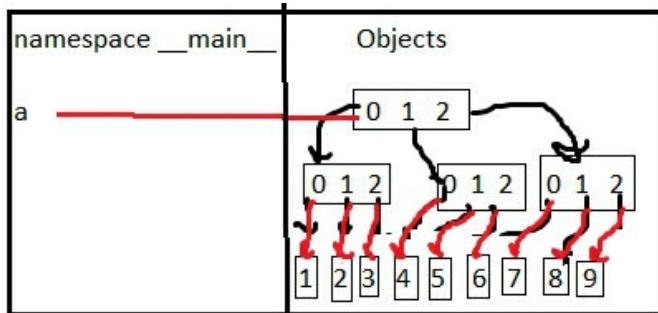
2° Evaluate the following code

```
>>> for i in a :  
    for j in i :  
        j  
>>> c = [j for i in a for j in i]  
>>> def sous_liste(l) :
```

```
return [j for in a for j in i]
```

Solutions

1°



```
>>> for i in a:
```

```
    i
```

```
[1, 2, 3]
```

```
[4, 5, 6]
```

```
[7, 8, 9]
```

iteration	i
1	[1,2,3]
2	[4,5,6]
3	[7,8,9]

```
>>> b = [i for i in a ]
```

```
>>> b
```

```
[[1,2,3], [4,5,6], [7,8,9]]
```

```
>>> def element_liste (l) :
```

```
    return [i for in l]
```

```
>>> element_liste(a)
```

```
[[1,2,3], [4,5,6], [7,8,9]]
```

```
>>> element_liste(b)
```

```
[[1,2,3], [4,5,6], [7,8,9]]
```

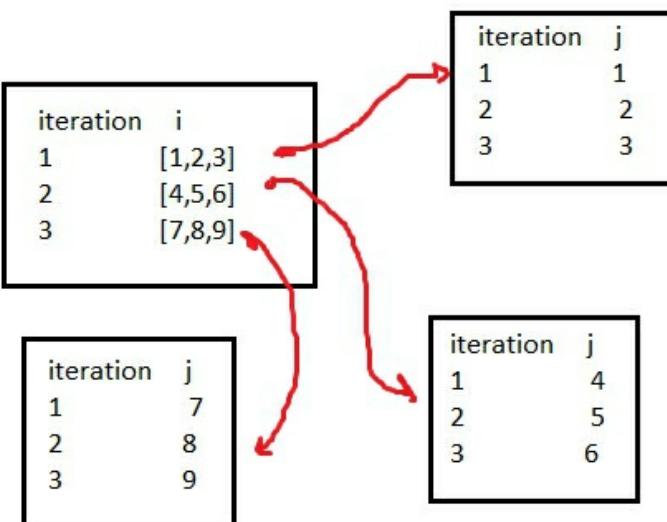
2°

```
>>> for i in a :
```

```
    for j in i :
```

```
        j
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9
```



```
>>> c = [j for i in a for j in i]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> def sous_liste(l) :
```

```
    return [j for i in a for j in i]
```

```
>>> sous_liste(a)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Tips:

```
[ expression loop_for_i loop_for_j]
```

Equals to

```
>>> for i in objet :  
    for j in i :  
        expression
```

i must be an iterable object

>>> Exercise 315

1° Evaluate the following code

```
>>> for a in ['Hello', 'salut']:  
    for b in ['world', 'bonjour']:  
        a+b  
>>> [a+b for a in ['Hello', 'salut'] for b in ['world','bonjour']]
```

2° Evaluate the following code

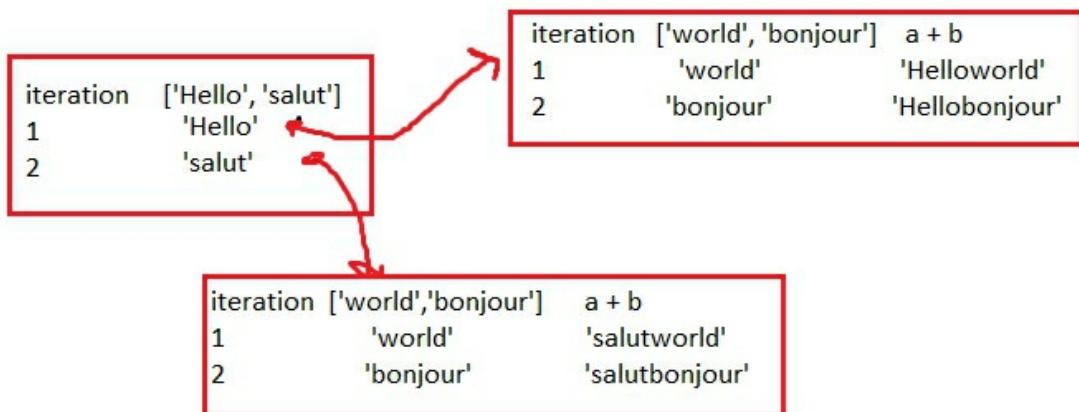
```
>>> for a in [[1,2,3],[4,5,6]] :  
    for b in [1,2,3] :  
        a*b  
>>> a = [a * b for a in [[1,2,3],[4,5,6]] for b in [1,2,3] ]
```

Solutions

1°

```
>>> for a in ['Hello', 'salut']:  
    for b in ['world', 'bonjour']:  
        a+b  
'Helloworld'  
'Hellobonjour'  
'salutworld'
```

```
'salutbonjour'
```



```
>>> [a+b for a in ['Hello', 'salut'] for b in ['world','bonjour']]
```

```
['Helloworld', 'Hellobonjour', 'salutworld', 'salutbonjour']
```

2°

```
>>> for a in [[1,2,3],[4,5,6]] :
```

```
    for b in [1,2,3] :
```

```
        a*b
```

```
[1, 2, 3]
```

```
[1, 2, 3, 1, 2, 3]
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
[4, 5, 6]
```

```
[4, 5, 6, 4, 5, 6]
```

```
[4, 5, 6, 4, 5, 6, 4, 5, 6]
```

```
>>> a = [a * b for a in [[1,2,3],[4,5,6]] for b in [1,2,3] ]
```

```
>>> a
```

```
[[1, 2, 3], [1, 2, 3, 1, 2, 3], [1, 2, 3, 1, 2, 3, 1, 2, 3], [4, 5, 6], [4, 5, 6, 4, 5, 6], [4, 5, 6, 4, 5, 6, 4, 5, 6]]
```

>>> **Exercise 316**

Isupper() doc

```
Return true if all cased characters in the string are uppercase and there is at
```

least one cased character, false otherwise.

1° Evaluate the following code

```
>>> phrase = 'Une Integrale de Comics'  
>>> 'U'.isupper()  
>>> 'n'.isupper()  
>>> 'hello'.isupper()  
>>> 'HELLO'.isupper()  
>>> phrase.isupper()
```

2°

```
>>> a = [ i for i in phrase if i.isupper() ]
```

- a) Give the expression ?
- b) is phrase an iterable object ?
- c) give the condition ?
- d) evaluate the code ?

Solutions

1°

```
>>> phrase = 'Une Integrale de Comics'  
>>> 'U'.isupper()  
True  
>>> 'n'.isupper()  
False  
>>> 'hello'.isupper()  
False  
>>> 'HELLO'.isupper()  
True  
>>> phrase.isupper()  
# 'Une Integrale de Comics'.isupper()
```

```
# False
```

```
False
```

2°

a) expression : i

```
>>> a = [ i for i in phrase if i.isupper() ]
```

b)

```
>>> type(phrase)
```

```
<class str>
```

Phrase is an iterable object

c) condition

isupper()

d)

```
>>> a = [ i for i in phrase if i.isupper() ]
```

```
>>> a
```

```
[U, T, 'C']
```

>>> **Exercise 317 (doc Python)**

1° Evaluate the following code

```
>>> liste = [ ]
```

```
>>> liste.append((1,2))
```

```
>>> liste.append((3,4))
```

2° Evaluate the following code

```
>>> liste = [ ]
```

```
>>> for a in [1,2,3] :
```

```
    for b in [4,5,6] :
```

```
        liste.append((a,b))
```

3° Evaluate the following code

```
>>> d = [(a,b) for a in [1,2,3] for b in [4,5,6] ]
```

4° Evaluate the following code

```
>>> a = [ (x,y) for x in [1,2,3] for y in [4,5,6] if x!=y]
```

Solutions

1°

```
>>> liste = [ ]  
>>> liste.append((1,2))  
[(1, 2)]  
>>> liste.append((3,4))  
[(1, 2), (3, 4)]
```

2°

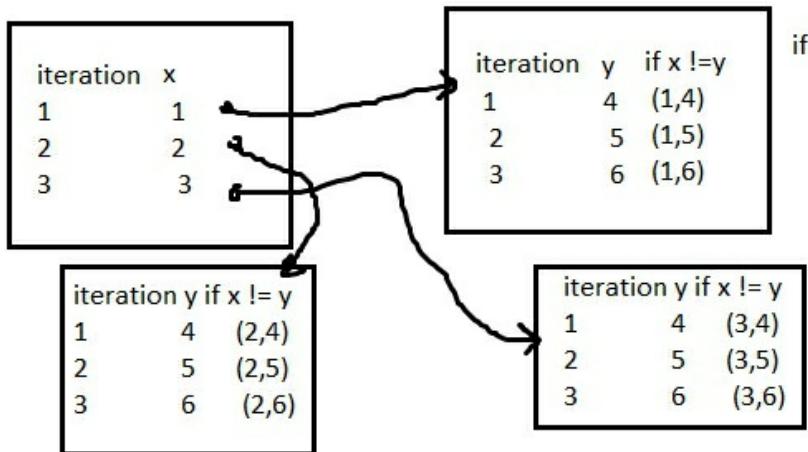
```
>>> liste = [ ]  
>>> for a in [1,2,3] :  
    for b in [4,5,6] :  
        liste.append((a,b))  
>>> liste  
[(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)]
```

3°

```
>>> d = [(a,b) for a in [1,2,3] for b in [4,5,6] ]  
>>> d  
[(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)]
```

4°

```
>>> a = [ (x,y) for x in [1,2,3] for y in [4,5,6] if x!=y]  
>>> a  
[(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)]
```



>>> 29.6.3.break

« heartbreak » chanson

« have a break, have a chocolate »

>>> **Exercise 317**

1° Evaluate the following code

```
>>> bool1 = True
>>> while(bool1) :
    nombre_entre = 41
    if nombre_entre == 42 :
        bool1 = False
```

2° Evaluate the following code

```
>>> bool1 = True
>>> while(bool1) :
    nombre_entre = 42
    if nombre_entre == 42 :
        bool1 = False
```

3° Evaluate the following code

```
>>> bool1 = True
```

```
>>> while (bool1) :  
    nombre_entre = 41  
    if nombre_entre == 42 :  
        bool1= False  
    break
```

Solutions

1° infinite loop= bad idea

```
>>> bool1 = True  
>>> while(bool1) :  
    # while(True) :  
    # nombre_entre = 41  
    # if nombre_entre == 42 :  
    # if 41 == 42 :  
    # if False :  
    # while(True) :  
    # while (True) :  
    # ...
```

2°

```
>>> bool1 = True  
>>> while(bool1) :  
    # while(True) :  
    # nombre_entre = 42  
    # if nombre_entre == 42 :  
    # if 42 == 42 :  
    # if True :  
    # bool1 = False
```

```
# while(False) :  
# end of the loop
```

3°

```
>>> bool1 = True  
>>> while (bool1) :  
    nombre_entre = 41  
    if nombre_entre == 42 :  
        bool1= False  
        break  
    # if nombre_entre == 42  
    # if False  
    # break  
# end of the loop^  
>>>
```

Tips

```
>>> while (True)  
    block  
    break
```

>>> Exercise 318

Evaluate the following code

```
>>> print(' menu :')  
>>> print('1.Pizza')  
>>> print('2.Hamburger')  
>>> while (True) :  
    choix = int(input('choice ?'))  
    if (choix == 1 or choix == 2) :  
        print('processing')
```

```
    break
else :
    print('please press 1 or 2')
```

Solutions

```
menu :
1.Pizza
2.Hamburger
choice ?0
please press 1 or 2
choice ?1
#     print('processing')
#     break
processing
>>>
```

>>> 29.6.4.for and list indexes

>>> **Exercise 319**

1° Evaluate the following code

```
>>> a = [1, 2, 3]
>>> for i in a :
    print(a[i])
```

2° Evaluate the following code

```
>>> range(len(a))
```

3° Evaluate the following code

```
>>> for i in range(len(a)) :
    print(a[i])
```

Solutions

1°

```
>>> a = [1, 2, 3]
>>> for i in a :
    print(a[i])
2
3
IndexError: list index out of range
```

2°

```
>>> range(len(a))
# range(3)
range(0,3)
```

3°

```
>>> for i in range(len(a)) :
    print(a[i])
1
2
3
```

iteration	i	print(a[i])
1	0	1
2	1	2
3	2	3

>>> **Exercise 320**

1° Evaluate the following code

```
>>> l = [[0,1], 1, 2]
>>> for i in l :
    for j in i :
        j
```

2° Evaluate the following code

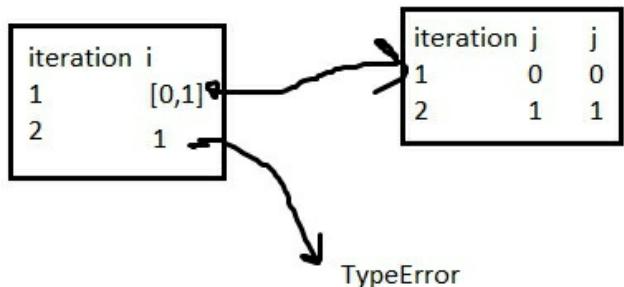
```
>>> l = [[0,1], 1, 2]
```

```
>>> for i in l :
    for j in (range(len(i)) :
        j
```

Solutions

```
>>> for i in l:
    for j in i:
        j

0
1
Traceback (most recent call last):
  File "<pyshell#60>", line 2, in <module>
    for j in i:
TypeError: 'int' object is not iterable
```



2°

```
>>> for i in l:
    for j in range(len(i)):
        j

0
1
Traceback (most recent call last):
  File "<pyshell#68>", line 2, in <module>
    for j in range(len(i)):
TypeError: object of type 'int' has no len()
```

>>>30.Dictionaries and Sets

>>> 30.1.Dictionaires

« un dictionnaire est une table de hash » un dealer

>>> 30.1.1.Couple (key,value)

>>> Exercise 321

Evaluate the following code

```
>>> dico = { }

>>> type(dico)

>>> bande_dessinee = { 'iron man' : 'marvel', 'tintin' : 'franco belge' : 'one
piece' : 'manga' }

>>> type(bande_dessinee)

>>> bande_dessinee['iron man']

>>> bande_dessinee['tintin']

>>> bande_dessinee['one piece']
```

Solutions

```
>>> dico = { }

>>> type(dico)

>>> bande_dessinee = { 'iron man' : 'marvel', 'tintin' : 'franco belge' : 'one
piece' : 'manga' }

>>> bande_dessinee['iron man']

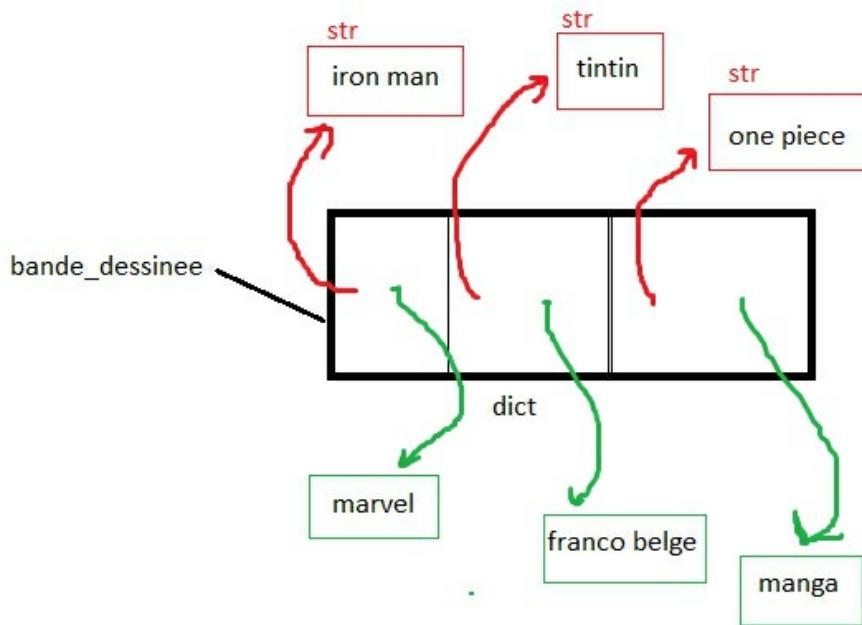
'marvel'

>>> bande_dessinee['tintin']

'franco belge'

>>> bande_dessinee['one piece']

'manga'
```



Tips

```
>>> dico = { key1 : value1 , key2 : value2 , key3 : valeur3 }
```

Key must be an immutable object such as str, numerical or tuple

```
>>> dico[key]
```

```
value
```

>>> Exercise 322

Evaluate the following code

```
>>> dico = { 1 : 'hello' }

>>> dico2 = { '1' : 'hello' }

>>> dico3 = { (1,) : 'hello' }

>>> dico4 = { [1] : 'hello' }
```

Solutions

```
>>> dico[1]
```

```
'hello'
```

```
>>> dico2['1']
```

```
'hello'
```

```
>>> dico3[(1,)]  
'hello'  
>>> dico4 = { [1] : 'hello' }  
TypeError: unhashable type: 'list'
```

>>> Exercise 323

1° Create a dictionary with the following (key,value)

```
cle1 : 1200, cle2 : 2430 et cle3 : 5463
```

2° Evaluate the following code

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' ,  
'numero' : 123456 }  
>>> agenda[nom]  
>>> agenda[numero]
```

Solutions

1°

```
>>> dico = { 'cle1' : 1200 , 'cle2' : 2430 , 'cle3' : 5463 }
```

2°

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' ,  
'numero' : 123456 }  
>>> agenda[nom]  
'derp'  
>>> agenda[numero]  
123456
```

>>> 30.1.2. Skills on dictionaries

>>> Exercise 324

Evaluate the following code

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' ,  
'numero' : 123456 }  
>>> agenda['prenom'] = 'Python3'
```

```
>>> agenda['numero'] = 11110001001000000
>>> agenda['travail'] = 'controleur'
>>> agenda
```

Solutions

```
>>> agenda['prenom'] = 'Python3'
# agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',
'numero' : 123456 }

# agenda = { 'nom' : 'derp' , 'prenom' : 'Python3', 'adresse' : '4 rue du Port',
'numero' : 123456 }

>>> agenda['numero'] = 11110001001000000
# agenda = { 'nom' : 'derp' , 'prenom' : 'Python3', 'adresse' : '4 rue du Port',
'numero' : 11110001001000000 }

>>> agenda['travail'] = 'controleur'
# agenda = { 'nom' : 'derp' , 'prenom' : 'Python3', 'adresse' : '4 rue du Port',
'numero' : 11110001001000000, 'travail' = 'controleur' }

>>> agenda
{ 'nom' : 'derp' , 'prenom' : 'Python3', 'adresse' : '4 rue du Port', 'numero' :
11110001001000000, 'travail' = 'controleur' }
```

Tips

- A dictionary is an iterable object.
- a dictionary is a mutable object
- we can add a (key,value) like

```
>>> dico [key] = value
```

>>> 30.1.2.1. Built-in function del ()

>>> **Exercie 324**

1° Evaluate the following code

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',
'numero' : 123456 }

>>> del agenda['nom']
```

```
>>> del agenda['adresse']  
>>> agenda
```

2° Evaluate the following code

```
>>> liste = [1,2,3,4,5]  
>>> del(liste[3])  
>>> del(liste[2])  
>>> liste
```

Solutions

1°

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' ,  
'numero' : 123456 }  
>>> del agenda['nom']  
# agenda = { 'nom' : 'derp' , 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' ,  
'numero' : 123456 }  
# agenda = { 'prenom' : 'rederp' , 'adresse' : '4 rue du Port' , 'numero' : 123456  
}  
>>> del agenda['adresse']  
# agenda = { 'prenom' : 'rederp' , 'numero' : 123456 }  
# agenda = { 'prenom' : 'rederp' , 'numero' : 123456 }  
>>> agenda  
{ 'prenom' : 'rederp' , 'numero' : 123456 }
```

2°

```
>>> liste = [1,2,3,4,5]  
>>> del(liste[3])  
#del([1,2,3,4,5])  
# [1,2,3,5]  
>>> del(liste[2])
```

```
#del([1,2,3,5])
```

```
#[1,2,5]
```

```
>>> liste
```

```
[1,2,5]
```

Tips

Function `del()` can delete

- list elements `del(liste[i])`
- dictionary items `del dico[key]`

```
>>> 30.1.2.2.Methods keys(), value()and items()
```

```
>>> Exercise 325
```

Evaluate the following code

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',  
'numero' : 123456 }  
  
>>> agenda.keys()  
  
>>> agenda.values()  
  
>>> agenda.items()
```

Solutions

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',  
'numero' : 123456 }
```

```
>>> agenda.keys()
```

```
dict_keys(['nom', 'prenom', 'adresse', 'numero'])
```

```
>>> agenda.values()
```

```
dict_values(['derp', 'rederp', '4 rue du Port', 123456])
```

```
>>> agenda.items()
```

```
dict_items([('nom', 'derp'), ('prenom', 'rederp'), ('adresse', '4 rue du Port'),  
( 'numero', 123456)])
```

```
>>> 30.1.2.3.Loop and dictionaries
```

```
>>> Exercise 326
```

Evaluate the following code

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',  
'numero' : 123456 }  
>>> for i in agenda :  
    print(i)
```

Solutions

```
>>> agenda = { 'nom' : 'derp' , 'prenom' : 'rederp', 'adresse' : '4 rue du Port',  
'numero' : 123456 }  
>>> for i in agenda :  
    print(i)
```

nom
prenom
adresse
numero

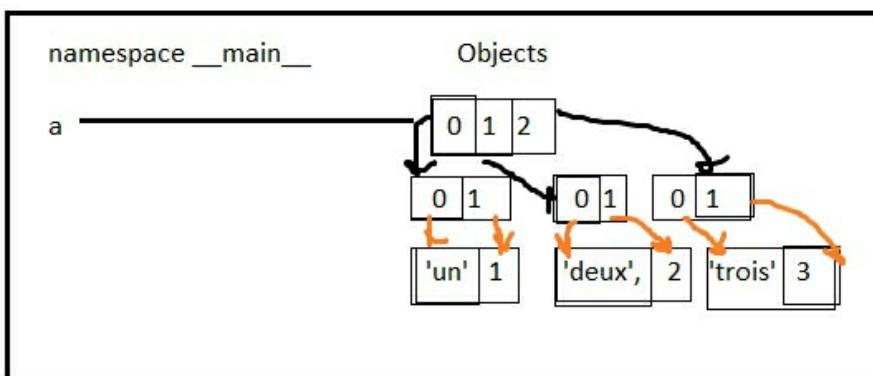
>>> 30.1.2.1.dict() iterator

>>> **Exercie 327**

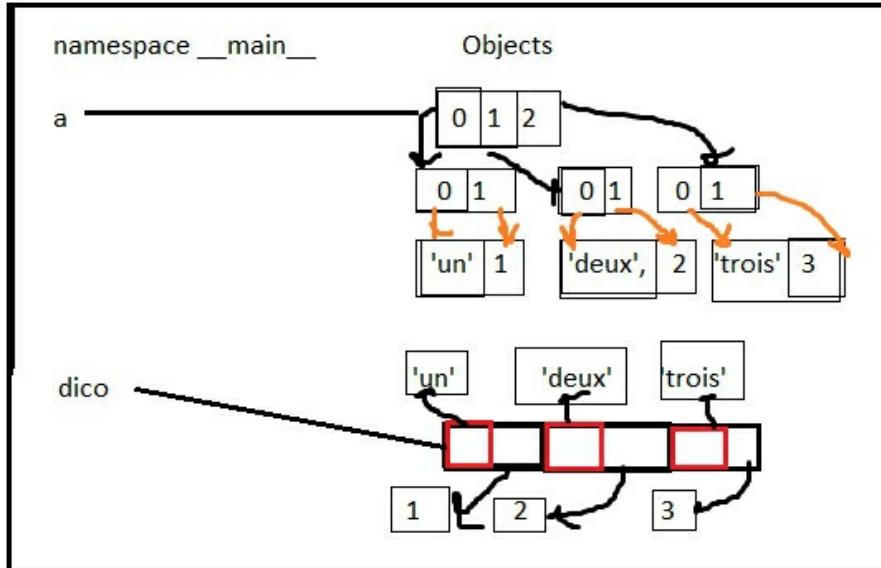
Evaluate the following code

```
>>> a = [ ('un ', 1), ('deux ', 2), ('trois ', 3) ]  
>>> dico = dict(a)  
>>> dico
```

Solutions



```
>>> dico = dict([('un', 1), ('deux', 2), ('trois', 3)])
# dico = {'un': 1, 'deux': 2, 'trois': 3}
>>> dico
{'un': 1, 'deux': 2, 'trois': 3}
```



Tips :

`dict(iterable)` -> new dictionary initialized

>>> 30.2.Sets

« ensemble on peut le faire » un politicien

>>> 30.2.1.Sets 101

>>> **Exercise 328**

Evaluate the following code

```
>>> ensemble_vide = { }
>>> type(ensemble)
>>> ensemble = {'A', 'B', 'C'}
>>> len(ensemble)
>>> 'A' in ensemble
```

Solutions

```
>>> ensemble_vide = {}
```

```
>>> ensemble_vide
```

```
{}
```

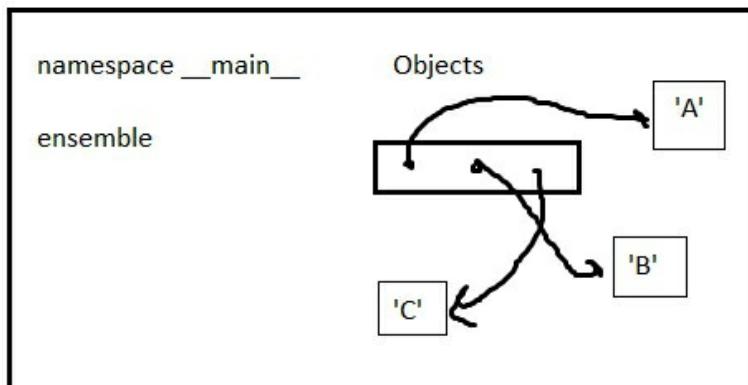
```
>>> len(ensemble)
```

```
3
```

```
>>> 'A' in ensemble
```

```
True
```

No indexes in a set



>>> **Exercise 329**

```
>>> taxes = { 'IR' , 'tva', 'IS'}
```

Evaluate the following code

```
>>> print(taxes)
```

```
>>> 'IR' in taxes
```

```
>>> 'OTD' in taxes
```

Solutions

Taxes variable references a set object

```
>>> print(taxes)
```

```
{ 'IR' , 'tva', 'IS'}
```

```
>>> 'IR' in taxes
```

```
True
```

```
>>> 'OTD' in taxes
```

```
False
```

Tips

- set object is an iterable object

>>> **Exercise 330**

Evaluate the following code

```
>>> ensemble = {1, 2, 3, 4, 'a', 'b', 'c'}  
>>> for i in ensemble :  
    i  
>>> ensemble2 = {'a', 1, 2, 'b', 'c','d', 3, 4}  
>>> ensemble == ensemble2
```

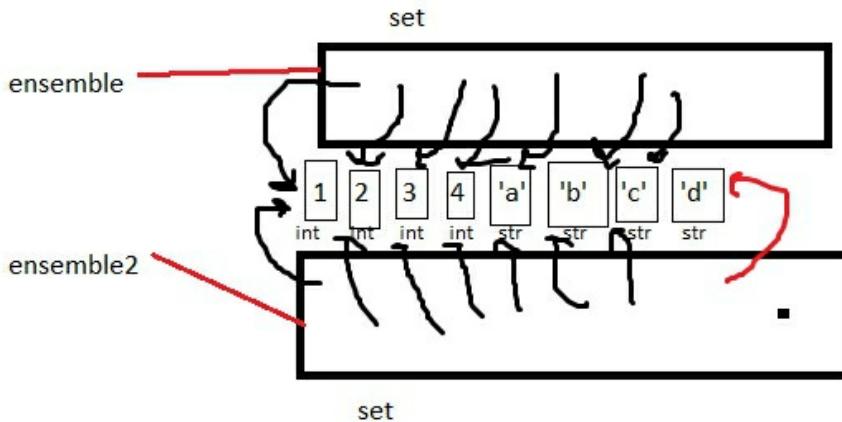
Solutions

```
>>> for i in ensemble :  
    i
```

```
'a'  
1  
2  
3  
4  
'c'  
'b'
```

```
>>> ensemble == ensemble2
```

```
False
```



Tips :

- {} is a an empty dictionary not an empty set
- like a mathematical set there's no order within

>>> 30.2.2.built-in fuction set()

>>> Exercise 331

Evaluate the following code

```
>>> liste =['A','B','C']
>>> a = set(liste)
>>> a
>>> chaine = 'aaaabbbbbbbcccccccddddd'
>>> b = set(chaine)
>>> b
>>> len(chaine) == len(b)
```

Solutions

set (iterable) → new set object

```
>>> a = set(liste)
# a = set(['A','B','C'])
# a = {'A','B','C'}
>>> a
```

```
{'C', 'A', 'B'}  
>>> b = set(chaine)  
# b = set('aaaabbbbbcccccddddd')  
# b = {'a','b','c','d'}  
>>> b  
{'a','b','c','d'}  
>>> len(chaine) == len(b)  
False
```

>>> **Exercise 332**

Evaluate the following code

```
>>> a = {1,2,3,4,5,'a','b','c'}  
>>> for i in a :  
    print(i, end = ' ')
```

Solutions

```
>>> for i in a :  
    print(i, end = ' ')
```

1 2 3 4 5 c b a

>>> 30.2.3.Methods add() and update()

>>> **Exercise 333**

Evaluate the following code

```
>>> ensemble = {'a','b','c','d'}  
>>> ensemble.add('e')  
>>> ensemble  
>>> ensemble.update([1,2,3,4,5])  
>>> ensemble
```

Solutions

```

>>> ensemble = {'a','b','c','d'}
>>> ensemble.add('e')
# {'a','b','c','d'}.add('e')
# {'e', 'c', 'd', 'a', 'b'}
>>> ensemble
{'e', 'c', 'd', 'a', 'b'}
>>> ensemble.update([1,2,3,4,5])
#{'e', 'c', 'd', 'a', 'b'}.update([1,2,3,4,5])
# {1, 'c', 2, 3, 'e', 4, 5, 'b', 'a', 'd'}
>>> ensemble
{1, 'c', 2, 3, 'e', 4, 5, 'b', 'a', 'd'}

```

>>> 30.2.4. Set Theory 101

>>> 30.2.4.1. Operator -
 >>> **Exercise 334**

Evaluate the following code

```

>>> ensemble1 = {'a','b','c'}
>>> ensemble2 = {'c','d','e'}
>>> ensemble1 - ensemble2
>>> ensemble2 - ensemble1

```

Solutions

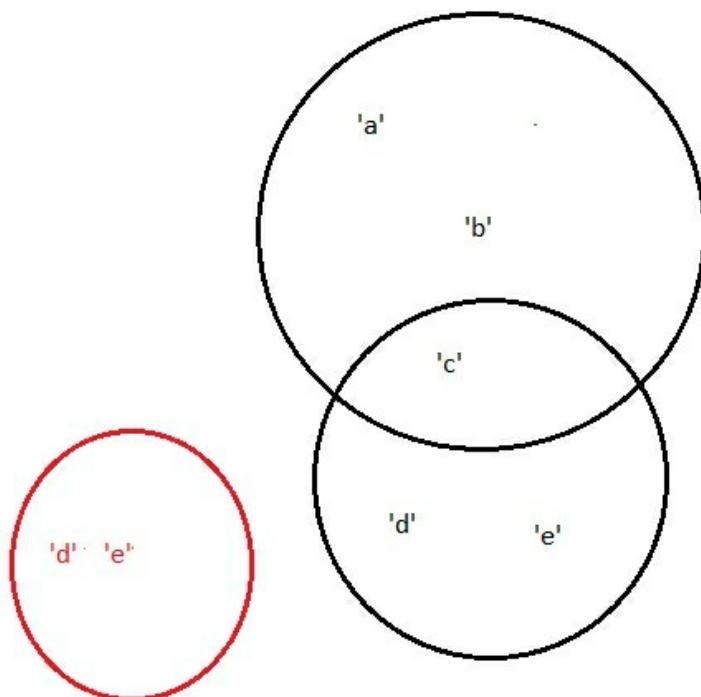
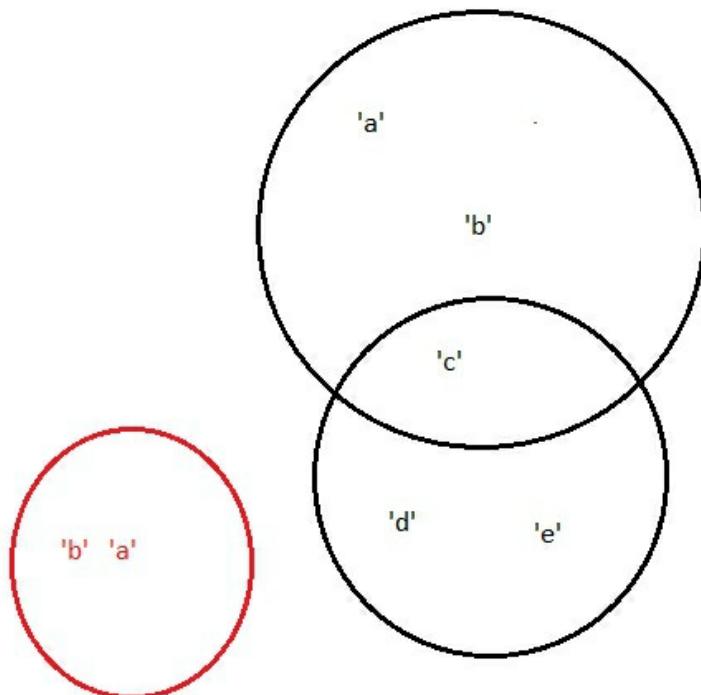
```

>>> ensemble1 = {'a','b','c'}
>>> ensemble2 = {'c','d','e'}
>>> ensemble1 - ensemble2
# {'a','b','c'} - {'c','d','e'}
# {'b', 'a'}
>>> ensemble2 - ensemble1

```

$\# \{ 'c', 'd', 'e' \} - \{ 'a', 'b', 'c' \}$

$\# \{ 'd', 'e' \}$



>>> 30.2.4.2. Set Union : Operator |

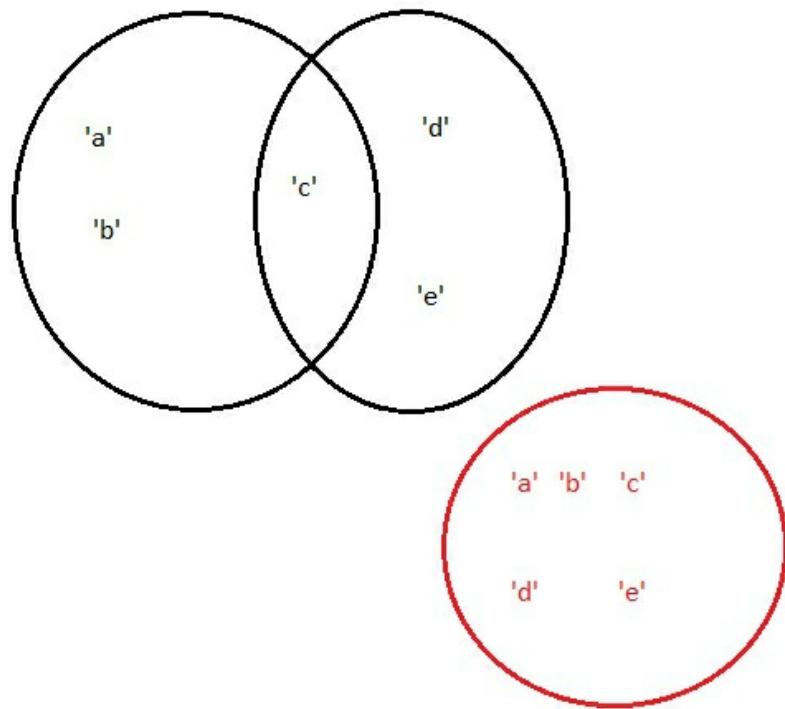
>>> **Exercise 335**

Evaluate the following code

```
>>> ensemble1 = {'a','b','c'}  
>>> ensemble2 = {'c','d','e'}  
>>> ensemble1 | ensemble2
```

Solutions

