FERREIRA Pablo
BERDERY Ambre
L1-INT3
2023/2028

# PYTHON PROJECT REPORT
My first chatbot

## INTRODUCTION:

Our project, a chatbot inspired by models found all over the Internet(such as ChatGPT that we all know), stands out for its specialization in the analysis of French political speeches.

Our central objective was to simplify access to political information by offering users a chatbot capable of analyzing speeches, selecting data out of it and answering specific questions in function of their content, and doing so in an accessible way. In other words, developing a system that can answer questions based on the frequency of words in the corpus of the speeches.

### Goals

- Understand and apply basic concepts we've seen in python class
- Discover the methods used to develop an IA based chatbot
- Bring concrete answers to users requests

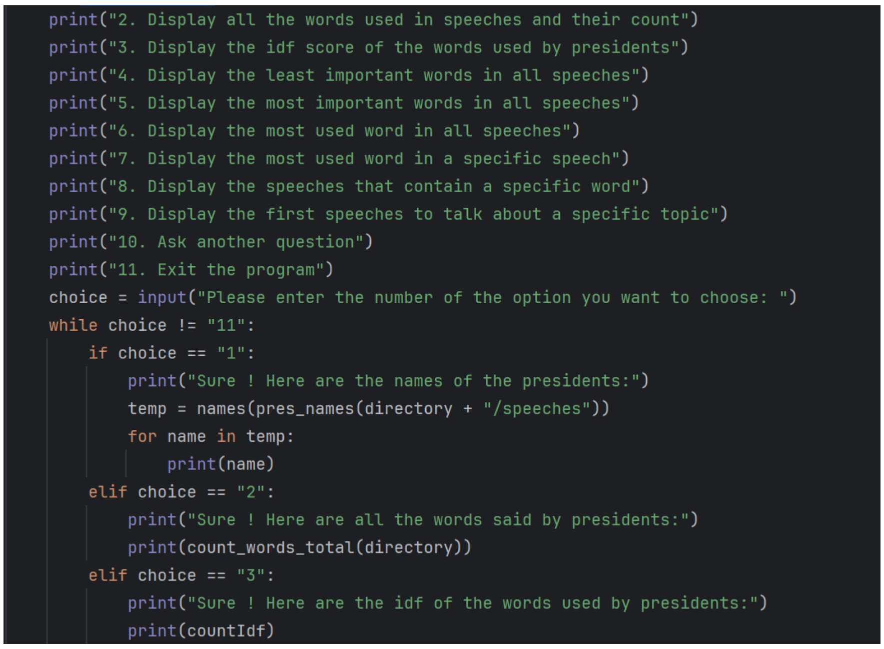
The chatbot is expected to act as a virtual companion, demystifying political speeches and making politics more understandable for the users. We've put the emphasis on user comprehension and simplicity, because politics can sometimes seem complex. Our project thus represents the crossroads between technology and politics, aiming to make the subject more accessible. We hope this approach will appeal to you as much as it did to us!

## TABLE OF CONTENTS

## I.1. Paradigms and method

To solve this problem, we had to use two main paradigms.

Our main.py is using the procedural-imperative concept by doing simple comparisons and function call:



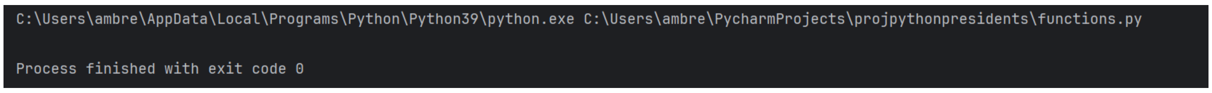
```
print("2. Display all the words used in speeches and their count")
print("3. Display the idf score of the words used by presidents")
print("4. Display the least important words in all speeches")
print("5. Display the most important words in all speeches")
print("6. Display the most used word in all speeches")
print("7. Display the most used word in a specific speech")
print("8. Display the speeches that contain a specific word")
print("9. Display the first speeches to talk about a specific topic")
print("10. Ask another question")
print("11. Exit the program")
choice = input("Please enter the number of the option you want to choose: ")
while choice != "11":
    if choice == "1":
        print("Sure ! Here are the names of the presidents:")
        temp = names(pres_names(directory + "/speeches"))
        for name in temp:
            print(name)
    elif choice == "2":
        print("Sure ! Here are all the words said by presidents:")
        print(count_words_total(directory))
    elif choice == "3":
        print("Sure ! Here are the idf of the words used by presidents:")
        print(countIdf)
```

main.py, line 25 to 47

It uses a lot of iterative methods (for I in range, for cell in tab…), comparisons and loops.

