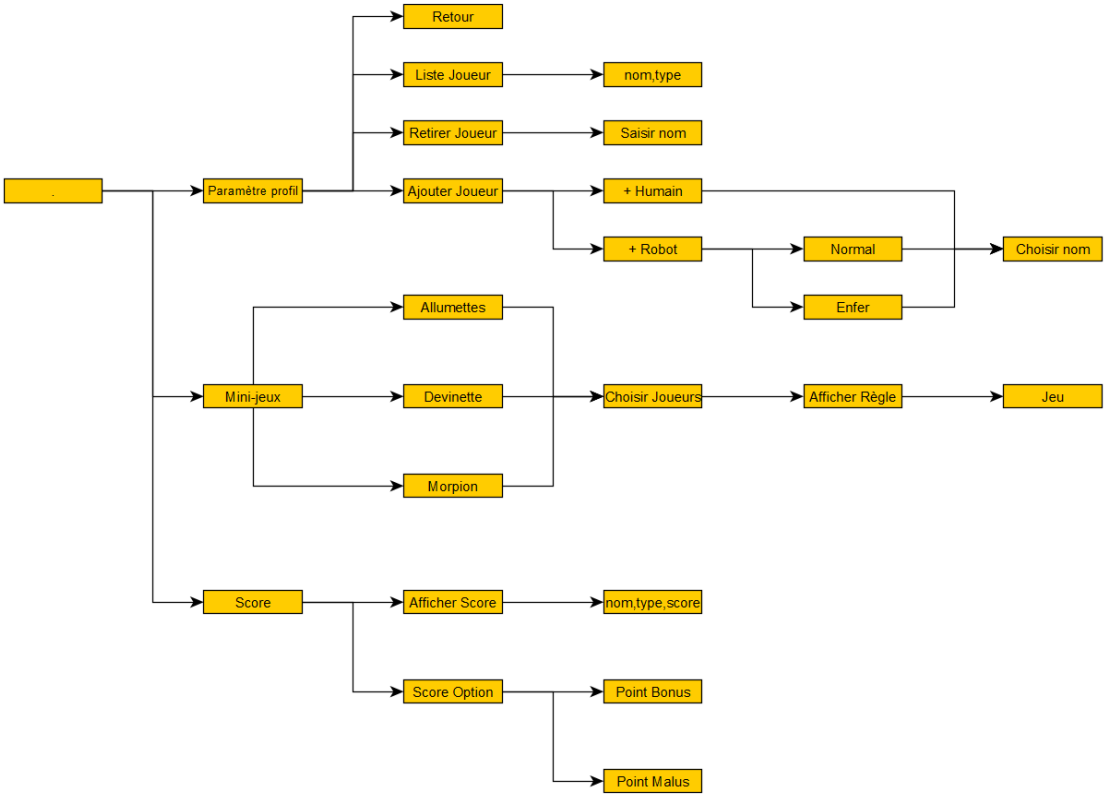
# SAé 1.01
# Développer des applications informatiques simples

Nous devons, pour cette SAé, réaliser une application complète qui regroupe un ensemble de 3 mini-jeux.
Cette application doit posséder un menu permettant de se déplacer.
Les jeux pourront être joués à deux, humain ou machine.
Les scores devront être stockés et pourront être affichés pour avoir la liste des joueurs, du meilleur au pire.

## Table des matières

# ***Présentation générale :***

## Schéma de présentation




Voici ci-dessus le schéma de présentation de la forme de notre algorithme. Nous avons décidé de le scinder en 3 parties “Paramètres Profil”, “Mini-jeux” et “Score” qui contiennent chacune différentes sections.

*On retrouve :*
- la section “Paramètres Profil” qui permet d’ajouter ou de retirer des joueurs et d’afficher la liste de ces joueurs.
- la section “Score” qui permet d’afficher le score et de le gérer, par exemple ajouter ou enlever des points à certains joueurs.
- Finalement la section “Mini-jeux” qui est la plus importante. C’est ici que l’on choisit le mini-jeu sur lequel on veut jouer.