```
>>> ensemble1 = {'a','b','c'}  
>>> ensemble2 = {'c','d','e'}  
>>> ensemble1 | ensemble2  
# {'a','b','c'} | {'c','d','e'}  
# {'a','b','c', 'c','d','e'}  
# {'a','b', 'c','d','e'}
```



>>> 30.2.4.2. Set Intersection : Operator &

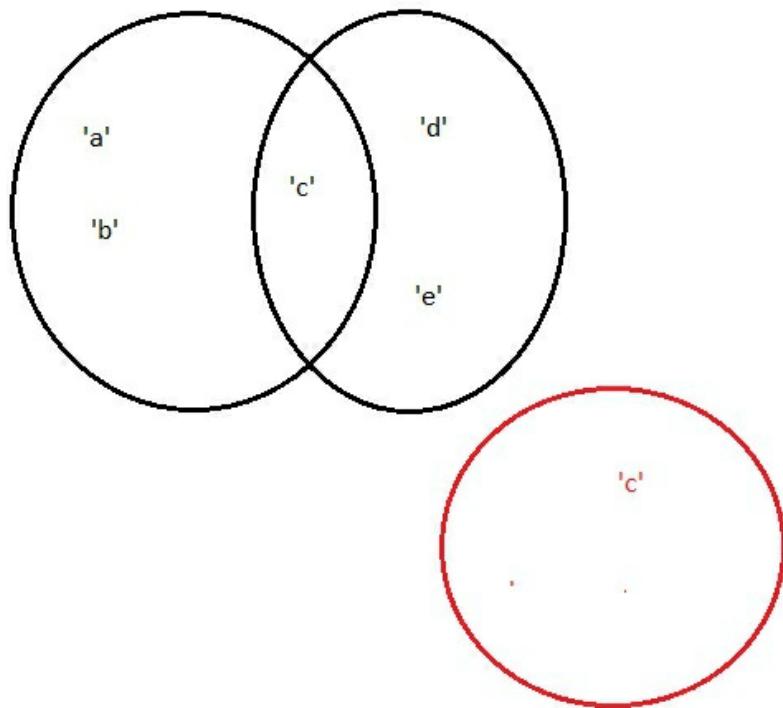
>>> **Exercise 336**

Evaluate the following code

```
>>> ensemble1 = {'a','b','c'}  
>>> ensemble2 = {'c','d','e'}  
>>> ensemble1 & ensemble2
```

Solutions

```
>>> ensemble1 = {'a','b','c'}  
>>> ensemble2 = {'c','d','e'}  
>>> ensemble1 & ensemble2  
# {'a','b','c'} & {'c','d','e'}  
# {'c'}
```



[**>>>31.Exceptions**](#)

>>> 31.1. There's a glitch
'ERROR 404' Windows

>>> 31.1.1.Syntax Errors

>>> **Exercise 337**

Evaluate the following code

```
>>> if (3 > 4) : print('ok')  
>>> 'l'air'
```

Solutions

```
>>> if (3 > 4) : print('ok')  
  
SyntaxError: illegal target for annotation
```

>>> 31.1.2.Semantic Errors

« il y a des choses qui sont impossibles pour moi » Python

>>> **Exercise 338**

Evaluate the following code

```
>>> 1 / 0  
>>> python
```

Solutions

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21  
1) ] on win32  
Type "help", "copyright", "credits" or "license()  
>>> 1/0  
Traceback (most recent call last):  
  File "<pyshell#0>", line 1, in <module>  
    1/0  
ZeroDivisionError: division by zero  
>>> python  
Traceback (most recent call last):  
  File "<pyshell#1>", line 1, in <module>  
    python  
NameError: name 'python' is not defined  
>>>
```

>>> **Exercise 339**

Evaluate the following code

```
>>> l = [0,1]  
>>> l[2]
```

Solutions

```
>>> l[2]
```

IndexError : list index out of range

```
>>> 31.2.try ... except
```

« Il faut toujours avoir un plan B » un espion

>>> Exercise 340

1° Evaluate the following code

```
>>> 1/0
```

```
>>> try :
```

```
    1/0
```

```
except ZeroDivisionError :
```

```
    print("No Zero division")
```

2° Evaluate the following code

```
>>> l = [0,1]
```

```
>>> l[2]
```

```
>>> try :
```

```
    l[2]
```

```
except IndexError :
```

```
    print("Index is missing")
```

Solutions

1°

```
>>> 1/0
```

ZeroDivisionError : Division by zero

```
>>> try :
```

```
    1/0
```

```
except ZeroDivisionError :
```

```
    print("No zero Division")
```

No zero Division

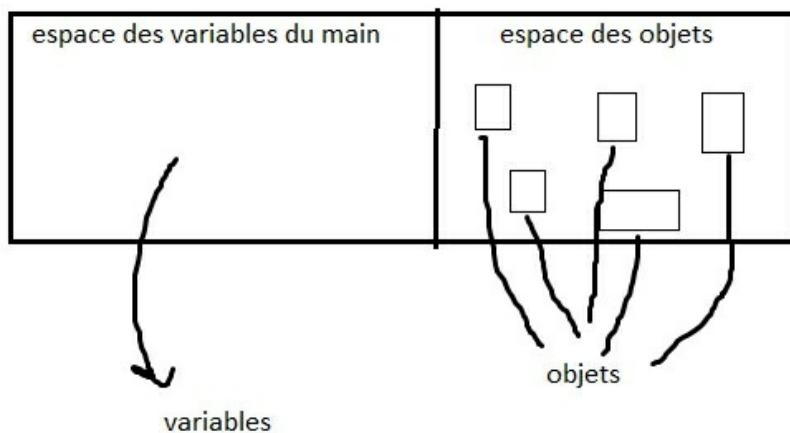
2°

```
>>> l = [0,1]
>>> l[2]
IndexError : list index out of range
>>> try :
    l[2]
except IndexError :
    print("Index is missing")
Index is missing
```

Tips

```
>>> try
    block
except ERROR_NAME :
    block
```

>>> 32.Namespaces and Objects space



>>> 32.1. Namespaces and Objects space for numerical type

>>> Exercise 341

Evaluate

```
>>> spam = 1
```

- 1° a) Give the value, the variable name ?
- b) Is 1 an object ?
- c) what does variable spam refer to ?

2° Do a schema

Solutions

1° a)

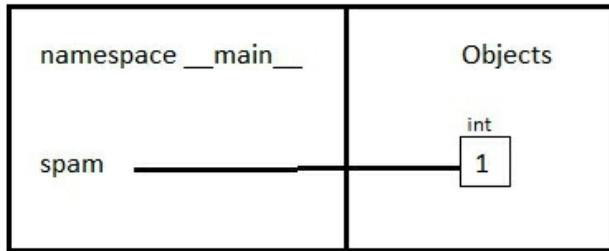
```
>>> spam = 1
```

Value : 1

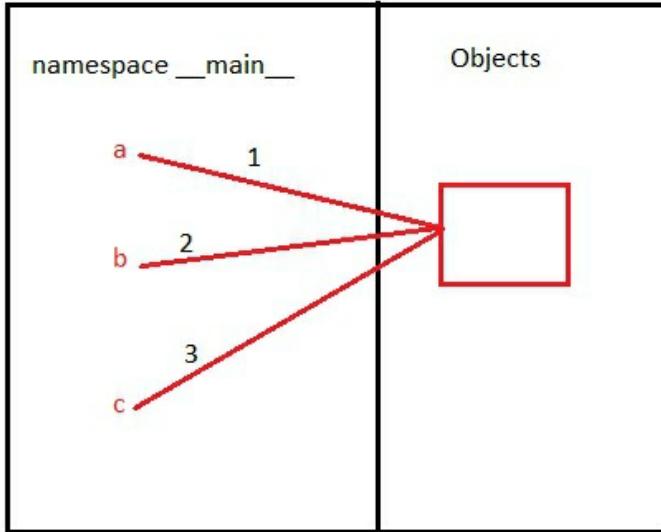
Variable name : spam

- b) 1 is an object.
- c) Variable spam refers to object 1

2°



Tips :



The object is referred by 3 names.

>>> Exercise 342

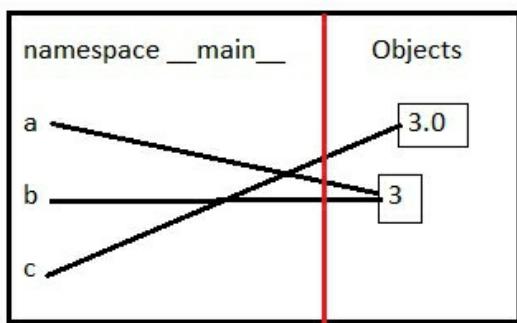
Evaluate the following code and draw the namespace and object space

```
>>> a = 3
>>> b = 3
>>> c = 3.0
>>> a = 2
```

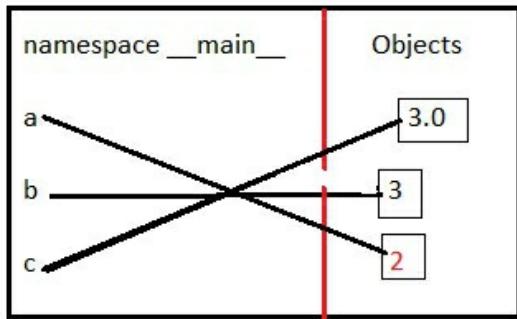
Solutions

Variables a and b refer to the same object : 3.

```
>>> a = 3
>>> b = 3
>>> c = 3.0
```



```
>>> a= 2
```



>>> Exercise 343

Evaluate the code and draw the spaces

```
>>> a = 1
```

```
>>> b= 2
```

```
>>> c= 3.0
```

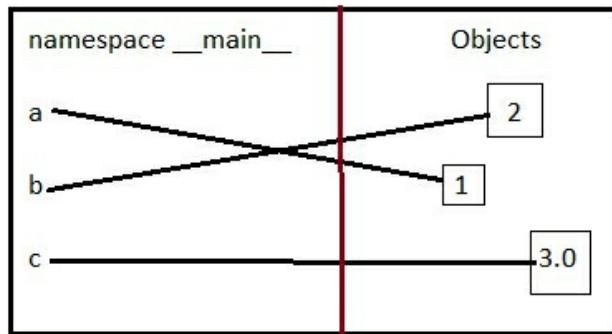
```
>>> b =1
```

Solutions

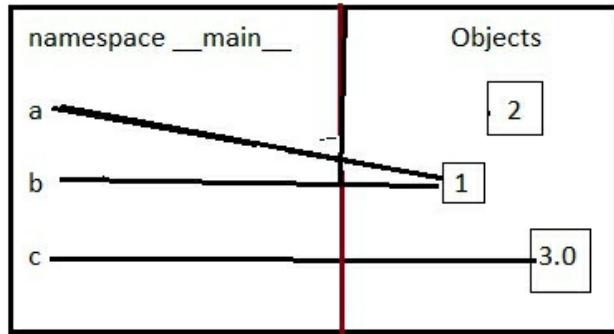
```
>>> a = 1
```

```
>>> b= 2
```

```
>>> c= 3.0
```



```
>>> b = 1
```



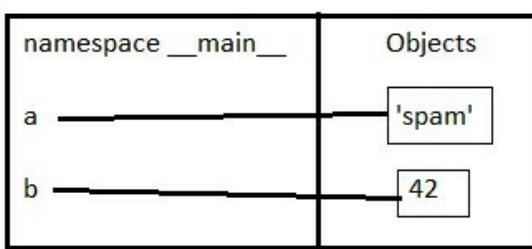
>>> Exercise 344

Evaluate the following code

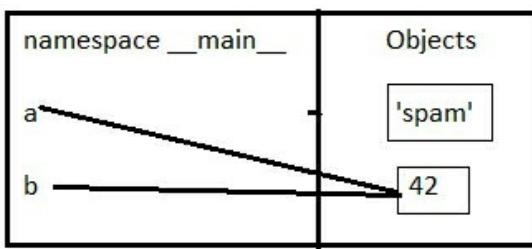
```
>>> 42
>>> a = 'spam'
>>> b = 42
>>> a = b
```

Solutions

```
>>> 42
>>> a = 'spam'
>>> b = 42
```



```
>>> a = b
# a = 42
```



>>> 32.2.List object dans indexes

>>> **Exercise 345**

Evaluate the following code and draw the spaces

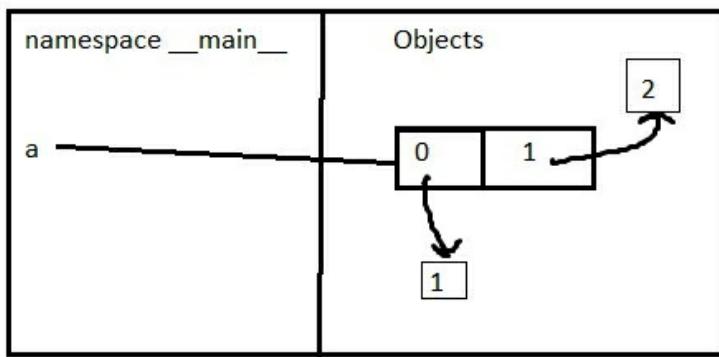
```
>>> a= [1,2]
```

Solutions

Variable refers to object list

Index 0 refers to object 1

Index 1 refers to object 2



Tips

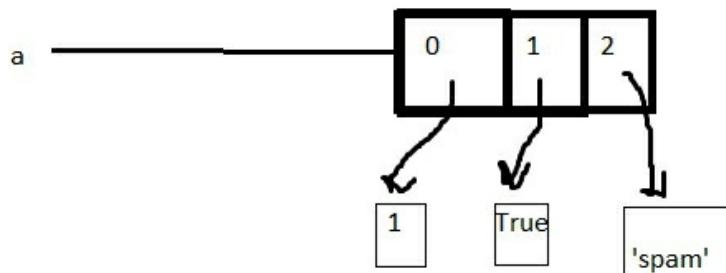
List indexes refer to objects

>>> **Exercise 346**

Evaluate the following code and draw the variable and the object

```
>>> a =[1,True, 'spam']
```

Solutions

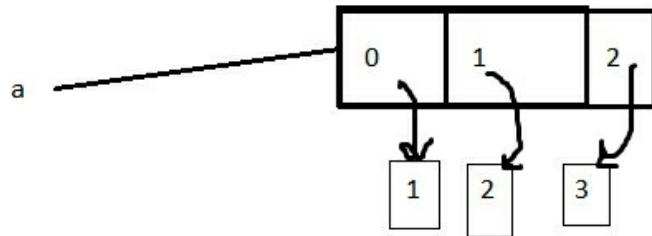


>>> **Exercise 347**

```
>>> a= [1,2,3]
```

Evaluate the following code and draw the variable and the object

Solutions



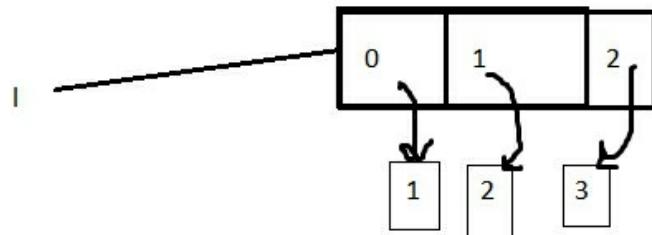
>>> 32.2.1.A list is a mutable objecy

>>> **Exercise 348**

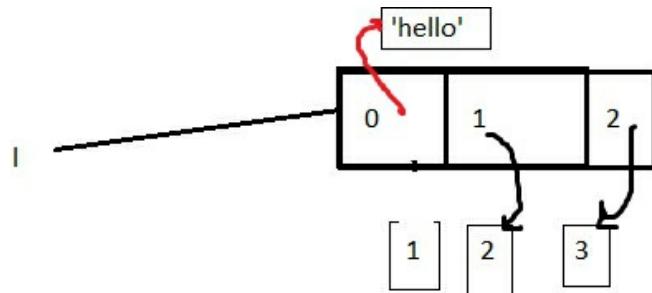
```
>>> l = [1,2,3]
>>> l[0] = 'hello'
```

Evaluate the following code and draw the variable and the object

Solutions



```
>>> l[0]= 'hello'
>>> l
['hello', 2, 3]
```

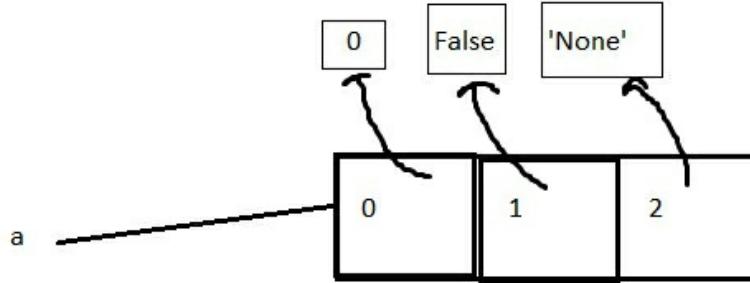


>>> **Exercise 349**

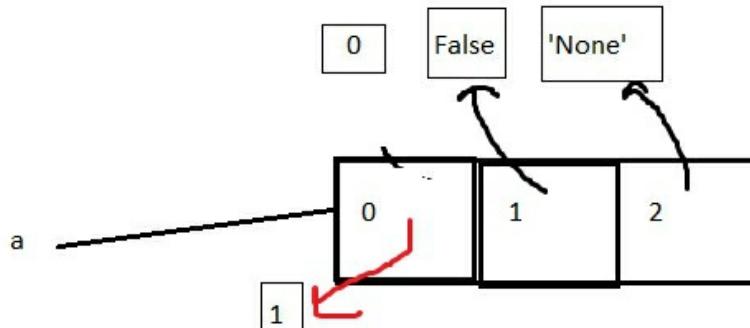
Evaluate the following code and draw the variable and the object

```
>>> a = [0, False, 'None']  
>>> a[0] = 1
```

Solutions



```
>>> a[0]=1
```



>>> 32.2.3.Lists Operations

>>> 32.2.3.1.Operation 1

>>> **Exercice 350**

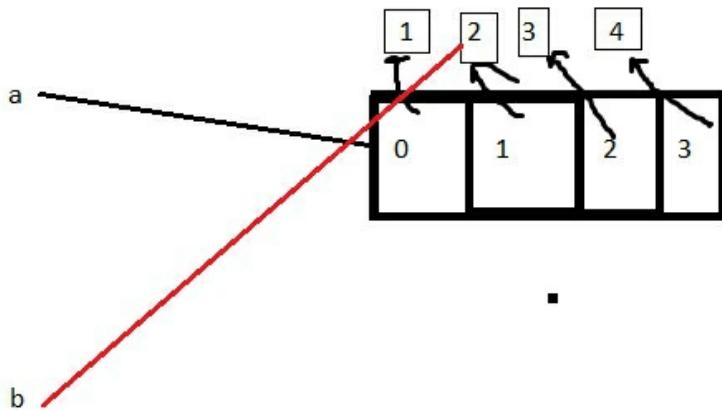
Evaluate the following code and draw the variable and the object

```
>>> a = [1,2,3,4]  
>>> b = 2  
>>> b
```

Solutions

```
>>> a = [1,2,3,4]  
>>> b = 2  
>>> b
```

2



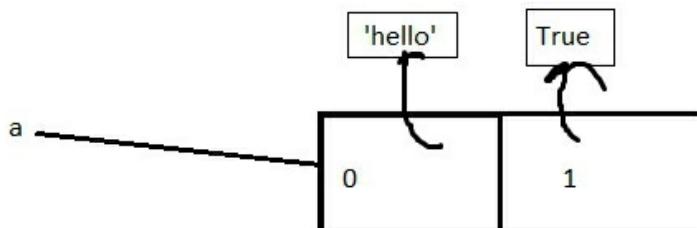
Variable b and index 1 refer to the same object : 2.

>>> Exercise 351

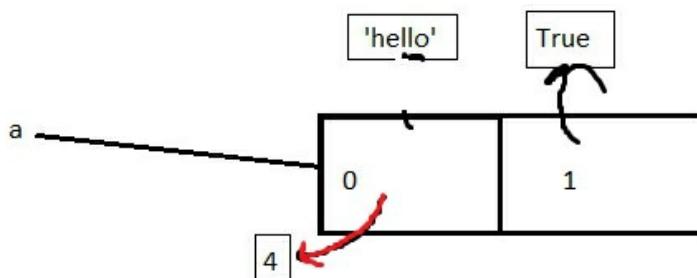
Evaluate the following code and draw the variable and the object

```
>>> a = ['hello', True]  
>>> b = 'hello'  
>>> a[0] = 4
```

Solutions



```
>>> a[0] = 4
```



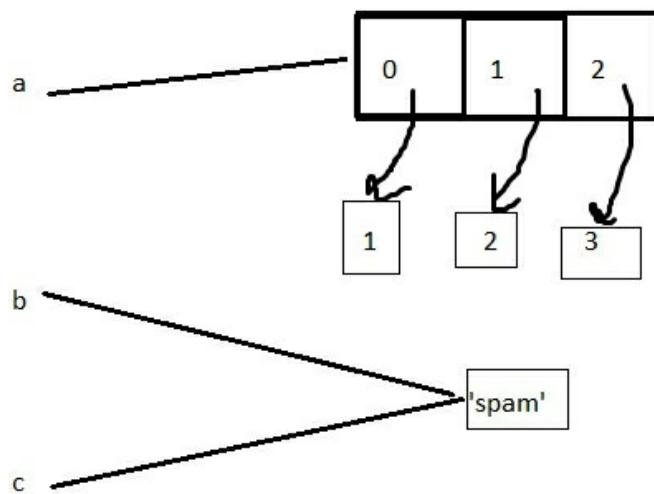
>>> Exercise 352

Evaluate the following code and draw the variable and the object

```
>>> a = [1,2,3]
>>> b = 'spam'
>>> c='sp'+ 'am'
>>> a[1]= 'spam'
```

Solutions

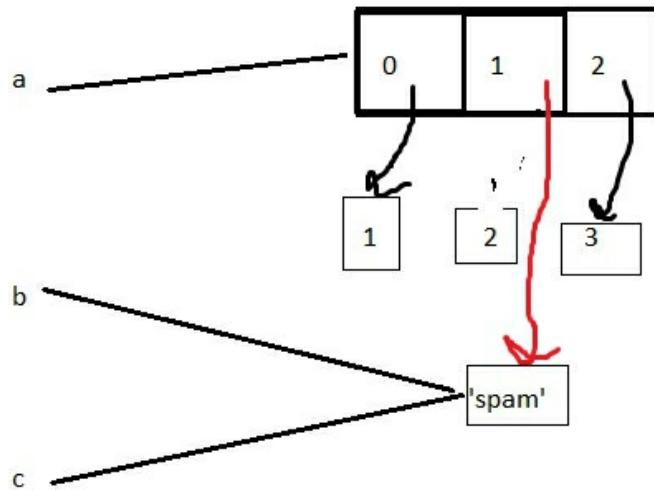
```
>>> a = [1,2,3]
>>> b = 'spam'
>>> c='sp'+ 'am'
# c = 'spam'
```



```
>>> a[1]= 'spam'
```

```
>>> a
```

```
[1, 'spam', 3]
```



String object `spam` is referred by :

- variable `b`
- variable `c`
- index 1

>>> 32.2.3.2.Copy lists

>>> 32.2.3.2.1. case study

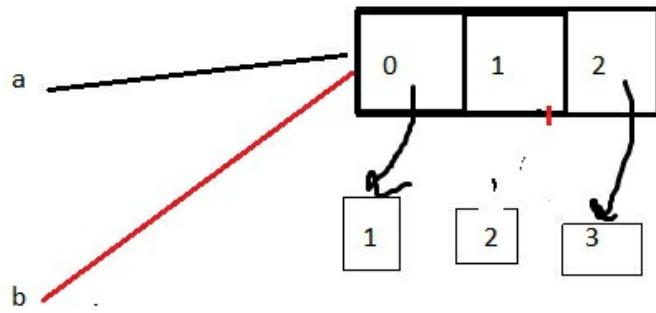
>>> **Exercise 353**

Evaluate the following code and draw the variable and the object

```
>>> a = [1,2,3]
>>> b = a
>>> b
```

Solutions

```
>>> a = [1,2,3]
>>> b = a
# b = a
# b = [1,2,3]
>>> b
[1, 2, 3]
```



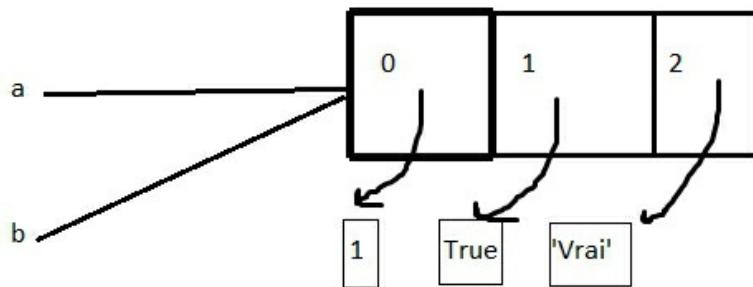
Variable `a` and variable `b` refer to same list object.

>>> Exercise 354

Evaluate the following code and draw the variable and the object

```
>>> a = [1, True, 'Vrai']
>>> b = a
```

Solutions

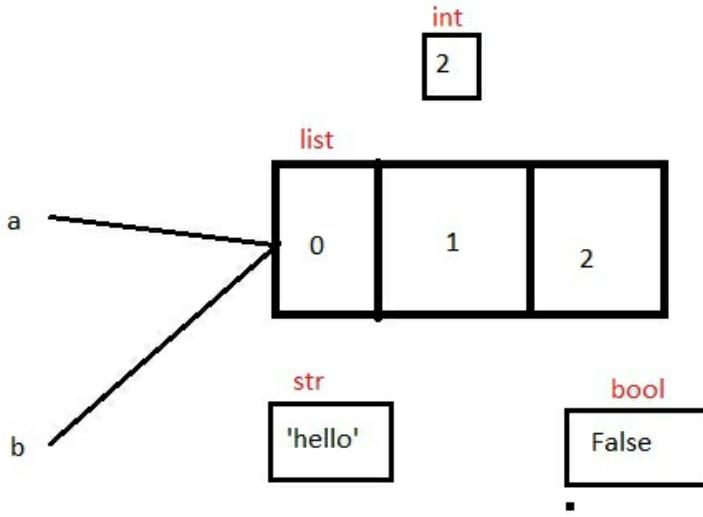


>>> Exercise 355

Evaluate the following code and draw the variable and the object

```
>>> a = ['hello', 2, False]
>>> b = a
```

Solutions



>>> 32.2.3.2.1. First world problem : list side effect

>>> Exercise 356

Evaluate the following code and draw the variable and the object

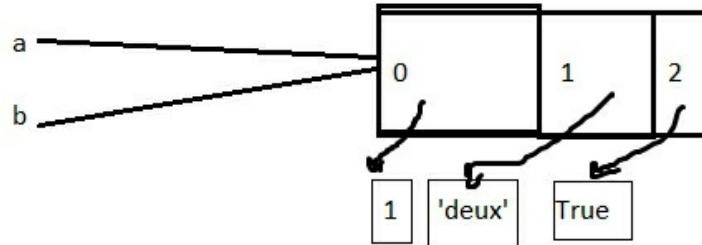
```
>>> a = [1, 'deux', True]
>>> b = a
>>> a[0] = 0
>>> a
>>> b
```

Solutions

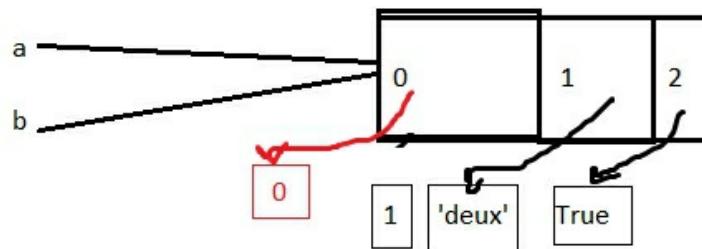
```
>>> a = [1, 'deux', True]
>>> b = a
# b = [1, 'deux', True]
>>> a[0] = 0
# a = [a[0], 'deux', True]
# a = [0, 'deux', True]
>>> a
[0, 'deux', True]
```

```
>>> b
```

```
[0, 'deux', True]
```



```
>>> a[0] = 0
```



Tips :

If two variables refer to the same list object there will be a side effect problem.

```
>>> Exercise 357
```

Evaluate the following code and draw the variable and the object

```
>>> a = [True, False, 3]
```

```
>>> b = a
```

```
>>> a[0] = 4
```

```
>>> b
```

```
>>> b[1] = 'deux'
```

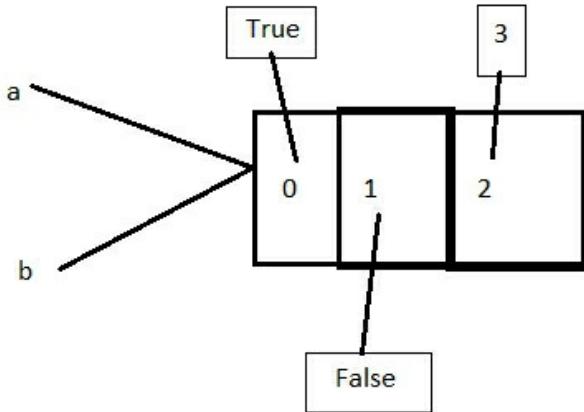
```
>>> a
```

```
>>> b
```

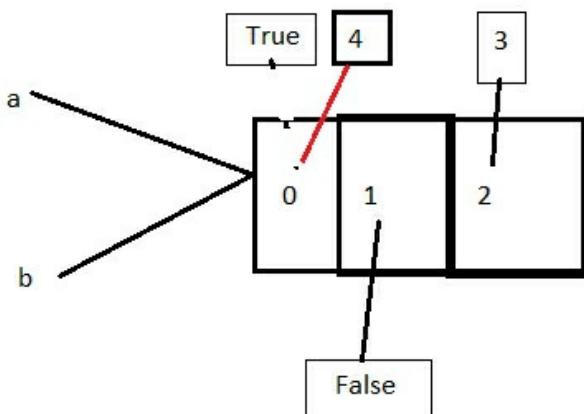
Solutions

```
>>> a = [True, False, 3]
```

```
>>> b = a  
# b = [True, False, 3]
```

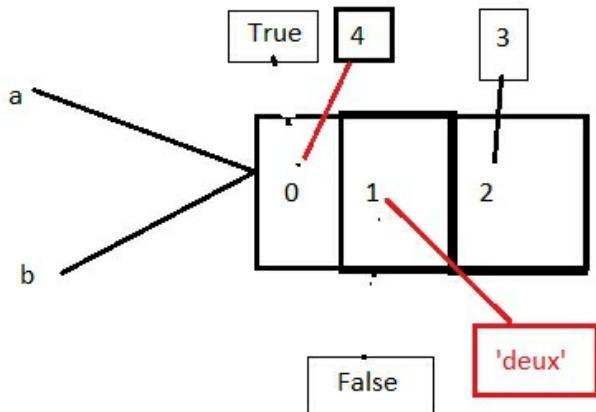


```
>>> a[0] = 4  
# a = [4, False, 3]  
>>> b  
[4, False, 3]
```



```
>>> b[1] = 'deux'  
# b = [4, 'deux', 3]  
>>> a  
[4, 'deux', 3]  
>>> b
```

```
[4, 'deux',3]
```



```
>>> 32.2.3.3.New list object from slicing
```

```
>>> 32.2.3.3.1.Slicing again
```

```
>>> Exercise 358
```

Evaluate the following code and draw the variable and the object

```
>>> a = [1,2,3,4,5]
```

```
>>> b = a[:3]
```

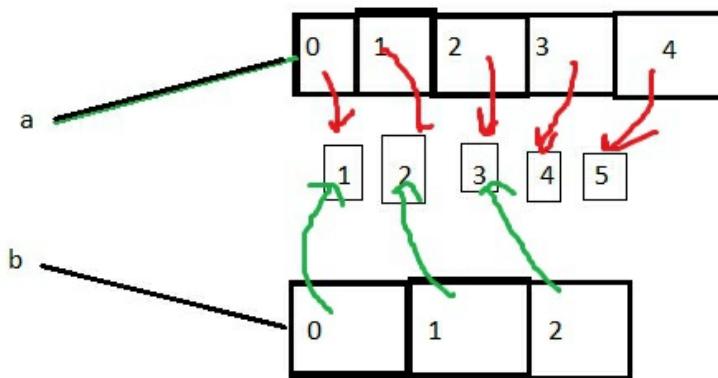
Solutions

```
>>> a = [1,2,3,4,5]
```

```
>>> b = a[:3]
```

```
# b = [a[0],a[1],a[2]]
```

```
# b =[1,2,3]
```



There's no side effect with slicing

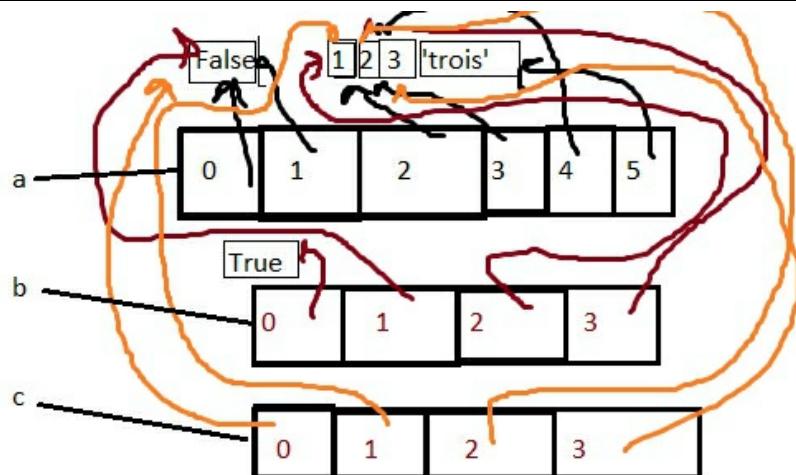
>>> Exercise 359

Evaluate the following code and draw the variable and the object

```
>>> a = [True, False, 1, 2, 3, 'trois']  
>>> b = a[:4]  
>>> c = a[1:5]  
>>> a[0] = False
```

Solutions

```
>>> b = a[:4]  
[True, False, 1, 2]  
>>> c = a[1:5]  
[False, 1, 2, 3]
```



>>> 32.2.3.3.2.New list object with [:]

>>> Exercise 360

Evaluate the following code and draw the variable and the object

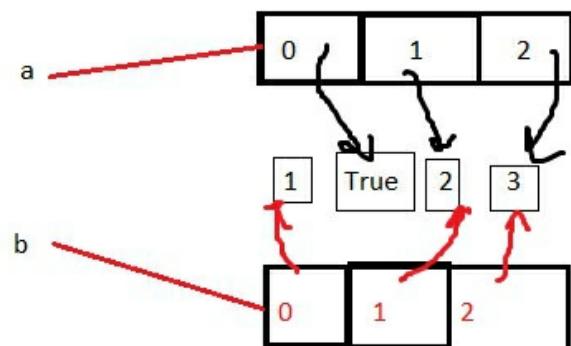
```
>>> a = [1,2,3]  
>>> b = a[:]  
>>> b  
>>> a[0] = True
```

Solutions

```

>>> a = [1,2,3]
>>> b = a[:]
# b = [a[0],a[1],a[2]]
# b =[1,2,3]
>>> b
[1, 2, 3]
>>> a[0] = True
# a = [a[0], 2, 3]
# a = [True, 2, 3]
>>> b
[1, 2, 3]

```



Tips

```

>>> liste2 = liste1[:]

```

- list copy
- new object created
- no side effect

>>> Exercise 361

Evaluate the following code and draw the variables and the objects

```

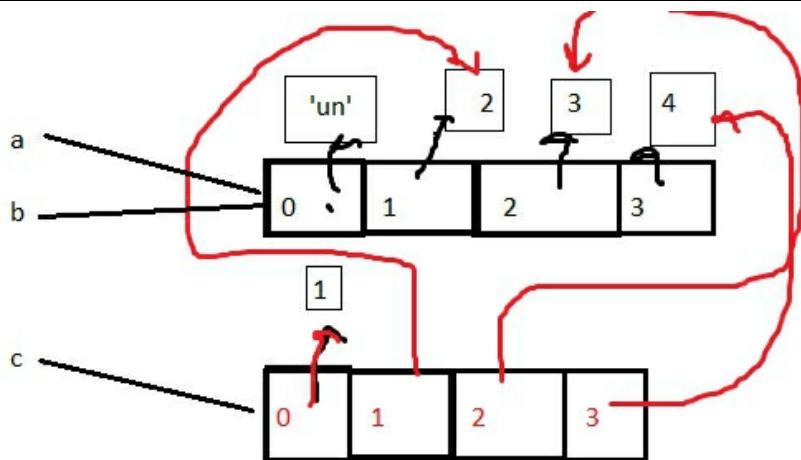
>>> a = [1,2,3,4]
>>> b = a

```

```
>>> c = a[:]  
>>> a[0] = 'un'  
>>> b  
>>> c
```

Solutions

```
>>> a  
['un', 2, 3, 4]  
>>> b  
['un', 2, 3, 4]  
>>> c  
[1,2,3,4]
```



