# 1ALG Exercices

## Summary

# Instructions

At the start of each new chapter, your instructor will specify the exercises you should complete.

There intentionally are more exercises than can be covered within the course duration. You are encouraged to independently work on exercises that were not addressed during the allocated time.

It is highly advisable to **take notes** and **pay close attention** to the corrections provided for each exercise during the course.

Sometimes, you may encounter challenges in the exercises that you find difficult to resolve. **There is no shame in searching the internet for solutions; professional developers do it ALL THE TIME**. However, it's important to emphasize that **you should avoid cheating and make a genuine effort to solve the exercise independently**.

If there is anything you don't understand, **don't hesitate to seek help with or clarification** with your instructor.

Upon the conclusion of each chapter or on the final day of the course, your instructor will share a link to a folder containing solutions for each exercise.

Each exercice is written in english as instructed by the school's administration. If you don't understand something ask your instructor or use a translator like https://www.deepl.com/translator.

## Exercice workspace

Unless asked, each exercice must be done in a new file.

You should use a new folder for each chapter.

Example :

```
chapter_3/
    .
    ├── 3.1.exercice_title.py
    ├── 3.2.exercice_title.py
    ├── ...
    └── 3.19.exercice_title.py
```

If you need to create multiple files for a single exercice you need to put all these files in a single folder whose name will be the one from the exercice.

Example :

```
7.5.using_multiple_files/
    .
    ├── venv/
    ├── main.py
    └── secret.py
```

# Chapter 2] Introduction to Python

## 2.1] Python Installation

Go to https://www.python.org/ and download python for your OS.

If you are using a linux distribution or macOS you might already have python installed on your system.

When installing python, do not press "NEXT" immediately, read what is written and ensure that you are installating to path.

## 2.2] IDE / Editor installation

To work with python you will need an environment. You have two choices for this course :

1. Download Pycharm using jetbrain toolbox : https://ecole-it.notion.site/JetBrains-fbcca50fc26c4e5c8b684b16a9daf5bd
2. Download Visual Studio Code : https://code.visualstudio.com/download

# Chapter 3] First steps with Python

## 3.1] Variable creations

Write a python program that contains a variable called `name` and prints it.

The variable should contain your own name.

## 3.2] Basic math - 1

Create a program that stores the result of `10 + 25` in a variable and prints it.

```
The sum of 10 and 25 is {result}
```

## 3.3] What did you say ? - 1

Write a program that asks the user to enter something and displays what he has written.

```
Enter something : Hello there
You wrote "Hello there" .
```

## 3.4] Basic math - 2

Create a program that asks the user to enter a number and display its double.

```
Enter a number : 25
The result of 25 * 2 is 50.
```

## 3.5] What's your name ?

Write a program that asks the user for his first and last names, then displays them with greetings.

```
Enter your first name : Bob
Enter your last name : Marley
Hello, Bob Marley !
```

## 3.6] What did you say ? - 2

Write a program that asks the user to enter something and displays what he has written in UPPERCASE.

```
Enter something : I am not yelling!
You wrote "I AM NOT YELLING!" .
```

### 3.7] Euros to USD

Write a program that converts a sum in euros to dollars.

You can use a rate of 1€ = 1.08$ for conversion.

Attention : you must round to the second number after the virgule.

```
Enter the value in euros : 2.49
2.49€ is equals to 2.69$.
```

### 3.8] Body Mass Index Calculator

Write a program that asks the user for his weight in kilograms (kg) and height in meters (m).

Calculate the user's BMI and display it using the formula BMI = WEIGHT / HEIGHT².

```
Enter your height in meters (m) : 1.8
Enter your weight in kilograms (kg) : 90
Your BMI is : 27.78
```

# Chapter 4] Flow control

## 4.1] Are you an adult ?

Ask the user to enter its age.
If the age is greater than 18 print "You are an adult", else print "You are not an adult".

```
Enter your age : 20
You are an adult !
```

## 4.2] Absolute value

Write a program that prompts the user to enter a number and then displays its absolute value.

```
Enter a number : -8
The absolute value is 8.
```

## 4.3] Number range

Print all integer numbers between 0 and 100

```
0
1
2
...
99
100
```

## 4.4] Number range with step

Print all numbers between 1 and 100 but with a step of 3.

```
1
4
...
97
100
```

### 4.5] Reverse number range

Print all integer numbers between 100 and 0.

```
100
99
...
1
0
```

### 4.6] Next n integers

Ask the user to enter a number and print the next 10 integers.

```
Enter a number : 4.58
5
6
...
13
14
```

### 4.7] Multiples of x

Ask the user for an integer x and print all multiples of x between 0 & 100.