Notre programme fonctionne alors sous forme de plusieurs menus qui s'enchaînent. Chacun de ces menus possède un dernier choix pour revenir en arrière "retour" :

```
def MenuJeux(listJ:list[joueur]):
    choix : int
    choix = 0

    while choix != 5:

        Transition()

        print()
        print("Mini-jeux :")
        print()
        print(bcolors.LightMagenta,"1 - ",bcolors.OKGREEN,"Allumettes")
        print(bcolors.LightMagenta,"2 - ",bcolors.OKGREEN,"Morpion")
        print(bcolors.LightMagenta,"3 - ",bcolors.OKGREEN,"Devinette")
        print(bcolors.LightMagenta,"4 - ",bcolors.OKGREEN,"Puissance 4")
        print(bcolors.LightMagenta,"5 - ",bcolors.OKGREEN,"Retour")
        print()

        choix = int(input("Choisissez une option : "+bcolors.OKYELLOW))
```

## La section “paramètres profils” :

Nous pouvons dans cette section ajouter des profils et leur donner un nom, un genre (humain ou machine) et s’il s’agit d’une machine, lui attribuer une intelligence.

```
listJ.append(le_joueur)
listJ[-1].nom = str(input("Saisir nom du joueur : "+bcolors.OKYELLOW))

listJ[-1].score = 0

while choix != 1 and choix != 2:
    print()
    print(bcolors.LightMagenta+"1 - ",bcolors.OKGREEN,"Le joueur est un Humain")
    print(bcolors.LightMagenta+"2 - ",bcolors.OKGREEN,"Le joueur est un Robot")

    choix = int(input("Choisissez une option : "+bcolors.OKYELLOW))

    if choix == 1:
        listJ[-1].typo = "Humain"
    elif choix == 2:
        listJ[-1].typo = "Robot"
    else:
        print(bcolors.OKRED,"Erreur.")
```

```
if listJ[-1].typo == "Humain":
    listJ[-1].mode = "Normal"
else:
    choix = 0
    while choix != 1 and choix != 2:
        print(bcolors.LightMagenta,"1 - ",bcolors.OKGREEN,"Le Robot est basique")
        print(bcolors.LightMagenta,"2 - ",bcolors.OKGREEN,"Le Robot est intelligent")

        choix = int(input("Choisissez une option : "+bcolors.OKYELLOW))

        if choix == 1:
            listJ[-1].mode = "Normal"
        elif choix == 2:
            listJ[-1].mode = "Intelligent"
        else:
            print(bcolors.OKRED+"Erreur.")
```

Nous avons aussi créé une section permettant, à la suite d'une recherche pour savoir si le personnage existe, d'effacer le profil en question.
Si le profil existe, le programme l'efface.

```
def AfficherJoueurs(listJ:list[joueur]):
    #Procédure
    i : int

    Transition()
    print("Liste des joueurs :")

    if len(listJ) > 0:

        for i in range(0,len(listJ)):
            if listJ[i].typo == "Robot":
                print(listJ[i].nom," - ",listJ[i].typo," - ",listJ[i].mode)
            else:
                print(listJ[i].nom," - ",listJ[i].typo)
    else:
        print("Aucun joueur d'enregistré.")
```

Il est alors possible d'afficher la liste des joueurs, pour chaque joueur les uns après les autres (avec pour chacun le type et le mode).

## La section “Score” :

```
def AfficherScore(listJ:list[joueur]):
    #Procédure
    i : int

    Transition()

    if len(listJ) == 0:
        print()
        print("Veuillez ajouter des joueurs.")
        print()
    else:
        print("Liste des joueurs :")

        for i in range(0,len(listJ)):
            if listJ[i].typo == "Robot":
                print(listJ[i].score," - ",listJ[i].nom," - ",listJ[i].typo," - ",listJ[i].mode)
            else:
                print(listJ[i].score," - ",listJ[i].nom," - ",listJ[i].typo)
```

