# Game : Message Locker

## Plan

1. One member creates the project
2. Do the product planning

* Plan game, rules,

1. Flowchart
2. Pseudo-code and repartition

* Definition of names of variables/lists/dicts

1. Coding
2. Presentation (with slides)

## Pseudocode

### Word highlight key

Green = agreed object names

Blue = agreed function names

* **Protocol message function** [1 person]
* Just in english
* Limit special characters
* Explain how the program works in message
* **Encryption key function:** [1 person]
* Create a dict\_encrypt 🡺 [character, encrypted\_character]
  + Challenge on deciding if we use more than 1 encryption key, and if so, the implications on the rest of the code
  + Challenge : random encryption needs to be unique to ensure the decryption
* **def encryption\_process (kaci)**

[1 person]

* msg = Input()
* Encryption process : iterate through each letter of the message (check if need to convert to a list first) + return value from encryption\_dict by maching its key
* Output1 : a string or list (if list, need to convert to string)d
* Display encrypted message
* Export txt file
* Name of the string variable encrypted\_msg

[1 person]

* **Username and password feature** (discussed but eventually dropped due to time constraints)
* Input : username + password
* Google Get.pass function
* Store username + password in a dictionary dict[key,value] 🡺 userdetails[username, password]

dict\_pwd

* **def decryption\_process**

challenge : need to iterate the input encrypted text by 3, to match the encryption key

[1 person]

* Input1 : incripted text encr\_txt=(input)
* Input 2 : username + password username, password (2 different values)
* Match username + password in dict\_pwd: (discussed but dropped)
* do you want to try again
* if not, end

[1 person]

* Decryption process
* Output : decrypted message (we do not store it)
* Goodbye message : thank you for using blablabla