FYI : By using a strictly mathematical point of view 0 is a multiple of no number and 1 is only a multiple of 1.

```
Enter a number : 3
3 is a multiple of 3
6 is a multiple of 3
...
99 is a multiple of 3
```

### 4.8] See the stairs

Write a program that prompts the user to enter a number.
Next, the program should display a staircase pattern of stars, where each step of the staircase is incremented according to the value entered by the user.

```
Enter the number of layers : 5
*
**
***
****
*****
```

### 4.9] Loop interest

Write a program that asks the user to enter a value in euros.

The program must then calculate and display the value after an annual growth of 2.5% over a 10-year period.

You need to round the value to the third decimal.

```
Enter a value : 1000
Year+1 : 1025.0
Year+2 : 1050.625
...
Year+10: 1280.085€
```

### 4.10] Multiplication table - 1

Write a program that asks the user to enter a number and displays the first 10 values of the multiplication table of this number.

```
Enter a number : 5

Multiplication Table for 5:

1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
...
10 x 5 = 50
```

### 4.11] See the pyramid

Write a program that prompts the user to enter a number.
The program should then display a pyramid of stars, where each row of the pyramid is incremented based on the value entered by the user.

Tips: if you don't understand how to center the pyramid, do it upside down

```
Enter the number of layers : 5
    *
   ***
  *****
 *******
*********
```

### 4.12] FizzBuzz

Create a program that implements the FizzBuzz algorithm.

Prompt the user to enter a positive integer, and then iterate through the numbers from 1 to the entered integer. For each number, follow these rules:

- If the number is divisible by 3, print "Fizz."
- If the number is divisible by 5, print "Buzz."
- If the number is divisible by both 3 and 5, print "FizzBuzz."
- Otherwise, print the number.

```
1
2
Fizz.
4
Buzz
```

### 4.13] Multiplication table - 2

Write a program that shows the multiplication of all numbers between 1 and 10.

There must a be at least two digits per number. Add a leading 0 so there are at least 2 digits.

```
01 02 03 04 05 06 07 08 09 10
02 04 06 08 10 12 14 16 18 20
03 06 09 12 15 18 21 24 27 30
04 08 12 16 20 24 28 32 36 40
05 10 15 20 25 30 35 40 45 50
06 12 18 24 30 36 42 48 54 60
07 14 21 28 35 42 49 56 63 70
08 16 24 32 40 48 56 64 72 80
09 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

### 4.14] Message loop

Create a program that prints in a loop "This is not a loop.".
After each print, ask if the user wishes to continue seeing the message.

```
This is not a loop.
Do you wish to continue seeing this message Y/N ? : Y
This is not a loop.
Do you wish to continue seeing this message Y/N ? : N
Goodbye!
```

### 4.15] Sum of n numbers

Ask the user to input a number. Then inquire if they want to continue. If they say no, stop the process and display the sum of all the entered numbers else keep asking for other numbers.

```
Enter a number : 5
Do you want to continue Y/N ? : Y
Enter a number : 48
Do you to continue Y/N ? : N
The sum of all numbers is 53.
```

## 4.16] Fibonacci

The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. It goes like this: 0, 1, 1, 2, 3, 5, 8, and so on.

Write a program that prompts the user to calculate the first 20 values of the Fibonacci sequence.

```
0
1
1
2
...
```

# Chapter 5] Collections and Containers

## 5.1] Simple list creation

Create and print a list containing the following values : 1, 2, 3, 5, 7, 11, 13

## 5.2] Simple list read

Create the following list :

```
demo_list = ["one", "two", "three", "four", "five"]
```

Print the following values :

- the size of the list using the function that returns it.
- the first value of the list using its index to read it.
- the last value of the list using its index to read it.
- the second value of the list using its index to read it.

## 5.3] List iteration

Use a loop to print all values in the following list :

```
demo_list = ["Alice", "Bob", "Charlie", "Michael"]
```

## 5.4] Sum of list values

Print the sum of all values in the following list

```
demo_list = [5, 1, 59, 42, 8, 3, 15, 9, 7]
```

## 5.5] Print odd numbers in a list

Use a loop to print all odd numbers in the following list :

```
demo_list = [5, 1, 59, 42, 8, 3, 15, 9, 7]
```

### 5.6] Check if value is in list

Ask the user to enter its name and check if it is in the given list.

- You must ensure that your check is case insensitive.
- If the user is in the list print "You are allowed".
- If the user is not in the list print "You are not allowed".