Elle permet d’afficher le score pour chaque joueur.

```
choix1 = int(input("Saisissez un joueur à qui modifier son score : "+bcolors.OKYELLOW))

if choix1 != len(listJ)+1 and choix1 <= len(listJ) and choix1 > 0:
    choix2 = 0

    while choix2 != 3 :
        Transition()
        print()
        print(bcolors.LightMagenta,"1 - ",bcolors.OKGREEN,"Augmenter")
        print(bcolors.LightMagenta,"2 - ",bcolors.OKGREEN,"Réduire")
        print(bcolors.LightMagenta,"3 - ",bcolors.OKGREEN,"Retour")
        print()

        choix2 = int(input("Voulez-vous augmenter ou réduire son score ? : "+bcolors.OKYELLOW))
        if choix2 == 1:
            listJ[choix1-1].score = listJ[choix1-1].score+int(input(bcolors.OKGREEN+"Saisissez une valeur : "+bcolors.OKYELLOW))
        elif choix2 == 2:
            listJ[choix1-1].score = listJ[choix1-1].score-int(input(bcolors.OKGREEN+"Saisissez une valeur : "+bcolors.OKYELLOW))
        elif choix == 3:
            print(bcolors.OKGREEN+"Retour au menu principal !")
        else:
            print(bcolors.OKRED+"Erreur.")
```

Mais aussi de gérer celui-ci, par exemple si un joueur mérite 1 point bonus pour une action hors des règles qui mérite un bonus.

## La section “Mini-jeux” :

Le menu sur les mini-jeux sert de lien avec les programmes que l'on a importé auparavant.

```
from objets import joueur
from objets import bcolors
import allumettes
import morpion
import devinette
import Puissance4
```

Ce menu permet alors de lancer différentes procédures de mini-jeux desquels nous parlerons plus tard.

```
if choix == 1:
    allumettes.Allumettes(listJ[choix1-1],listJ[choix2-1])
elif choix == 2:
    morpion.Morpion(listJ[choix1-1],listJ[choix2-1])
elif choix == 3:
    devinette.Devinette(listJ[choix1-1],listJ[choix2-1])
elif choix == 4:
    Puissance4.ChoixP4(listJ[choix1-1],listJ[choix2-1])
```

Finalement nous enregistrons tous les scores dans un fichier "profil.txt". Cela nous permet de ne pas perdre de données. Ce fichier possède aussi les noms et les types de chaque joueur (machine ou pas).

```
def SauvegardeJoueurs(listJ:list[joueur]):

    profiles = open("profiles.txt", "w")

    for i in range(len(listJ)):
        profiles.write(",")
        profiles.write(listJ[-i].mode)
        profiles.write(",")
        profiles.write(listJ[-i].typo)
        profiles.write(",")
        profiles.write(str(listJ[-i].score))
        profiles.write(",")
        profiles.write(listJ[-i].nom)
        profiles.write(";\n")

    profiles.close()
```

Les profils sont donc sauvegardés et utilisés au lancement.

# ***Allumettes :***

## Rappel de l'énoncé :

On dispose d'un tas de 20 allumettes. Chaque joueur à tour de rôle peut en prélever 1, 2 ou 3. Le perdant est celui qui prend la dernière allumette.

## Analyse du travail à faire :

Nous devons découper le travail en plusieurs parties. Tout d'abord, il y aura le programme principal du jeu qui enclenchera le tour de J1 puis de J2 avant de recommencer encore et encore jusqu'à ce qu'il y a un gagnant. Ici, nous pouvons déjà identifier 3 fonctions :

- Une fonction gérant le tour d'un humain.
- Une fonction géant le tour d'un robot.
- Une fonction vérifiant si l'un des deux joueurs a gagné.

Il faudra faire attention à ce que la fonction vérifiant si l'un des deux joueurs a gagné soit enclenché entre le tour de J1 et de J2, puisque J2 ne peut pas jouer si J1 a déjà gagné.

Pour ce mini-jeu, une information est importante à savoir :
Il y a une méthode infaillible pour gagner, il s'agit de laisser à son adversaire un nombre d'allumettes multiples de 4 +1, par exemple 1, 5 ou 9. Comme cela, quand il restera 5 allumettes, celui-ci va devoir enlever au moins une allumette, il se retrouvera alors dans tous les cas perdant.
Nous allons nous servir de ça pour créer notre algorithme intelligent. En effet, si la machine n'est pas intelligente, elle choisit un nombre au hasard alors que si elle est intelligente, elle essaie de choisir un nombre avantageux

Pour savoir qui a gagné, nous nous servons du nombre d'allumettes prélevés, ainsi si celui si est égal ou supérieur à 20, la personne ayant joué en dernier est perdante.
Finalement 1 point est ajouté au score du gagnant. Celui-ci sera enregistré quand l'utilisateur quittera l'algorithme.

Nous utilisons en parallèle les procédures printAllumettes et constructor afin d'obtenir une représentation graphique. Les allumettes déjà enlevées prennent alors une couleur différente.