>>> 32.2.4.Nested list

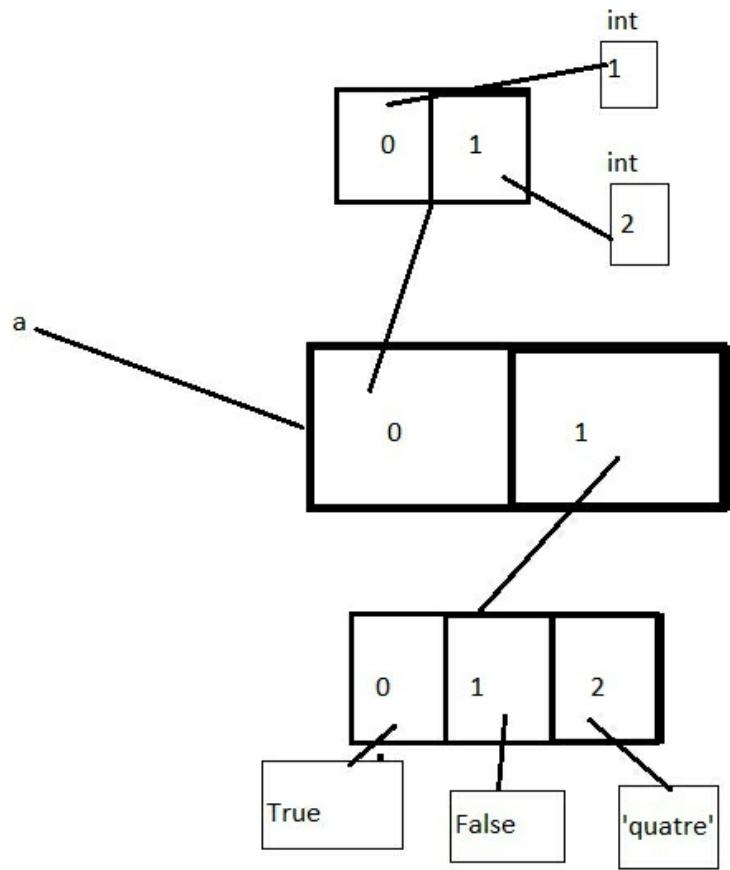
>>> **Exercise 362**

Evaluate the following code and draw the variables and the objects

```
>>> a = [[1,2],[True, False, 'quatre']]
```

Solutions

```
>>> a = [[1,2],[True, False, 'quatre']]
```

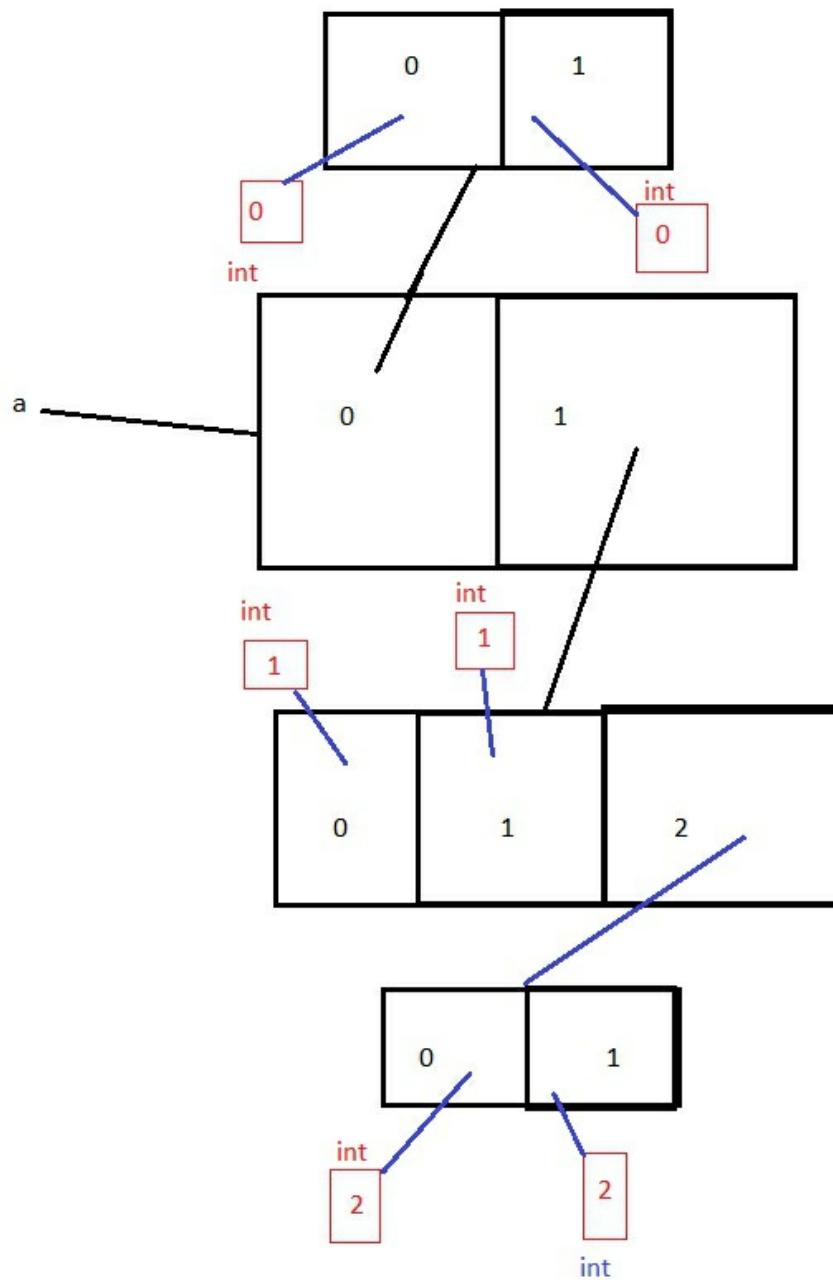


>>> Exercise 363

Evaluate the following code and draw the variables and the objects

```
>>> a = [[0,0],[1,1,[2,2]]]  
>>> a[1][2][1]
```

Solutions



```
>>> a[1][2][1]
```

```
2
```

>>> 33.Classes

« toujours avoir la classe » hitch

>>> 33.1.Classes 101

« au début était la classe object » un concepteur de Python

>>> 33.1.1.How to create a class object

>>> **Exercise 364**

1° Evaluate the following code

```
>>> class football:
```

```
    """la classe du football """
```

```
    print('football')
```

2° Evaluate the following code

```
>>> type(football)
```

```
>>> print(football)
```

Solutions

1°

Keword : class

Docstring : """la classe du football """

Block : print('football')

```
>>> class football:
```

```
    """la classe du football """
```

```
    print('football')
```

```
football
```

2°

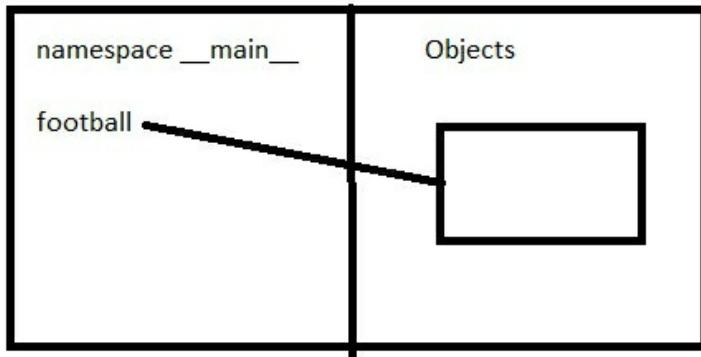
```
>>> type(foobtall)
```

```
<class 'type'>
```

```
>>> print(foottall)
```

```
<class '__main__.football'>
```

Variable football refers to class object



Tips

```
>>> class class_name :  
    """docstring"""  
    block
```

>>> Exercise 365

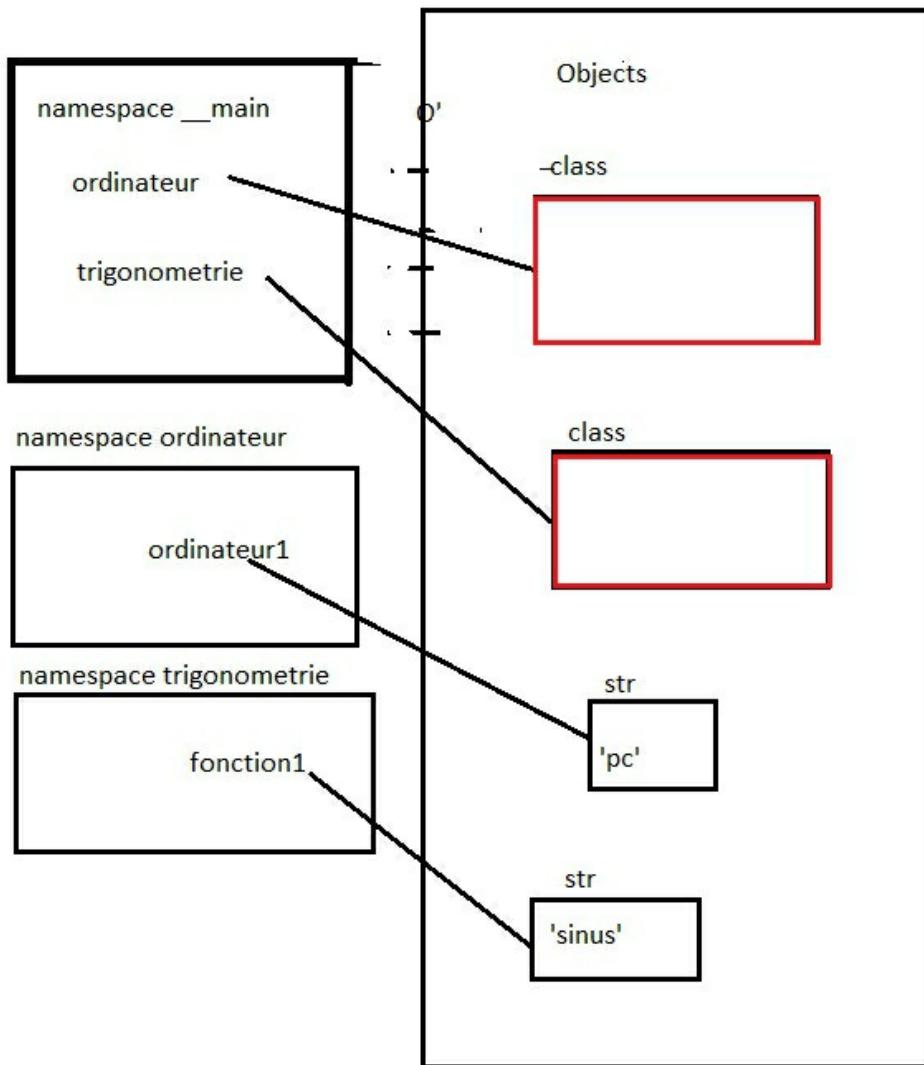
Evaluate the following code and draw the variables and the objects

```
>>> class ordinateur :  
    """la classe des ordinateurs """  
    ordinateur1 = 'pc'  
>>> class trigonometrie :  
    """classe des fonctions trigonometriques"""  
    fonction1 = 'sinus'
```

Corrigé

Each class has its own namespace

```
>>> class ordinateur :  
    """la classe des ordinateurs """  
    ordinateur1 = 'pc'  
>>> class trigonometrie :  
    """classe des fonctions trigonometriques"""  
    fonction1 = 'sinus'
```



>>> 33.1.2.Operations on classes

>>> 33.1.2.1. Variables within a class : attribute reference

>>> **Exercise 365**

1° Evaluate the following code

```
>>> class classe1 :
    """ This is a docstring """
    variable1 = 42
    variable2 = 27
```

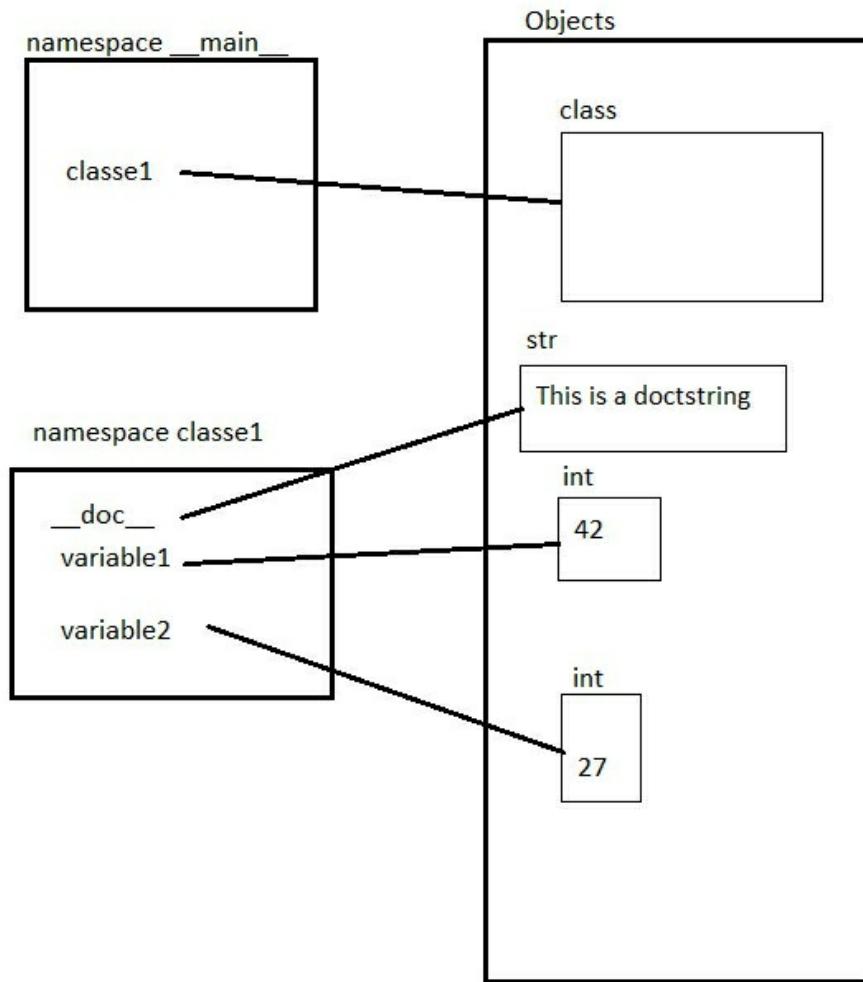
2° Evaluate the following code

```
>>> classe1.variable1
>>> classe1.variable2
>>> classe1.variable1 = 55
>>> classe1.variable1
>>> classe1.variable3
```

Solutions

1°

```
>>> class classe1 :
    """ This is a doctstring """
    variable1 = 42
    variable2 = 27
```



2° The dot operator for finding in the right namespace

```

>>> classe1.variable1
# classe1.variable1
# classe1.42
# 42
42
>>> classe1.variable2
# classe1.variable2
# classe1.27
# 27

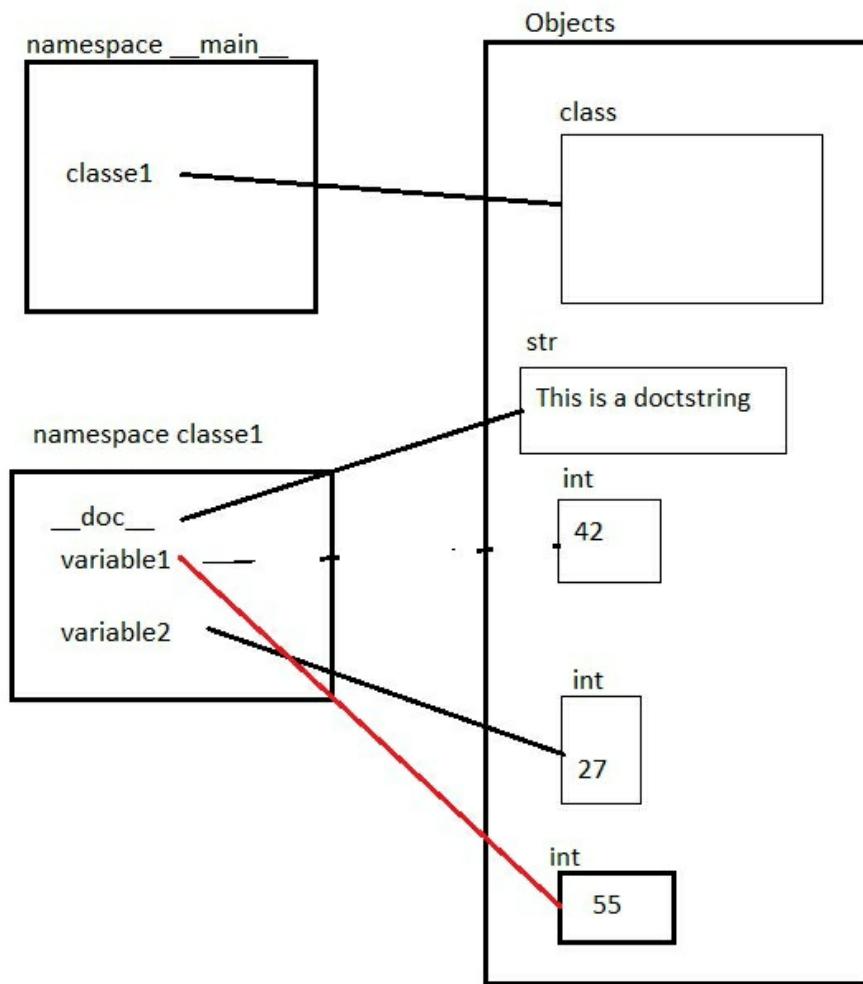
```

27

Variable1 and variable2 are called class attributes

```
>>> classe1.variable1 = 55
# classe1.variable1 = 55
# class classe1 :
#"""" ceci est une docstring """
#variable1 = 55
#variable2 = 27
>>> classe1.variable1
# classe1.variable1
# classe1.55
# 55
```

55



```
>>> classe1.variable3
```

```
AttributeError : type object 'classe1' has no attribute variable3
```

Tips

```
>>> class class_name :  
      """ docstring """  
      variable_class_attribute = value1  
>>> class_name.variable_attribute  
value1
```

>>> Exercise 366

Evaluate the following code

```
>>> class c1 :  
    donnee = 42  
>>> c1.donnee  
>>> c1.variable  
>>> c1.donnee = 43.0  
>>> c1.donnee
```

Solutions

```
>>> class c1 :  
    donnee = 42  
>>> c1.donnee  
# c1.42  
# 42  
42  
>>> c1.variable  
AttributeError: type object 'c1' has no attribute 'variable'  
>>> c1.donnee = 43.0  
# c1.donnee = 43.0  
>>> c1.donnee  
43.0
```

>>> Exercise 367

Evaluate the following code

```
>>> donnee = 28  
>>> class c2 :  
    """ a class with two attributes """  
    donnee = 57.0  
    donnee2 = True
```

```
>>> donnee
>>> c2.donnee
>>> c2.donnee2
>>> donnee == c2.donnee
>>> c2.__doc__
```

Solutions

```
>>> donnee = 28
>>> class c2 :
    """ une classe qui contient deux attributs """
    donnee = 57.0
    donnee2 = True
>>> donnee
28
>>> c2.donnee
57.0
>>> c2.donnee2
True
>>> donnee == c2.donnee
# 28 == 57.0
# False
False
>>> c2.__doc__
""" une classe qui contient deux attributs """
```

>>> 33.1.2.1.1.Add attributes

>>> **Exercise 368**

Evaluate the following code

```
>>> class c1 :
```

```
donnee1 = 42.0
>>> c1.donnee1
>>> c1.donnee2
>>> c1.donnee2 = 7
>>> c1.donnee2
```

Solutions

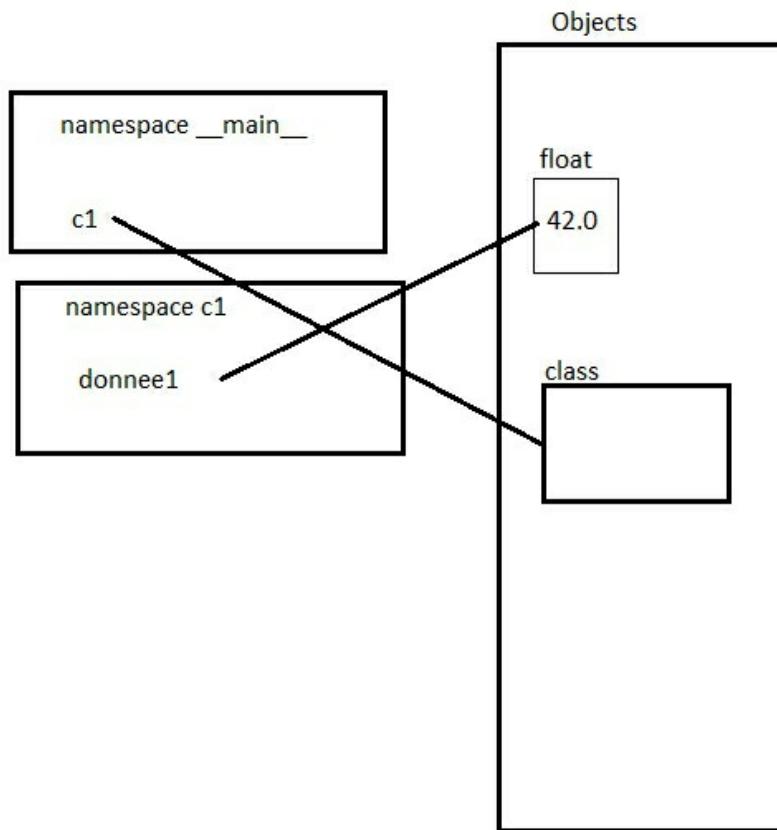
Class name : c1

Block : one attribute

Docstring : none

```
>>> class c1 :
    donnee1 = 42.0
>>> c1.donnee1
42.0
>>> c1.donnee2
```

AttributeError: type object 'c1' has no attribute 'donnee2'



```
>>> c1.donnee2 = 7
```

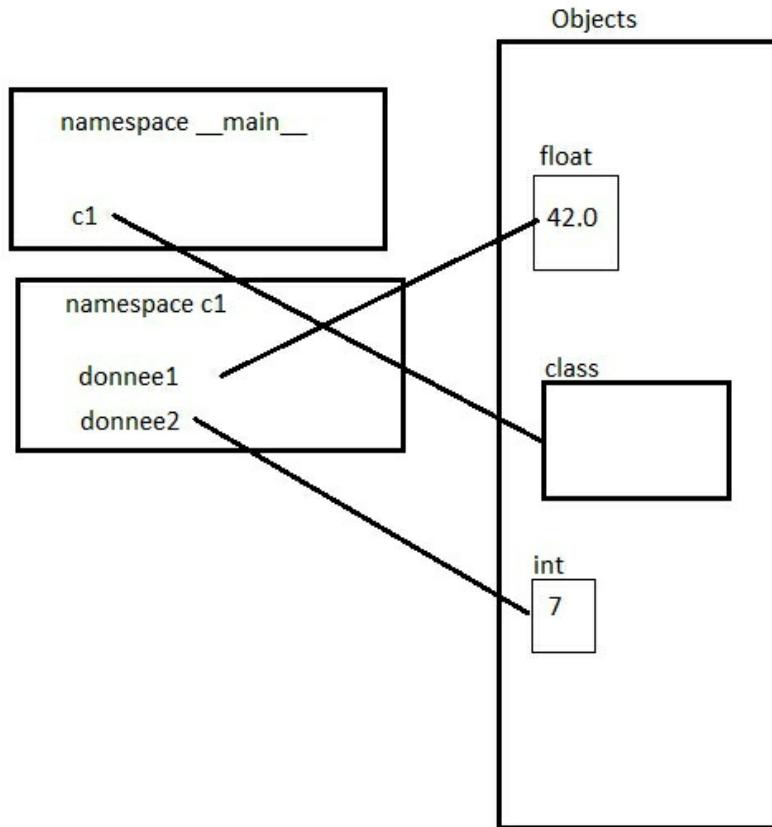
```
# class c1 :
```

```
    donnee1 = 42.0
```

```
    donnee2 = 7
```

```
>>> c1.donnee2
```

```
7
```



Tips

```
>>> classe_name.attribut_donnee = value
```

>>> 33.1.2.2.Function within a class : attribute method

>>> **Exercise 369**

Evaluate the following code

```
>>> class classe3 :
    donnee = 43
    def f(x) :
        return x*2
    def affichage() :
        return 'pas hello'
```

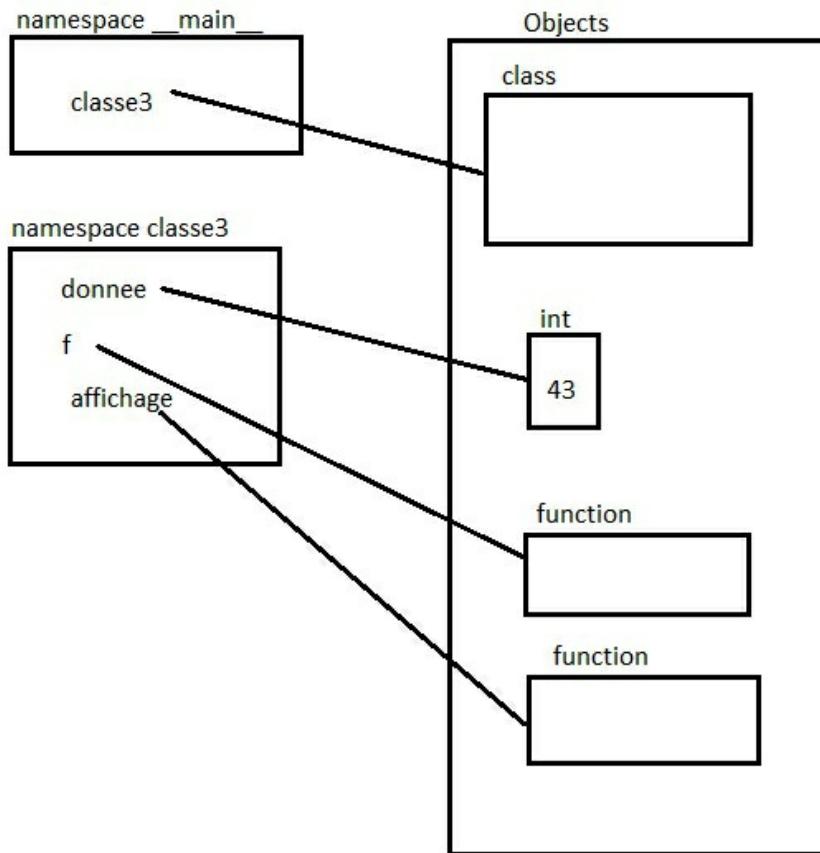
2° Evaluate the following code

```
>>> classe3.donnee
```

```
>>> classe3.f(8)  
>>> classe3.affichage()
```

Solutions

1°



2°

```
>>> classe3.donnee  
# classe3.donnee  
# classe.43  
# 43  
43  
>>> classe3.f(8)
```

```
# classe3.f(8)
# return 8 * 2
# return 16
# classe3.16
# 16
16
>>> classe3.affichage()
# classe3.affichage()
# return 'pas hello'
# classe3.'pas hello'
# 'pas hello'
'pas hello'
```

>>> **Exercise 370**

Evaluate the following code

```
>>> class c1 :
    def f(x) :
        return x**2
    donnee = 7
>>> c1.f(4)
>>> c1.f(10)
>>> c1.donnee
>>> c1.donnee = 13
>>> c1.donnee
```

Solutions

```
>>> class c1 :
    def f(x) :
```

```
    return x**2
```

```
donnee = 7
```

```
>>> c1.f(4)
```

```
# c1.f(4) :
```

```
# return 4**2
```

```
# retrun 16
```

```
16
```

```
>>> c1.f(10)
```

```
100
```

```
>>> c1.donnee
```

```
7
```

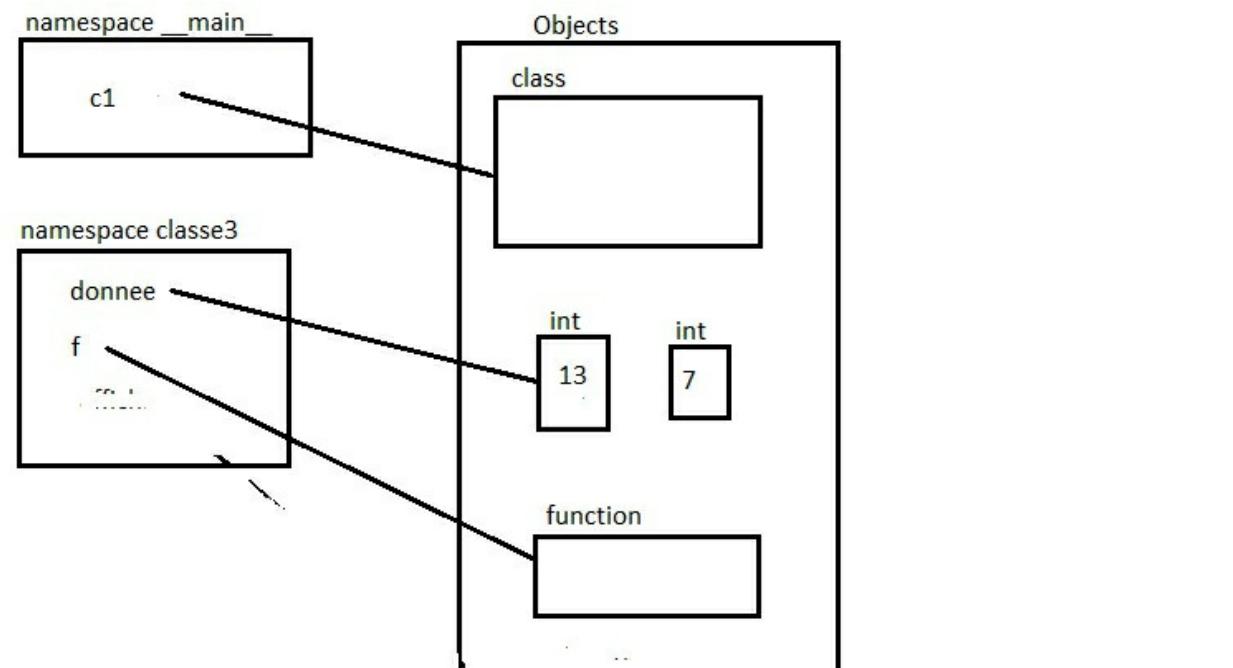
```
>>> c1.donnee = 13
```

```
# donnee = 13
```

```
# 13
```

```
>>> c1.donnee
```

```
13
```



>>> 33.1.3.Instantiation

>>> **Exercise 371**

1° Evaluate the following code

```
>>> class c :  
    donnee = 7  
    def methode1(x) :  
        return x**2  
>>> c1 = c()
```

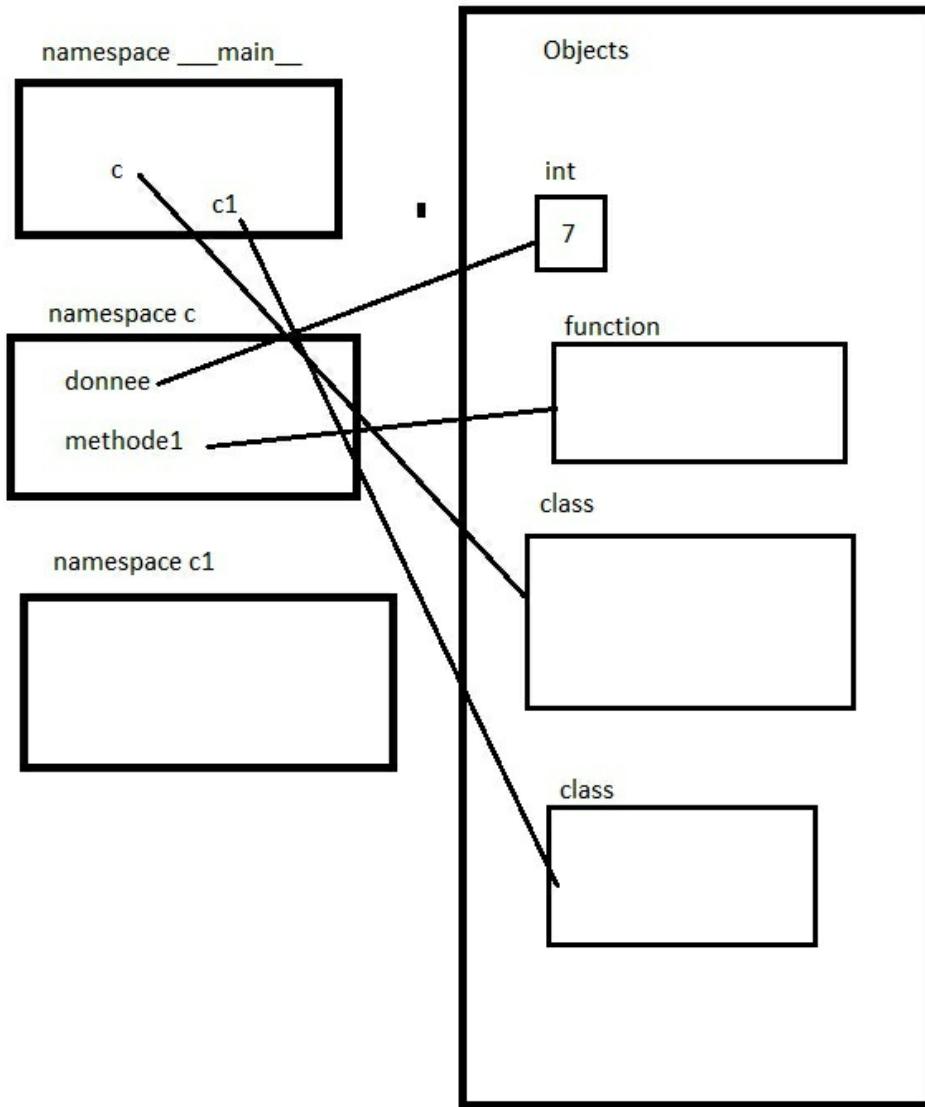
2° Evaluate the following code

```
>>> c1.donnee  
>>> c1.methode(16)
```

Solutions

1°

```
>>> class c :  
    donnee = 7  
    def methode(x) :  
        return x**2
```



Instance name : c1

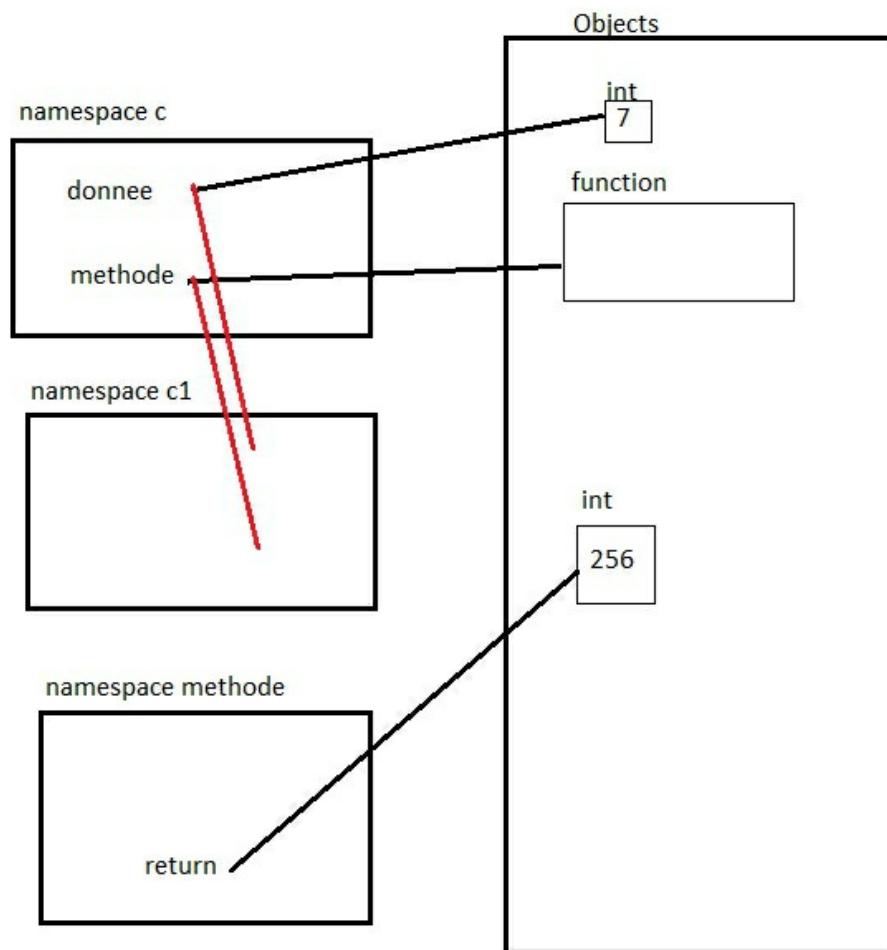
An instance has its own namespace like module, function, class.

```
>>> c1 = c()
```

2° As seen on the previous schema the c1 seems empty but we can access to class c attributes with the dot operator.

```
>>> c1.donnee
# c1.donnee
# c1.7
# 7
```

```
# 7
7
>>> c1.methode(16)
# c1.methode(16)
# return 16**2
# return 256
# c1.256
# 256
256
```



Tips

```
>>> instance_name = class_name()
```

There is a new object created from the class : the instance

From the instance we can access to class attributes with the dot operator

```
>>> instance.attribut
```

>>> **Exercise 372**

1° Evaluate the following code

```
>>> class c :  
    donnee1 = 34  
    def methode1(x) :  
        return x/2  
>>> c1 = c()  
>>> c1.donnee1  
>>> c1.methode1(10)
```

2° Evaluate the following code

```
>>> c2 = c()  
>>> c2.donnee1  
>>> c2.methode1(20)
```

Solutions

1°

Class name : c

Attribute variable : donnee (int)

