# Python Exercises

# Exercises

## 1. Types & Syntax

### Exercise: Print the type

Can you print the type of each of those?

```
first_name = "Peter"
age = 10
```

### Exercise: Integer value

Can you tell what value *a* and *b* have at the end?

```
a = 3
b = 6
a = b
b = 7
```

- If you multiply their value together what will be the output? Do it.
- What type would it have if you divide *b* with *a*? Do it.
- What happens if we set *a* at 0 and divide *b* with *a*? Do it.

### Exercise: Convert a variable from one type to another

Can you fix this code?

```
number = 15
print("The number is " + number)
```

Expected output: "The number is 15".

### Exercise: Print with separator

Print the value of a, b and c separated with a "+".

```
a = 2
b = 6
c = 3
```

Expected output: 2+6+3

## Exercise: Error in loop

What is the error in this code? Can you fix it?

```
wd = "Python"
for i in len(wd):
    print(i)
```

## Exercise: Create a list of numbers from 5 to 15

Write code for creating a list of numbers between 5 and 15.

Expected result: `[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]`

## Exercise: Create a list of even numbers from 1 to 100

Write code for creating a list of only even numbers between 1 and 100.

Hint: check the argument 'step' of *range*, read `help(range)`.

Expected result: `[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]`

## Exercise: Count the number of occurrences of a letter in a sentence

Find the number of occurrences of letter "l" with a loop.

```
s = "Hello Ladies and Gentlemen"
```

## Exercise: Remove duplicates from a list

Write code for removing duplicate from a list.

```
numbers = [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 7, 8, 9, 10]
numbersOnlyOnce = []
```

Expected result: `[1, 2, 3 ,4, 5, 6, 7, 8, 9, 10]`

## Exercise: Error in function

What is the error in this code? Can you fix it?

```
def addition(a, b):
    c = a + b

result = addition(5, 10)
print(result)
```

## Exercise: Sum of a list of number

Create a function which calculates the sum of a list of numbers.

```
mysum([1, 2, 3, 5]) # 11
```

## Exercise: Multiplication tables

Write a function which prints the multiplication table of a number. It should take the number in argument and the maximum number to which to go as an optional argument with a default value of 10.

```
multiTable(num, max)
```

## Exercise: Retrieve elements common to two lists using a loop

Write a function which returns a new list which contains only the numbers common to the two lists.

```
list1 = [1, 5, 6, 7, 9, 10, 11]
list2 = [2, 3, 5, 7, 8, 10, 12]
mycommon(list1, list2) # [5, 7, 10]
```

## Exercise: Adding up the numbers of a number

Write a function which adds all the numbers of a number:

```
number = 209812
numaddnum(number) # 22
#results: 2+0+9+8+1+2=22
```

## Exercise: Even number of a list

Write a function which, given a list of arbitrary numbers, will return a new list containing only the numbers which are even.

```
evenNum([1, 2, 3, 4]) # [2, 4]
evenNum([100, 0, -2, 3, 8, 9, 0]) # [100, 0, -2, 8, 0]
```

## Exercise: Clean a list

Write a function which takes a list in argument and return a new list with only the elements which are strings.

```
l = [10, 2329, 5, "Pierre", 203, "Marie", 867, "Adrien"]
cleanList(l) # ["Pierre", "Marie", "Adrien"]
```

## Exercise: half-diamond

Write a function which prints a half-diamond using a loop and print. The argument should be the maximum length of the half diamond.

```
halfDiamond(8)
*
**
***
****
*****
******
*******
********
*******
******
*****
****
***
**
*
```

## Bonus

- Why do we use underscores before the name of variables, functions, … in python?
- Is it possible to force the type of a parameter in a function argument?
- Look at sphinx documentation, setup one for your project. Write documentation for each of your function for using the autodoc feature of sphinx.

# 2. Standard Data Types

### Exercise: dictionary value

Retrieve the "last_name" of the dictionary:

```
employees = {"01": {"id": {"first_name": "Peter", "last_name": "Smith"}}}
```

### Exercise: Replace a word by another

Hint: look at the *replace* method.

```
sentence = "Good morning, everybody."
#result: Good evening, everybody.
```

### Exercise: Ordering a string

Sort names in alphabetical order.

```
str1 = "Peter, Julian, Mary, Luke"
```

Hints:

- For separating a string, use the *split* method.
- For concatenate string with a separator, use the *join* method.

Expected result: `"Julian, Luke, Mary, Peter"`

### Exercise: Count the number of occurrences of a letter in a sentence

Write a function which finds the number of occurrences of a letter in a sentence. Handle the case for upper and lower case.

