**FIRST PROJECT: THE HANGMAN**

After some courses about how programming in python, we were asked to do our first project. All along this report, you will discover which project we had to do, how we did it, what were our difficulties encountered, an analysis of our project and a conclusion at the end.

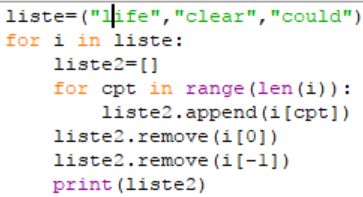
We were asked to do a hangman in python. In addition, we had to do 15 algorithms to train ourselves so that we can do the project. It was not only a hangman, in fact we had to do the different steps:

-Let the user choose if we wanted a man or a woman

-Display the letter he has already used and the evolution of the word that the user had to discover

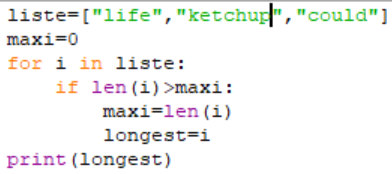
-Display the man or the woman being hanged

-Let the user decide if he wants to play again or not



3-A

We convert each string of liste in a list, then we remove the first and the last character of the corresponding list. Finally, we print the list corresponding to the string. After each modified string we reset the second list for the next string.

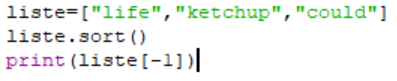


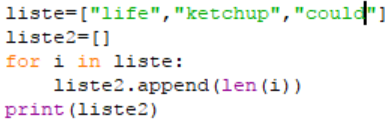
3-B

At first, we initialize maxi at 0. Then we compare the length of each word compare to our variable “maxi”. If the length of this word is greater than our variable maxi, we enter its length in maxi and then we enter i in the variable “longest”. Finally, we print longest.

3-C

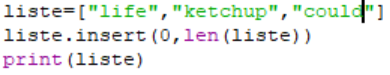
Thanks to the function “liste.sort()”, we sort the word of our liste in the alphabetic order and finally, we print the last word of our list.





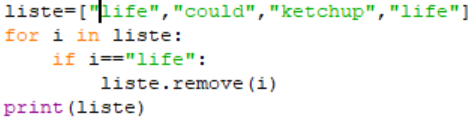
3-D

At first, we create a new list, then we add to this list the length of each word in our first list. Finally, we print the second list and the length of each word appear.



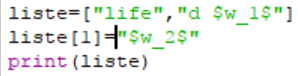
3-E

We use the fonction « liste.insert » so that we add at the first position of our initial list the number of items. Then we print the list which contains now the number of items and the items which were already there.



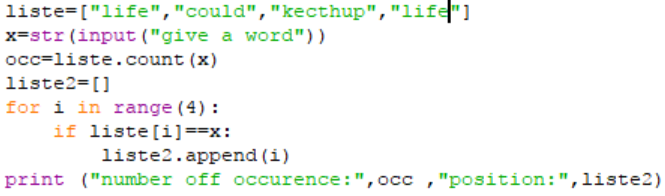
3-F

We create a loop, and we compare each word to the word we want to remove from the list. If the word compared is the same, it is removed from the list. Finally, we print the list.



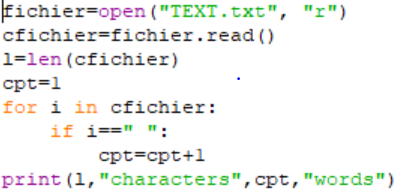
3-G

We change the word we want by specifying the index of this word and then we enter the new word we want. Finally, we print the list.



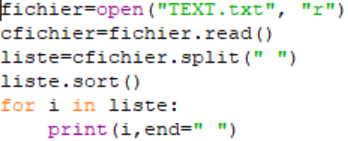
3-H

At first, we ask the user on which word does we want to know the occurrences and their position. Then, we create a variable “occ” which correspond to the number of occurrences of the word the user entered thanks to the function “liste.count()”. Then we create a second list and we make a loop in which we compare every word of our list with the word the user entered. If they are similar, we add the position of the word in our second list. Finally, we print the variable “occ” and the second list so that it displays the occurrences of the given word and their position.



4-A

First, we open the file, then we compute his length with len function, so we have the number of characters. Then we create a counter of space at 1, because there is no space before the first word and then we add 1 to this counter when there is a space in the file, the value of the counter will be the number of words

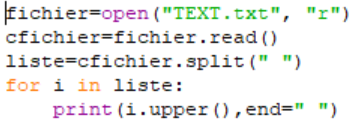


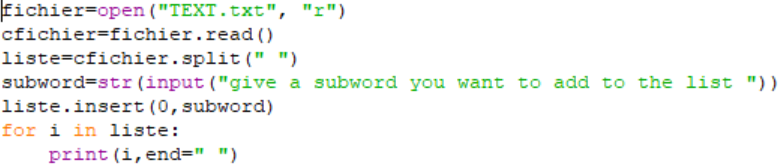
4-B

We open the file, then we create a list of this file with the. split function, we consider that each term of the file separate by a space will be a term of the list. Finally, we sort the list and print the words which have been sorted.