Attribute method : method1

Instance name : c1

```
>>> class c :  
    donnee1 = 34  
    def methode1(x) :  
        return x/2
```

```
>>> c1 = c()  
>>> c1.donnee1  
7  
>>> c1.methode1(10)  
# return 10 / 2  
# return 5.0  
# 5.0  
5.0
```

2°

```
>>> c2 = c()  
>>> c2.donnee1  
7  
>>> c2.methode1(20)  
# return 20 / 2  
# return 10.0  
10.0
```

>>> 33.1.3.1.Instance variables

>>> **Exercise 373**

1° Evaluate the following code

```
>>> class c :  
    donnee1 = 7  
>>> c1 = c()  
>>> c1.donnee1
```

2° Evaluate the following code

```
>>> c1.a = 10  
>>> c1.b = [1,2]  
>>> a
```

```
>>> c.a
```

Solutions

1° class name : c

Attribute : donnee1 (int)

Instance name : c1

```
>>> class c :
```

```
    donnee1 = 7
```

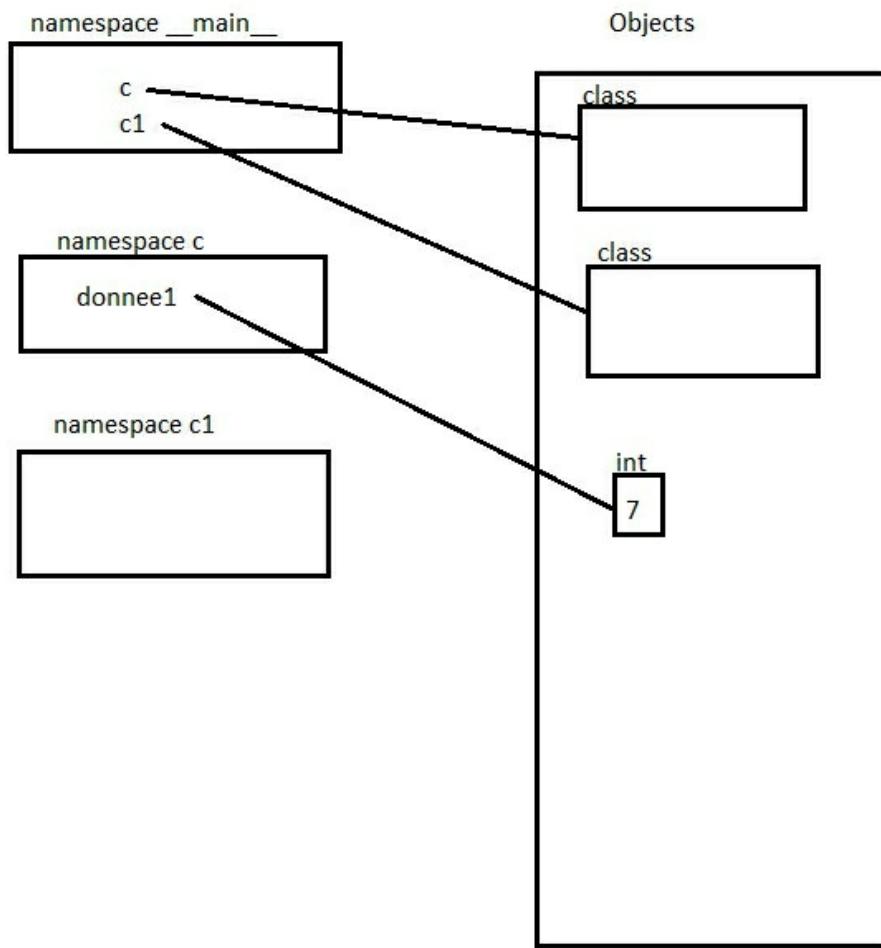
```
>>> c1 = c()
```

```
>>> c1.donnee1
```

```
# c1.donnee1
```

```
# c1.7
```

```
# 7
```



2°

```
>>> c1.a = 10
>>> c1.b = [1,2]
>>> a
```

NameError : name 'a' is not defined

```
>>> c.a
```

AttributeError : type object 'c' has no attribute 'a'

```
>>> c1.a
```

10

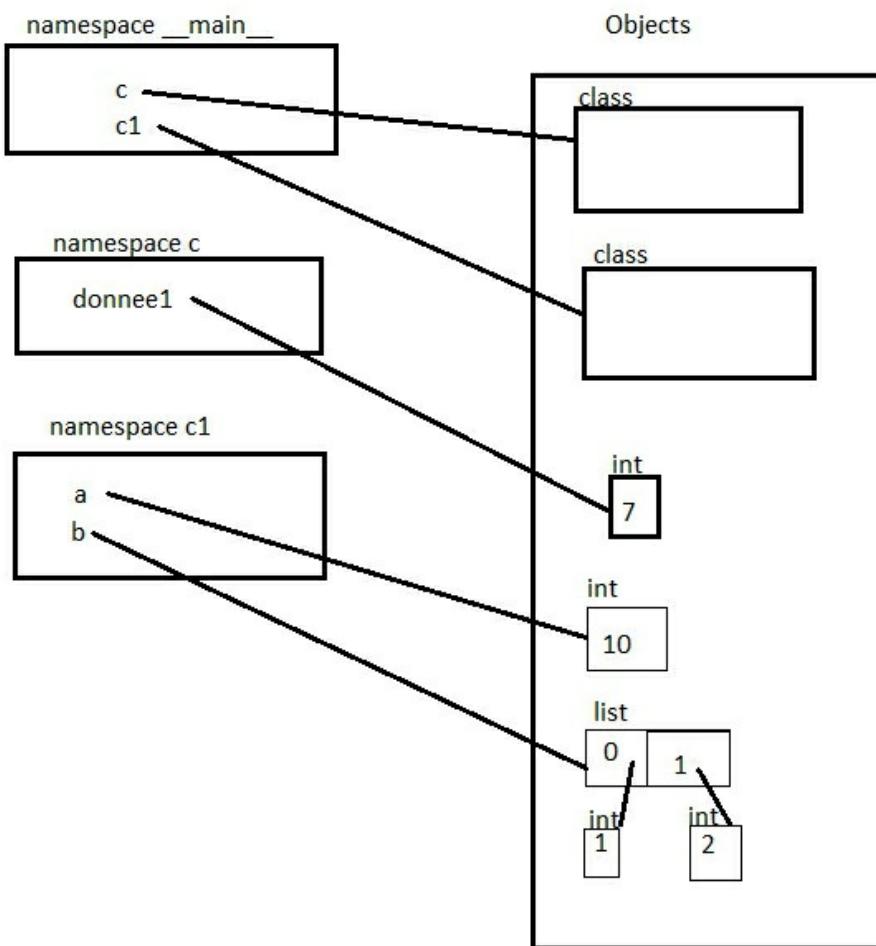
```
>>> c1.b
```

```
[1,2]
```

```
>>> c1.donnee1
```

```
7
```

The class can't access to instance c1 variables a and b



Tips

```
>>> instance.variable = value
```

```
>>> instance.variable
```

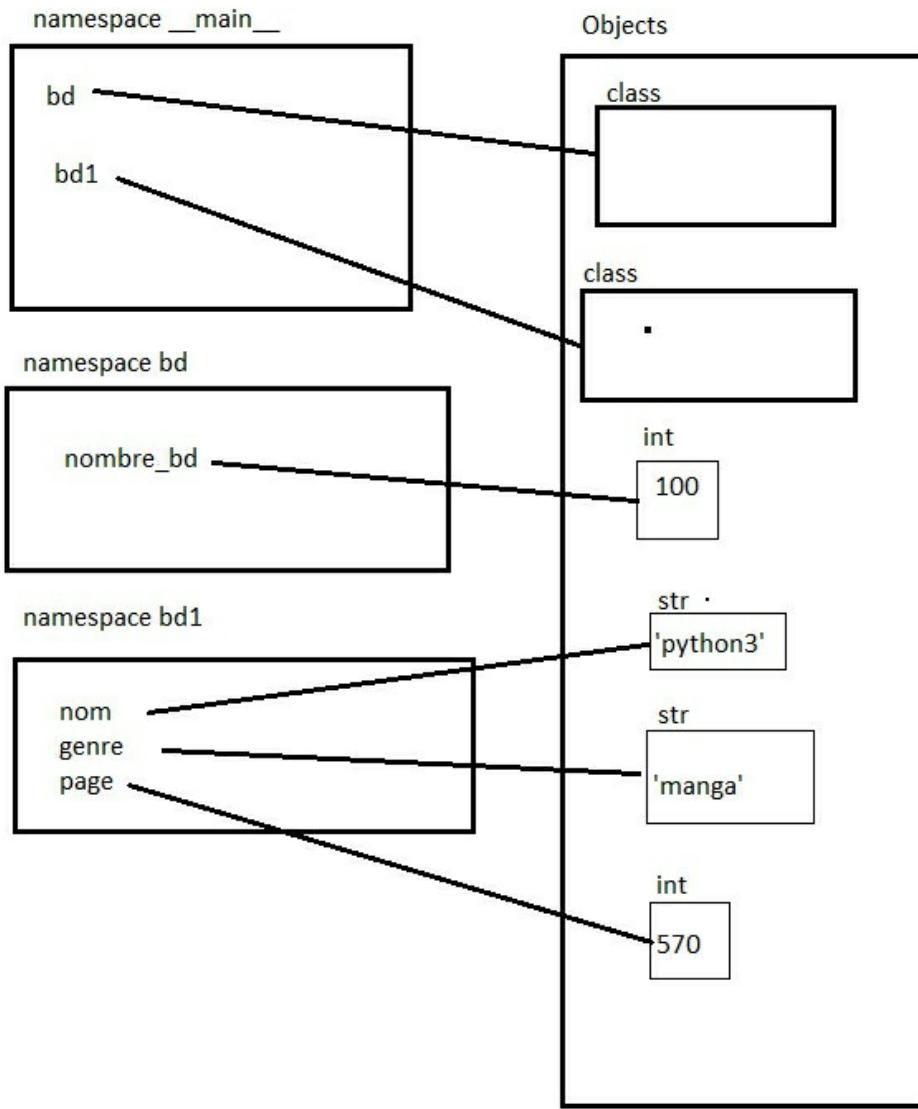
```
value
```

>>> **Exercise 374**

Evaluate the following code

```
>>> class bd :  
    nombre_bd = 100  
>>> bd1 = bd()  
>>> bd1.nom= 'python 3'  
>>> bd1.genre = 'manga'  
>>> bd1.page = 570  
>>> bd.nom
```

Solutions



```
>>> bd.nom
```

```
AttributeError: type object 'bd' has no attribute 'nom'
```

```
>>> 31.3.1.2built-in function isinstance ( )
```

```
>>> Exercise 375
```

Evaluate the following code

```
>>> help(isinstance)
```

```
isinstance (obj, class)
```

```
    return whether an object is an instance of a class [...]
```

```
>>> class c :
```

```
attribut_donnee = 53
>>> class d :
    attribut_donnee = 65.0
>>> c1= c()
>>> c2 = c()
>>> d1 = d ()
>>> d2 = d()
>>> isinstance (c1, c)
>>> isisntance(c2, c)
>>> isinstance(d1,d)
>>> isinstance(d2,d)
>>> isinstance(c1,d)
>>> isinstance(d1,c)
```

Solutions

```
>>> isinstance (c1, c)
```

```
True
```

```
>>> isisntance(c2, c)
```

```
True
```

```
>>> isinstance(d1,d)
```

```
True
```

```
>>> isinstance(d2,d)
```

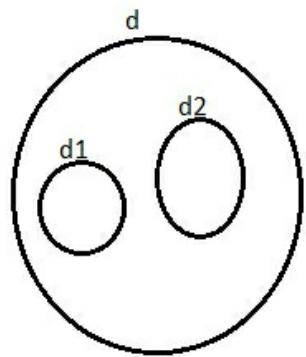
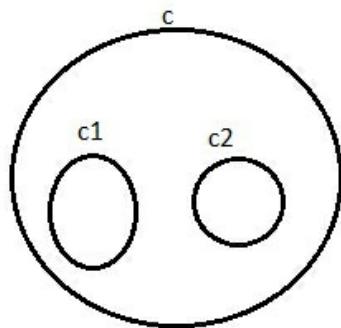
```
True
```

```
>>> isinstance(c1,d)
```

```
False
```

```
>>> isinstance(d1,c)
```

```
False
```



>>> Exercise 376

Evaluate the following code

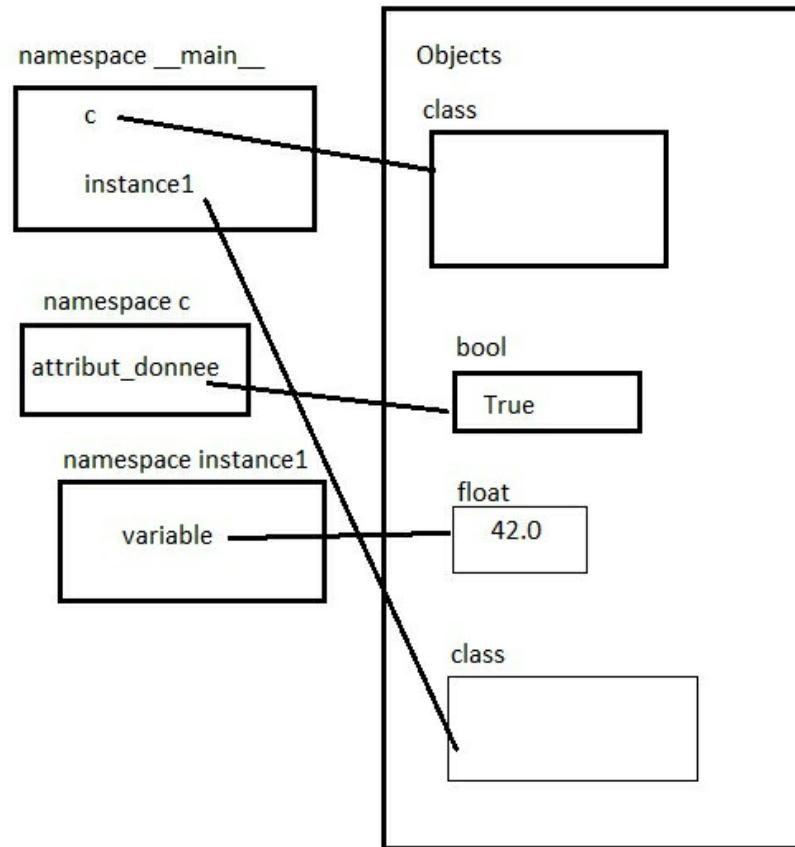
```
>>> class c():
    attribut_donnee = True
>>> instance1 = c()
>>> c.donnee
>>> instance1.variable = 42.0
>>> isinstance(instance1, c)
```

Solutions

```
>>> class c():
    attribut_donnee = True
>>> instance1 = c()
>>> c.donnee
AttributeError: type object 'c' has no attribute 'donnee'
>>> instance1.variable = 42.0
```

```
>>> isinstance(instance1, c)
```

```
True
```



```
>>> isinstance(instance1, c)
```

```
True
```

```
>>> 31.3.1.2.Builtins-types
```

```
>>> Exercise 377
```

1° Evaluate the following code

```
>>> class c :  
    donnee = 0  
>>> type(c)  
>>> type(int)
```

2° Evaluate the following code

```
>>> c1=c()  
>>> type(c1)  
>>> type(3)  
>>> type(3.0)  
>>> type('spam')  
>>> type(True)
```

3° Evaluate the following code

```
>>> isinstance(3,int)  
>>> isinstance(3.0, float)  
>>> isinstance('spam', float)  
>>> isinstance('spam', str)  
>>> isinstance(3.0, float)
```

Solutions

1°

```
>>> type(c)  
<class 'type'>  
>>> type(int)  
<class 'type'>  
>>> help(int)  
class int(object) :[...]
```

2°

```
>>> type (c1)  
<class '__main__.c'>  
>>> type (3)  
<class 'int'>  
>>> type(3.0)
```

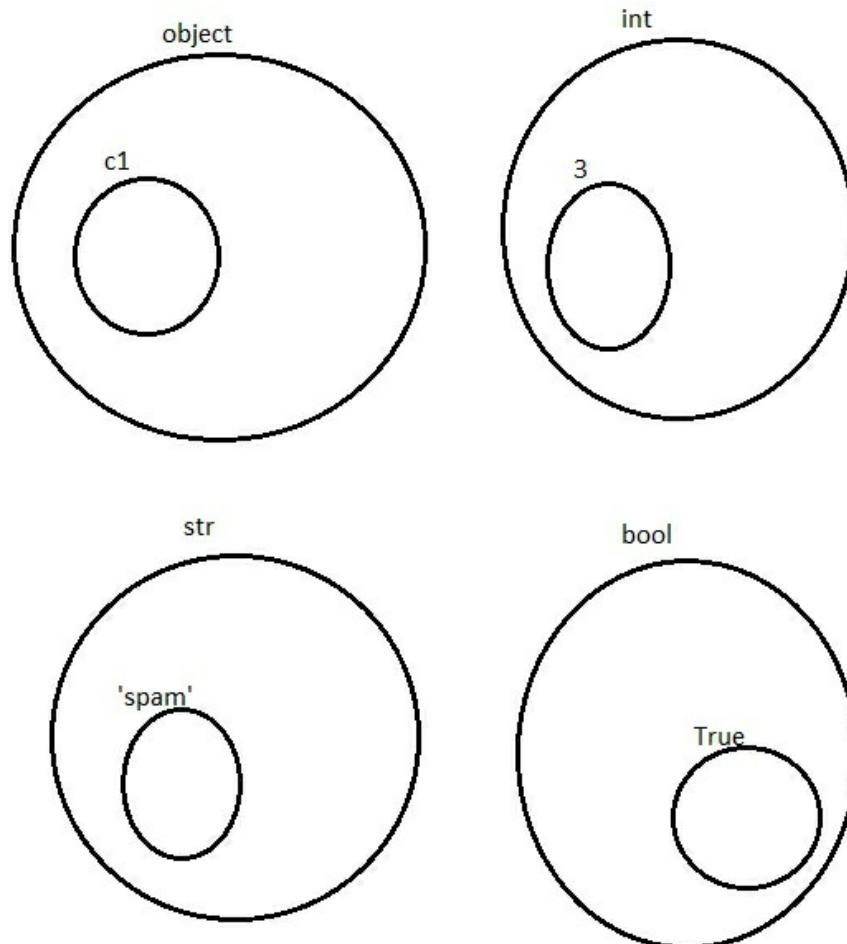
```
<class 'float'>
```

```
>>> type('spam')
```

```
<class 'str'>
```

```
>>> type(True)
```

```
<class 'bool'>
```



3°

```
>>> isinstance(3,int)
```

```
True
```

```
>>> isinstance(3.0, float)
```

```
True
```

```
>>> isinstance('spam', float)
```

```
False  
>>> isinstance('spam', str)
```

```
True  
>>> isinstance(3.0, float)
```

```
True  
>>> Exercise 378
```

Evaluate the following code

```
>>> class c :  
    donnee = 2  
>>> c1=c()  
>>> type(c1)  
>>> isinstance(c1,c)  
>>> type([1,2])  
>>>isinstance([1,2],list)
```

Solutions

```
>>> class c :  
    donnee = 2  
>>> c1=c()  
>>> type(c1)  
<class '__main__.c'>  
>>> isinstance(c1,c)  
True  
>>> type([1,2])  
<class 'list'>  
>>>isinstance([1,2],list)  
True
```

```
>>> 31.3.1.2.Superclass object
```

```
>>> Exercise 379
```

Evaluate the following code

```
>>> help(object)
```

Help on class object in module builtins

class object

The most base type

```
>>> isinstance(int, object)
```

```
>>> isinstance(str, object)
```

```
>>> isinstance(set, object)
```

```
>>> def f(x) :
```

return x

```
>>> isinstance(f,object)
```

Solutions

```
>>> help(object)
```

Help on class object in module builtins

class object

The most base type

```
>>> isinstance(int, object)
```

True

```
>>> isinstance(str, object)
```

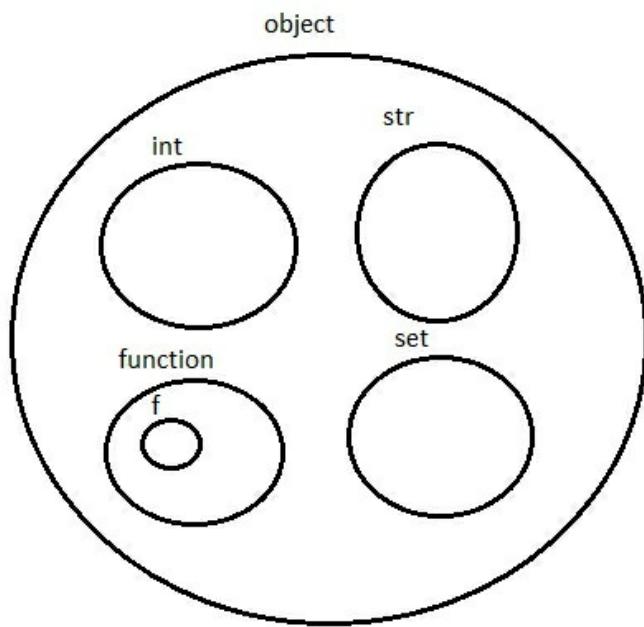
True

```
>>> isinstance(set, object)
```

True

```
>>> isinstance(f,object)
```

True



>>> Exercise 380

Evaluate the following code

```

>>> variable = 14
>>> def f(x) :
    return 2*x
>>> class c :
    donnee = 7
>>> isinstance(variable, int)
>>> isinstance(f, object)
>>> isinstance(variable, object)
>>>> isinstance(donnee, object)
>>> isinstance(c, object)

```

Solutions

```

>>> variable = 14
>>> def f(x) :

```

```
    return 2*x

>>> class c :
    donnee = 7

>>> isinstance(variable, int)
# isinstance(14,int)

# True

True
>>> isinstance(f, object)

True
>>> isinstance(variable, object)

True
>>>> isinstance(donnee, object)

NameError: name 'donnee' is not defined
>>> isinstance(c, object)

True
```

>>> 33.2.Methods again
« basé sur un fameux discours » anon123
>>> 33.2.1.The instance is the argument
>>> **Exercise 381**

1° Evaluate the following code

```
>>> def f(x) :
    return 'void'

>>> f(1)

>>> f()
```

2° Evaluate the following code

```
>>> class MaClasse :
```

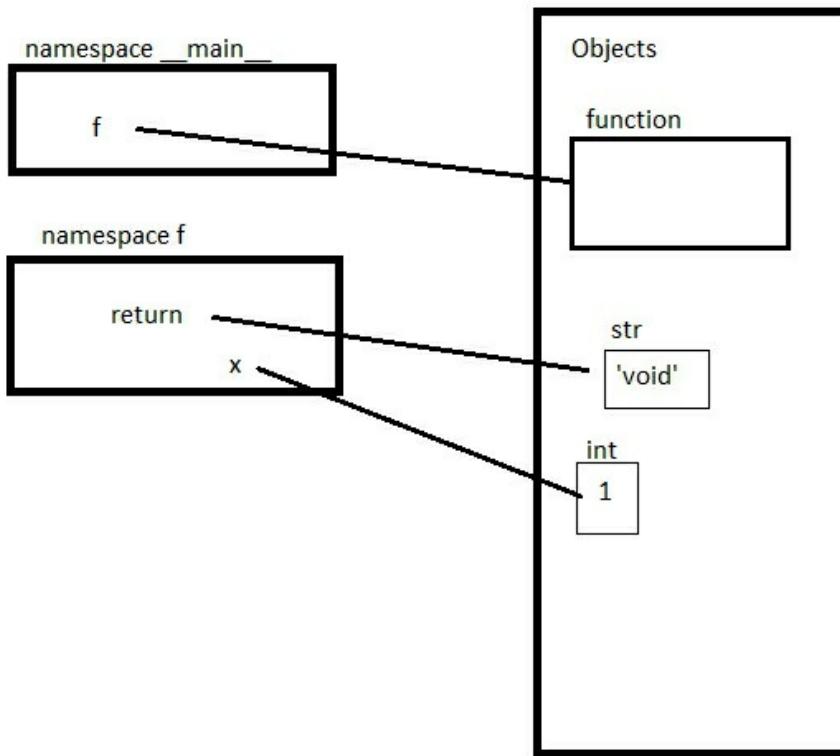
```
def f(x) :  
    return 'void'  
>>> a = MaClasse()  
>>> MaClasse.f(2)  
>>> MaClasse.f()  
>>> a.f()
```

Corrigé

1°

```
>>> def f(x) :  
    return 'void'  
>>> f(1)  
# f(1)  
# return 'void'  
'void'  
>>> f()
```

TypeError: f() missing 1 required positional argument: 'x'



2°

Class name : MaClasse

Attribute method : f

>>> class MaClasse :

 def f(x) :

 return 'void'

>>> MaClasse.f(2)

f(x) :

f(2)

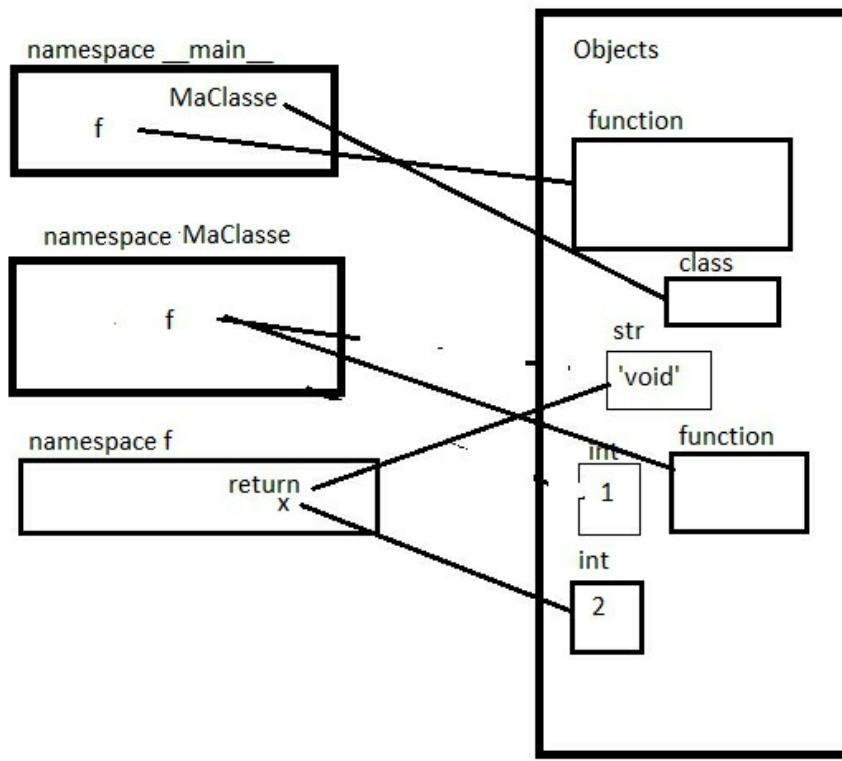
return 'void'

'void'

>>> MaClasse.f()

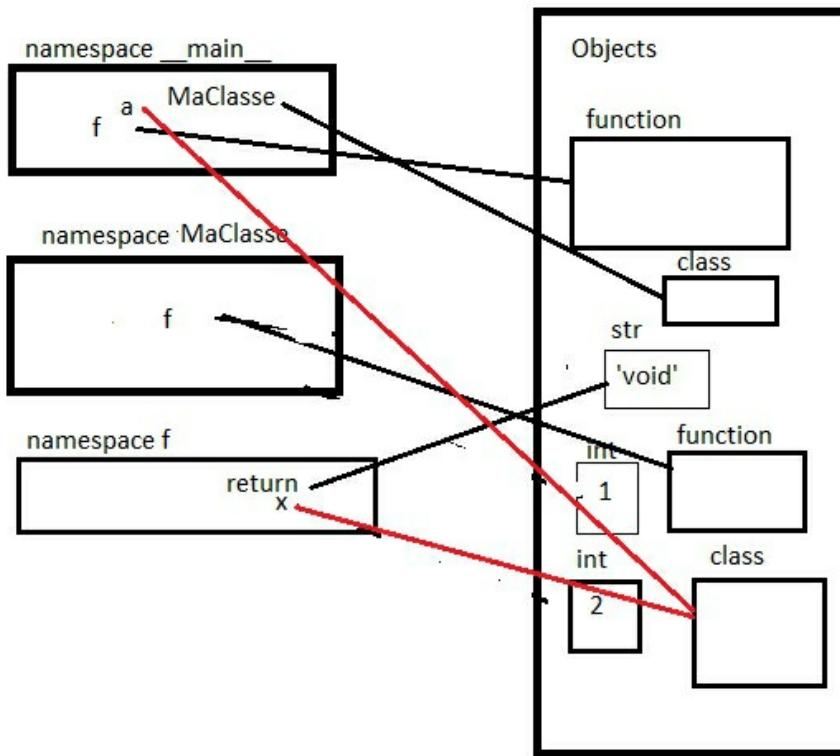
f()

TypeError: f() missing 1 required positional argument: 'x'



Let's create an instance of MaClasse

```
>>> a = MaClasse()  
>>> a.f()  
# a.f()  
# f(a)  
# return void  
'void'
```



Tips

Class method has n arguments

```
>>> methode(arg1, arg2, arg3, ..., argn)
```

If we call the method with the instance, instance will passing by reference as an argument

```
>>> instance.methode()
# methode(instance,arg2, arg3, ..., argn )
```

>>> Exercise 382

Evaluate the following code

```
>>> class c :
```

```
    def f(x, y) :
```

```
        return 'hello'
```

```
>>> c.f(1,2)
```

```
>>> c1=c()
```

```
>>> c1.f()  
>>> c1.f(2)
```

Solutions

```
>>> class c :  
    def f(x, y) :  
        return 'hello'
```

```
>>> c.f(1,2)  
# return 'hello'  
hello  
>>> c1=c()  
>>> c1.f()  
# c1.f()  
# f(c1)  
# TypeError : f() missing 1 required positional argument : 'y'
```

```
>>> c1.f(2)  
# c1.f(2)  
# f(c1, 2)  
# return 'hello'
```

```
hello'
```

>>> 33.2.2.Method __init__

>>> **Exercise 383**

1° Evaluate the following code

```
>>> class Compte:  
    """ Compte Bancaire d'un client """  
>>> compte_client1 = Compte()  
>>> compte_client1.nom = 'derp'
```

```
>>> compte_client1.adresse = 'Rome'  
>>> compte_client1.montant = 56
```

2° Evaluate the following code

```
>>> class Compte:  
    """ Compte Bancaire d'un client """  
    def __init__(self, nom, adresse, montant):  
        self.nom = nom  
        self.adresse = adresse  
        self.montant = montant  
>>> Compte.__doc__  
>>> c1 = Compte('derp', 'rome', 42)  
>>> c1.nom  
>>> c1.adresse  
>>> c1.montant  
>>> c2 = Compte('derpina', 'tokyo', 100)  
>>> print(c2.nom, c2.adresse, c2.montant)
```

Solutions

1°

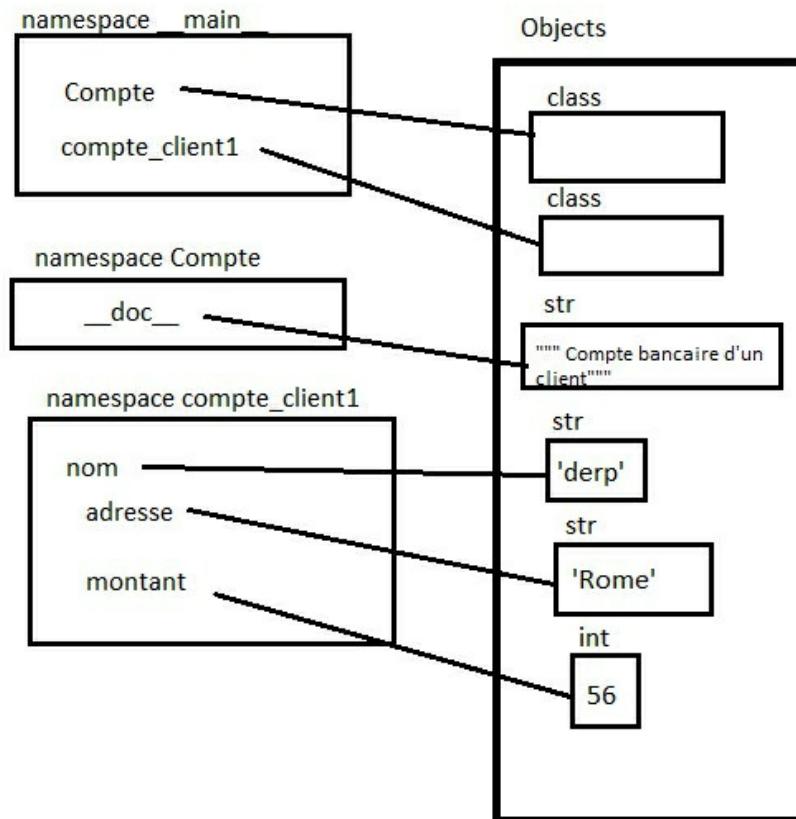
```
>>> class Compte:  
    """ Compte Bancaire d'un client """  
>>> compte_client1 = Compte()  
>>> compte_client1.nom = 'derp'  
>>> compte_client1.adresse = 'Rome'  
>>> compte_client1.montant = 56  
>>> compte_client1.nom  
'derp'
```

```
>>> compte_client1.adresse
```

```
'Rome'
```

```
>>> compte_client1.montant
```

```
56
```



2°

```
>>> class Compte:
```

```
    """ Compte Bancaire d'un client """
```

```
    def __init__(self, nom, adresse, montant):
```

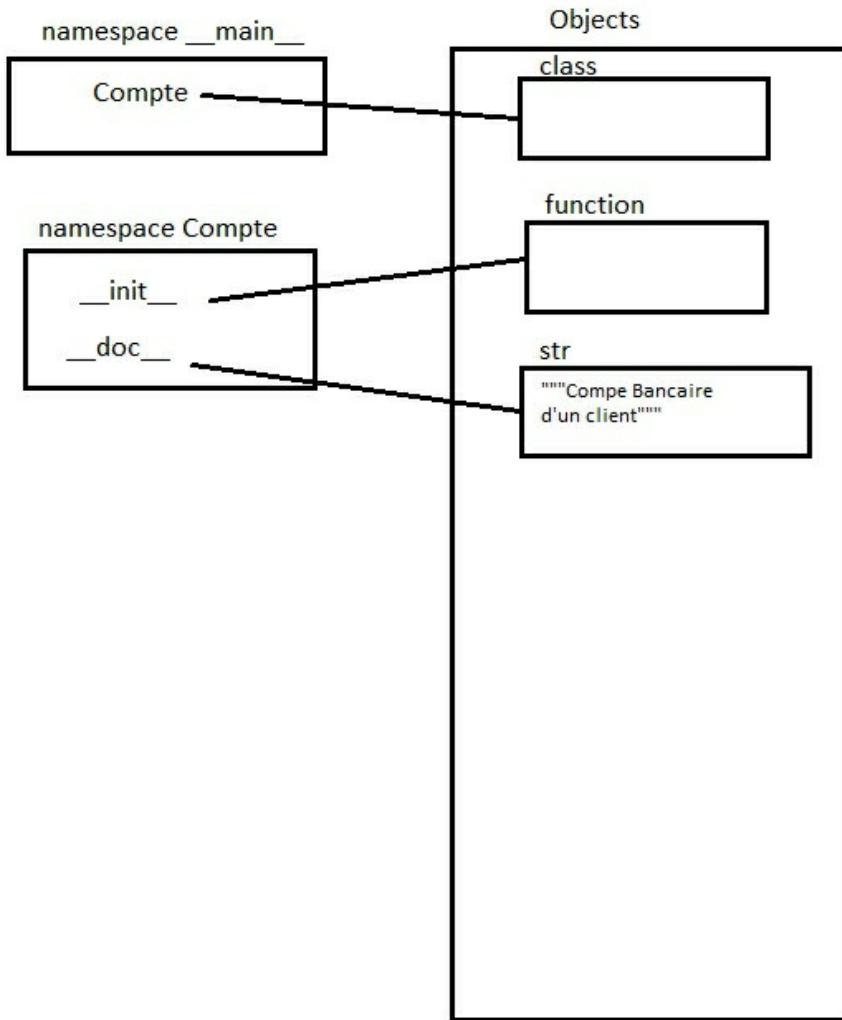
```
        self.nom = nom
```

```
        self.adresse = adresse
```

```
        self.montant = montant
```

```
>>> Compte.__doc__
```