---

"function.py" has a functional paradigm as its structure and does nothing alone: its sole purpose is to be called by main, like a library.

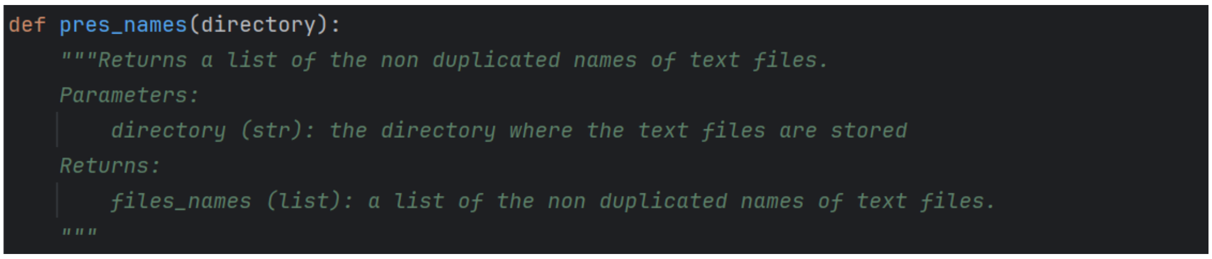


```
C:\Users\ambre\AppData\Local\Programs\Python\Python39\python.exe C:\Users\ambre\PycharmProjects\projpythonpresidents\functions.py

Process finished with exit code 0
```

when executing functions.py

## I.2. Functions used

These functions can call other functions, but don't do recursive or object-oriented computations for harmonization purposes. All these functions are rigorously documented with this structure:



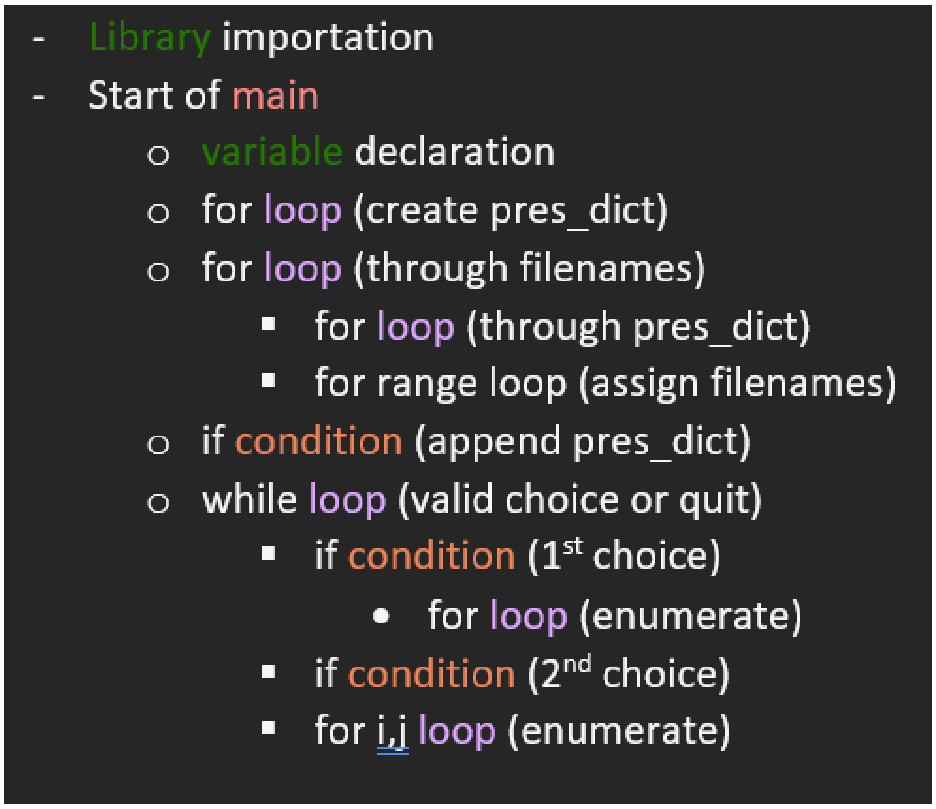
```python
def pres_names(directory):
    """Returns a list of the non duplicated names of text files.
    Parameters:
        directory (str): the directory where the text files are stored
    Returns:
        files_names (list): a list of the non duplicated names of text files.
    """
```