Finalement, durant toutes ces procédures, les variables des deux joueurs j1 et j2 sont transportées, elles passent alors partout.

## Algorithme :

```
Procédure Transition()

      Afficher((couleur.rouge+"►"+couleur.bleu+"►"couleur.rouge+"►"+c
      ouleur.bleu+"►"+couleur.rouge+"►"+couleur.bleu+"►"+"
      ")*4+couleur.vert)

procédure constructor(text:chaîne de
caractère,nbAlumJ1:entier,nbAlumJ2:entier):
      niv <- ""
      Pour i de 0 à 20 faire
            Si i >= 20-nbAlumJ2 alors
                  niv <-
niv+bcolors.OKRED+bcolors.BOLD+text+bcolors.OKGREEN
            Sinon
                  Si i < nbAlumJ1 alors
                        niv <-
niv+bcolors.OKRED+bcolors.BOLD+text+bcolors.OKGREEN
                  Sinon
                        niv <- niv+text
            afficher(niv)
                  FinSi
            FinSi
      FinFaire

Procédure PrintAllumettes(nbAlumJ1:entier,nbAlumJ2:entier)
      Afficher()
      constructor("@@@ ",nbAlumJ1,nbAlumJ2)
      constructor("@@@ ",nbAlumJ1,nbAlumJ2)
      constructor("@@@ ",nbAlumJ1,nbAlumJ2)
      constructor("@@@ ",nbAlumJ1,nbAlumJ2)
      constructor("@@@ ",nbAlumJ1,nbAlumJ2)

Fonction TourHumain(nbAlumJ1:entier,nbAlumJ2:entier) -> entier:
      choix : entier

      choix <- 0
```

```
      Afficher()
      Transition()
      Afficher()

      Afficher "Combien retirez-vous d'alumettes ?"
      Saisir choix

      Si choix != 1 et choix != 2 et choix !=3 alors
            Afficher("Erreur.")
            Retourne(TourHumain(nbAlumJ1,nbAlumJ2))
      Sinon
            Si nbAlumJ1+nbAlumJ2+choix>20 alors
                  choix <- 2
            FinSi
            Si nbAlumJ1+nbAlumJ2+choix>20 alors
                  choix <- 1
            FinSi
      FinSi

      Retourne(choix)

Fonction TourRobot(player:joueur,nbAlumJ1:entier,nbAlumJ2:entier) ->
entier
      choix : entier

      choix <- 0

      Afficher()
      Transition()

      Si player.mode = "Intelligent" alors
            Si 20-(nbAlumJ1+nbAlumJ2) = 3 alors
                  choix <- 2
            FinSi
            Si 20-(nbAlumJ1+nbAlumJ2) = 2 alors
                  choix <- 1
            FinSi
            Si 20-(nbAlumJ1+nbAlumJ2) = 1 alors
                  choix <- 1
            FinSi
            Si 20-(nbAlumJ1+nbAlumJ2) = 4 alors
                  choix <- 3
            FinSi
            Si (20-nbAlumJ1+nbAlumJ2-1-3)%4 != 0 alors
                  choix <- 3
            FinSi
            Si (20-nbAlumJ1+nbAlumJ2-1-2)%4 != 0 alors
                  choix <- 2
```

```
            FinSi
            Si (20-nbAlumJ1+nbAlumJ2-1-1)%4 != 0 alors
                  choix <- 1
            Sinon
                  choix <- aléatoire(1,3)
            FinSi

      Sinon
            choix <- aléatoire(1,3)
      FinSi

      Afficher()
      Afficher("Le Robot décide de prendre ",choix," allumette(s) !")

      Retourne(choix)

Procédure Allumettes(J1:joueur,J2:joueur):

      nbAlumJ1 : entier
      nbAlumJ2 : entier

      nbAlumJ1 <- 0
      nbAlumJ2 <- 0

      win <- 0

      Transition()
      Transition()

      PrintAllumettes(nbAlumJ1,nbAlumJ2)

      Tant que nbAlumJ1+nbAlumJ2 < 20 faire

            Afficher()
            Afficher("Tour de ",J1.nom," !")

            Si J1.typo = "Humain":
                  nbAlumJ1 <- TourHumain(nbAlumJ1,nbAlumJ2)+nbAlumJ1
            Sinon
                  nbAlumJ1 <- TourRobot(J1,nbAlumJ1,nbAlumJ2)+nbAlumJ1
            FinSi
            PrintAllumettes(nbAlumJ1,nbAlumJ2)

            Si nbAlumJ1+nbAlumJ2 < 20 alors

                  Afficher()
                  Afficher("Tour de ",J2.nom," !")
```

```
            Si J2.typo = "Humain" alors
                  nbAlumJ2 <-
TourHumain(nbAlumJ1,nbAlumJ2)+nbAlumJ2
            Sinon
                  nbAlumJ2 <-
TourRobot(J2,nbAlumJ1,nbAlumJ2)+nbAlumJ2
                  PrintAllumettes(nbAlumJ1,nbAlumJ2)
            FinSi

            Si nbAlumJ1+nbAlumJ2 >= 20 alors
                  win <- 1
            Sinon
                  win <- 2
            FinSi

Si win = 1 alors
      Afficher()
      Afficher(J1.nom," gagne ! ",J1.nom," gagne 1 point !")
      J1.score <- J1.score+1
FinSi

Si win = 2 alors
      Afficher()
      Afficher(J2.nom," gagne ! ",J2.nom," gagne 1 point !")
      J2.score <- J2.score+1
FinSi

Transition()
Transition()
Transition()

Retourne(J1,J2)
```