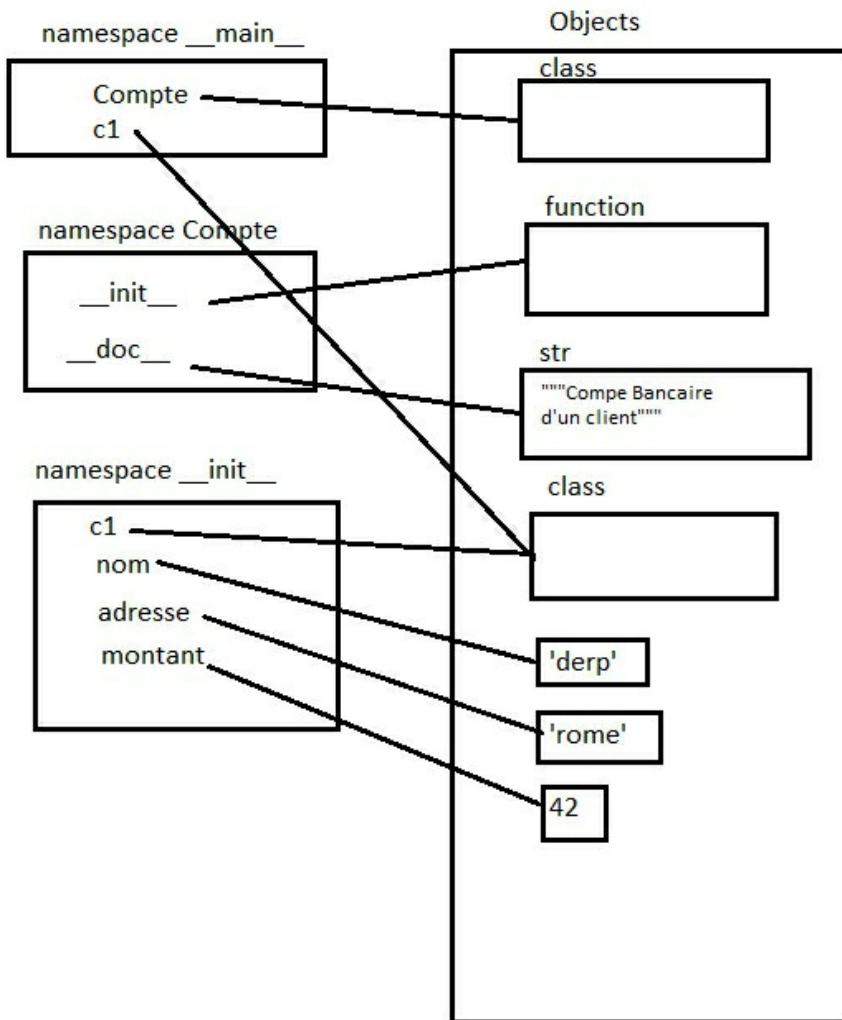
```
" Compte Bancaire d'un client "
```



```

>>> c1 = Compte('derp', 'rome', 42)
# def __init__(c1, nom, adresse, montant):
    c1.nom = nom
    c1.adresse = adresse
    c1.montant = montant
# def __init__(c1, 'derp', 'rome', 42):
    c1.nom = 'derp'
    c1.adresse = 'rome'
    c1.montant = 42

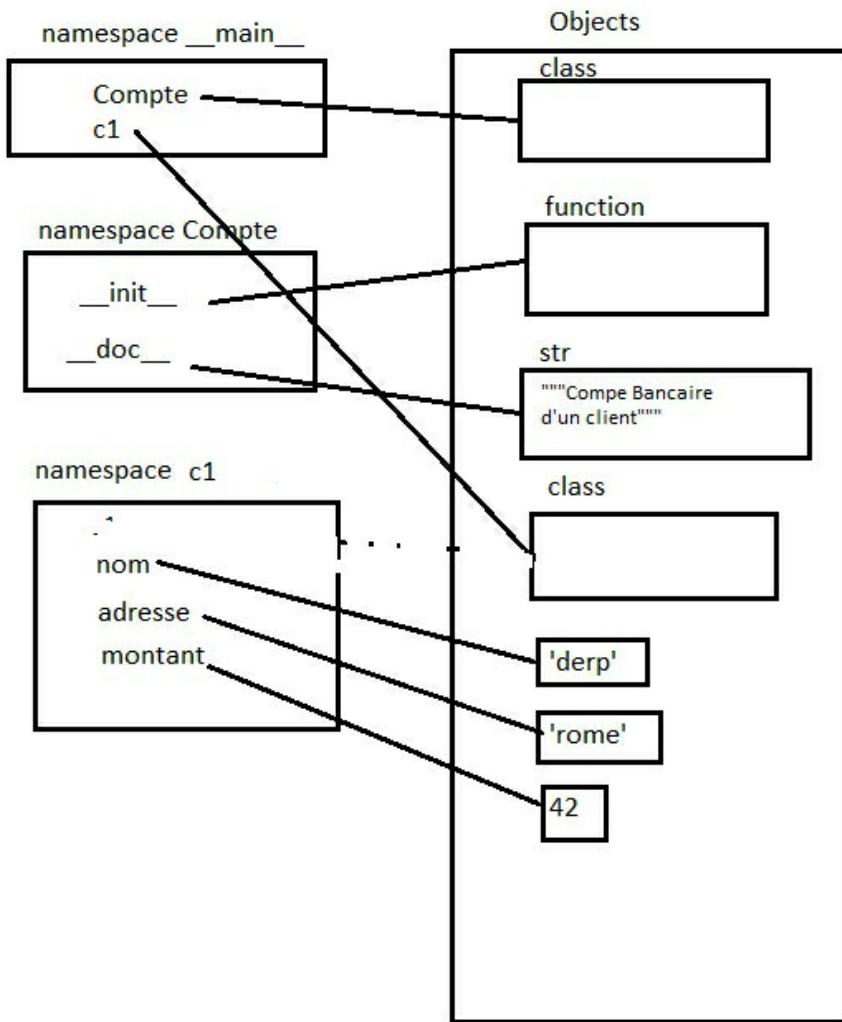
```



```

>>> c1.nom
'derp'
>>> c1.adresse
'rome'
>>> c1.montant
42
>>> c2 = Compte('derpina', 'tokyo', 100)
>>> print(c2.nom, c2.adresse, c2.montant)
derpina tokyo 100

```



>>> Exercise 384

1° Evaluate the following code

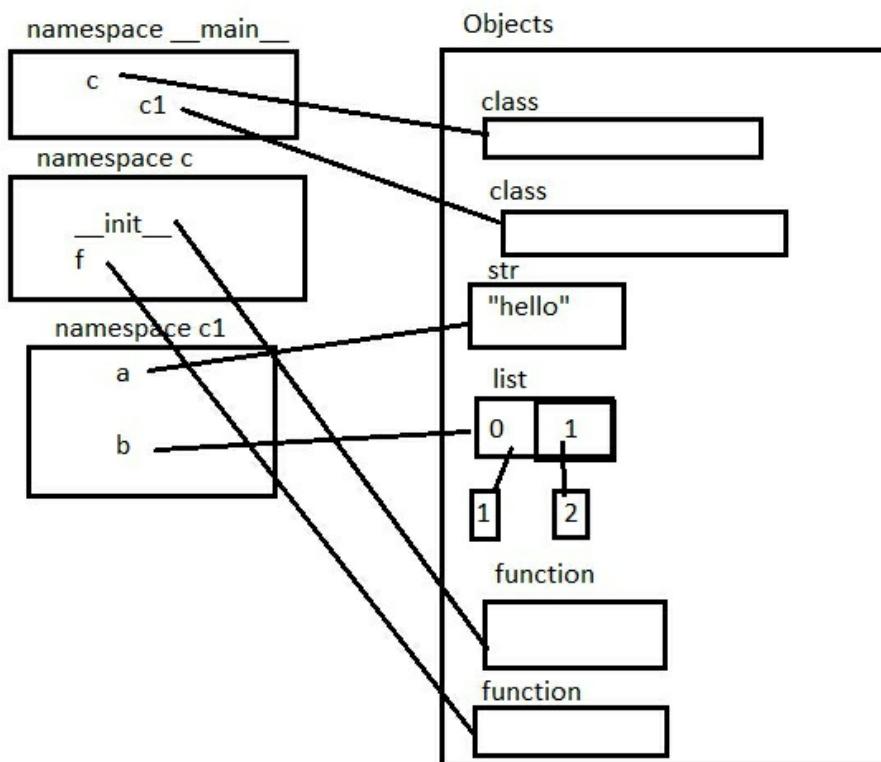
```
>>> class c():
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def f(c,d):
        return c + d
>>> c1=c("hello", [1,2])
```

2° Evaluate the following code

```
>>> c1.a  
>>> c1.b  
>>> c1.c = 'salut'
```

Solutions

1°



2°

```
>>> c1.a  
'hello'  
>>> c1.b  
[1,2]  
>>> c1.c = 'salut'  
>>> c1.c  
'salut'
```

>>> 34.Fonctions:partie 2

« soit $f(x)$ la fonction ... » nightmare

>>> 34.1.Function arguments

>>> Exercise 385

Evaluate the following code

```
>>> def f(a) :
```

```
    return a
```

```
>>> f(1)
```

```
>>> f('1')
```

```
>>> f([1])
```

```
>>> f((1,))
```

```
>>> f({ 'un':1})
```

```
>>> f({1})
```

```
>>> a = 1
```

```
>>> f(a)
```

Solutions

Function name : f

Argument : a

Return expression : a

```
>>> def f(a) :
```

```
    return a
```

```
>>> f(1)
```

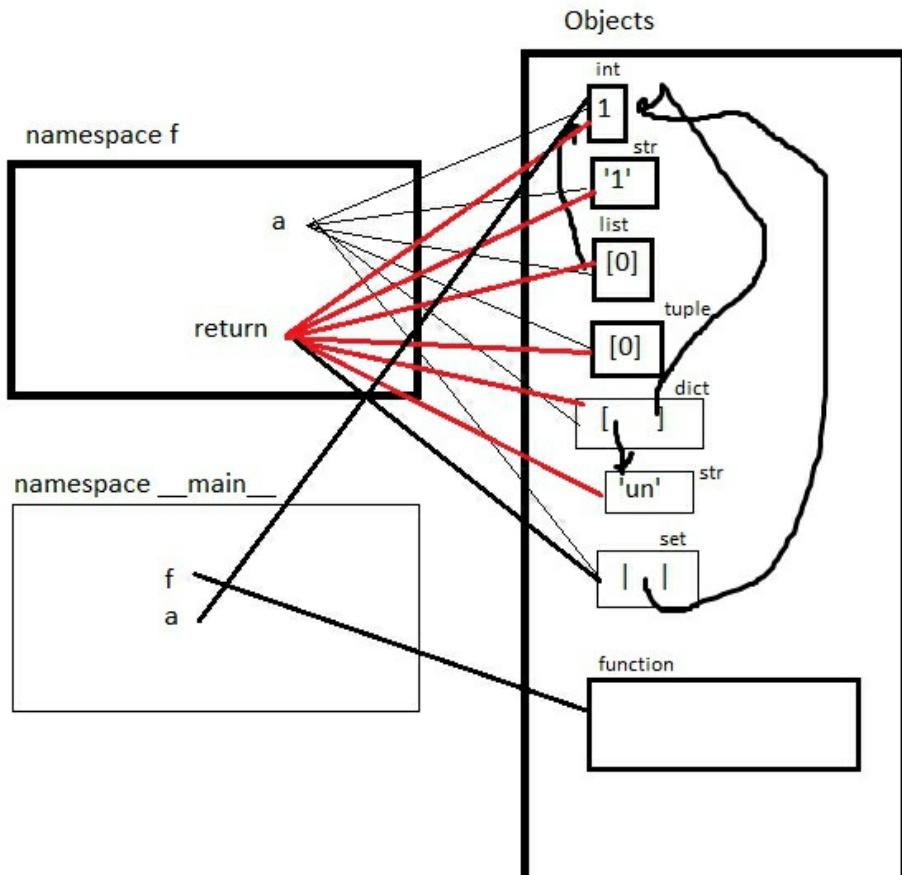
```
1
```

```
>>> f('1')
```

```
'1'
```

```
>>> f([1])
```

```
[1]
>>> f((1,))
(1,)
>>> f({'un':1})
{'un':1}
>>> f({1})
{1}
>>> a = 1
>>> f(a)
# f(1)
# return 1
# 1
1
```



>>> Exercice 386

Que fait le code suivant

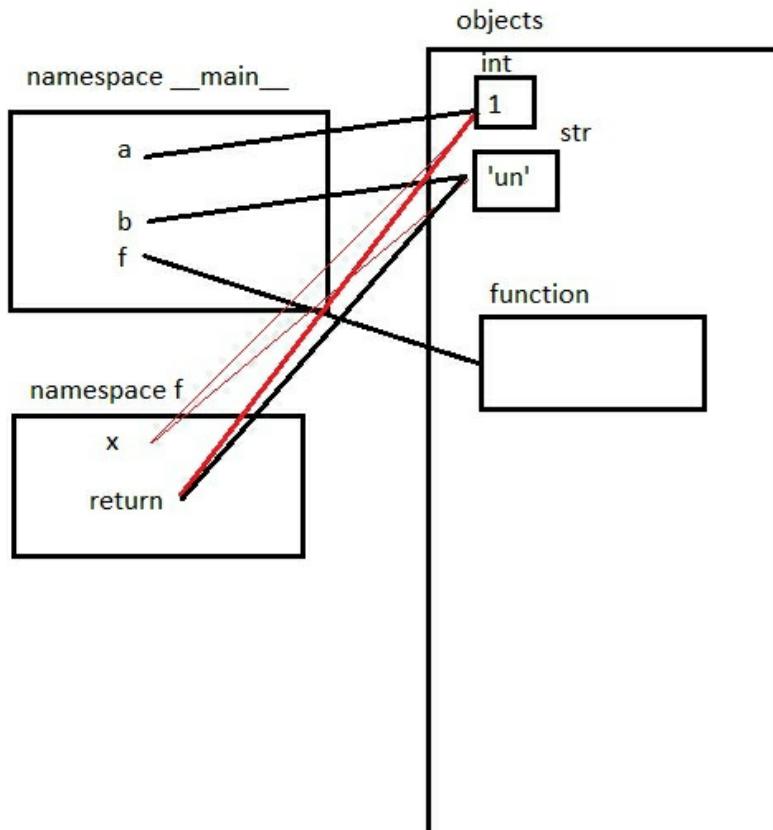
```
>>> a, b = 1, 'un'

>>> def f(x) :
    return x

>>> f(a)

>>> f(b)
```

Solutions



```

>>> f(a)
# f(1)
# return 1
1
>>> f(b)
# f('un')
# return 'un'
'un'

```

>>> 34.1.1. Non mutable arguments

« du mutagène » une tortue

>>> **Exercise 387**

1° Evaluate the following code

```
>>> def f(x) :
```

```
    return x
>>> f(1)
>>> f(1+1)
>>> f(1*4)
>>> f(2/0)
>>> f()
```

2° Evaluate the following code

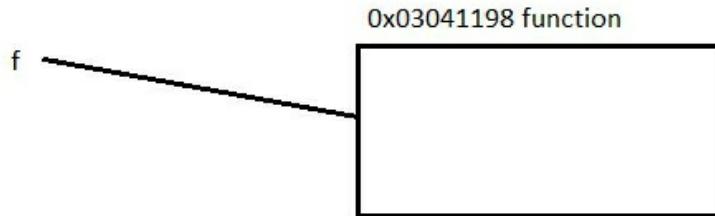
```
>>> f('1')
>>> f('1' + '1')
>>> f('1' + 1)
```

Solutions

1°

```
>>> def f(x) :
    return x
>>> f(1)
# return 1
# 1
1
>>> f(1+1)
# f(1+1)
# f(2)
# return 2
# 2
2
>>> f(1*4)
# f(1*4)
```

```
# f(4)
# return 4
# 4
4
>>> f(2/0)
# f(2/0)
# ZeroDivisionError: division by zero
ZeroDivisionError: division by zero
>>> f()
TypeError: f() missing 1 required positional argument: 'x'
>>> f
<function f at 0x03041198>
```



2°

```
>>> f('1')
'1'
>>> f('1'+'1')
'11'
>>> f('1'+1)
TypeError: can only concatenate str (not "int") to str
```

Tips

```
>>> expression
```

```
value
```

```
>>> f(expression)
# f(value)
```

>>> 34.1.2.Mutables arguments

>>> 34.1.2.1.A list as an argument

>>>**Exercise 388**

Evaluate the following code

1°

```
>>> def f(x) :
    return x
>>> f([1,2])
```

2° Evaluate the following code

```
>>> def f(x) :
    i=0
    while(i<2) :
        x.append(i)
        i=i+1
    return x
```

3° Evaluate the following code

```
>>> f([])
>>> f([1])
```

Solutions

1°

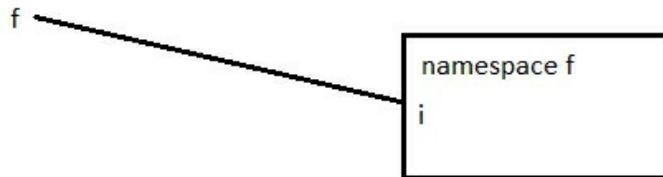
```
>>> def f(x) :
    return x
>>> f([1,2])
# f([1,2])
# return [1, 2]
```

[1, 2]

2° On a

a) i est définie dans le block de f c'est une variable locale

b) On a

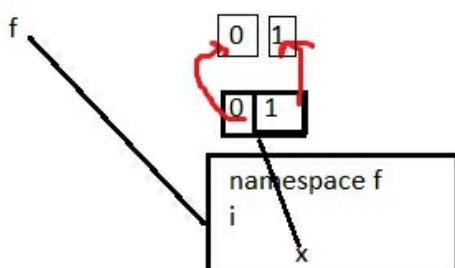


On a

```
iteration x = []    i
1      x=[0]    0
2      x=[0,1]  1
```

```
iteration x =[1] i
```

```
1      x=[1,0] 0
2      x=[1,0,1] 1
```



```
>>> f([])
```

```
[0,1]
```

```
>> f([1])
```

```
[1,0,1]
```

>>> **Exercice 389**

1° Evaluate the following code

```
>>> L = []
```

```
>>> def f(x) :
```

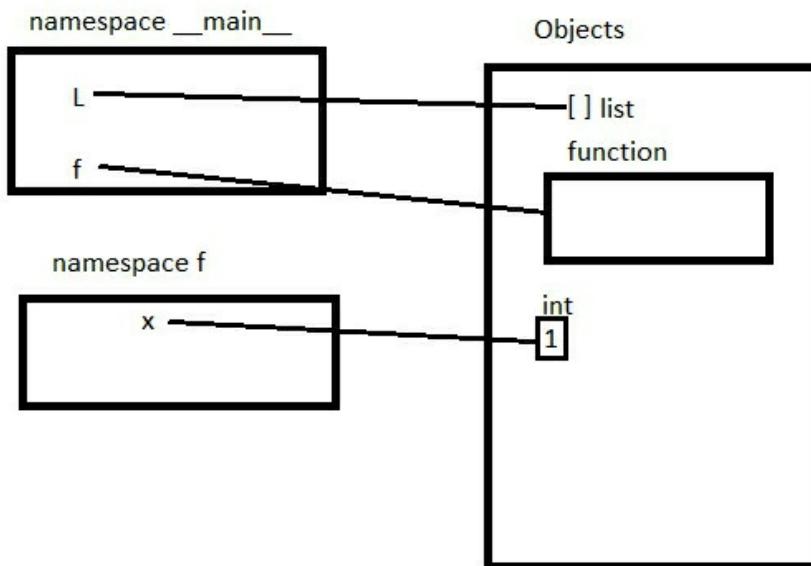
```
for i in range(2):  
    x.append(i)  
>>> f(1)
```

2° Evaluate the following code

```
>>> f(L)  
>>> L
```

Solutions

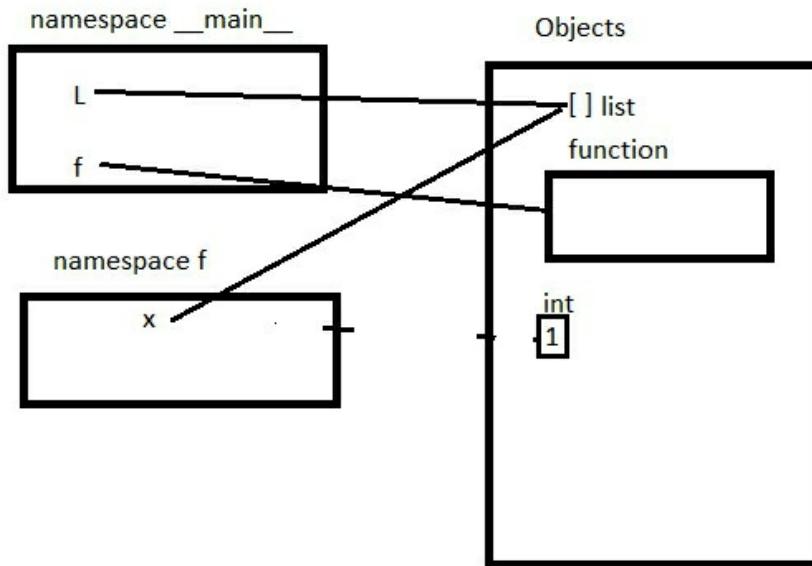
1°



```
>>> f(1)  
# 1.append(1)  
# AttributeError: 'int' object has no attribute 'append'  
AttributeError: 'int' object has no attribute 'append'
```

Int object hasn't method append

2° side effect



Iteration $x = L$ $x.append(i)$ $x = L$

1 $x = L$ $[].append(0)$ $[0]$ $[0]$

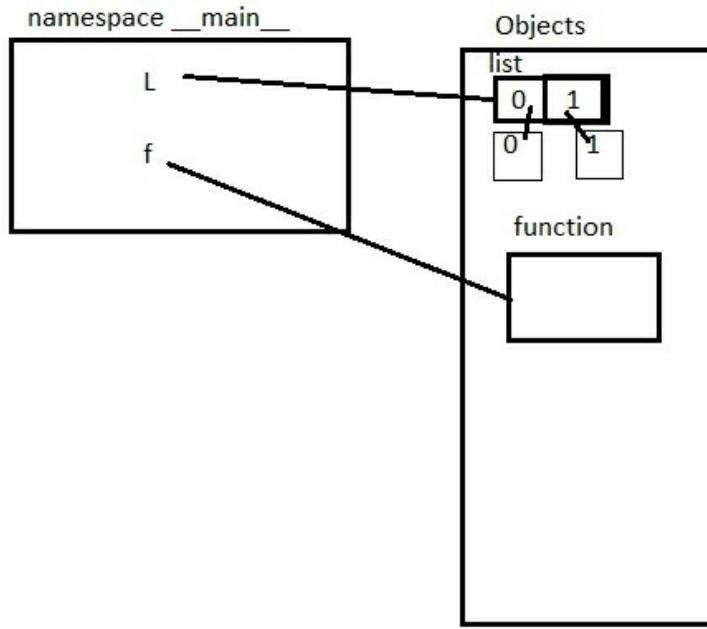
2 $x = L$ $[0].append(1)$ $[0,1]$ $[0,1]$

>>> L

[0, 1]

>>> x

NameError : name 'x' is not defined



>>> 34.1.2.2.A dictionary as argument

>>> **Exercise 390**

1° Evaluate the following code

```
>>> def f(x) :
    return x + 1
>>> f({'dico':1})
```

2° Evaluate the following code

```
>>> def f(x) :
    return {'dico' : x}
>>> f(1)
>>> f('11')
>>> f({'mot' : 42})
```

Solutions

1° function name : f

Argument : x

Return expression : x + 1

```

>>> def f(x) :
    return x + 1
>>> f({'dico':1})
# f({'dico':1})
# return {'dico':1} + 1
# TypeError: unsupported operand type(s) for +: 'dict' and 'int'
TypeError: unsupported operand type(s) for +: 'dict' and 'int'

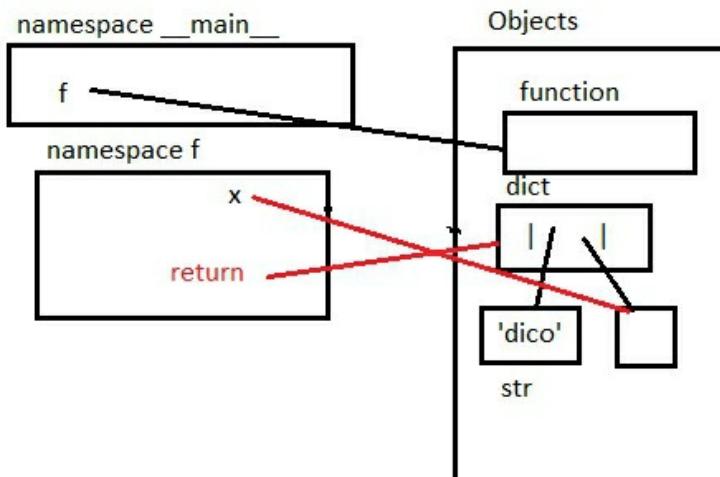
```

{'dico':1} + 1 is not a valid expression

2° function name : f

Argument : x

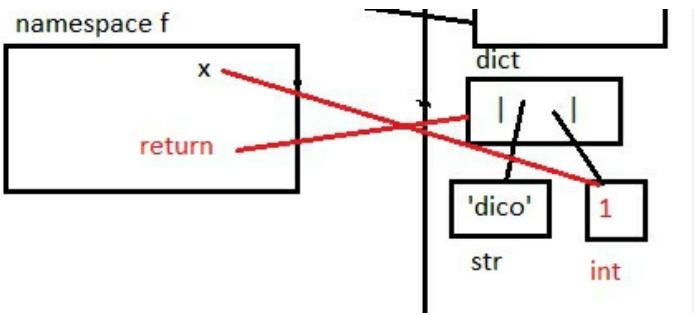
Return expression : { 'dico': x }



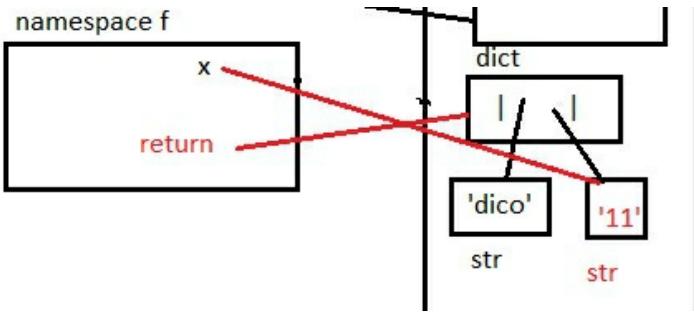
```

>>> def f(x) :
    return {'dico' : x}
>>> f(1)
# return {'dico' : 1}
{'dico':1}

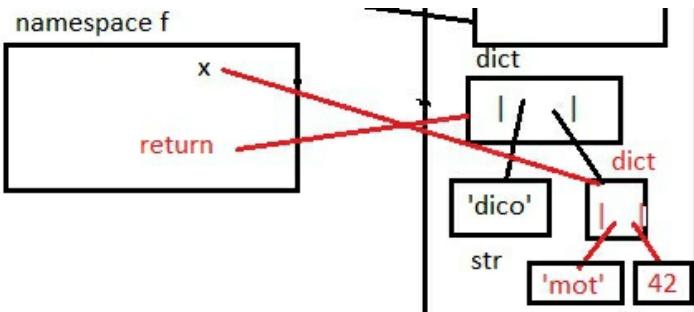
```



```
>>> f('11')
# return {'dico':11}
{'dico':11}
```



```
>>> f({'mot' : 42})
# return {'dico' : {'mot':42}}
{'dico' : {'mot':42}}
```



>>> **Exercise 391**

Evaluate the following code

```
>>> def f(x) :
    return {x : 108 }

>>> f(1)
>>> f([1,2])
```

```
>>> f('1')
```

Solutions

```
>>> f(1)
```

```
# return {1:108}
```

```
{1:108}
```

```
>>> f([1,2])
```

```
# return {[1,2]:108}
```

```
# TypeError: unhashable type: 'list'
```

```
TypeError: unhashable type: 'list'
```

```
>>> f('1')
```

```
# return {'1':108}
```

```
{'1':108}
```

The dictionary key must be an immutable object, list isn't.

>>> 34.2.Default arguments and keyword arguments

« votre plus grand défaut ? nommez-le ! »

>>> 34.2.1.Default arguments

>>> **Exercise 392**

1° Evaluate the following code

```
>>> def f(a, b) :
```

```
    return a
```

```
>>> f(1)
```

2° Evaluate the following code

```
>>> def f(a, b = 'hello') :
```

```
    return a
```

```
>>> f(1)
```

Solutions

1°

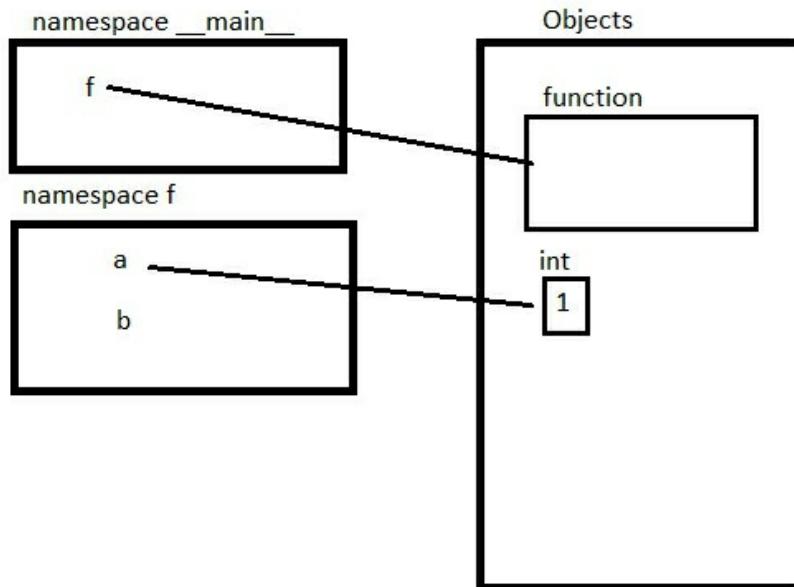
```
>>> def f(a, b):
```

```
    return a
```

```
>>> f(1)
```

```
TypeError: f() missing 1 required positional argument: 'b'
```

Even if b does not appear in block function, he's required as positional argument



2° a) b s'appelle un **argument par défaut**

```
>>> def f(a, b = 'hello'):
```

```
    return a
```

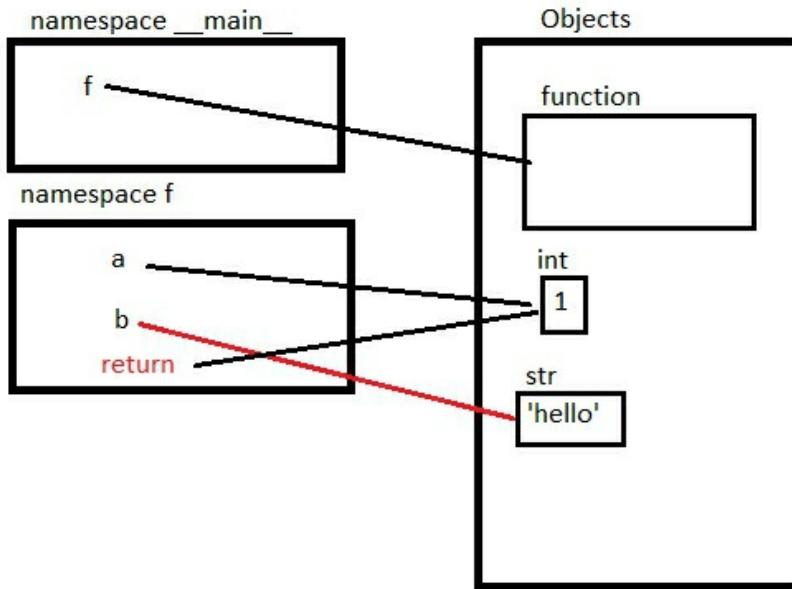
```
>>> f(1)
```

```
# def f(1, b='hello'):
```

```
# return 1
```

```
# 1
```

```
1
```



Tips

```
>>> def f(argument1, argument2 = default_value) :
    block
```

>>> Exercise 393

1° Evaluate the following code

```
>>> def f(a, b= 'world') :
    return a + b
>>> f('hello')
>>> f('hello', 'python')
```

2° Evaluate the following code

```
>>> f(1)
>>> f(1,2)
```

3° Evaluate the following code

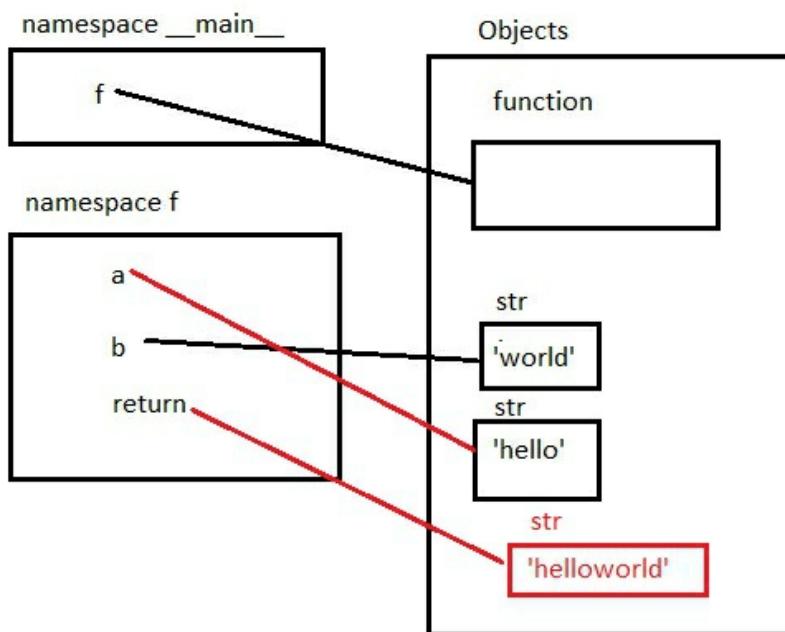
```
>>> f([1])
>>> f([1],[2])
```

Solutions

1°

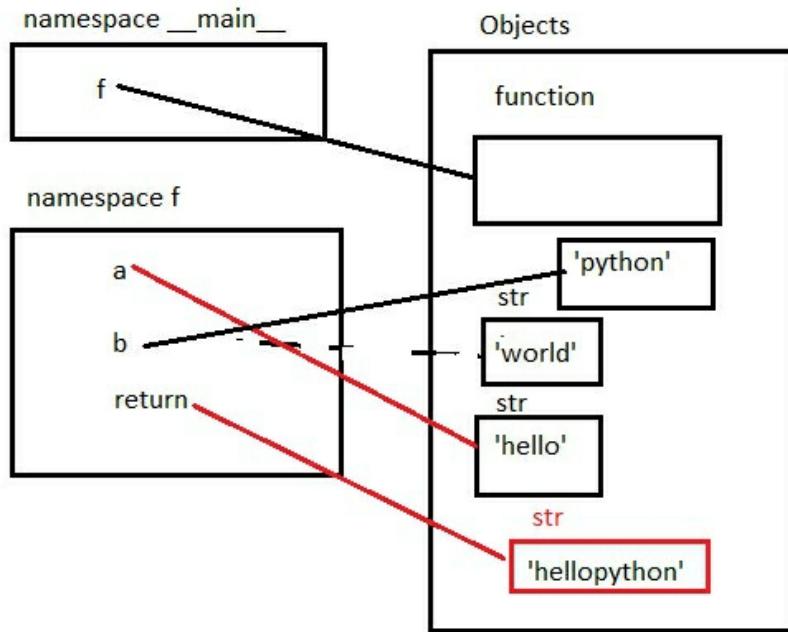
```
>>> f('hello')
# f('hello', b = 'world')
# return 'hello' + b
# return 'hello' + 'world'
# return 'helloworld'
```

'helloworld'



```
>>> f('hello', 'python')
# f('hello', 'python')
# return 'hello' + 'python'
# return 'hellopython'
```

'hellopython'



2°

```

>>> f(1)
# f(1, b='world')
# return 1 + 'world'

# TypeError: unsupported operand type(s) for +: 'int' and 'str'
TypeError: unsupported operand type(s) for +: 'int' and 'str'

1 + 'world' isn't a valid expression

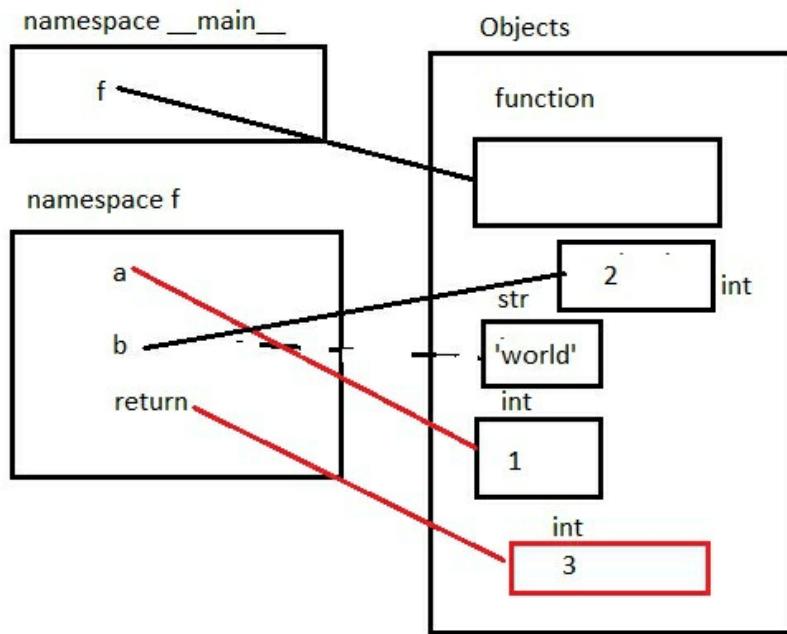
```

```

>>> f(1,2)
# f(1, 2):
# return 1 + 2
# return 3