- The 1st paragraph indicates the main goal of the function
- The 2nd paragraph is the necessary parameters or inputs, with the type needed written between parenthesis
- The 3rd one gives what will the function return or display at the end or if it doesn't return anything

We have used functions such as `os, listdir, getcwd,path` to access the files in the directory containing the speeches, the `math` function to help with the logarithm that we needed to develop the matrix and calculate the tf-idf scores and the `remove` function to remove some elements in lists or dictionaries that we've created in our functions.

## II.1. Code structure

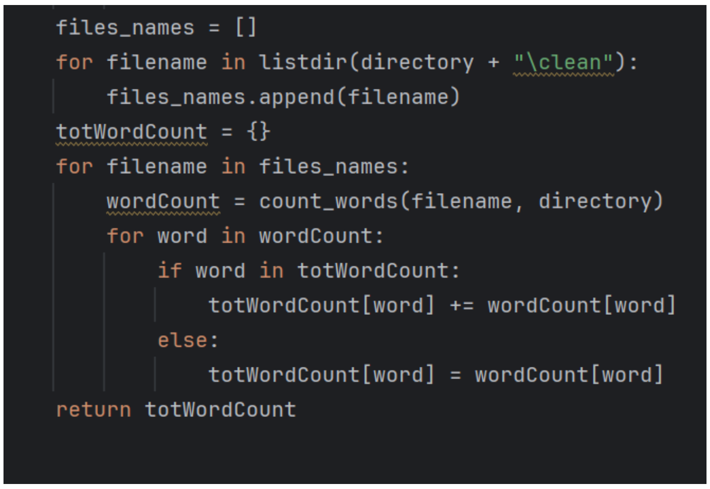
Here is a sample of the code structure of our main:



- Library importation
- Start of main
    - variable declaration
    - for loop (create pres_dict)
    - for loop (through filenames)
        - for loop (through pres_dict)
        - for range loop (assign filenames)
    - if condition (append pres_dict)
    - while loop (valid choice or quit)
        - if condition (1st choice)
            - for loop (enumerate)
        - if condition (2nd choice)
        - for i,j loop (enumerate)

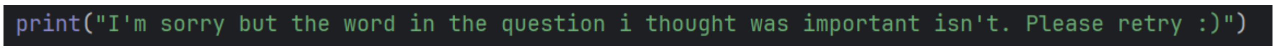
## II.2. Data structure

We used lists and dictionaries to store and manipulate data that was linked to the words in each speech. For example, here we have used a list of strings to store each word in each file, which allowed us to count the words of each file in a dictionary.

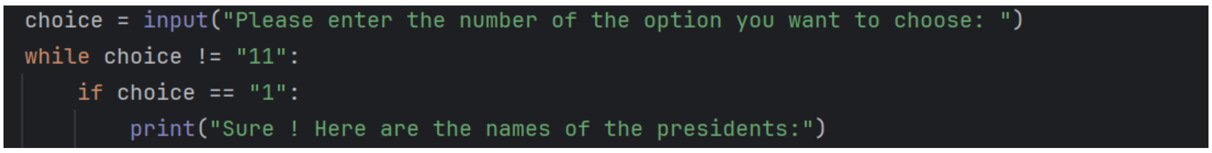


```python
    files_names = []
    for filename in listdir(directory + "\clean"):
        files_names.append(filename)
    totWordCount = {}
    for filename in files_names:
        wordCount = count_words(filename, directory)
        for word in wordCount:
            if word in totWordCount:
                totWordCount[word] += wordCount[word]
            else:
                totWordCount[word] = wordCount[word]
    return totWordCount
```

We also used strings as a way to make the chatbot more interactive, to communicate with the user, enable them to make requests and understand the main functionalities of the chatbot.