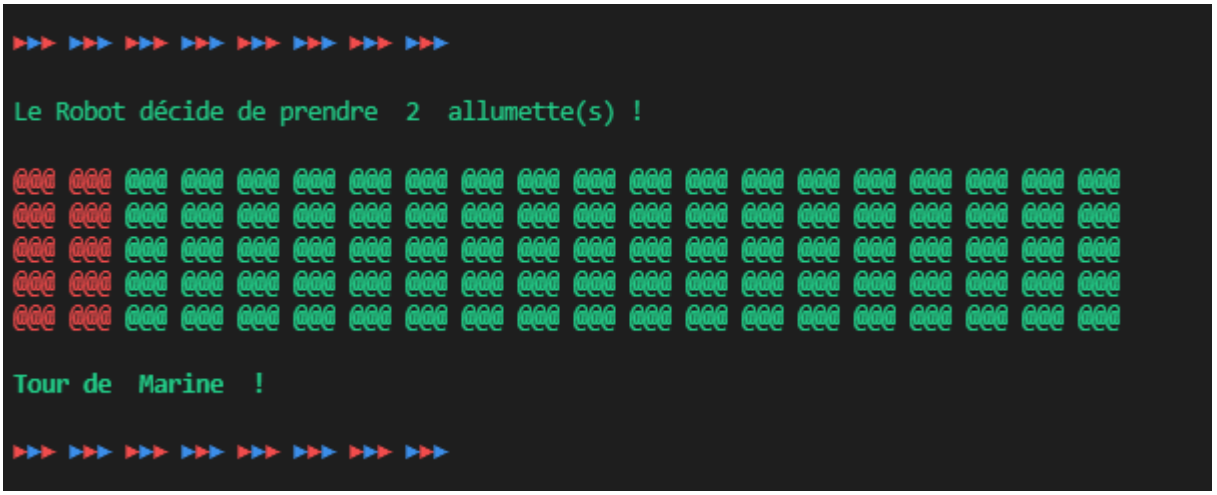
## **Programme :**

Le code Python sera lui entièrement envoyé dans un dossier zip à part.

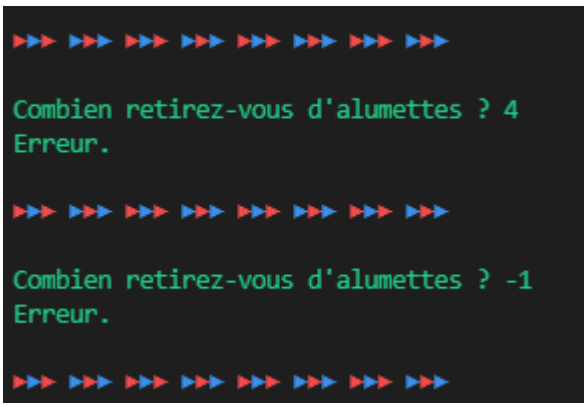
# Jeux d'essais :