```

3



3°

```
>>> f([1])
# f([1], b = 'world')
# return [1] +'world'
# TypeError: can only concatenate list (not "str") to list
TypeError: can only concatenate list (not "str") to list
```

[1] +'world' isn't a valid expression

```
>>> f([1],[2])
# f([1], [2])
# return [1] + [2]
# return [1,2]
```

[1,2]

>>> **Exercise 394**

Evaluate the following code

```
>>> def f(a='hello', b):
```

```
    return a + b
>>> def f(a, b = 'hello') :
    return a + b
>>> f()
```

Solutions

```
>>> def f(a='hello', b) :
    return a + b
```

SyntaxError: non-default argument follows default argument

```
>>> def f(a, b = 'hello') :
    return a + b
>>> f()
```

TypeError: f() missing 1 required positional argument: 'a'

>>> **Exercise 395**

Evaluate the following code

```
>>> def f(a, b = 'hello')
    if a == 0 :
        return 'salut'
    else :
        return b
>>> f()
>>> f(1)
>>> f(0)
```

Solutions

```
>>> f()
```

TypeError: f() missing 1 required positional argument: 'a'

```
>>> f(1)
```

```
# if 1 == 0 :
```

```
# if False :  
# else :  
# return b  
# return 'hello'  
'hello'  
>>> f(0)  
# if 0 == 0 :  
# if True :  
# return 'salut'  
'salut'
```

>>> 34.2.1.1.Mutable default argument

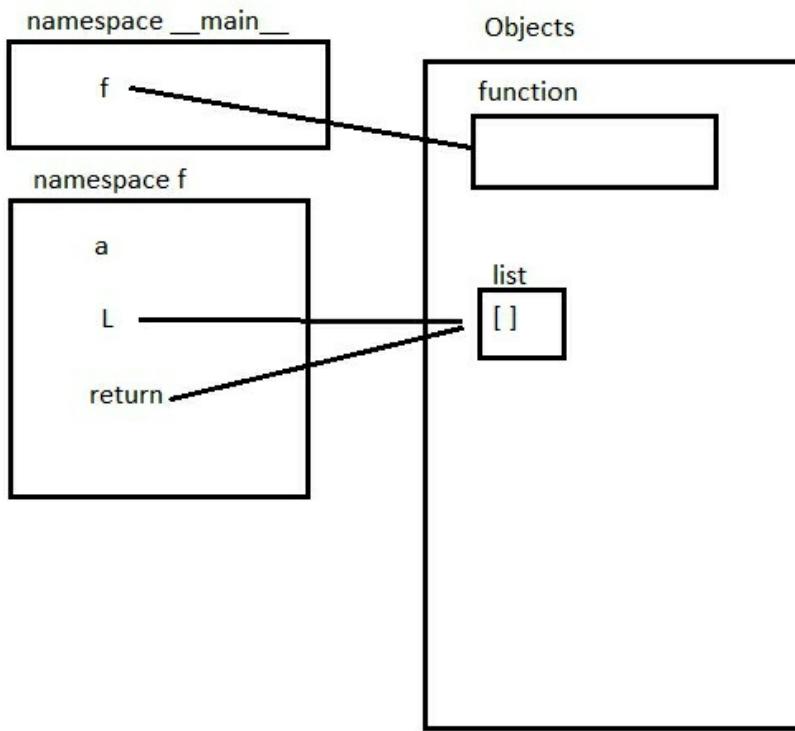
>>> **Exercise 396 (from Python doc)**

Evaluate the following code

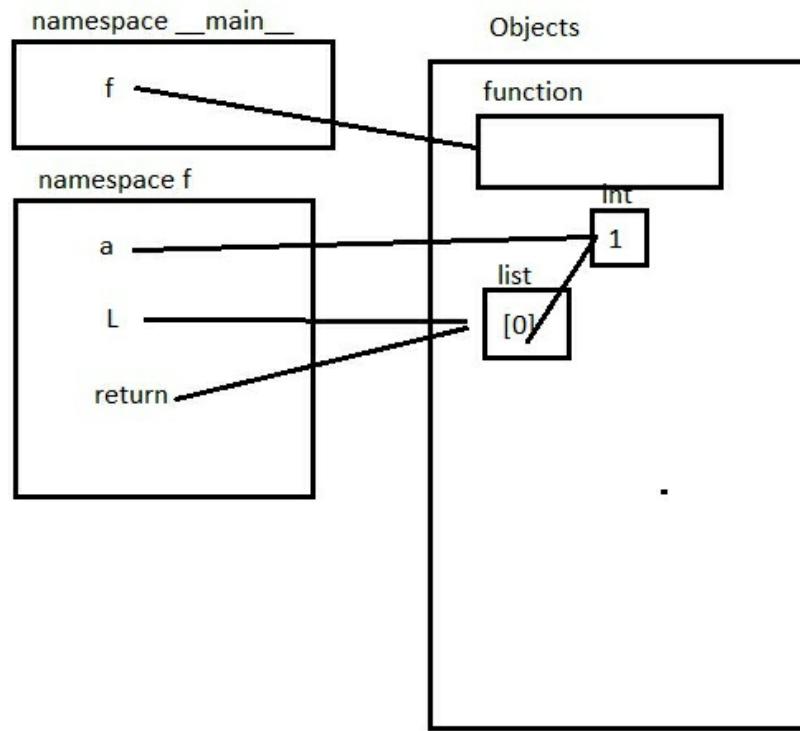
```
>>> def f(a, L=[]):  
    L.append(a)  
    return L  
>>> f(1)  
>>> f(2)  
>>> f(3)
```

Solutions

There will be a side effect



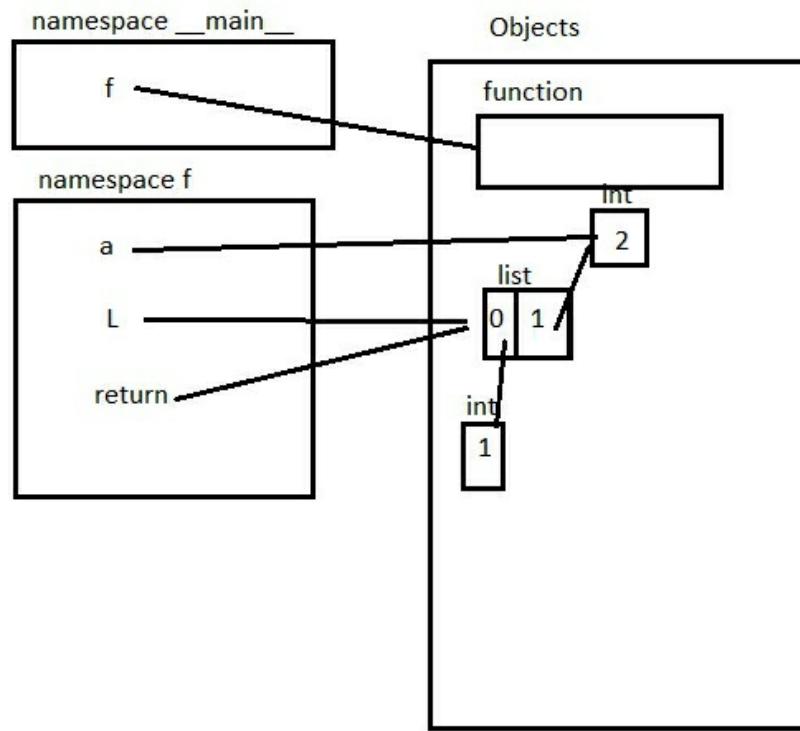
```
>>> f(1)
# f(1, L = []):
# [].append(1)
# [1]
# return [1]
[1]
```



```

>>> f(2)
# f(2, L = [1]):
# [1].append(2)
# [1, 2]
# return [1, 2]
[1, 2]

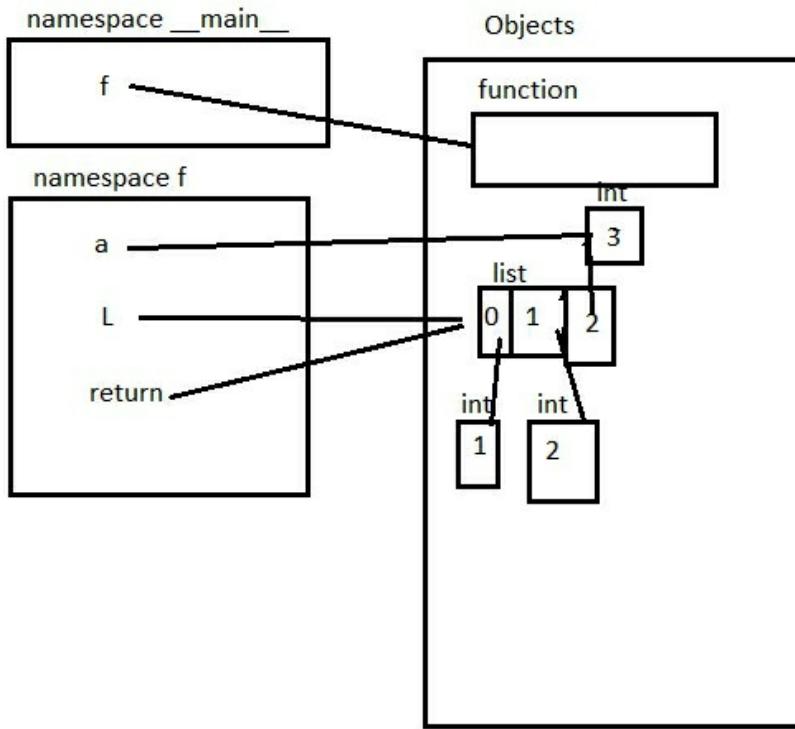
```



```

>>> f(3)
# f(3, L = [1, 2]) :
# [1, 2].append(3)
# [1, 2, 3]
# return [1, 2, 3]
[1, 2, 3]

```

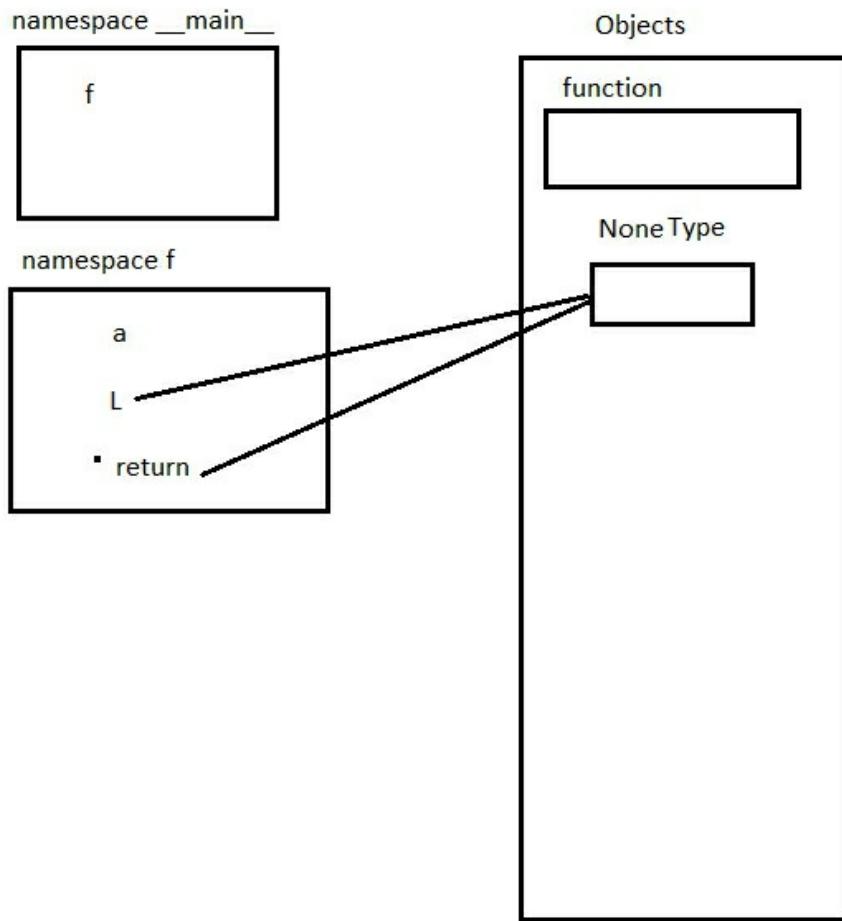


>>> **Exercise 397 (from python doc)**

Que fait le code suivant

```
>>> def f(a, L= None) :
    if L == None :
        L = []
    L.append(a)
    return L
>>> f(1)
>>> f(2)
>>> f(3)
```

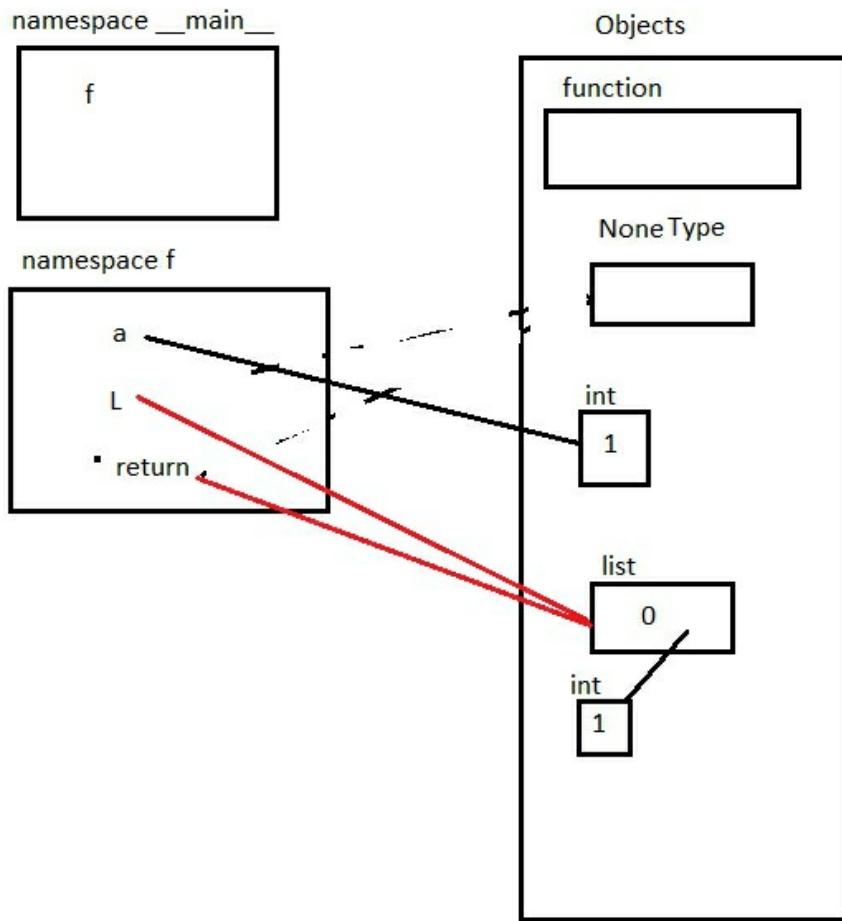
Corrigé exercice 397



```

>>> f(1)
# f(1, L= None) :
# if L == None :
# if None == None :
# if True :
# L = []
# [].append(1)
# [1]
# return [1]
[1]

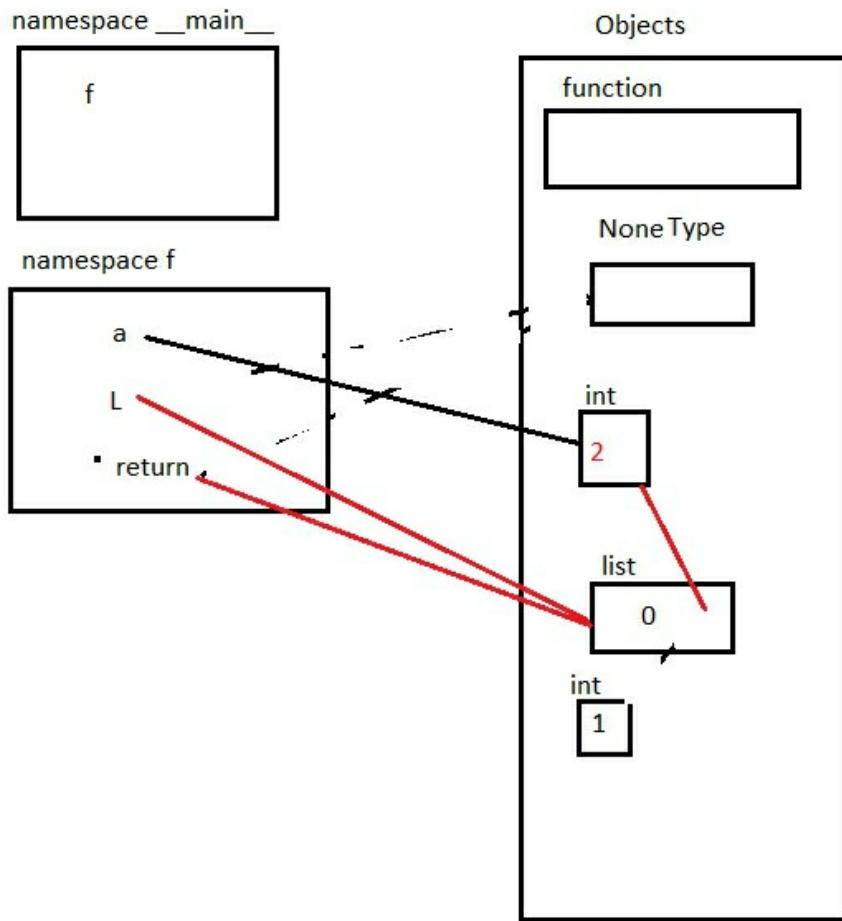
```



```

>>> f(2)
# f(2, L = None) :
# if L == None :
# if None == None :
# if True :
# L = []
# [].append(2)
# [1]
# return [2]
[2]

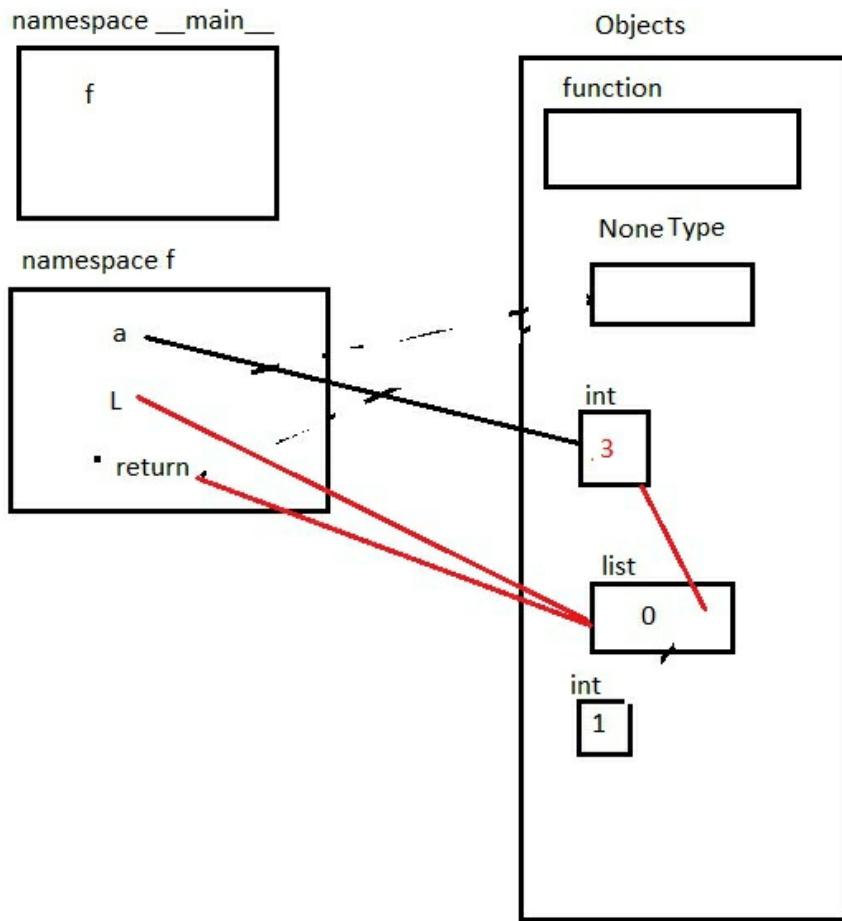
```



```

>>> f(3)
# f(3, L = None) :
# if L == None :
# if None == None :
# if True :
# L = []
# [].append(3)
# [1]
# return [3]
[3]

```



>>> 34.2.2.keyword arguments

>>> Exercise 398

1° Evaluate the following code

```
>>> def f(a= 'un', b= 'deux') :
    return a + b
>>> f()
```

2° Evaluate the following code

```
>>> f(a = 'deux ', b= 'un ')
```

3° Evaluate the following code

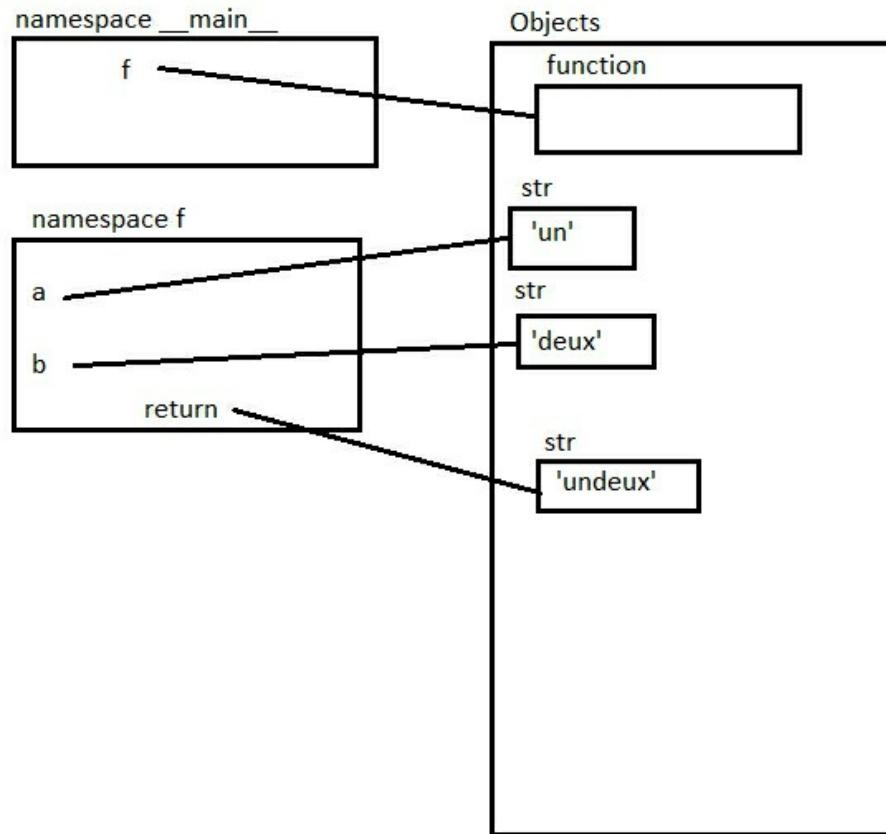
```
>>> f(b= 'deux ', a = 'un ')
```

4° Evaluate the following code

```
>>> f(c= 'un', d= 'deux')  
>>> f('three', 'four')
```

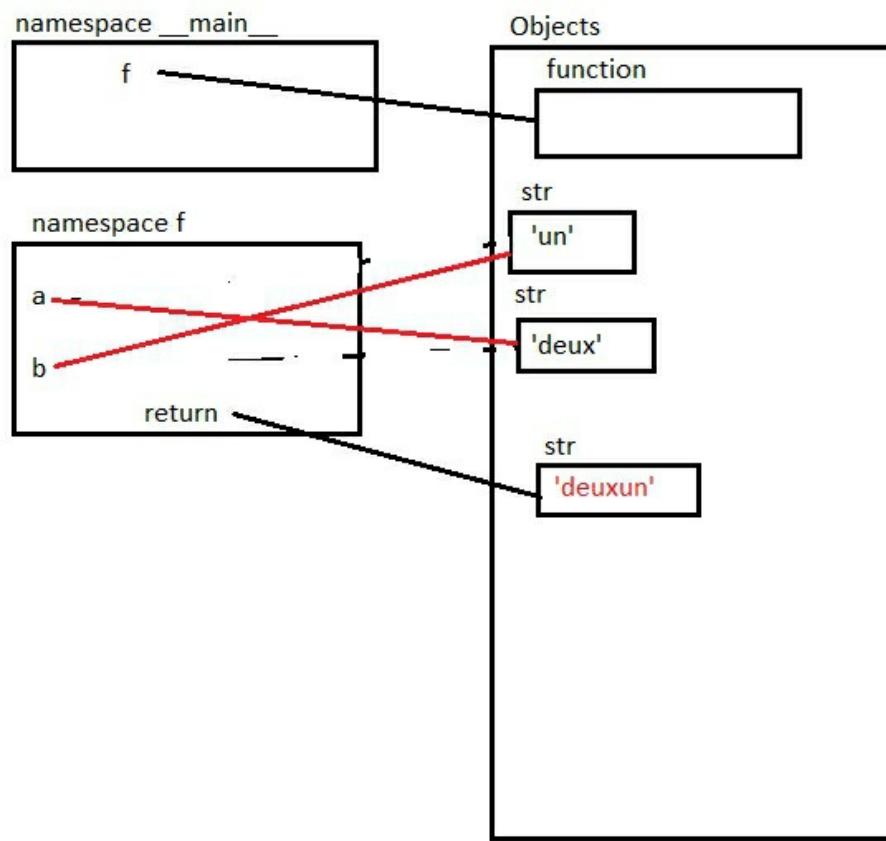
Solutions

1°



```
>>> f()  
# f('un', 'deux'):  
# return 'un' + 'deux'  
# return 'undeux'  
'undeux'
```

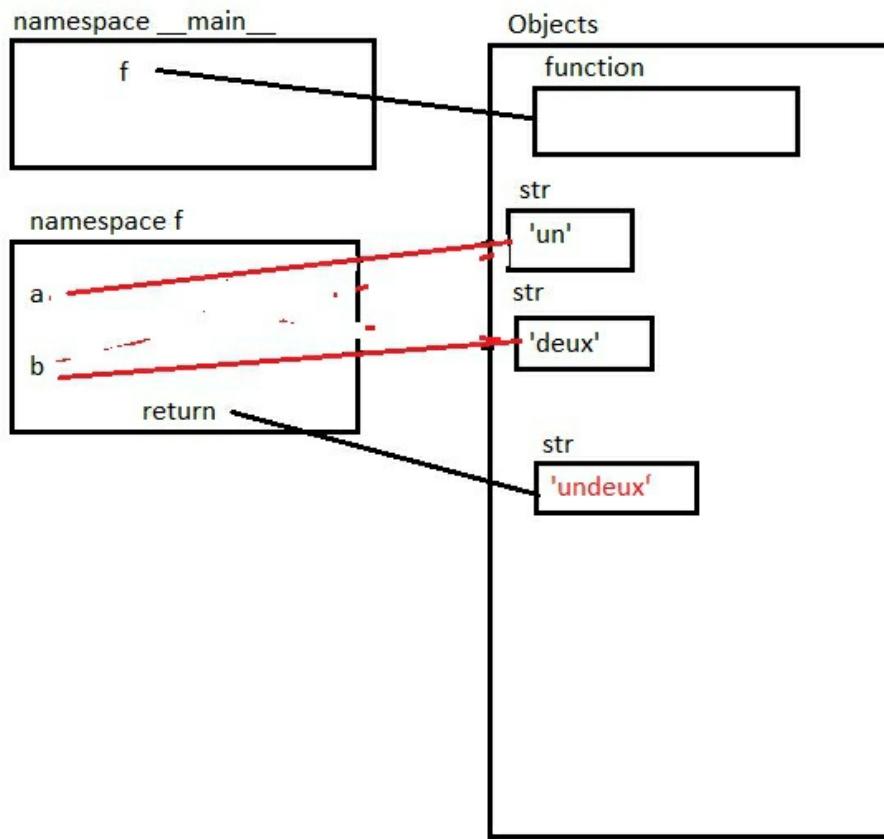
2°



```
>>> f(a = 'deux ', b= 'un')
# f(a = 'deux', b= 'un')
# return 'deux' + 'un'
# return 'deuxun'
```

'deuxun'

3°



```
>>> f(b= 'deux', a = 'un ')
# return a + b
# return 'un' + 'deux'
# return 'undeux'
'undeux'
```

4°

```
>>> f(c= 'un', d= 'deux')
TypeError: f() got an unexpected keyword argument 'c'
>>> f('three', 'four')
# return 'three' + 'four'
'threefour'
```

Tips

```
>>> def f(argument1 = value1, argument2 = value2, ..., argumentn =  
valuen):  
    block
```

we can call the way we want the arguments with the value we want

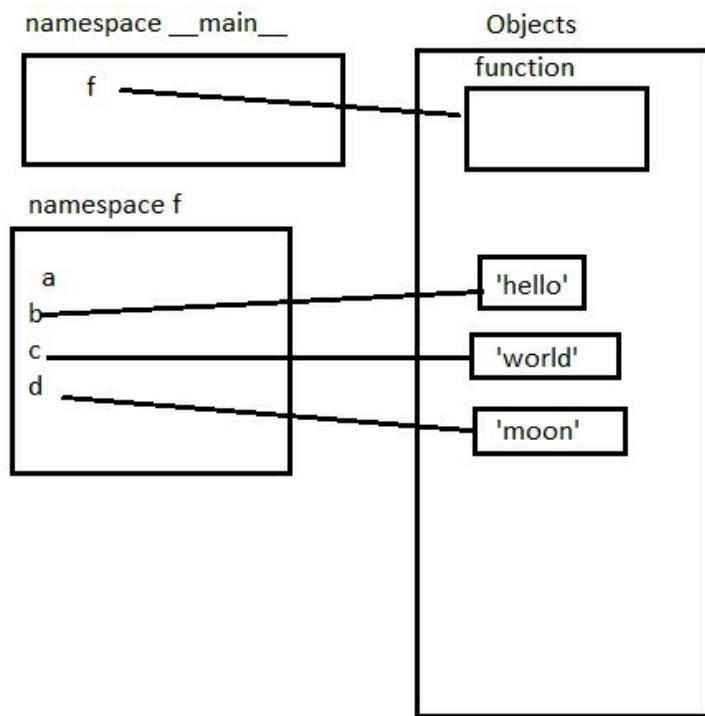
```
>>> f(argumentn = value1, argument2 = valuen, ..., argument1 = value3)
```

>>> Exercise 399

Evaluate the following code

```
>>> def f(a, b= 'hello ', c= 'world ', d= 'moon '):  
    print(c)  
    print(a)  
    print(d)  
    print(b)  
>>> f(1000)  
>>> f(a=1000)  
>>> f(a= 'hello', c= 'zorglub')  
>>> f()
```

Solutions

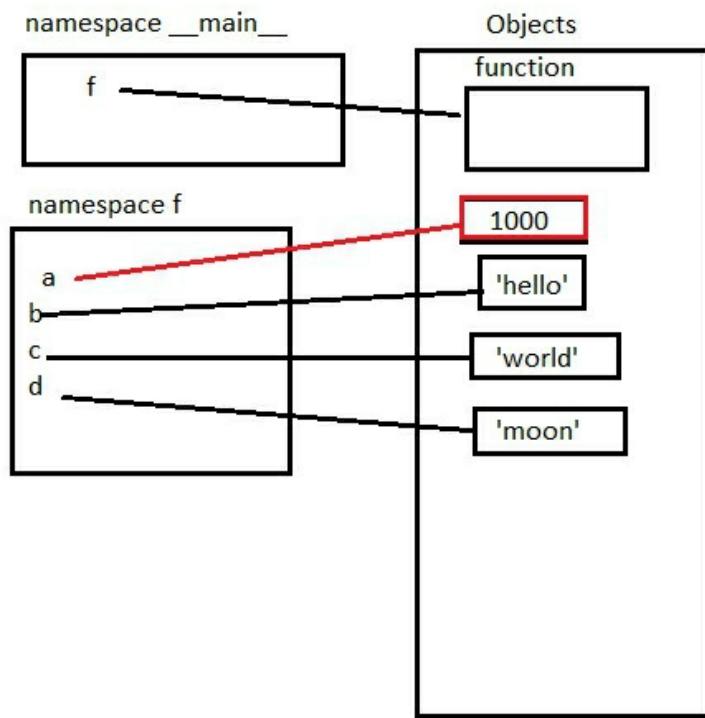


```
>>> f(1000)
# f(1000 , b= 'hello', c= 'world', d= 'moon')
```

```
world
1000
moon
hello
```

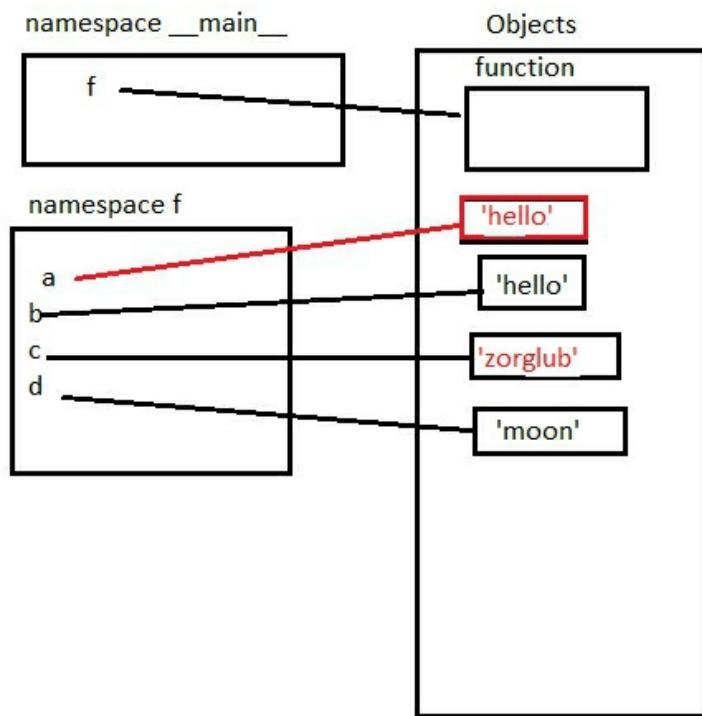
```
>>>f(a=1000)
# f(a = 1000, b= 'hello', c= 'world', d= 'moon')
# f (1000, b= 'hello', c= 'world', d= 'moon')
```

```
world
1000
moon
hello
```



```
>>> f(a= 'hello ', c= 'zorglub ')
# f(a= 'hello ', b= 'hello ', c= 'zorglub ', d= 'moon ')
```

```
zorglub
hello
moon
hello
```



```

>>> f()
# f( , b= 'hello ' , c= 'zorglub ' , d= 'moon ')
TypeError: f() missing 1 required positional argument: 'a'

```

>>> **exercise 400**

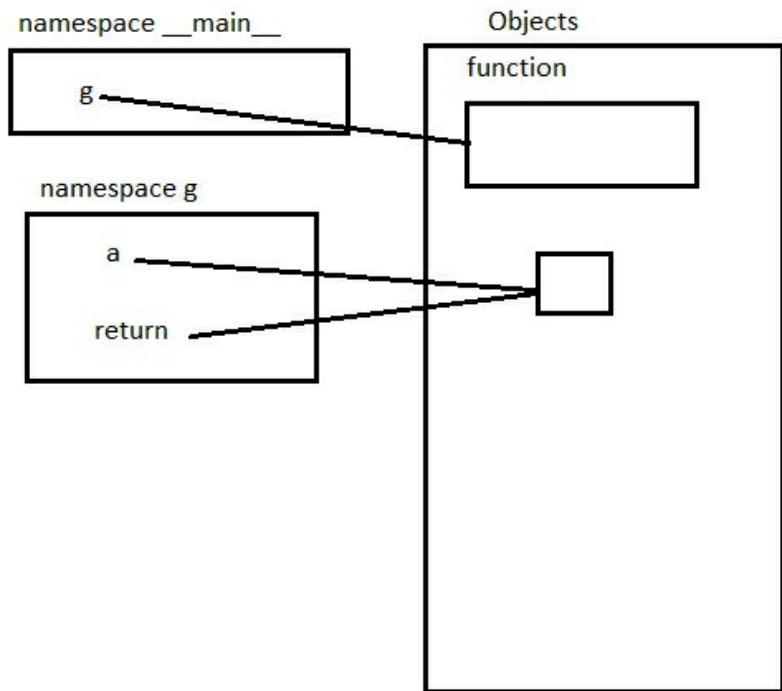
Evaluate the following code

```

>>> def g(a) :
    return a
>>> g(1)
>>> g(b=1)
>>> g(a= '1')
>>> g(b=1)

```

Solutions



```
>>> g(1)
# return 1
1
>>> g(a=1)
# return 1
1
>>> g(a= '1 ')
# g('1 ')
# return '1 '
'1'
>>> g(b=1)
TypeError: g() got an unexpected keyword argument 'b'
```