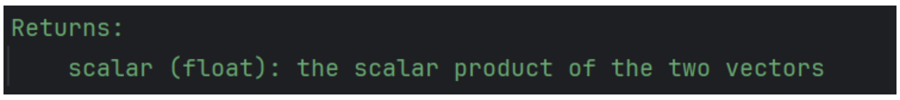


```python
print("I'm sorry but the word in the question i thought was important isn't. Please retry :)")
```



```python
choice = input("Please enter the number of the option you want to choose: ")
while choice != "11":
    if choice == "1":
        print("Sure ! Here are the names of the presidents:")
```

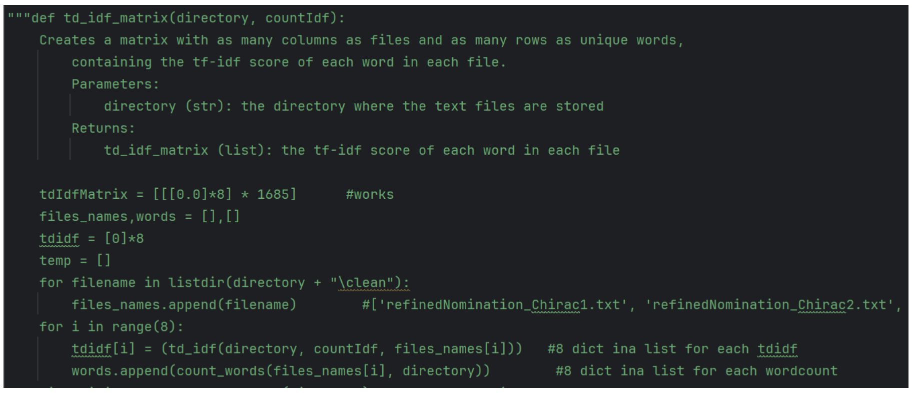
Floats were also useful to make some calculations, e.g. when we wanted to calculate vector coordinates.



```
Returns:
    scalar (float): the scalar product of the two vectors
```

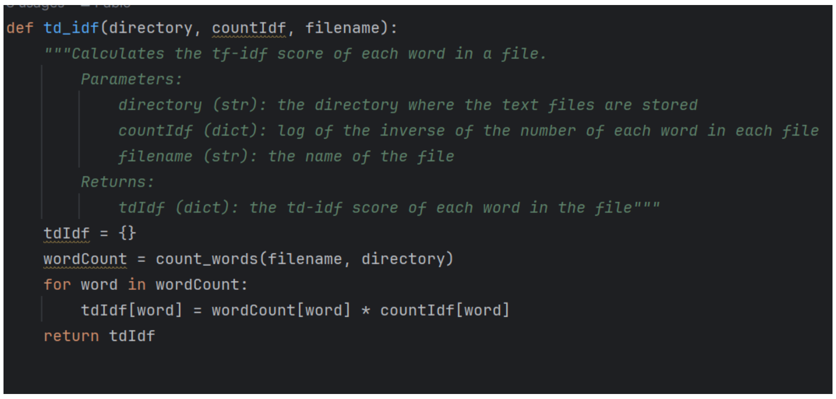
## II.3 Technical difficulties

We’ve encountered some difficulties at some point in the realization of the code. Indeed, we struggled with the tf-idf matrix: there was a problem with the previous indexes being replaced by the last value computed when displaying the elements of the matrix.



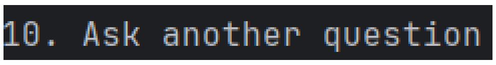
```
"""def td_idf_matrix(directory, countIdf):
    Creates a matrix with as many columns as files and as many rows as unique words,
        containing the tf-idf score of each word in each file.
        Parameters:
            directory (str): the directory where the text files are stored
        Returns:
            td_idf_matrix (list): the tf-idf score of each word in each file

    tdIdfMatrix = [[[0.0]*8] * 1685]      #works
    files_names,words = [],[]
    tdidf = [0]*8
    temp = []
    for filename in listdir(directory + "\clean"):
        files_names.append(filename)        #['refinedNomination_Chirac1.txt', 'refinedNomination_Chirac2.txt',
    for i in range(8):
        tdidf[i] = (td_idf(directory, countIdf, files_names[i]))   #8 dict ina list for each tdidf
        words.append(count_words(files_names[i], directory))        #8 dict ina list for each wordcount
```