```
allowed_users = ["Alice", "Bob", "Charlie", "Michael"]
```

### 5.7] List remove duplicates

Write a program that shows all unique values in the given list :

```
demo_list = [4, 85, 6, 45, 85, 1, 2, 5, 1, -5, 96, -4 ,58, 14, 23, 6, 7,
58, 85]
```

### 5.8] List keep duplicates

Write a program that shows only all non unique values in the given list :

```
demo_list = [4, 85, 6, 45, 85, 1, 2, 5, 1, -5, 96, -4 ,58, 14, 23, 6, 7,
58, 85]
```

### 5.9] List comprehension

Use list comprehension to generate a list containing all squared values of the given list.

```
demo_list = [1, 2, 3, 5, 7, 11, 13]
```

### 5.10] List append / pop

Write a program that initializes an empty list and repeatedly asks the user for a number.

- If the number is even, add it to the list.
- If the number is odd, remove the last element from the list.
- The program should stop when the user enters 0.

### 5.11] Dict iteration

Use different loops with the given dict to :

- print all keys
- print all values
- print all key-value couples

```
product_stocks = {
    "screwdriver": 4,
    "hammer": 3,
    "glue": 7,
    "paper": 8
}
```

### 5.12] Count letters

Based on the provided text, create a program that counts and prints the occurrence of each character. Ensure that each character's count is printed only once.

```
famous_citation = "I think therefore, I am."
```

Example :

```
There are 2 'i' in the text.
There are 2 ' ' in the text.
There are 2 't' in the text.
```

### 5.13] User login

Write a program that asks for a **username** and a **password** then compare those values with the content of **user_logins**.

- username check should be **case insensitive**.
- password check should be **case sensitive**.
- if there is a match print "Access granted".
- if there is no match print "Could not login".

```
# association username=>password
user_logins = {
    "alice": "12345678",
    "bob": "p@ssw0rd",
    "charlie": "qwertyuiop123",
    "michael": "superP@ssword123",
}
```

### 5.14] Show unique letters

Using the given text, write a program that displays all unique characters in the text.

```
famous_citation = "I think therefore, I am."
```

### 5.15] Count words in text

Write a program that counts all words in the given text.

If you have two words that have the same letters but not the same case (uppercase/lowercase) you should consider that they are the same.

Characters that are in the list **chars_to_remove** should be removed from the text before counting words.

```
chars_to_remove = ".,;:!?/'"
lorem_ipsum = "Lorem Ipsum is simply dummy text of the printing and
typesetting industry. Lorem Ipsum has been the industry's standard dummy
text ever since the 1500s, when an unknown printer took a galley of type
and scrambled it to make a type specimen book. It has survived not only
five centuries, but also the leap into electronic typesetting, remaining
essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with
desktop publishing software like Aldus PageMaker including versions of
Lorem Ipsum."
```

# Chapter 6] Using functions

Starting from this chapter all **functions must use type hint**.

**Docstring is optional** unless explicitely asked.

You should **use functions whenever feasible and reasonable**, even in cases where explicit instructions do not specifically request their usage.

If function names are unspecified, opt for clear and explicit choices.

If there is no test value provided, you are free to use anything to test the function execution.

## 6.1] Greet

Write a function called **greet** that takes a name as input and returns "Hello, {name}!" as a str.

## 6.2] Power of x

Create a function called **power** that takes a base and an exponent.

- If the exponent is not provided, it should default to 2.
- The function should return the result of raising the base to the exponent.

## 6.3] Multiple return values

Write a function called **divide_numbers** that takes two numbers as arguments and returns both their quotient and remainder.

## 6.4] List average

Write a function that returns the average of all numbers in a list and test it against the given list.