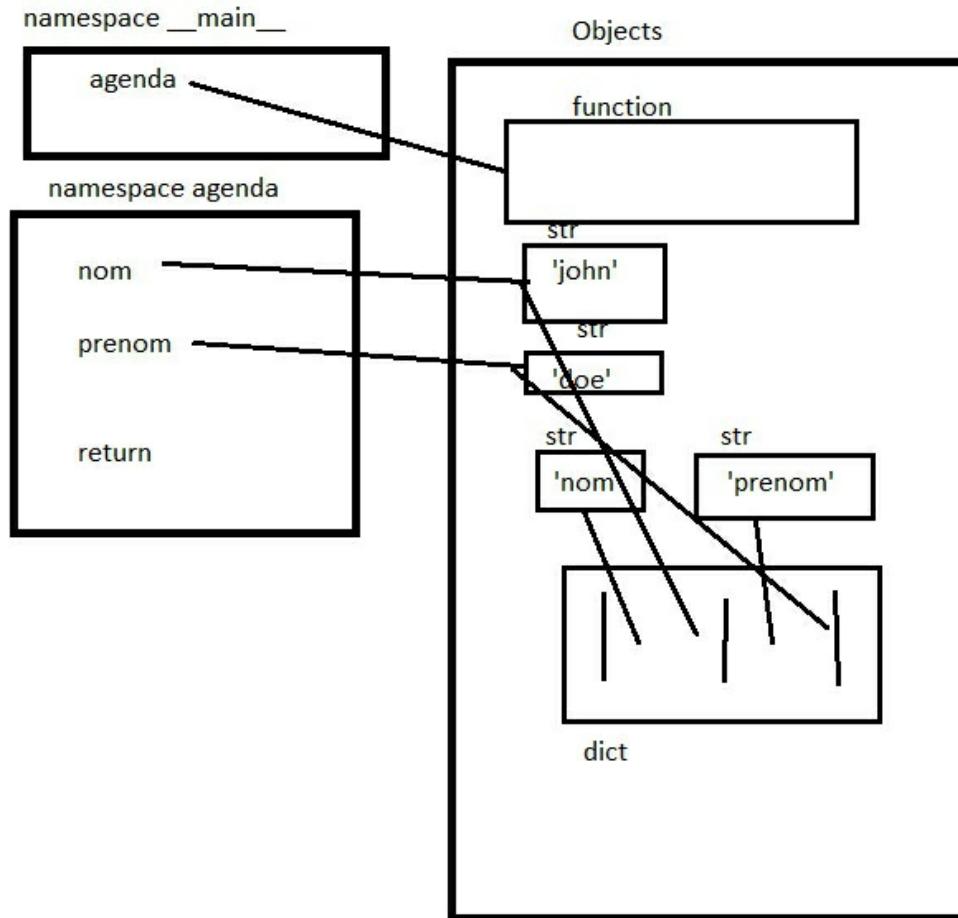
>>> Exercise 401

```

>>> def agenda(nom = 'john' , prenom = 'doe') :
    return { 'nom' : nom , 'prenom' : prenom}
>>> agenda()
>>> agenda('python', 3)

```

Solutions

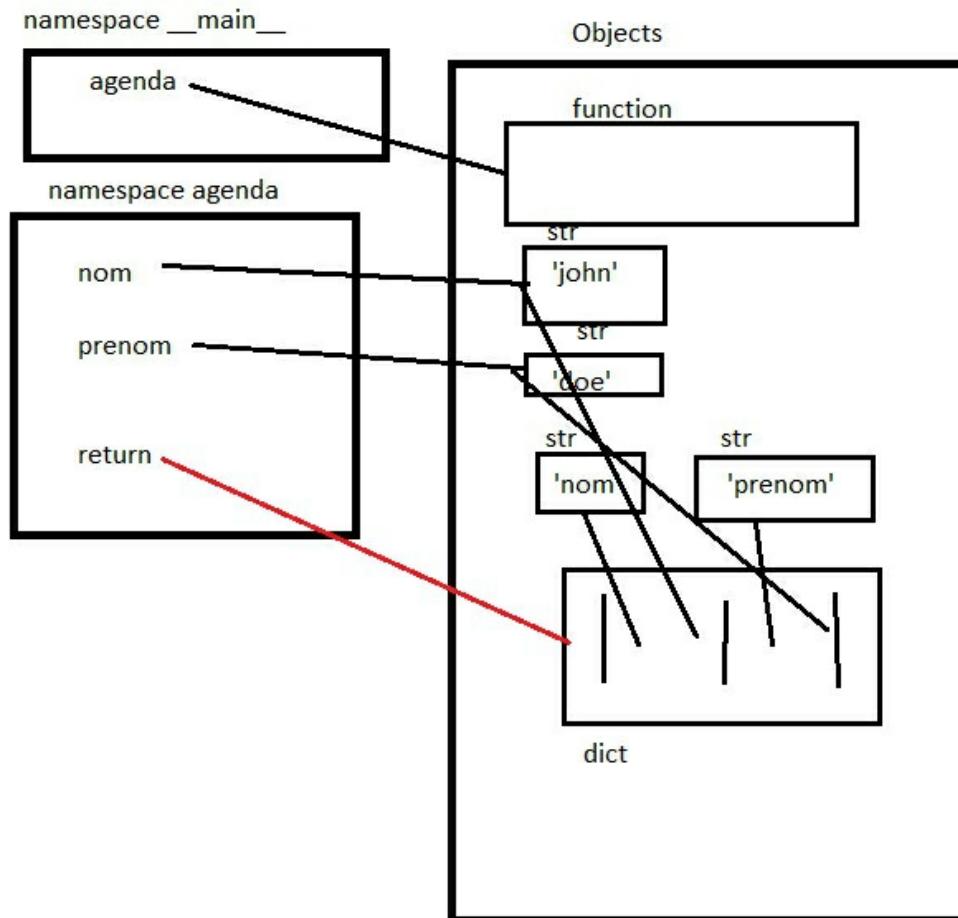


```

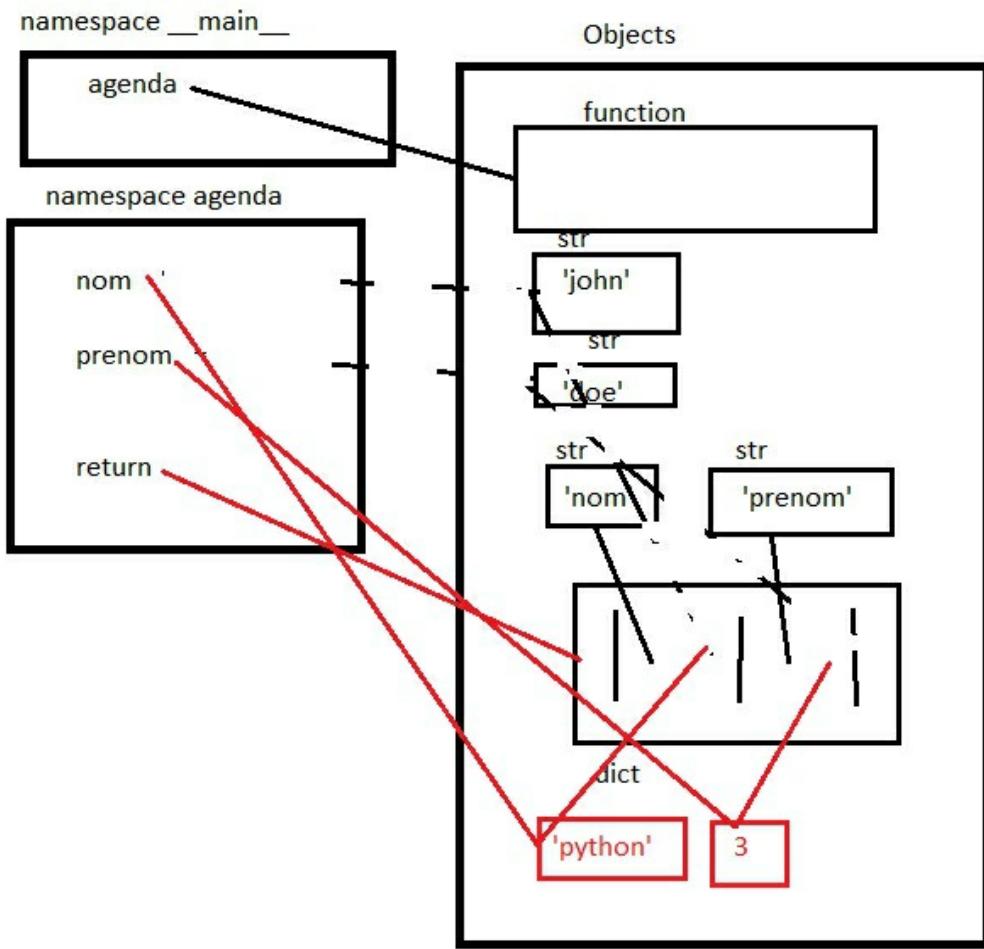
>>> def agenda(nom = 'john' , prenom = 'doe') :
    return { 'nom' : nom , 'prenom' : prenom}
>>> agenda()
# agenda (nom = 'john' , prenom= 'doe ')
# return { 'nom ' : 'john ' , 'prenom ' = 'doe ' }

```

```
{ 'nom' : 'john', 'prenom' = 'doe' }
```



```
>>> agenda('python', 3)  
# return { 'nom' : 'python', prenom = '3 '}  
{ 'nom' : 'python', 'prenom' = '3 '}
```



>>> Exercise 402

Evaluate the following code

```
>>> def f(x) :
    return x+1

>>> x = 1

>>> f(1)

>>> x
```

Solutions

```
>>> f

<function f at 0x03E5C390>
```

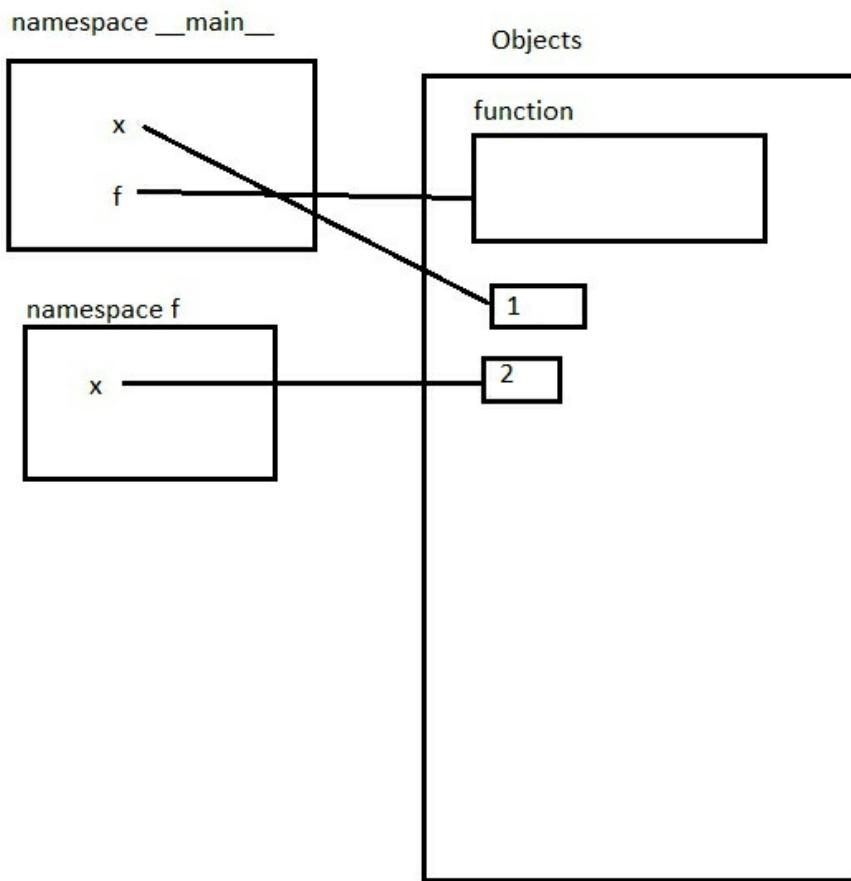
```
>>> x  
1  
>>> f(1)  
# f(x) :  
# f(1) :  
# return 1+1  
# return 2  
2  
>>> x  
1
```

>>> **Exercise 403**

Evaluate the following code

```
>>> x = 1  
>>> def f():  
    x = 2  
    print(x)  
>>> x
```

Solutions



```

>>> x = 1
>>> def f():
>>>     x = 2
>>>     print(x)
>>> x
1
>>> f()
# x = 2
# print(x)
# 2
2

```

>>> Exercise 404

1° Evaluate the following code

```
>>> def f(x):
```

```
    return x + 1
```

```
>>> g = f
```

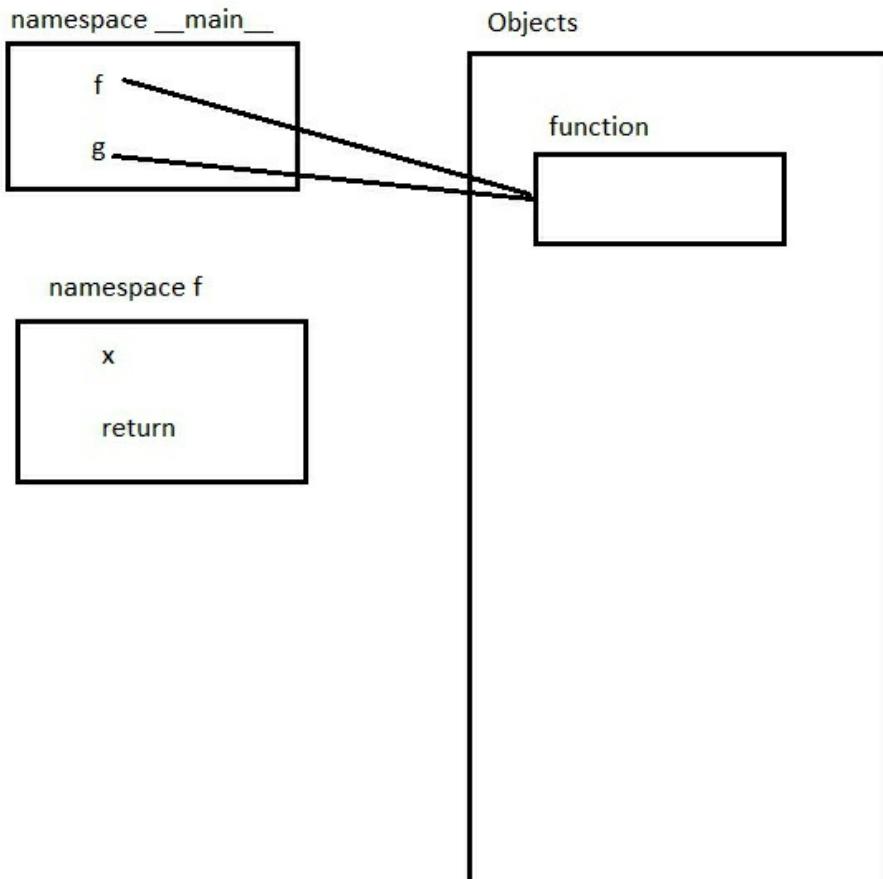
2° Evaluate the following code

```
>>> f(3)
```

```
>>> g(3)
```

Solutions

1° variables f and g refer to the same function object



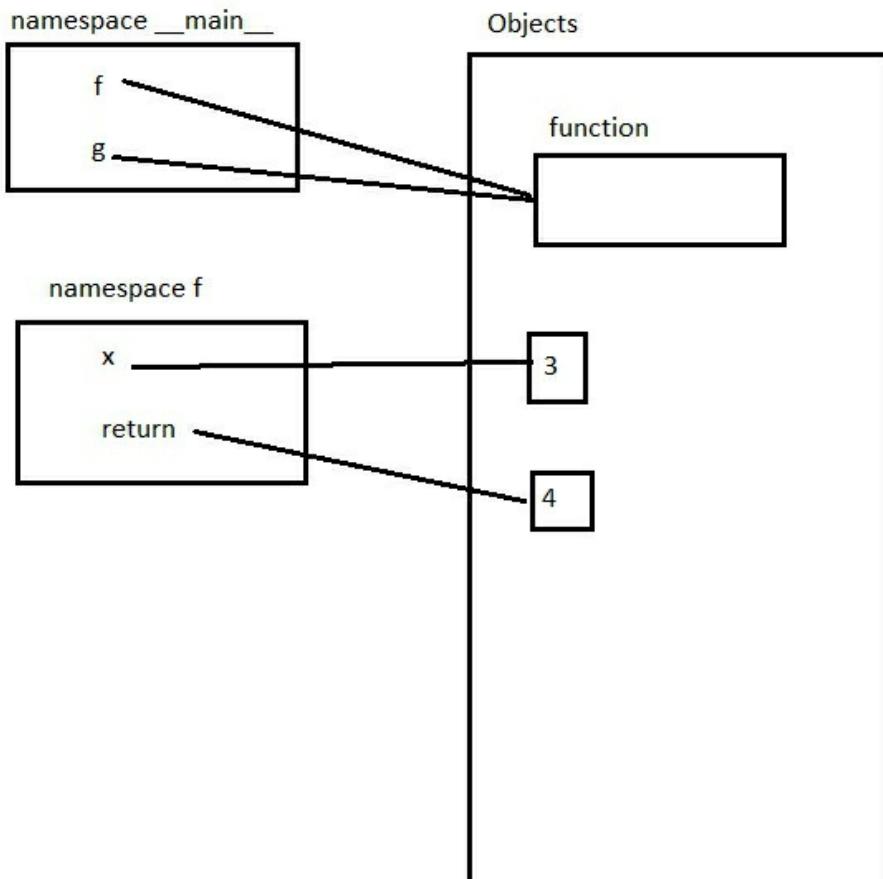
2°

```
>>> f(3)
```

```
#return 3 + 1
```

```
4
```

```
>>> g(3)
# return 3 + 1
4
```



>>> Exercise 405

Evaluate the following code

```
>>> L = [1, 2, 3]
>>> def f(x) :
    x.pop()
    return L
>>> help(list.pop)
pop(self, index=-1, /)
```

Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

```
>>> f(1)
```

```
>>> f(L)
```

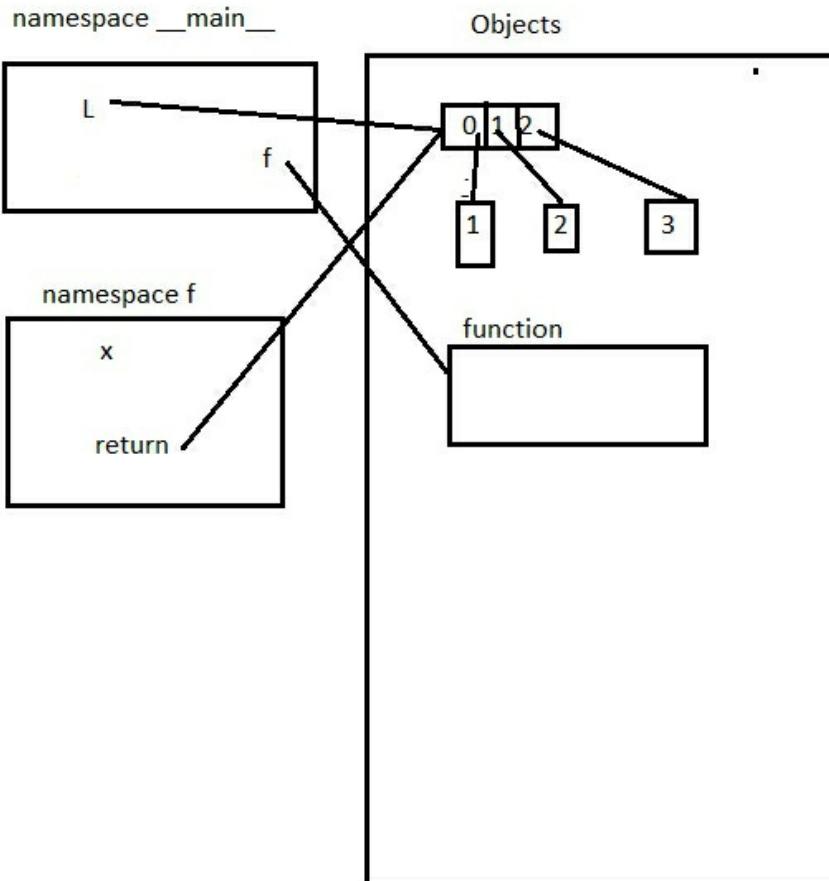
```
>>> f(L)
```

```
>>> f(L)
```

```
>>> f(L)
```

Solutions

Side effect



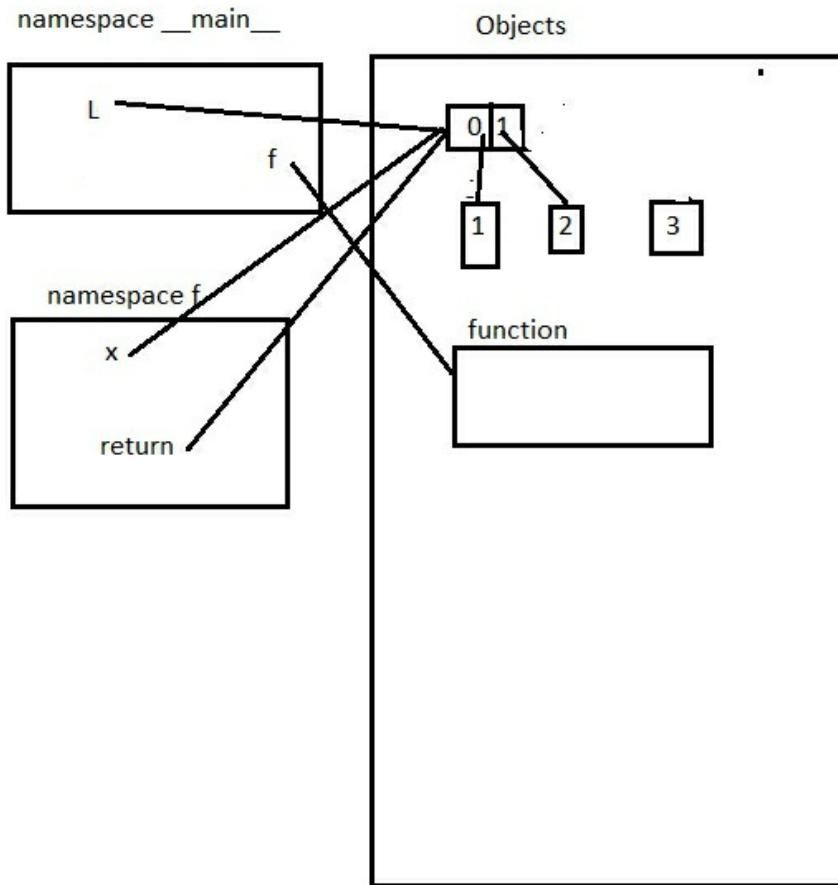
```
>>> f(1)
```

```
# 1.pop()
```

```
# AttributeError: 'int' object has no attribute 'pop'
```

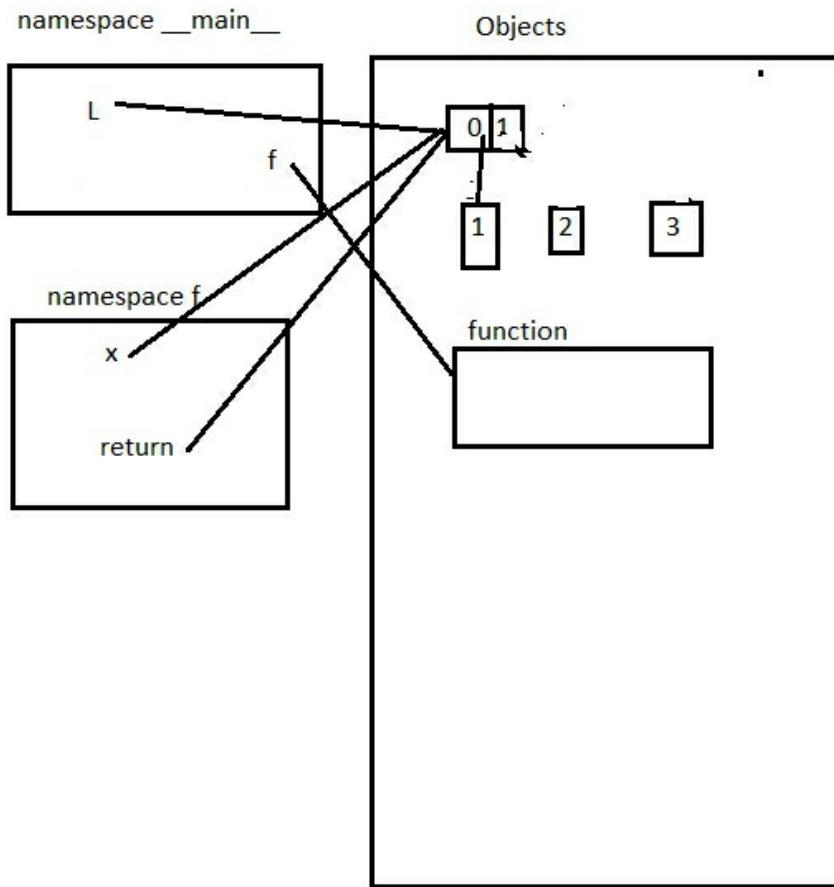
```
AttributeError: 'int' object has no attribute 'pop'
```

```
>>> f(L)
# f([1,2,3])
# [1,2,3].pop()
# pop([1,2,3], index = -1)
# 3
# L= [1,2]
# return L
[1, 2]
```



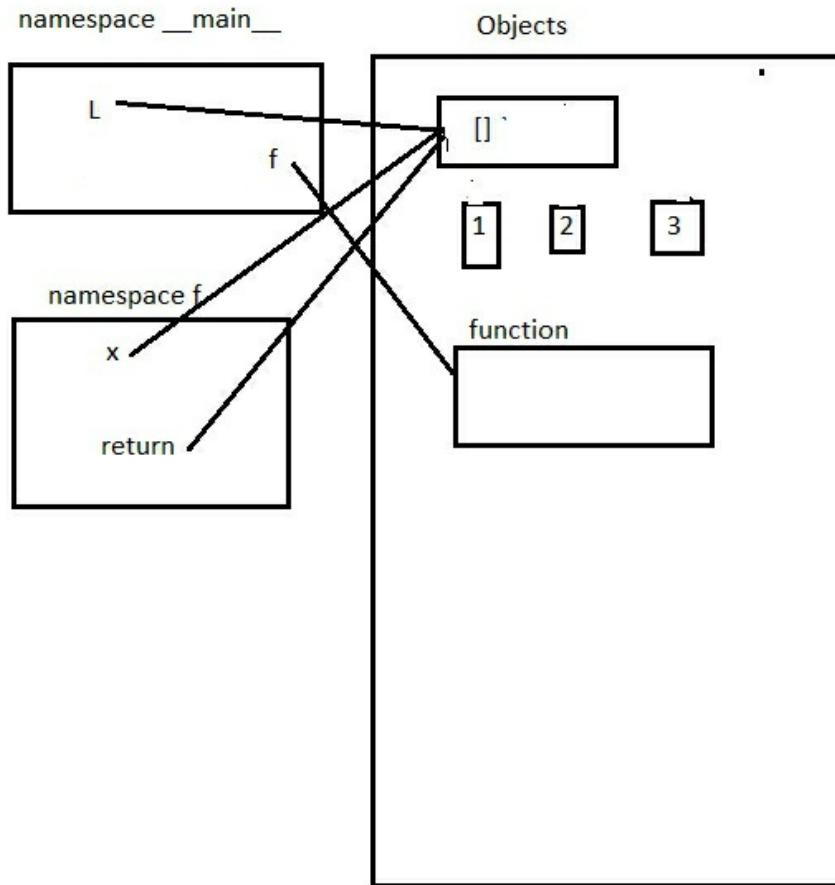
```
>>> f(L)
# f([1,2])
# [1,2].pop()
```

```
# pop([1,2], index = -1)
# 2
# L= [1]
# return L
[1]
```



```
>>> f(L)
# f([1])
# [1].pop()
# pop([1], index = -1)
# 1
# L= []
```

```
# return L
[]
```



```
>>> f(L)
# f([])
# [].pop()
# pop([], index = -1)
# IndexError: pop from empty list
IndexError: pop from empty list
```

>>> 35. Variables scopes

« i am the ... variable »

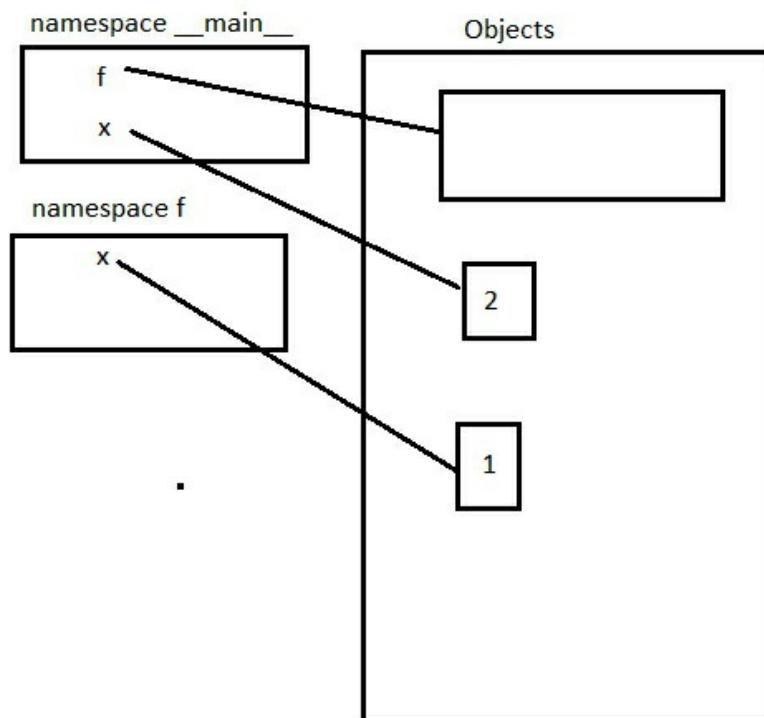
>>> 35.1. Local variable

>>> exercise 406

Evaluate the following code

```
>>> def f():
    x = 1
    print(x)
>>> x = 2
>>> f()
>>> x
```

solutions



```
>>> f()
# x = 1
# print(1)
# 1
1
```

```
>>> x
```

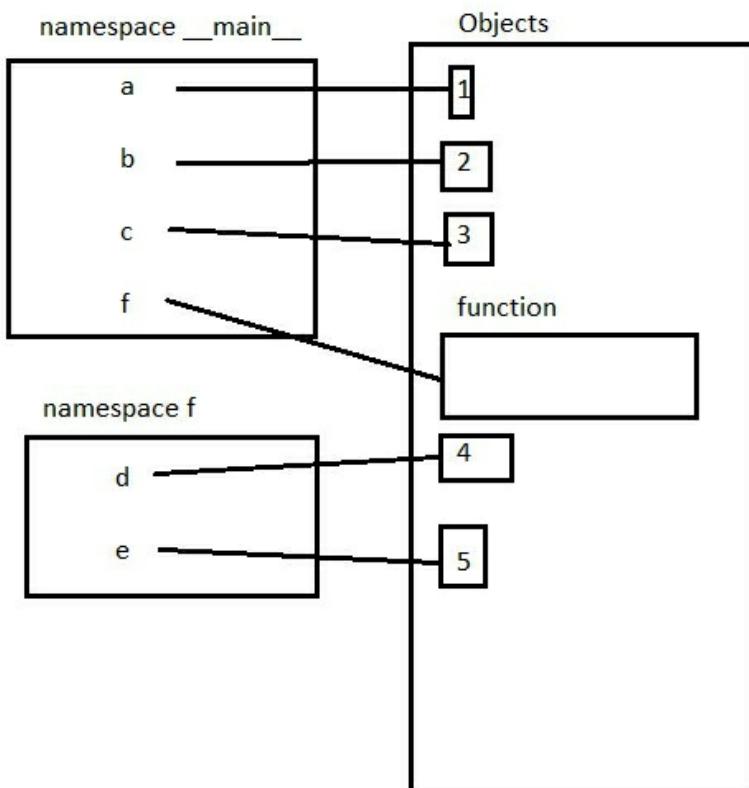
```
2
```

>>> Exercise 407

Evaluate the following code

```
>>> a = 1
>>> b = 2
>>> c = 3
>>> def f():
    d = 4
    e = 5
>>> f()
```

Solutions



>>> 35.1.1.local variable and global variable

>>> Exercise 408

1° Evaluate the following code

```
>>> def f( ) :  
    x = 1 / 0  
    return x  
>>> x = 1  
>>> f()
```

2° Evaluate the following code

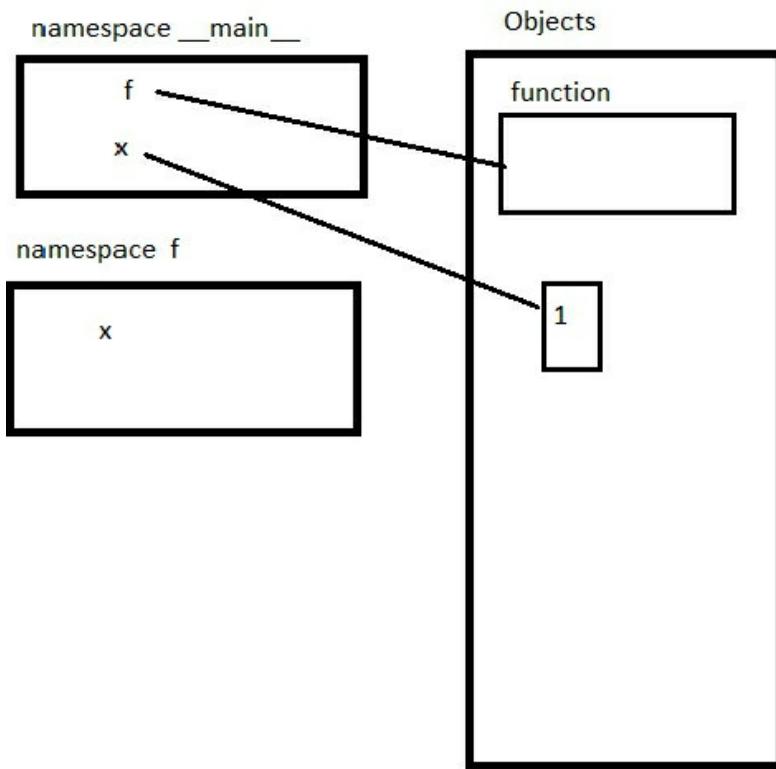
```
>>> def f() :  
    print(x)  
>>> f()  
>>> x = 1  
>>> f()
```

3° Evaluate the following code

```
>>> def f( ) :  
    x = 1  
    print(x)  
>>> x = 2  
>>> f()
```

Solutions

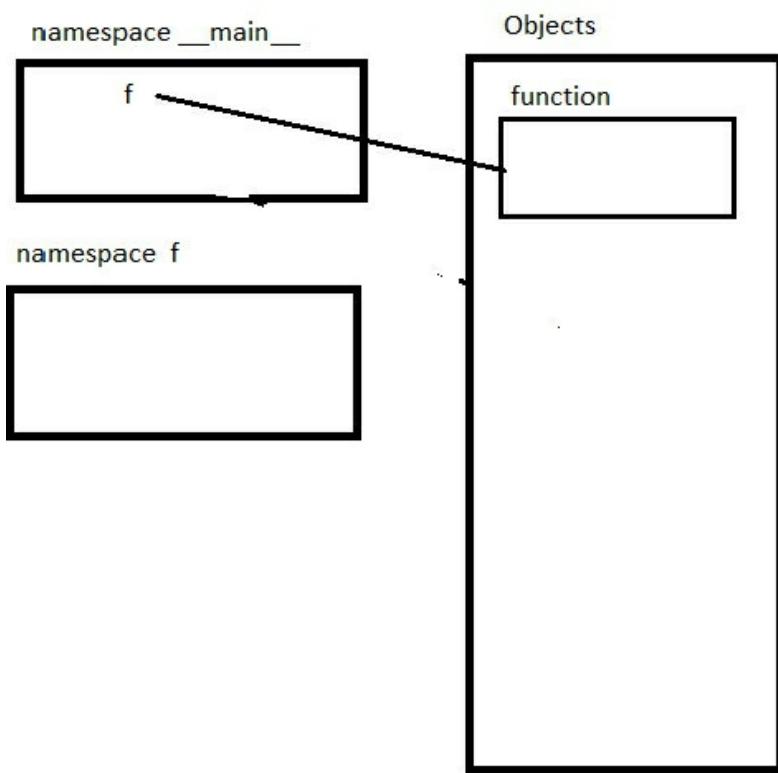
1°



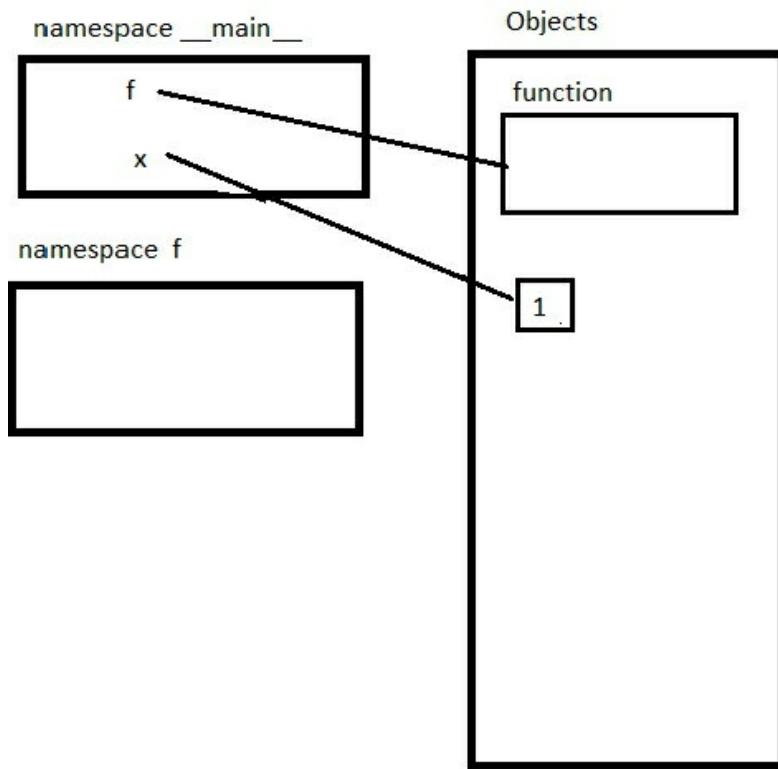
```
>>> f()
ZeroDivisionError: division by zero
```