*Nous commençons par regarder ce qu'il se passe entre deux joueurs robots l'un étant intelligent et l'autre basique :*

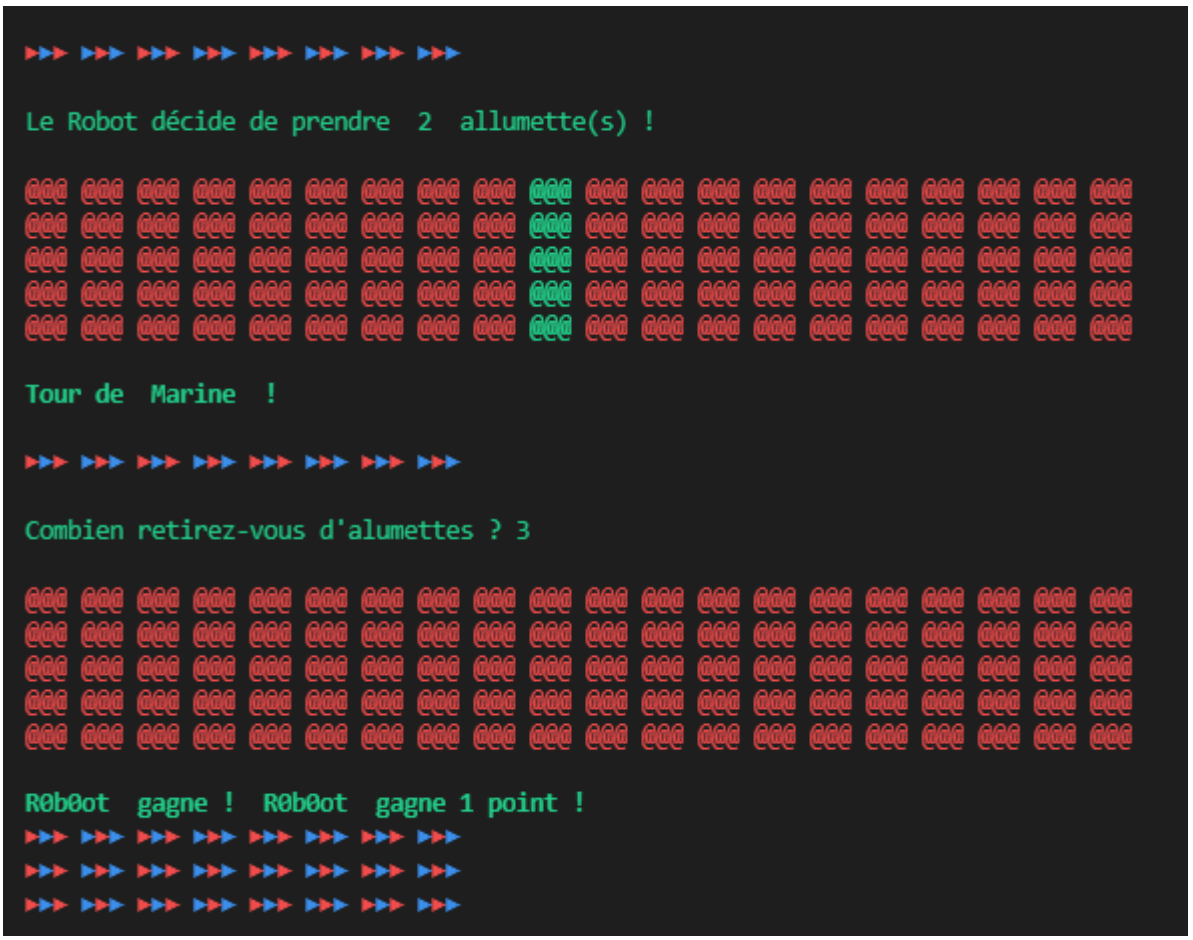

Les allumettes se représentent sous forme de barre de "@". Pour l'instant, personne n'a encore joué. Le tour du robot a été annoncé.




Le robot prend entre 1 et 3 allumettes, il choisit aléatoirement. Ici, il choisit deux allumettes.




Puis, c'est à notre tour. Si on rentre un nombre invalide, une erreur s'affiche.




Lorsqu'on veut récupérer plus d'allumette qu'il en reste, pas d'erreur s'affiche, mais le programme modifie la valeur entrée pour qu'elle soit égale au nombre d'allumette restant.