Hints:

- To get lowercase of a string, use the *lower* method.
- To get number of characters in a string, use the *count* method.

```
s = "Hello Ladies and Gentlemen"
mycount(s)
```

### Exercise: Replacing an item in a list

Replace an element in a list. Write a function for doing this.

```
my_list = ["Pierre", "Marie", "Julie", "Adrien", "Julie"]
name_to_find = "Julie"
new_name = "Julien"
```

## Exercise: Retrieve elements common to two lists using a set

Using a set find the common elements. There is no need to write a function.

```
list1 = [1, 5, 6, 7, 9, 10, 11]
list2 = [2, 3, 5, 7, 8, 10, 12]
# [5, 7, 10]
```

## Exercise: Sort a list of tuples

Sort the list of tuples by the number which is the second element of the tuple.

```
l = [("Harry Potter", 5), ("Wall-E", 3), ("Blade Runner", 4)]
print(sort_tuples(l))
# [('Wall-E', 3), ('Blade Runner', 4), ('Harry Potter', 5)]
```

## Exercise: From a list of tuples create a dictionary

Write a function which transforms a list of tuples to a dictionary. Use the second elements of the tuples as keys.

```
l = [("Harry Potter", 5), ("Wall-E", 3), ("Blade Runner", 4)]
print(tu2dict(l))
# {3: 'Wall-E', 4: 'Blade Runner', 5: 'Harry Potter'}
```

## Exercise: Sum the values of the dictionary

Write a function which calculates the sum of all the value of a dictionary if those are integers.

```
employees = {"Peter": 2500, "Mary": 5000, "Julian": 1200, "Julia": "NotANumber"}
```

## Bonus

- What is a decorator?
- Can you find the function arguments name when using a decorator?

# 3. File Manipulation, Input & Outputs

### Exercise: manipulate path

Find:

- The basename of the file.
- The extension of the file.
- The directory in which the file is.

```
import os
my_path = "/xxx/yyy/myfile.txt"
```

### Exercise: Parse a CSV file

Write a function which parse a CSV file and return its content as a matrix (a list of list). A CSV file separate each line by a newline and each value separated by a comma.

For the file:

```
hello,world,!
how,are,you?
```

the expected result is: `[["hello", "world", "!"], ["how", "are", "you?"]]`

### Exercise: Would you like to continue?

Write a function which ask the user if he wishes to continue, if the answer is "yes" the function return True, if the answer is "no" the function return False, otherwise it asks again.

# 4. Mini-project: Tic-tac-toe

The goal of this mini-project is to implement a tic-tac-toe game where two players can play. This is not guided and you have to choose how you make the development by yourself.

Before starting, ask yourself a few questions:

- How are you going to store information on the table?
- How many states are possible for each square?
- What information are you going to keep about players?
- What does a round of tic-tac-toe looks like? What are the steps to do?
- When and how will you determine who wins?

You should probably start by taking a paper and think of the different steps and functions you have to implement.

Bonus:

- Develop a simple "Artificial Intelligence" for the game.
- Develop a client-server architecture for playing across a network.
- Implement a graphical or web interface for your game.
- Implement another simple game.

# 5. Object Oriented Programing

## Exercise: database of books

Write simple classes for representing authors and books.

The book class should have the following properties:

- Title
- Authors (list of authors)
- ISBN number
- a boolean indicating if it has been read
- the possibility to make a comment.

The author class should have the following properties:

- Name
- First name
- birth date
- death date
- A is_alive property

Implement:

- A method for printing a book.
- A method for storing the book information in a file.
- A method for loading the book information from a file.
- A method for adding a comment.
- A method for storing the author information in a file.
- A method for loading the author information in a file.

Bonus:

- Implement a library class which can contain books and allow to search by authors.
- Support the iteration on the books using the __*iter*__ method.
- Write a way to access books and authors using the __*dict*__ method in your library.

## Exercise: write classes for transport

Write simple classes for representing different transport. You should have a main transport class from which other inherit. You should represent:

- Car
- Motorcycle
- Plane
- Boat

Each of those should have a name, a speed, and method for printing them. Some of those should have a *fly* method, other a *drive* method.

## Exercise: complex number

Write a class which implement complex numbers. Support printing the number. Support mathematical operation (addition, subtraction and multiplication) with a normal integer or another complex number.

## Exercise: Tic-tac-toe improvement

Retake your tic-tac-toe game and create classes for players, game/boards and cases.

## Bonus:

- What is a class decorator? Try implementing one.
- What is a metaclass? When can it be of use? Try using one.

# 6. Modules

### Exercise: Random

1. Write a loop to get 10 rolls of dice, use the *random* module to generate a random number.
2. Create a new IA for your tic-tac-toe which plays at a random place.

### Exercise: Store & load with json

Using the *json* module, implement functions for storing and loading books from files containing json. Handle the exceptions correctly and the cases when not enough information is given.

### Exercise: MyList Module

Take all the functions you have written about list and create a module *mylist* which contains all of them.

### Bonus

- Look at the *request* module, try to use it for making internet request.
- Look at the *pytest* module, try to implement some unit testing for your tic-tac-toe.
- Look at the *sqlite* or *sqlalchemy* module, write a function for your database of books which allow to store and read into a *sqlite* database.
- Look at the *numpy* module, try to use it.

# Project: Sudoku Solver

The goal of this project is to write a Sudoku Solver. This is not guided and you have to choose how you make the development by yourself.

The sudoku are stored in files that you have to read. It should be possible to give the path to the file(s) to your script. Once your algorithm found the solution, it should write it in a corresponding file.

Before starting, ask yourself a few questions:

- How are you going to store information on the sudoku?
- How are you going to find the possible answer for each case of the sudoku?
- What is your overall algorithm for solving a sudoku will look like?
- What functions and class do you need to write?
- What existing modules can you use?

The *grid.py* file contains some cases which can help you for testing.

Bonus:

- Make a generator of sudoku.
- Allows a user to play on your sudoku.
- Make a web interface for it.
- Using *sqlalchemy* try to store the result of your sudoku.
- Make a chess game with an IA for it.