```
demo_list = [5, 1, 59, 42, 8, 3, 15, 9, 7]
```

## 6.5] min/max/abs

Create two functions for finding the minimum and maximum values of a sequence of numbers without relying on built-in functions.

Those functions must return **None** if their given input is an **empty sequence** or **None**.

## 6.6] Find index in list

Write a function that takes **a sequence** and **a target** value as inputs.

The function should find and return the index of the target value in the sequence without using any built-in functions/method designed for this purpose.

The function should return **None** if the element is not found or the sequence is **empty** or **None**.

## 6.7] Reverse list

Write a function that reverses the elements of a given list. The function should take a list as input and return a new list with the elements in reverse order.

## 6.8] Reverse number

Write a function that takes a number as input and return its "mirror value" as a float.

If the number is negative, its "mirror value" should also be negative.

Examples :

- 854 => 458
- 123.456 => 654.321
- 0 => 0
- -12 => -21

## 6.9] Removing duplicates

Write a function that removes duplicates from a list while preserving the order of the first occurrence of each element.

The function should take a list as input and return **a new list** with duplicates removed

Examples :

- [1, 1, 2, 2, 3, 3] => [1, 2, 3]
- [1, 2, 1, 3, 2, 1, 4] => [1, 2, 3, 4]

## 6.10] List common elements

Write a functions that identifies and returns the common elements between two lists.

It should take two lists as input and produce **a new list** containing only the elements that appear in both original lists.

### 6.11] Prime number

Create a function to determine whether a given positive integer is a prime number.

The function should take **an integer** as input and return **True** if it is prime and **False** otherwise.

### 6.12] Safe read from index

The objective of this exercise is to replicate the functionality of the **dict.get** method. Create a function that receives three parameters:

- A dictionary (`dict`)
- A key (string)
- A default return value if the key does not exist (defaulting to None)

The function should return the value associated with the key if it exists in the dictionary; otherwise, it should return the third input value.

In this exercice you are expected to also write the docstring.

### 6.13] Price reduction

Write a function named **compute_discounted_price** that takes two parameters:

- **original_price**: The original price of an item (float or int).
- **discount_percentage**: The discount percentage to be applied (float).

The function should calculate and return the discounted price based on the provided original price and discount percentage.

Exemple :

- original_price=100, discount_percentage=30 => return value 70

### 6.14] Sorting lists - 1

Write a function named **basic_list_sort** that takes a **list of numbers** as input and returns a **new list** containing the elements sorted in **ascending order**.

### 6.15] Sorting lists - 2

Create a function that accepts two parameters:

- A list of numbers, named **numbers**
- A boolean value, named **ascend**

If **ascend** is **True**, the function should return the values of **numbers** sorted in ascending order.

If **ascend** is **False**, the function should return the values in descending order.

### 6.16] The final countdown

Develop a function named **final_countdown** that takes an integer as a parameter and counts down from that number to 0.

If the parameter is less than or equal to 0, display the message: "This, is already finished."

### 6.17] Greatest common divisor

Write a function named **compute_gcd** that calculates the **Greatest Common Divisor** (GCD) of **two positive integers**.

The GCD is the largest positive integer that divides both numbers without leaving a remainder.

There are different ways to compute the GCD of two integers, you can find some algorithm here

In this exercice you are expected to also write the docstring.

### 6.18] Temperature conversion

Write a program to convert between celsius and fahrenheit temperatures.

The program must first ask the user to enter the mode he wants to use:

- 1] celsius -> fahrenheit
- 2] fahrenheit -> celsius

If the user enters a value that is neither **1** nor **2**, the program will stop and display an invalid entry.

For your information, here are the conversion formulas:

- tCelsius = (tFahrenheit - 32) / 1.8
- tFahrenheit = tCelsius * 1.8 + 32

```
1] fahrenheit -> celsius
2] celsius -> fahrenheit
Choose the conversion mode : 2

You choose celsius -> fahrenheit.
Enter the temperature in celsius : 37.5
37.5° celsius is 99.5] fahrenheit
```

In this exercice you are expected to also write the docstring.

## 6.19] Is password secure

Create a function called **is_password_secure** that takes a str called **password** as input.

It will return a boolean value that is True if :

- the password has at least 12 characters
- the password contains at least one uppercase
- the password contains at least one lowercase
- the password contains at least one number
- the password contains at least one "special char" that is in `./@!?;()[]`.

In this exercice you are expected to also write the docstring.

# Chapter 7] Working with modules

Starting from this chapter, for each exercise ensure that your solution includes a `main()` function, which contains the main code to execute. This function should be called by the following block:

```
if __name__ == '__main__':
    main()
```

This block is commonly used to check whether the Python script is being run directly or imported as a module. Including the `main()` function within this block allows for clean and organized execution of your code.

Example:

```
def main():
    # Your main code for the exercise goes here
    print("Hello, this is the main function!")

if __name__ == '__main__':
    main()
```

By following this structure, your code will be well-organized, and it allows for better readability and better reusability when importing functions from your script into other Python files.

If you need to create multiple files for a single exercice you need to put all these files in a single folder whose name will be the one from the exercice.

The same rule applies if you need to create a virtual env for an exercice.

Example :

```
7.25.using_multiple_files/
    .
    ├── venv/
    ├── main.py
    └── secret.py
```

When switching exercice, **do not forget to deactive the virtual env**.

### 7.1] Virtual env - basis

Create a new virtual environment named `venv_demo` and activate it.

Using the virtual env, install the package `request` with pip.

Create a requirements.txt file in the current folder.

### 7.2] Virtual env - CLI installation

A virtual environment not only provides isolation for Python projects but can also be utilized for the installation of command-line programs. In this exercise, we will install a specific command-line tool.

Begin by creating a new virtual environment named `demo_cli` and activating it. This step ensures that our installations remain isolated from the system-wide Python environment.

Once the virtual environment is active, proceed to install our command-line tool.

TLDR Pages is a community-driven collaborative collection of cheatsheets for command-line programs. It serves as a valuable resource for quickly finding examples on how to use various commands. TLDR Pages also offers a Python client that you can install.

The client is available on PyPI at https://pypi.org/project/tldr/.

Install it within the virtual environment and then execute the following commands to verify that it works as expected :