# ***Morpion :***

## Rappel de l'énoncé :

Chaque joueur pose sa marque (un O ou un X) à tour de rôle dans les cases d'une grille 3x3. Le premier qui aligne 3 marques a gagné.

## Analyse du travail à faire :

Tout comme pour les allumettes, nous découpons le travail en plusieurs parties avec notamment le programme principal du jeu qui enclenchera le tour de J1 puis de J2 avant de recommencer encore et encore jusqu'à ce qu'il y a un gagnant. Encore une fois, nous aurons ces trois fonctions :

- Une fonction gérant le tour d'un humain.
- Une fonction géant le tour d'un robot.
- Une fonction vérifiant si l'un des deux joueurs a gagné.

Pour gérer le marquage, la grille sera gérée par une matrice, soit une liste contenant 3 listes qui elles-même contiennent les différents marquages.

Pour créer une machine intelligente, nous nous servirons d'une liste brute de jeux à jouer en fonction du jeu de l'adversaire.
Nous nous servirons aussi d'une procédure permettant d'afficher le morpion comme pour les allumettes.

Finalement pour vérifier si quelqu'un a gagné nous vérifions de manière brute une succession de scénario de victoire possible. Si la partie fait partie de ces scénarios, cela veut dire que le joueur a gagné.

# Algorithme :

La première procédure permet d'afficher le morpion :

```
Procédure PrintMorpion(tab:liste[liste[chaîne de caractère]])

      tabtemp : liste[liste[chaîne de caractèrestr]]
      tabtemp <- [["-","-","-"],["-","-","-"],["-","-","-"]]

      pour i de 0 à len(tab) faire
            pour j de 0 à len(tab[i])) faire
                  Si tab[i][j] = "X" alors
                        tabtemp[i][j] <-
      couleur.rouge+couleur.gras+tab[i][j]+couleur.fin
                  FinSi
                  Si tab[i][j] = "O" alors
                        tabtemp[i][j] <-
      couleur.bleu+couleur.gras+tab[i][j]+couleur.fin
                  FinSi
            FinFaire
      FinFaire

      Afficher(couleur.fin)
      Afficher(" ",tabtemp[0][0]," ‖ ",tabtemp[0][1]," ‖
",tabtemp[0][2])
      Afficher("═══════╫═══════╫═══════")
      Afficher(" ",tabtemp[1][0]," ‖ ",tabtemp[1][1]," ‖
",tabtemp[1][2])
      Afficher("═══════╫═══════╫═══════")
      Afficher(" ",tabtemp[2][0]," ‖ ",tabtemp[2][1]," ‖
",tabtemp[2][2])
      Afficher()

La seconde procédure permet à un joueur humain de jouer :

Procédure TourHumain(tab:list[list[str]],Jnb:str)

      choixX : entier
      choixY : entier
      tabtemp : liste[liste[entier]]

      choixX <- 0
      choixY <- 0
      tabtemp <- [["-","-","-"],["-","-","-"],["-","-","-"]]
```

```
     pour i de 0 à longueur(tab) faire
          pour j de 0 à longueur(tab[i]) faire
               tabtemp[i][j] <- tab[i][j]
          FinFaire
     FinFaire

     Tant que choixX != 1 et choixX != 2 et choixX !=3 faire
          Afficher()
          choixX <- entier(Saisir("Indiquez la position de la case
     sur l'axe X : "))
          Si choixX != 1 et choixX != 2 et choixX !=3 faire
               Afficher("Erreur.")
          FinSi
     FinFaire

          Tant que choixY != 1 et choixY != 2 et choixY !=3 faire
               Afficher()
               choixY <- entier(Saisir("Indiquez la position de la
          case sur l'axe Y : "))
               Si choixY != 1 et choixY != 2 et choixY !=3 faire
                    Afficher("Erreur.")
               FinSi

               Afficher("La position de la case est de X :
          ",choixX,", Y : ",choixY)
          FinFaire

     Si tab[choixY-1][choixX-1] = "X" ou tab[choixY-1][choixX-1] =
"O" alors
          Afficher("Erreur. La case est déjà marqué par un symbole
     !")
          tabtemp <- TourHumain(tab,Jnb)
     Sinon
          Si Jnb = "J1" faire
               tabtemp[choixY-1][choixX-1] <- "X"
          Si Jnb = "J2" faire
               tabtemp[choixY-1][choixX-1] <- "O"
     FinSi

     Retourne(tabtemp)
```

