# C1 - Python

C-COD-140

# Day 01

Learn Python

1.3

{EPITECH.}

# Day 01

**repository name:** python_d01
**repository rights:** ramassage-tek

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

## EXERCISE 01 (1PT)

**File to hand in:** python_d01/ex_01/script.py

Create a script with the adapted shebang and UNIX rights.newline It must count the number of alphanumeric characters contained in the program parameters.
The script must display on the standard output the count of alphanumeric characters.

**Example:**

Terminal

```
~/C-COD-140> ./script.py 'Hello World!' '@_@' 'Geckos'
16
```

## EXERCISE 02 (1PT)

**File to hand in:** python_d01/ex_02/my_args_handler.py

Create a function named "my_args_handler" that takes several strings as parameter.
This function returns **one** string which is the concatenation of every argument separated with ", ".

**Example:**

Terminal

```
~/C-COD-140> python3
>>> import my_args_handler
>>> print(my_args_handler.my_args_handler("a", "b", "c", "poke","jour" "nuit"))
a, b, c, poke, jour, nuit
```

## EXERCISE 03 (1PT)

**File to hand in:** python_d01/ex_03/double_return.py

Make a function “double_return”.
It takes a dictonary as parameter, and returns two lists.

Those returned Lists must contains differents values.

In the first one, there is the list of the keys and the second the list of the values contained in the Dictionary parameter.

**Example:**

Terminal

```
~/C-COD-140> python3
>>> import double_return
>>> print(double_return.double_return({"a": 1, "b": 2}))
([‘a’, ‘b’], [1, 2])
```

## EXERCISE 04 (OPT)

**File to hand in:** python_d01/ex_04/loader.py
python_d01/01_basics_04/required.py
**Restriction:** loader.py must not contain “=”

In “required.py”, create a constant”REQUIRED" set to true.
In the loader, you must include and have access to this constant, whatever the way we load your file “loader.py”

## EXERCISE 05 (1PT)

**File to hand in:** python_d01/ex_05/show_methods.py

Make a function “show_methods”.
It takes one parameter and display every methods of the object.

**Example:**

Terminal

```
~/C-COD-140> python3
>>> from show_methods import show_methods
>>> show_methods("a string object")
{[‘add’, ‘class’, ‘contains’, ‘delattr’, ‘dir’, ‘doc’,
‘eq’, ‘format’, ‘ge’, ‘getattribute’, ‘getitem’, ‘getnewargs’,
‘gt’, ‘hash’, ‘init’, ‘init_subclass’, ‘iter’, ‘le’, ‘len’,
‘lt’, ‘mod’, ‘mul’, ‘ne’, ‘new’, ‘reduce’, ‘reduce_ex’,
‘repr’, ‘rmod’, ‘rmul’, ‘setattr’, ‘sizeof’, ‘str’, ‘subclasshook’,
‘capitalize’, ‘casefold’, ‘center’, ‘count’, ‘encode’, ‘endswith’, ‘expandtabs’,
‘find’, ‘format’, ‘format_map’, ‘index’,
‘isalnum’, ‘isalpha’, ‘isdecimal’, ‘isdigit’, ‘isidentifier’, ‘islower’,
‘isnumeric’, ‘isprintable’, ‘isspace’, ‘istitle’,
‘isupper’, ‘join’, ‘ljust’, ‘lower’, ‘lstrip’, ‘maketrans’, ‘partition’,
‘replace’, ‘rfind’, ‘rindex’, ‘rjust’,
‘rpartition’, ‘rsplit’, ‘rstrip’, ‘split’, ‘splitlines’, ‘startswith’, ‘strip’,
‘swapcase’, ‘title’, ‘translate’,
‘upper’, ‘zfill’]}
```

## EXERCISE 06 (1PT)

**File to hand in:** python_d01/ex_06/my_show_platypus.py

Create a function named "my_show_platypus" that takes a number as parameter and that displays "Platypus" as many times as indicated by the parameter.
Each display of "Platypus" will be followed by a new line.