```
tldr --version

# You can find examples about commands by running
tldr <command>

# examples
tldr python
tldr firefox
tldr google-chrome
```

### 7.3] Pythagorean theorem

Ask the user to enter two floats, x and y, representing the length of a right-angled triangle.

Use the Pythagorean theorem to calculate the length of the hypotenuse.

If we define the length of the hypotenuse as z, then we have `z = sqrt(x² + y²)`.

To compute a square root, you can use the `sqrt` function from the `math` module.

### 7.4] Color code generation

Colors can be represented by a hexadecimal code.

This consists of 6 characters chosen from the following "0123456789ABCDEF".

Create a function that returns a string corresponding to a color code.

Examples:

- AB84CD
- 0451DE
- FF00FF

FYI: you will need to use the `random` built-in module.

### 7.5] Guess the number

Create a program that selects a random integer between 0 and 100. The program will prompt the user to guess the chosen number.

Upon each guess, the program provides feedback by indicating whether the guessed number is greater or lesser than the secret number.

If the user correctly identifies the number, the program prints "You found the number!"

FYI: you will need to use the `random` built-in module.

### 7.6] Password generation

The goal of the exercise is to write a function that randomly generates and returns a password that meets the following criteria:

- At least one lowercase letter
- At least one uppercase letter
- At least one digit
- At least one special symbol from the sequence: `@|{}=&*^!`
- At least 10 characters
- Maximum 64 characters

Hints:

- You can use `random.choice()` to select a character to place.
- Feel free to use `random.random()` or `random.randint()` to determine the password's length.

### 7.7] Current date

Write a Python function that retrieves the current date and returns it in the ISO format (ISO 8601).

FYI: You will need to use the `datetime` module.

### 7.8] Text shuffle

Write a program that asks the user to enter a message.

Use the `shuffle` function from the `random` module to mix up the characters, then display the result.

Exemple :

- user_input="Hello there!" => output="ehHl! roeelt"

### 7.9] Import file simple

Create a variable named `SECRET_PASSWORD` with a value of your choice in a file named `secret.py`.

Import `secret.py` and display the variable `SECRET_PASSWORD`.

### 7.10] A simple calculator

Create a file called `calc.py`, it should contains four functions :

- add : takes two floats x and y and returns x + y
- sub : takes two floats x and y and returns x - y
- mul : takes two floats x and y and returns x * y
- div : takes two floats x and y and returns x / y

In a file called `main.py` import `calc.py` and use it to create a simple calculator where the user can choose an operation (+-*/) then enter values to compute.

You should handle cases where you don't understand the user input.

Example :

```
Welcome to the calculator.
Please choose an operation + - * / : !!!
Sorry, the operation '!!!' was not understood.
Please choose an operation + - * / : +
Doing additions, enter the first value : 5
Doing additions, enter the second value : 9.5
The result is : 5 + 9.5 => 14.5
```

## 7.11] Virtual env - external module

Wikipedia is an online and collaborative encyclopedia containing information about many different things.

It has an API (a sort of communication interface) that allows you to programmaticaly read data from it.

Inside a new virtual env called `wikipedia_reader` install the `wikipedia` package.

Write a program that asks the user to enter a topic and use the `wikipedia` module to print its summary.

Hints :

- you will need to read the documentation of the wikipedia python api wraper.
- if you are stuck don't hesitate to search on internet a way to do what is asked.

Links :

- Wikipedia Python API Wraper Source : https://github.com/goldsmith/Wikipedia
- Wikipedia Python API Wraper Documentation : https://wikipedia.readthedocs.org
- Wikipedia Python API Wraper Pypi : https://pypi.org/project/wikipedia/

# Chapter 8] Handling errors

## 8.1] Raising exception

Write a program with the following instructions :

1. Use the `raise` keyword to generate a custom exception with a message of your choice.
2. Implement a `try/except` block to catch this exception.
3. Print a message within the `except` block, incorporating the caught exception's message, such as "There is an error: {e}".

## 8.2] atof conversion

Attempt to convert a string with the value `This is a text` to a float. Capture the exception and display `an error occurred.`.

## 8.3] Divide or fail

Write a program that asks the user to input two numbers and compute the division of the first number by the second.

Use exception handling to address the possibility of a ZeroDivisionError.

In the event of such an error, instruct the user to input the second number again until a non-zero value is provided.

Example :