4-C

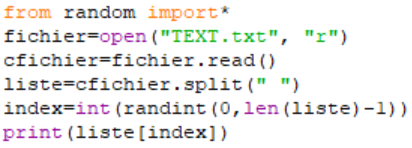
We open the file, then we create a list of this file with the. split function, we consider that each term of the file which is separated by a space will be a term of the list. Then we use function. upper to convert each string of the list in uppercase





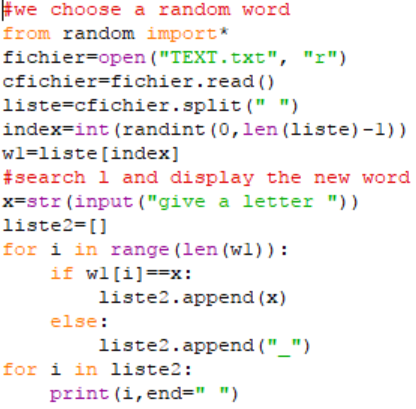
4-D

We open the file, create a list of the file, then we ask the user to give us a subword to add to this list and we insert it at the first position of the file.



4-E

We import random function, we open the file, we convert it in list. Then we use randint function to choose the index of the word to print. Finally, we print the random word.



4-F

First part is the last program, we use that to choose the word we will use in the new program.

Now we have a word, we ask the user to give a letter, then we create a list of the length of the word choose randomly with only “\_” and we search if the letter given by the user is in the word, if we find it we don’t add a: ”\_” but we add the letter to the list. Then we print the list.

The main difficulties that we have encountered:

-The layout of all the algorithms

-The paths to import the files to display the evolution of the hangman, at first, we didn’t understand how to make it well

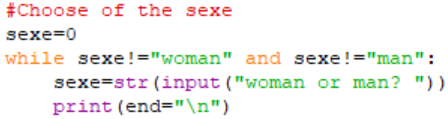
-The algorithm which permits to print the letter at the place of a “- “was particularly hard to do.

-For the algorithms of the beginning (to train ourselves) we had to search on internet to find some functions we didn’t know before, for instance; “. split” and “. upper”

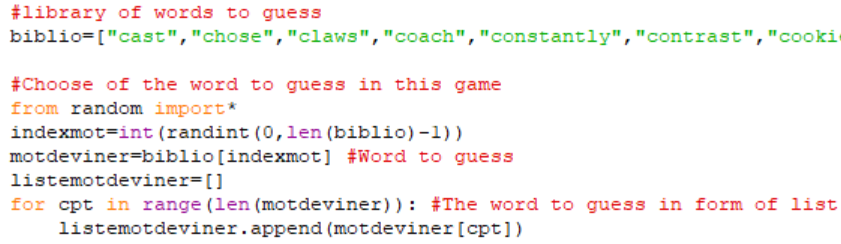
Analysis of the project:

1. We initialize a boolean condition so that all the algorithm will run again or not.

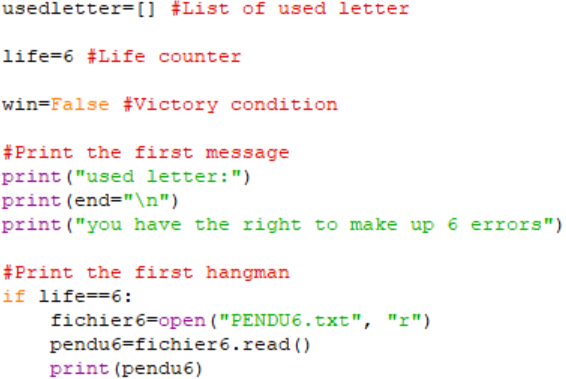




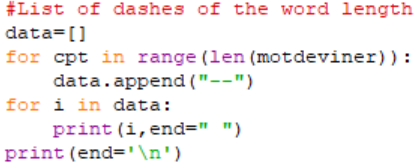
2) We ask the user if he wants to play with a man or a girl thanks to a while loop. And because of this while loop, even if the user enters any words different than “man” or “woman” the algorithm will ask him again to choose the sexe.



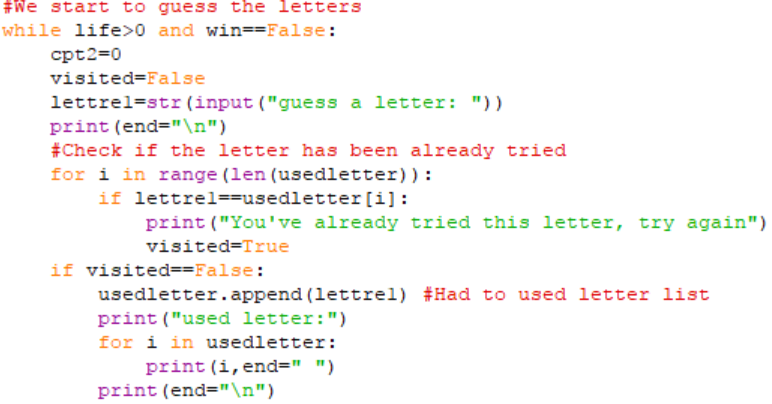
3) We initialize our library of words. Then, we choose a random number in the range of the length of the biblio list, with the randint function. This random number will be used as the index of the word that will be the word to guess in the game(motdeviner). Then, we convert the chosen word in a list(listemotdeviner), to make the following step easier.