**Example:**

```
Terminal
~/C-COD-140> python3
>>> import my_show_platypus
>>> my_show_platypus.my_show_platypus(5)
Platypus
Platypus
Platypus
Platypus
Platypus
```

## EXERCISE 07 (1PT)

**File to hand in:** python_d01/ex_07/display_all.py

Create a function "display_all".
It takes a Dictionary as parameters and display every elements of the Dictionary, formatted as:

- "[class]": must be the class of the element.
- "[value]": its value converted into a String.
- Example : "([class]: [value]) new line"

## Exercise 08 (1pt)

**File to hand in:** python_d01/ex_08/display_all.py

Create a function "display_all".
It takes a Dictionary as parameter and display every element of the Dictionary formatted as:

"[key] -> ([class]: [value]) new line"

## Exercise 09 (2pts)

**File to hand in:** python_d01/ex_09/display_all.py

Create a function "display_all".

It takes a list as parameter and displays every element of the list formatted as:
"[id] -> ([class]: [value]) new line"

"[id]" is the index, from 0 to the list's length.

*Note: "=" is forbidden only for assignations, not for default parameters. Same rule for every following exercise. this means you cannot create or change values of variables/arguments*

## EXERCISE 10 (1PT)

**File to hand in:** python_d01/ex_10/increment.py
**Mandatory:** Use python's list/dict comprehension capacities

Make a function "increment" which takes a list as parameter and returns the same list with all integer elements increased by one:

**Example:**

Terminal

```
~/C-COD-140> python3
>>> import increment
>>> increment.increment([3, 4, 9, 'a', "qvb", {}])
=> [4, 5, 10, 'a', "qvb", {}]
```

## EXERCISE 11 (2PT)

**File to hand in:** python_d01/ex_11/shows.py
**Mandatory:** Use python's list/dict comprehension capacities

Make a function "shows". It takes a dictionary as parameter.

For each element of the dictionary, it has to convert it into a string following the format: "([key]) -> ([class]: [value])".

The function will print a list with every converted value.

**Example:**

Terminal

```
~/C-COD-140> python3
>>> import shows
>>> shows.shows({"test": "hello", "t": 43, "pi": 3.14})
[t => (int: 43), pi => (float: 3.14), test => (str: hello)]
```

## Exercise 12 (2pts)

**File to hand in:** python_d01/ex_12/count_words.py
**Mandatory:** Use python's list/dict comprehension capacities

Create a function "my_count_words" which takes a list as parameter and returns a dictionnary.

The List passed as argument is filled of strings.

The returned dictionnary must be formatted to respect the following rules:

- Every key is a string.
- Every value is an integer.
- The associated value is the count of occurrences of the key in the dictionary parameter.

## Exercise 13 (3pts)

**File to hand in:** python_d01/ex_13/sum_sub.py
**Mandatory:** Use reduce from functools

Create a python script which will add or subtract the values passed as arguments and based on 3 simple rules:

- If there is no arguments, then display 0.
- Else, the first value to use is the first argument.
- For each next argument, if the current value is odd, then add the next argument to the current value, else subtract it.

## EXERCISE 14 (3PTS)

**File to hand in:** python_d01/ex_14/regexp.py

Create a global constant EMAIL. It should be an instance of Regexp to capture an email.
An email should respect the Rfc5322.
**But your exercise will be easier.**

You will not implement a complete Rfc.
Follow the specified grammar:

- EMAIL = [ USER**@**DOMAIN ]
- USER = [ any(1, 1023) ]
- DOMAIN = [ SUBDOMAIN [ "." SUBDOMAIN ]* ] & [ any(1,253) ]
- SUBDOMAIN = [ alpha alpha | num ]

- * = 0, 1, or n times
- (a,b) = between a and b times
- alpha = any alphabetic character
- num = any numeric (0-9) character
- any = any non null character
- = a logic union (LOGIC OR)
- & = a logic intersection (LOGIC AND)

The full domain and user should be captured and named.

Valid:
user**@**domain.domain.domain.domain

Not valid:
user**@**domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.
domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.
domain.domain.domain.domain.domain.domain.domain.domain.domain.domain.domain