```
We will divide x by y.
Enter a first number (x) : 10.8
Enter a second number (y) : 0
Caught a ZeroDivisionError, please enter another number y : 2
Result is : 10.8 / 2.0 = 5.4
```

### 8.4] Function safe input number

Design a function named `safe_float_input` intended for soliciting a numerical input from a user. The function should accept a message to prompt the user and return a **float** value.

The function should persistently prompt the user with the specified message until a valid number is entered.

Example usage of the function:

```
Enter a number: no
This is not a valid number.
Enter a number: don't want to
This is not a valid number.
Enter a number: 42
```

After that the function should return the user's input value (42.0).

### 8.5] Index read fail

Given the list:

```
my_list = ["a", "b", "c", "d"]
```

Ask the user to input the index of the value they want to read from the list. Attempt to display the value at the requested index and if you catch an error print it.

Example:

```
The values are ["a", "b", "c", "d"].
Enter the index of the value you want to read: 5
An error occurred: IndexError: list index out of range
```

### 8.6] List divisions

Write a program that asks the user to enter 6 numbers. The first 3 numbers will be put in a list called `dividends` and the next three in a list called `divisors`.

Try to divide each value of `dividends` by the value at the same index in `divisors`.

Ensure you handle all potential exceptions that may arise during the program's execution.

Example :

```
Please enter the three numbers to use as dividends :
First number : 5
Second number : Michael Jackson
This is not a valid number.
Second number : 12.2
Third number : 8

The dividends are : [5.0, 12.2, 8.0]

Please enter the three numbers to use as divisors :
First number : 2
Second number : 3
Third number : 16

The divisors are : [2.0, 3.0, 16.0]

Compute result :
First number => 5.0 / 2.0 => 2.5
Second number => 12.2 / 3.0 => 4.06
Third number => 8.0 / 16.0 => 0.5

The result is [2.5, 4.06, 0.5]
```

You are expected to handle exceptions like ValueError and ZeroDivisionError.

## 8.7] Function with ValueError

Write a function that compute the simple interest of an investment for the year X where X is the current after the investment..

It should take three inputs :

- an intial value (float)
- a time period in years (float)
- a rate (float)

The function must raise a ValueError if the rate is bigger than 100%.

To calculate the interest you can use this formula :

`Interest = InitialAmount * TimePeiod * Rate / 100`

Example of the formula :

```
InitialAmount = 1000 (in euros)
TimePeriod = 10 (10 year)
Rate = 25 (25%)

Interest = 1000 * 10 * 25 / 100
Interest = 2500
```

Formulation explanation : https://en.wikipedia.org/wiki/Interest#Simple_interest

# Chapter 09] Reading and Writing files

To complete the exercises in this chapter, you will require certain pre-existing files. These files are located in the `resource/chapter_9` folder.

Ensure that you duplicate this folder and place it in the directory where you are creating the Python files for this chapter.

## 9.1] Simple file read

Write a program that displays to the user the content of the file `9.1.readme.txt`.

## 9.2] Simple file write

Develop a program that generates a file named `9.2.i_can_write.txt` in the current directory and writes the phrase `I can write!` into it.

## 9.3] Reversed file lines

Develop a program that reads each line from the file named `9.3.file_to_reverse.txt`, prints the reversed version of each line to the user, and also writes these reversed lines to a file named `9.3.reversed_file.txt`.

Example :

```
Hello world
This is an example
```

Becomes

```
dlrow olleH
elpmaxe na si sihT
```

### 9.4] Count words in a text file

Develop a program that counts the number of words in each line of the file named `9.4.count_words.txt`.

In this context, consider a word to be anything separated by a space, even if it is only one character long or doesn't contain any letter.

The program should display each count to the user. For example (note: this is not the actual output):

```
Line 1 has 14 words.
Line 2 has 18 words.
Line 3 has no word.
...
```

### 9.5] User input to text file

Write a program that reads three user inputs and writes each one in a file called `9.5.user_inputs.txt`.

Example :

```
Enter a first input : Hello there
Enter a second input : General Kenobi !
Enter a third input : Noooooo
```

The file will contains these lines :

```
Hello there
General Kenobi !
Noooooo
```

### 9.6] Dict to JSON

Develop a program that writes the provided dictionary in JSON format to a file named `9.6.dict_to_json.json`.