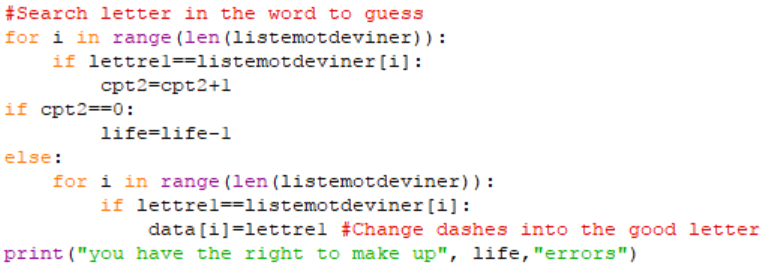
4) We initialize some variable for the future algorithm, and we print the beginning menu out of the while because we want it to be here at the start of the program.



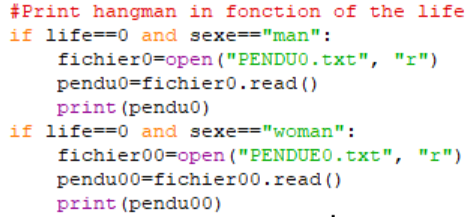
5) Now we create a list of dashes of the length of the word to guess, we will modify this list if the user find a good letter in the following of the algorithm.



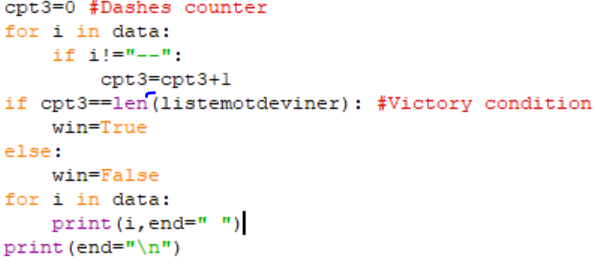
5) We create a while loop which will work since the user is not hanged and while he doesn’t win. We start make user guess letters, first we search if the letter has been already used by searching if this letter is in the list of usedletter, if the letter is in this list the visited become true, and we ask the user for another letter. Else we add this letter to the usedletter list. Last, we print the list of used letter in way that the user know what letters he already used.



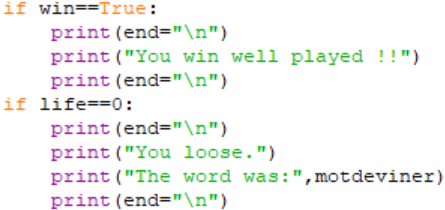
6) Now we search if the letter given by the user is in the word to guess, by comparing the letter given with all the letters in the word. If the letter is in the word, we change the dash in data list by the letter at the index where the letter is in the word to guess. Else the user loses a life. Last, we print the number of errors that the user can still make.



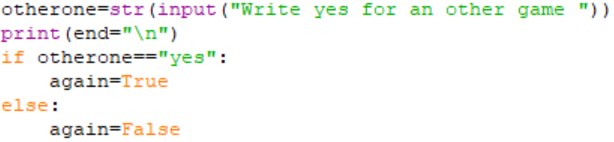
7) Now we print the step of the hangman in function of the number of life and the sexe. To do this we open a file with the hangman drawn. We do this for all the number of life with different file.



8) We initialize a counter to count the number of dashes in the list of dashes, if at the end of the count it is equal to 0 then win become true and the while loop in 5) stop, win stay false. Then we print the list of dashes with letter or not.



9) We check if the boolean condition is true, if it is true we print a message of victory. We also check the number of life and if it is equal to zero we print a message of defeat and what was the word to guess.



10) We ask the user if he wants to play again or not, if he enters yes then the boolean condition of 1) stay true and the algorithm restart. Else again become false and the algorithm stops.

To put in a nutshell, during this project we made our first huge program. To do this we have trained ourselves with small programs which have been useful to make the hangman. So, for us the hardest part of this project has been the 1st part of the project because we had to use tricks and functions that we have never seen before. Therefore, after this part of discovery, making the hangman was not so long and hard, in fact it was the most exciting part of the project. We have learnt so many things thanks to this project. We learnt for instance to work in team, share tasks, not being upset to the first difficulty. These behaviours are important for a such project and we have adopted them in the best way we could. Finally, we didn’t take this project like a homework we have to do, it was a real pleasure to make this project and we hope that the second one will be even more exciting than the first.