As the problem was not coming from the code, we decided to replace this matrix by another process:



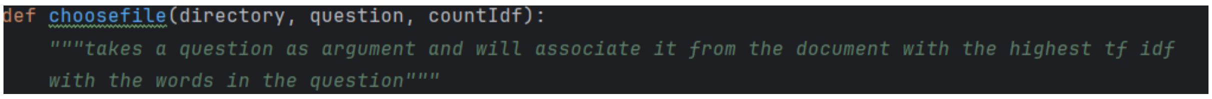
```
def td_idf(directory, countIdf, filename):
    """Calculates the tf-idf score of each word in a file.
        Parameters:
            directory (str): the directory where the text files are stored
            countIdf (dict): log of the inverse of the number of each word in each file
            filename (str): the name of the file
        Returns:
            tdIdf (dict): the td-idf score of each word in the file"""
    tdIdf = {}
    wordCount = count_words(filename, directory)
    for word in wordCount:
        tdIdf[word] = wordCount[word] * countIdf[word]
    return tdIdf
```

Furthermore, one of the options proposed in the menu, (more precisely the 10th) is supposed to do the following:



```
10. Ask another question
```

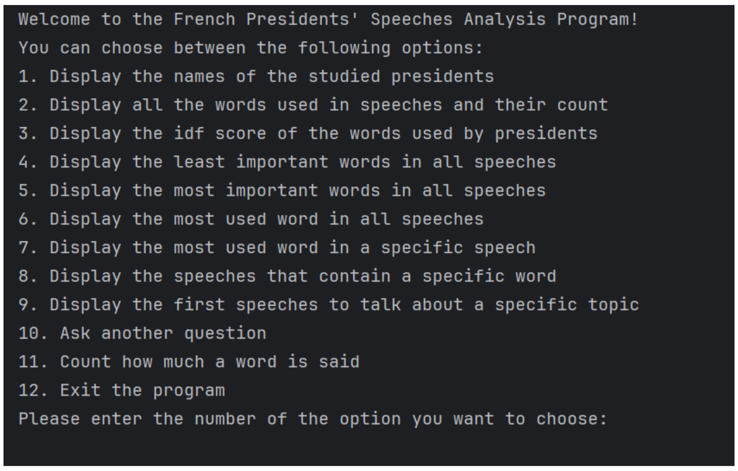
It allowed the user to ask a question about a subject, and the program had to understand with the help of tf-idf matrix hat was the important word in the question So, we had to remove unimportant words during the computing of the question, to target the exact topic the users was asking about:



```
def choosefile(directory, question, countIdf):
    """takes a question as argument and will associate it from the document with the highest tf idf
    with the words in the question"""
```

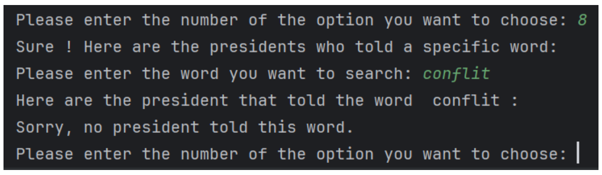
## III.Results presentation

We were finally able to offer accessible functionalities to the user. Here are some tests of the options given to the user:

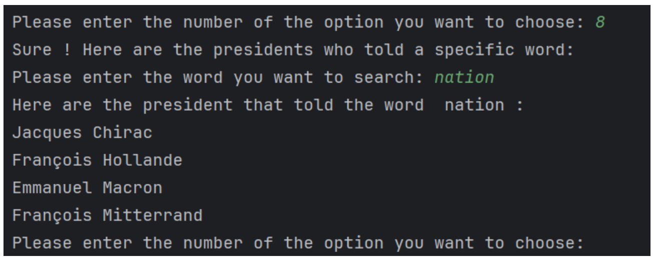


```
Welcome to the French Presidents' Speeches Analysis Program!
You can choose between the following options:
1. Display the names of the studied presidents
2. Display all the words used in speeches and their count
3. Display the idf score of the words used by presidents
4. Display the least important words in all speeches
5. Display the most important words in all speeches
6. Display the most used word in all speeches
7. Display the most used word in a specific speech
8. Display the speeches that contain a specific word
9. Display the first speeches to talk about a specific topic
10. Ask another question
11. Count how much a word is said
12. Exit the program
Please enter the number of the option you want to choose:
```