```
dict_to_convert = {
    "application_id": "54a32cba-994f-44ba-957b-77d156b3207b",
    "developper_id": "20e2662b-c39d-4327-b4c3-96013c4645f9",
    "name": "Super Python JSON debugger.",
    "paid": True,
    "price": 42.2,
    "price_unit": "euros",
}
```

### 9.7] CSV movie ratings

You are provided with a CSV file called `9.7.movie_ratings` that contains ratings from a famous movie critic.

Write a program that reads each of its line and prints it to the user in the format of this example :

```
Movie called "Star Wars" has been rated with a note of 4 by the critic.
Movie called "Indiana Jones" has been rated with a note of 5 by the critic.
...
```

### 9.8] JSON to CSV

You were provided with a JSON file called `9.8.server_players.json`. It contains information about the players of an online game.

You are asked to write a program that converts the data to a CSV format using the following column headers :

```
GAME_SERVER,PLAYER_NAME,PLAYER_CLASS,PLAYER_LEVEL
```

### 9.9] Product Inventory CSV

Your enterprise's product team has supplied you with the file `9.9.product_inventory.csv`, containing status information for various shops in your company.

The file has the following columns:

- `PRODUCT_ID`: the reference id of a sellable product
- `SHOP_ID`: the id of a physical shop
- `STOCK_QUANTITY`: the quantity of items in store for the referenced product

The marketing team has requested specific data extracted from the inventory:

1. The stock of the product with `PRODUCT_ID` 42 for each shop.
2. The `PRODUCT_ID` of all products that have at least a stock of 0 in any store.
3. The `SHOP_ID` of all stores that have more than 15 products with a stock quantity less than 3.

Write a Python program that reads the inventory file and provides the marketing team with the requested data.

## 9.10] Base64 encoding

Base64 is a method of encoding binary-to-text, enabling the representation of binary or text data in ASCII format using a set of 64 characters.

Develop a Python program utilizing the built-in `base64` package to encode the content of the file `9.10.image_to_convert.jpg` to base64 and print the result to the user.

# Chapter 10] Object Oriented Programming

In this section, you'll be tasked with crafting several classes. For optimal organization, it is strongly recommended to structure the exercise by placing it within its designated folder. Each class should be implemented in its individual file, and these files can then be imported into the main file for cohesive execution.

Example :

```
10.9.simple_banking_system/
    .
    ├── venv/
    ├── main.py
    ├── bank.py
    ├── account.py
    └── transaction.py
```

## 10.1] Simple class

Create a class to represent a **cat**. This class must encompass the following attributes:

- **name**: a string denoting the cat's name.
- **age**: a float representing the cat's age.
- **color**: a string specifying the cat's color.
- **is_lazy**: a boolean indicating whether the cat is lazy.

Instantiate the class with the following values:

```
name="boubou"
age=8
color="orange"
is_lazy=True
```

## 10.2] Rectangle class

Design a class that models a **rectangle**. This class should possess two attributes: **width** and **height**. Additionally, include two methods:

- **area()**: This method calculates and returns the area of the rectangle, determined by multiplying the width and height.
- **perimeter()**: This method computes and returns the perimeter of the rectangle using the formula 2 * (width + height).

### 10.3] Class inheritance

Construct a class named `Person` with two attributes:

- **name**: a string.
- **age**: a float.

This class should include a method named `display_infos()` that produces the following output:

```
My name is {name} and I am {age} years old.
```

Create another class called `Student` that inherits from `Person`. This subclass should accept the necessary attributes to pass to its parent (`name` and `age`) and include two additional attributes:

- **student_id**: an integer.
- **study_topic**: a string.

The `Student` class should override the `display_infos()` method. Within the overridden method, utilize `super()` to invoke the parent class's `display_infos()` method and then generate the following output:

```
My student_id is {student_id} and I study {study_topic}.
```

This setup allows the `Student` class to customize the display message while leveraging the functionality of the `Person` class.

Example code :

```
alice = Person("Alice", 21)
alice.display_infos()

bob = Student("Bob", 24, student_id=424242, study_topic="Computer Science")
bob.display_infos()
```

It produces the following output :

```
My name is Alice and I am 21 years old.
My name is Bob and I am 24 years old.
My student_id is 424242 and I study Computer Science.
```

### 10.4] Shapes classes

Create an abstract class named `Shape` featuring a sole abstract method named `area()`, which is anticipated to yield a float as a result.

Define a class named `Circle` that inherits from the `Shape` class. This class should accept a single attribute: its radius as a float.

Implement the `area()` method within the `Circle` class using the formula `Area = 2 * math.pi * radius**2`.

Create another class called `Rectangle` that extends the `Shape` class.

It should take two attributes : **width** and **height**. Its area method should use the formula `Area = width * height`.

You can test your classes with this code :

```
circle = Circle(4.5)
print(circle.area()) # 127.234502

rectangle = Rectangle(3, 5)
print(rectangle.area()) # 15
```

### 10.5] Vector class

Define a class named `Vector2` to portray a two-dimensional vector. This class must encompass two attributes:

- **x**: a float indicating the x-component of the vector.
- **y**: a float indicating the y-component of the vector.

Override the magic methods to enable addition and subtraction operations between two instances of `Vector2`. Additionally, override the `__repr__` method to provide a clear and informative representation.

Check that the class is functional with the following code:

```
vec1 = Vector2(4, 5)
vec2 = Vector2(6, 5)

print(f"{vec1 + vec2 =}")
print(f"{vec1 - vec2 =}")
```

### 10.6] Simple dataclass

Use a `dataclass` to represent a **cat**. It should have the following attributes :

- name: str
- age: float
- color: str
- is_lazy: bool

Create an instance of the class using the following values :

```
name="boubou"
age=8
color="orange"
is_lazy=True
```

Print the created instance to the user.

### 10.7] Counter class

Design a class named `Counter` tailored for counting items within sequences such as dictionaries, lists, tuples, and strings. The class constructor should accept a single attribute called "sequence," representing the sequence on which item counting will be performed.

Implement the following methods:

- **get(item)**: This method returns an integer representing how many instances of the specified item were counted within the sequence.
- **counts()**: This method returns a dictionary containing associations of each unique item to its corresponding count within the sequence.

Example code:

```
counter = Counter("this is a str")
counter.get("i")  # Output: 2 (as there are only two "i")
counter.count()   # Output: {'s': 3, ' ': 3, 't': 2, 'i': 2, 'h': 1, 'a':
1, 'r': 1}
```

Test this class on the following sequences:

```
citation = "I think there I am... Or I think, I don't know"
random_numbers = [0, 1, 1, 2, 2, 3, 3, 4, 8, 5, 1, 2, 6, 4, 5]
```

Feel free to use and adapt the provided example code for testing.

## 10.8] Simple Banking system

Create three classes: `Bank`, `Account`, and `Transaction`.

1. **Bank Class:**
    - The `Bank` class is designed with a list to manage accounts.
    - It provides a `create_account()` method for generating and adding accounts to the list.
    - The class includes `get_account_balance(account_number)` to fetch the balance of a specific account.
    - Additionally, it features `make_transaction(account_number, amount)` to facilitate account transactions (deposits or withdrawals).
2. **Account Class:**
    - The `Account` class encompasses attributes: `account_number`, `account_holder`, and `balance`.
    - Its `__init__` method initializes these attributes.
    - It incorporates a method `display_account_info()` to print the account details.
3. **Transaction Class:**
    - The `Transaction` class holds attributes: `transaction_id`, `account_number`, `amount`, and `transaction_type` (deposit or withdrawal).
    - Its `__init__` method initializes these attributes.
    - It provides a method `display_transaction_info()` for printing transaction specifics.

**Example Usage:**

```
# Create a Bank
bank = Bank()

# Create Accounts
alice_account = bank.create_account("Alice")
bob_account = bank.create_account("Bob")

# Display Account Information
alice_account.display_account_info()

# Make Transactions
bank.make_transaction(alice_account.account_number, 500)  # Deposit €500
bank.make_transaction(bob_account.account_number, 200)  # Deposit €200
bank.make_transaction(alice_account.account_number, -300)  # Withdraw €300

# Display Account Balances
print(f"Account 1 Balance: €
{bank.get_account_balance(alice_account.account_number)}")
print(f"Account 2 Balance: €
{bank.get_account_balance(bob_account.account_number)}")

# Display Transactions
```

```
bank.display_transactions()

# Display Transaction History for an Account
bank.display_account_transactions(alice_account.account_number)
```

## 10.9] Tic tac toe game

Create a Tic Tac Toe game with the following classes:

1. **Board:**
    - The `Board` class represents the game board.
    - It should have a 3x3 grid to store X, O, or an empty space.
    - Implement a method to display the current state of the board.
    - Include a method to check if a player has won.
    - Create a method to check if the board is full (a tie).
2. **Player:**
    - The `Player` class represents a player in the game.
    - It should have attributes for the player's name and symbol (X or O).
3. **Game:**
    - The `Game` class orchestrates the game.
    - It should have instances of the `Board` and two `Player` objects.
    - Implement a method to start the game and alternate turns between players.
    - Include a method to accept player moves and update the board.
    - Create a method to check for a winner or a tie.
    - Display the winner or declare a tie at the end of the game.

**Example Usage:**

```
# Create a Tic Tac Toe Game
game = Game("Player 1", "Player 2")

# Start the Game
game.start_game()

# Game Moves
game.make_move(0, 0)  # Player 1 moves to (0, 0)
game.make_move(1, 1)  # Player 2 moves to (1, 1)
game.make_move(0, 1)  # Player 1 moves to (0, 1)
game.make_move(1, 0)  # Player 2 moves to (1, 0)
game.make_move(0, 2)  # Player 1 moves to (0, 2)

# Display the Board
game.display_board()

# Check for Winner or Tie
game.check_winner_or_tie()
```