2°

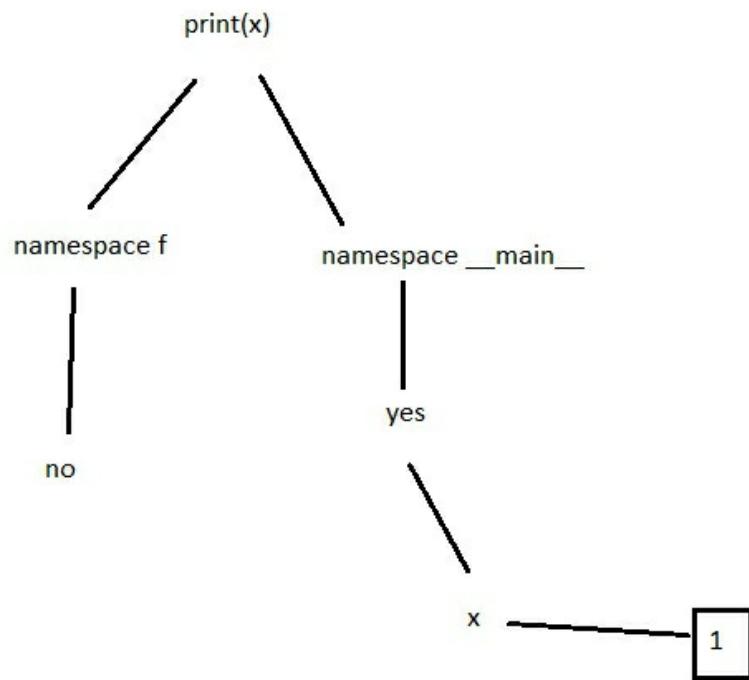
```
>>> def f( ):
    print(x)
>>> f()
# print(x)
# NameError: name 'x' is not defined
NameError: name 'x' is not defined
```



```
>>> x = 1
```

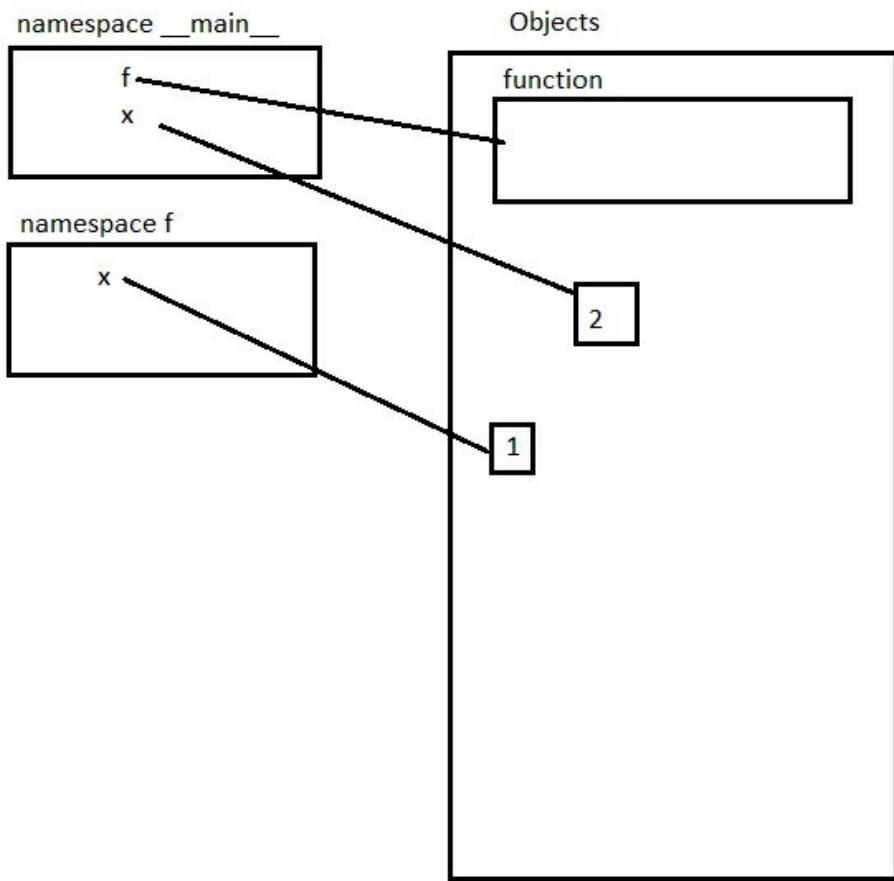


```
>>> f()
# print(x)
# aucune variable dans l'espace de f
# on va dans l'espace des variables du __main__
# print(x)
# print(1)
# 1
1
```

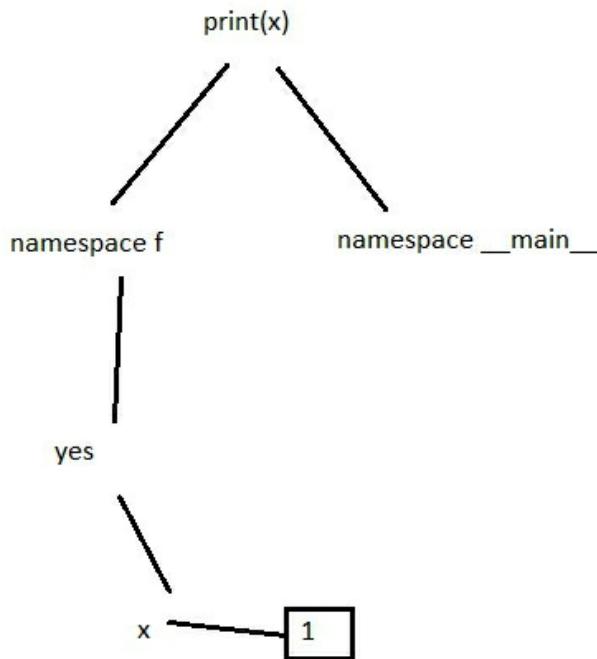


3°

```
>>> def f():
    x = 1
    print(x)
>>> x = 2
```



```
>>> f()
# x = 1
# print(x)
# print(1)
1
```



Tips :

Namespaces precedence :

- function namespace (local variables)
- __main__ namespace (global variables)

>>> Exercise 409

Evaluate the following code

```

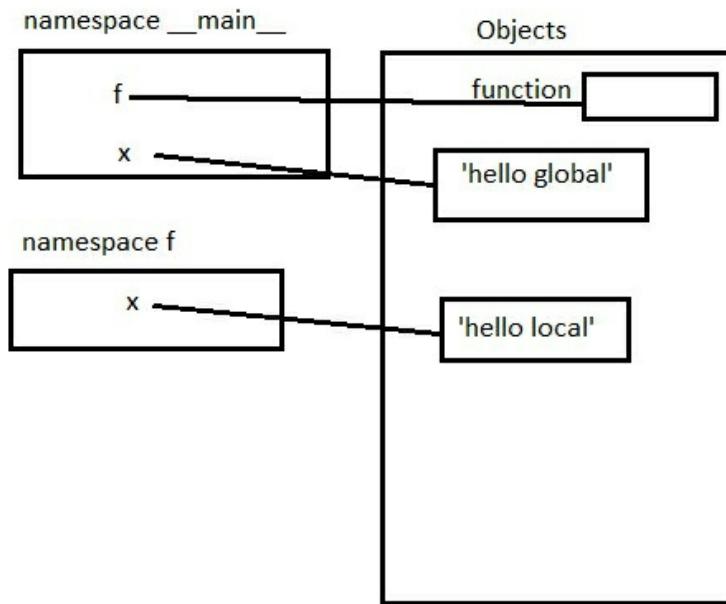
>>> def f():
    x = 'hello local'
    print(x)
>>> x = 'hello global'
>>> f()
>>> print(x)
  
```

Solutions

```

>>> f()
# x = 'hello local'
  
```

```
# print(x)  
# print('hello local')  
# hello local  
hello local
```



```
>>> print(x)  
# print('hello global')  
# hello global  
hello global
```

>>> 35.2.Function within a function

>>> **Exercise 410**

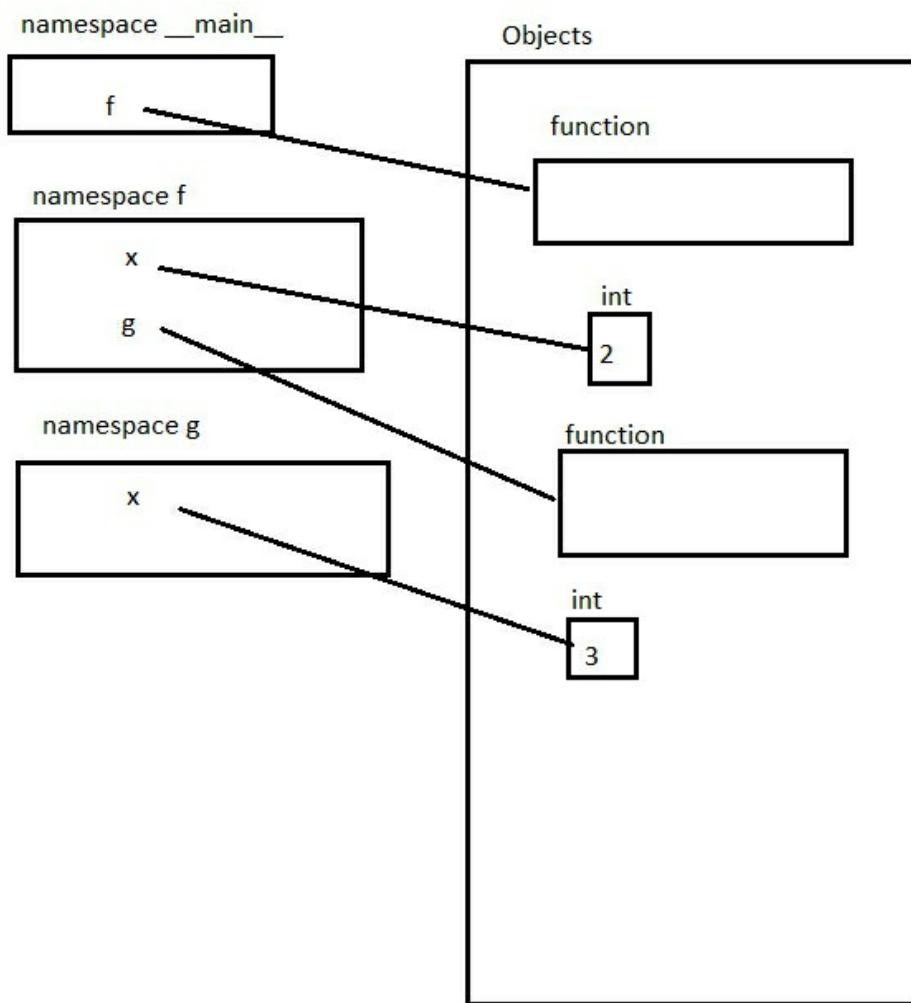
Evaluate the following code

```
>>> def f( ) :  
    x = 2  
    def g( ) :  
        x = 3
```

```
    print(x)
g()
>>> f()
```

Solutions

```
>>> f()
# x = 2
# def g( )
# g()
# g()
# x = 3
# print(x)
```



```
# print(x)
```

```
# print(3)
```

```
3
```

>>> Exercise 411

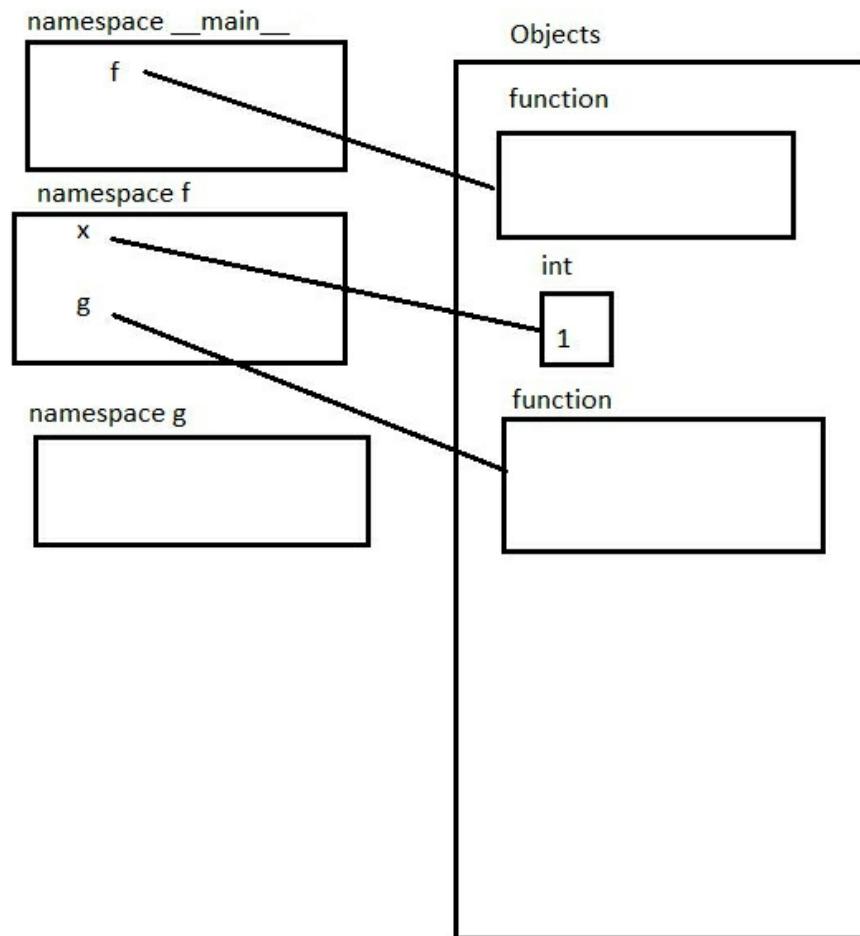
Evaluate the following code

```
>>> def f( ) :
    x = 1
    def g() :
        print(x)
    g()
```

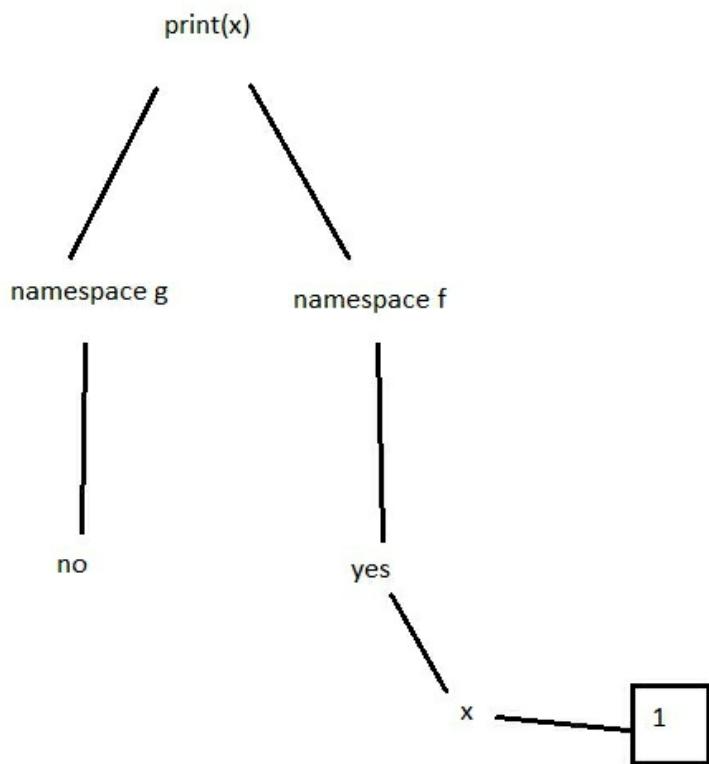
```
>>> f()
```

Solutions

```
# x = 1
# def g():
#     g()
#     print(x)
```



```
# print(1)
# 1
1
```



>>> Exercise 412

1° Evaluate the following code

```

>>> print(x)
>>> def f() :
    def g() :
        print(x)
    >>> f()
  
```

2° Evaluate the following code

```

>>> def f( ) :
    def g() :
        print(x)
    g()
    >>> f()
  
```

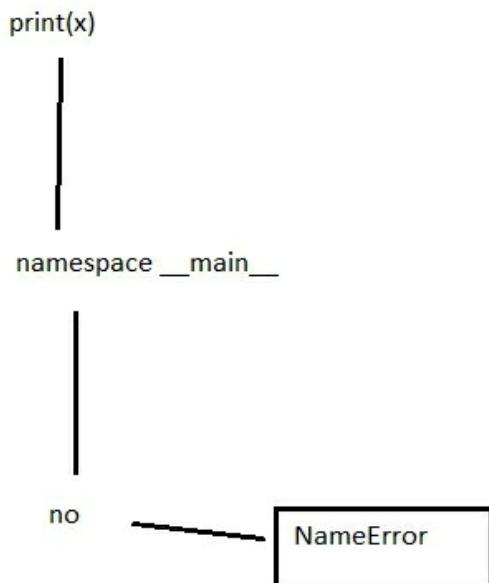
3° Evaluate the following code

```
>>> x = 1
>>> def f( ):
    def g():
        print(x)
    g()
>>> f()
```

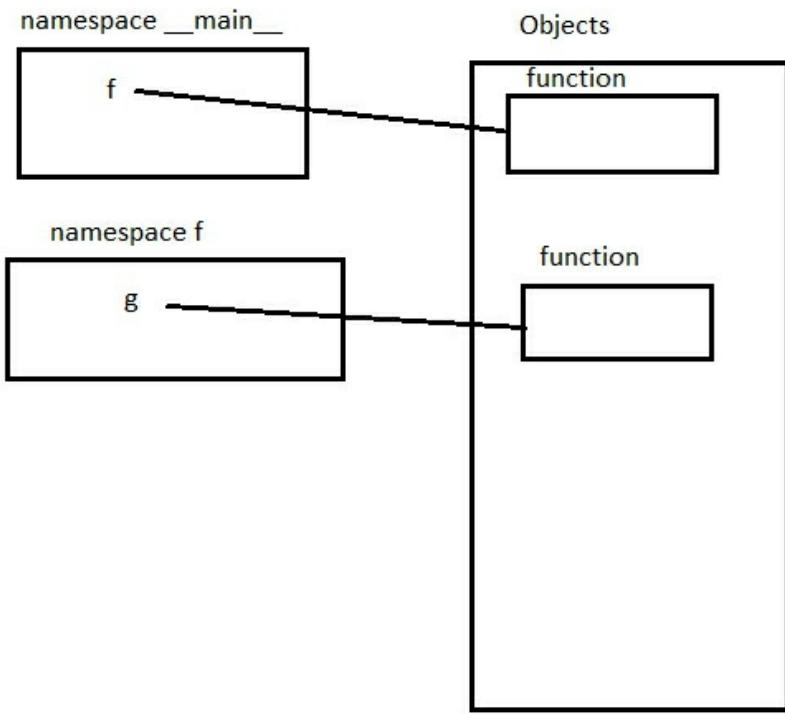
Solutions

1°

```
>>> print(x)
NameError: name 'x' is not defined
```

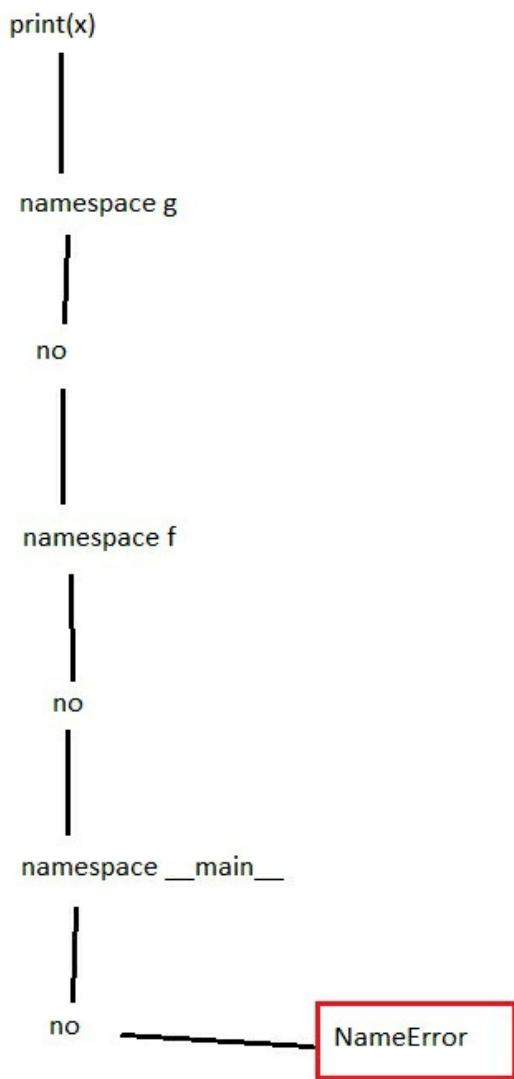


```
>>> f()
# def g()
>>>
```

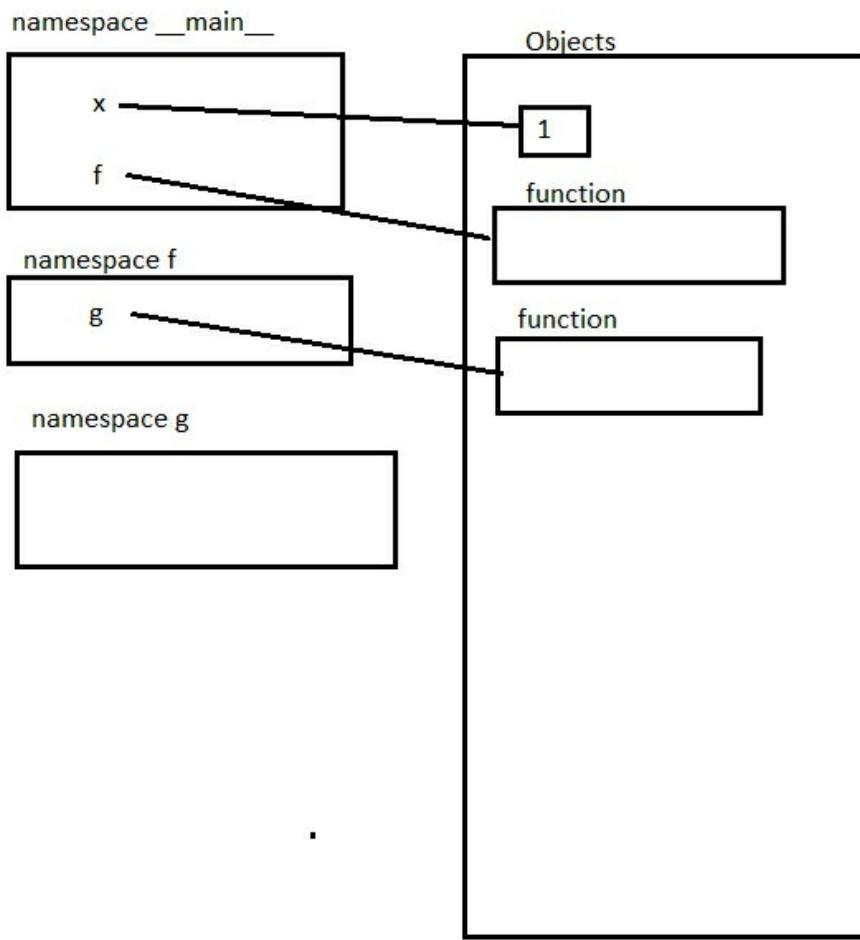


2°

```
>>> def f( ) :  
    def g() :  
        print(x)  
    g()  
>>> f()  
NameError: name 'x' is not defined
```



3°



```

>>> x = 1
>>> def f( ):
    def g():
        print(x)
    g()
>>> f()
# def g()
# print (x)
# print(1)
# 1

```

1

```
print(x)
```

 |
 |
 | namespace g
 |
 |

 | no
 |
 |

 | namespace f
 |
 |

 | no
 |
 |

 | namespace __main__
 |
 |

 | yes

 | x

 | 1

>>> 36.Modules : part 2

« module 1 : hello »

>>> 36.1.Module namespace

>>> Exercice 413

file fichier.py

```
a = 1
```

```
b = 3
```

1° Evaluate the following code

```
>>> import fichier
```

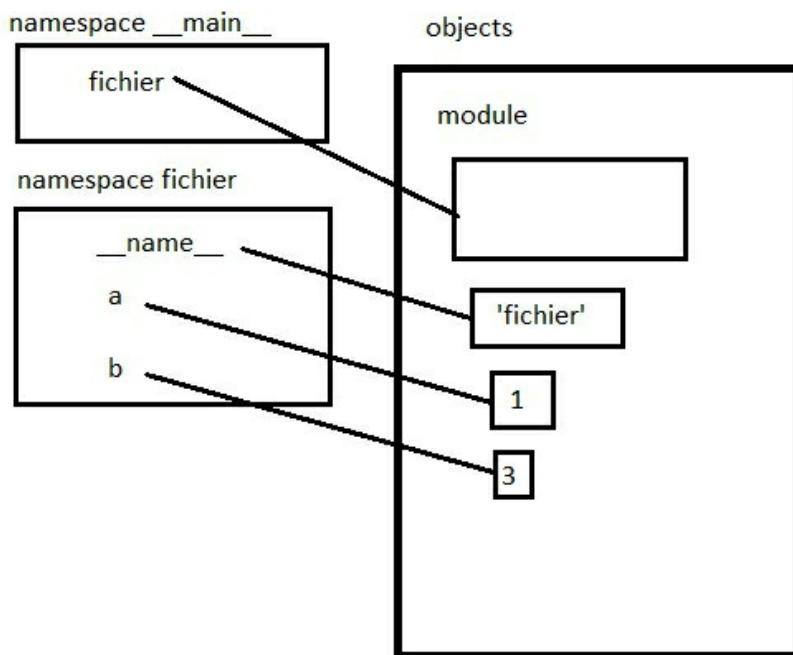
```
>>> fichier.__name__
```

2° Evaluate the following code

```
>>> a, b = 2, 4
>>> print(a,b)
>>> fichier.a
>>> fichier.b
```

Solutions

1°



```
>>> fichier.__name__
```

`'fichier'`

2°

```
>>> a, b = 2, 4
>>> print(a,b)
# print(2,4)
# 2 4
```

2 4

```
>>> fichier.a
```

```
# fichier.a
```

```
# fichier.1
```

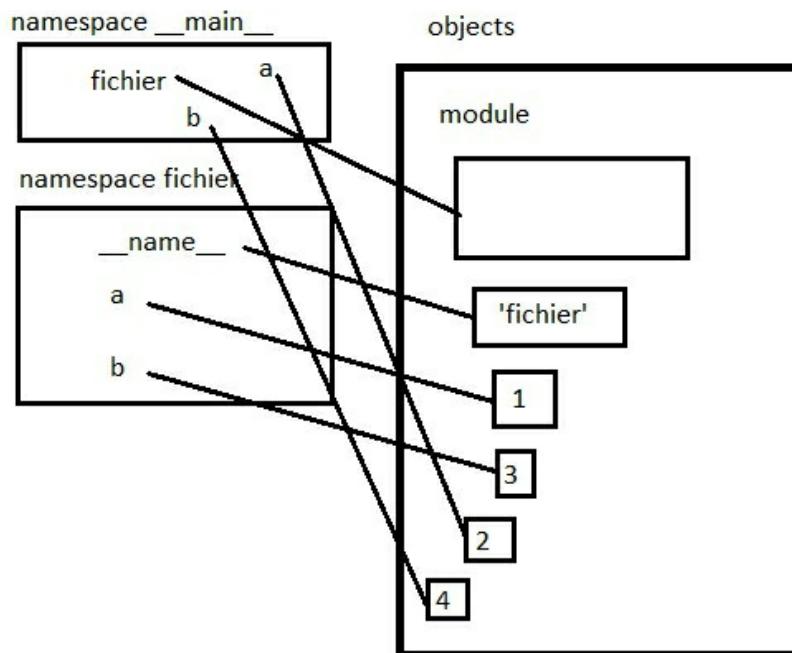
1

```
>>> fichier.b
```

```
# fichier.b
```

```
# fichier.4
```

4



>>> **Exercise 414**

file **module.py**

```
a = 2
```

```
b = True
```

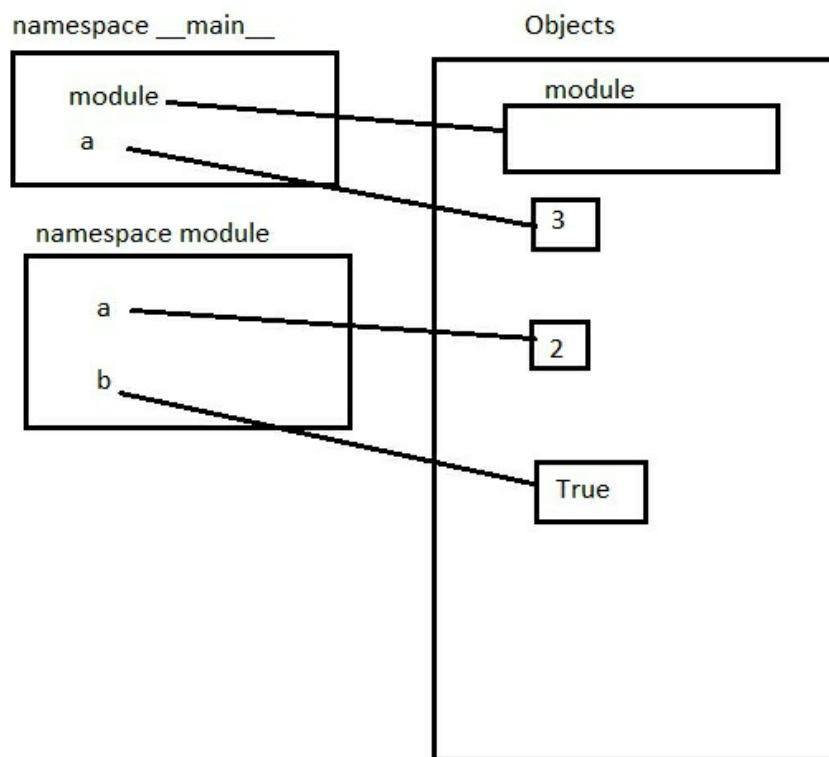
Evaluate the following code

```
>>> import module
```

```
>>> a = 3
>>> module.a
>>> print(module.b)
```

Solutions

```
>>> module.a
# module.a
# module.2
2
>>> print(module.b)
# print(module.True)
# print(True)
True
```



>>> 36.1.1.Fonctions defined within a module
>>> **Exercise 415**

file module1.py

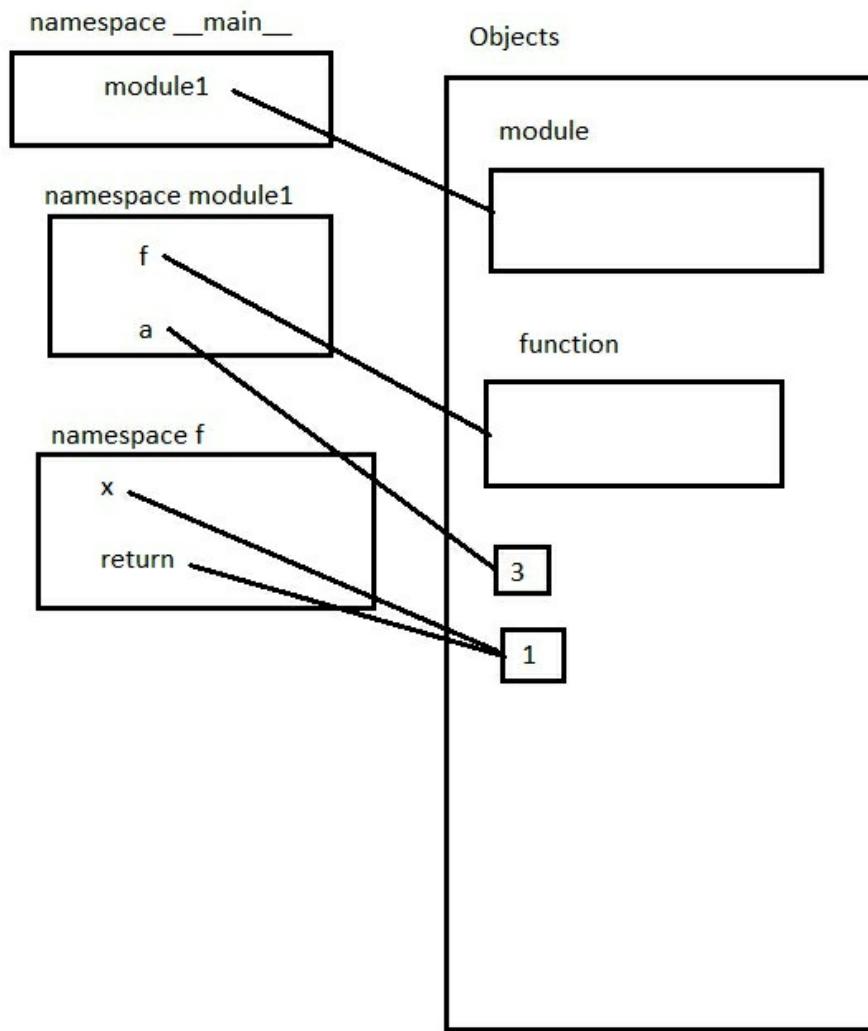
```
def f():
    x = 1
    return x
a = 3
```

Evaluate the following code

```
>>> import module1
>>> module1.f()
>>> module1.x
>>> module1.a
```

Solutions

```
>>> module1.f()
# x = 1
# return 1
1
```



```
>>> module1.x
# module1.x
AttributeError: module 'module1' has no attribute 'x'

>>> module1.a
# module1.a
#module1.3
3
```

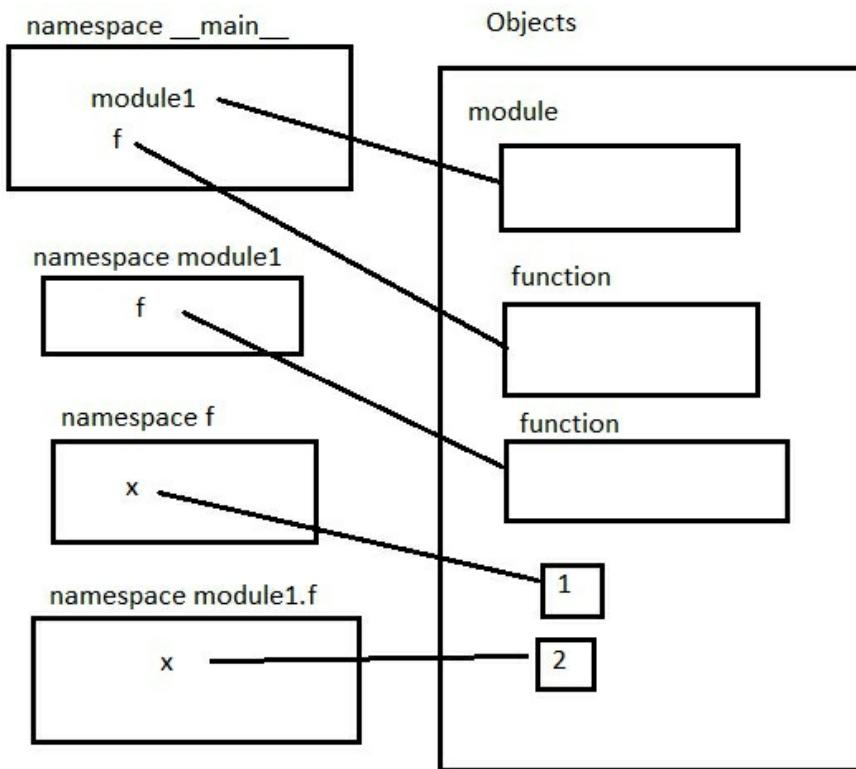
>>> **Exercise 416**
 file module1.py

```
def f() :  
    x = 2  
    print(x)
```

Evaluate the following code

```
>>> import module1  
>>> def f() :  
    x = 1  
    print(x)  
>>> f()  
>>> module1.f()
```

Solutions



```
>>> import module1  
>>> def f() :
```

```
x = 1
print(x)
>>> f()
# x = 1
# print(1)
1
>>> module1.f()
# module1.f()
# x = 2
# print(x)
# print(2)
2
```

```
>>> 'end'
```