EX: cases of unused and used words by the presidents with the the 8th option:

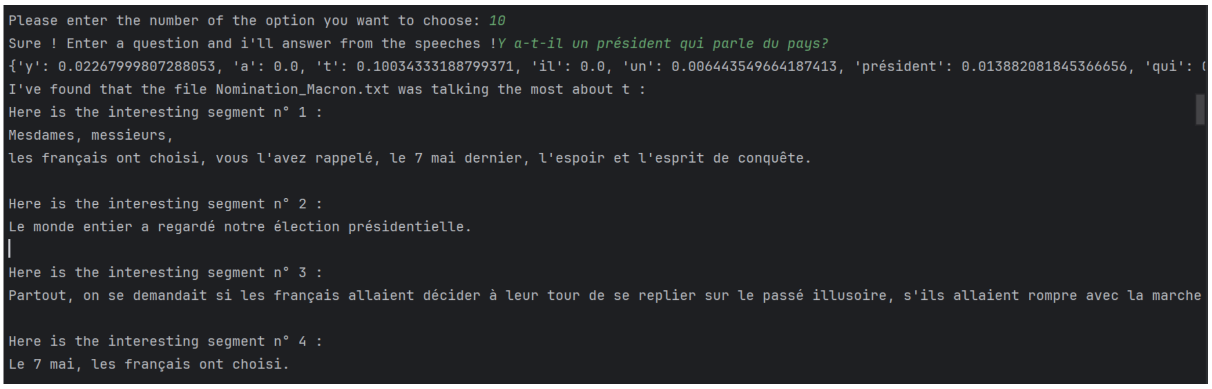


```
Please enter the number of the option you want to choose: 8
Sure ! Here are the presidents who told a specific word:
Please enter the word you want to search: conflit
Here are the president that told the word  conflit :
Sorry, no president told this word.
Please enter the number of the option you want to choose:
```



```
Please enter the number of the option you want to choose: 8
Sure ! Here are the presidents who told a specific word:
Please enter the word you want to search: nation
Here are the president that told the word  nation :
Jacques Chirac
François Hollande
Emmanuel Macron
François Mitterrand
Please enter the number of the option you want to choose:
```

Test of the 10th option:



```
Please enter the number of the option you want to choose: 10
Sure ! Enter a question and i'll answer from the speeches !Y a-t-il un président qui parle du pays?
{'y': 0.02267999807288053, 'a': 0.0, 't': 0.10034333188799371, 'il': 0.0, 'un': 0.006443549664187413, 'président': 0.013882081845366656, 'qui': (
I've found that the file Nomination_Macron.txt was talking the most about t :
Here is the interesting segment n° 1 :
Mesdames, messieurs,
les français ont choisi, vous l'avez rappelé, le 7 mai dernier, l'espoir et l'esprit de conquête.

Here is the interesting segment n° 2 :
Le monde entier a regardé notre élection présidentielle.

Here is the interesting segment n° 3 :
Partout, on se demandait si les français allaient décider à leur tour de se replier sur le passé illusoire, s'ils allaient rompre avec la marche

Here is the interesting segment n° 4 :
Le 7 mai, les français ont choisi.
```

## IV. Conclusion

We learned to work in groups, which included understanding each other's work, helping each other out and sharing tasks via github.
We then had to make our work more clear, so that we could understand each other's code and the changes made to it. One of the main problems was the use of certain variables, and now we understand their crucial use.
In addition, we essentially formed a group of hybrid pairs, so one of us learned a lot about the main python concepts such as libraries, functions, lists: their uses and purposes.
We also learned to constantly test our code with normal and unexpected values and variable types. This made our work a lot easier and saved us a lot of time, and we think this is one of the main lessons we've learned: we actually do a lot more debugging than coding.
Finally, this project has been a gain in experience and knowledge, in all its aspects: this pointer issue we encountered, a memory optimization one that we resolved and the organization and this long code allowed us to learn things that cannot be written in books.