Procédure permettant à un joueur machine de jouer :

```
Procédure TourRobot(tab:liste[liste[chaîne de caractère]],Jnb:chaîne
de caractère,player:joueur,tour:entier)
```

```
choixX : entier
choixY : entier
tabtemp : liste[liste[chaîne de caractère]]

choixX <- 0
choixY <- 0
tabtemp <- [["-","-","-"],["-","-","-"],["-","-","-"]]

Pour i dans(longueur(tab))faire
     pour j dans (longueur(tab[i]))faire
          tabtemp[i][j] <- tab[i][j]
Finfaire

si player.mode = "Intelligent" alors

     si tour = 1 et tab[1][1] = "-" alors
          choixX <- 2
          choixY <- 2

     sinon
          si tour = 2 et tab[1][1] != "-" alors
               choixX <- random.choice([1,3])
               choixY <- random.choice([1,3])
     sinon
          pour i de 0 à longueur(tab) faire
               si tab[i][0] = tab[i][1] et tab[i][2] = "-"
alors
                    choixX <- 3
                    choixY <- i+1
               sinon
                    si tab[i][2] = tab[i][1] et tab[i][0] =
               "-" alors
                         choixX <- 1
                         choixY <- i+1
                    finsi
               sinon
                    si tab[i][2] = tab[i][0] et tab[i][1] =
               "-" alors
                         choixX <- 2
                         choixY <- i+1
                    #Vérifie les colonnes
                    finsi
               sinon
                    si tab[0][i] = tab[1][i] et tab[2][i] =
               "-" alors
                         choixX <- i+1
                         choixY <- 3
                    finsi
```

```
                    sinon
                         si tab[2][i] = tab[1][i] et tab[0][i] =
                    "-" alors
                              choixX <- i+1
                              choixY <- 1
                         finsi
                    sinon
                         si tab[2][i] = tab[0][i] et tab[1][i] =
                    "-" alors
                              choixX <- i+1
                              choixY <- 2
                         finsi
                    finsi

               si choixX = 0 et choixY = 0 alors
                    #Diagonale d'en haut à gauche de en bas à
     droite
                    si tab[0][0] = tab[1][1] et tab[2][2] = "-":
                              choixX <- 2
                              choixY <- 2
                    sinon
                         si tab[2][2] = tab[1][1] et tab[0][0] =
                    "-" alors
                              choixX <- 0
                              choixY <- 0
                         finsi
                    sinon
                         si tab[2][2] = tab[0][0] et tab[1][1] =
                    "-" alors
                              choixX <- 1
                              choixY <- 1
                    #Diagonale d'en haut à droite de en bas à
     gauche
                         finsi
                    sinon
                         si tab[0][2] = tab[1][1] et tab[2][0] =
                    "-" alors
                              choixX <- 0
                              choixY <- 2
                         finsi
                    sinon
                         si tab[2][0] = tab[1][1] et tab[0][2] =
                    "-" alors
                              choixX <- 2
                              choixY <- 0
                         finsi
                    sinon
```

```
                        si tab[2][0] = tab[0][2] et tab[1][1] =
                    "-" alors
                              choixX <- 1
                              choixY <- 1
                        finsi
                    sinon
                        choixX <- random.randint(1,3)
                        choixY <- random.randint(1,3)
                    finsi

    sinon
        si player.mode = "Normal" alors
            choixX <- random.randint(1,3)
            choixY <- random.randint(1,3)
    finsi

    si tab[choixY-1][choixX-1] = "X" ou tab[choixY-1][choixX-1] =
"O" alors
        tabtemp <- TourRobot(tab,Jnb,player,tour)
    sinon
        si Jnb = "J1" alors
            tabtemp[choixY-1][choixX-1] <- "X"
        sinon
            si Jnb = "J2" alors
                tabtemp[choixY-1][choixX-1] <- "O"
            finsi
        finsi
    finsi

    return(tabtemp)


Procédure permettant de vérifier si une partie est gagnée :

procédure VerifWin(tab:list[list[chaine de caractères]]):
    Pour i de 0 à longueur tab faire
        si tab[i][0] = tab[i][1] = tab[i][2] et tab[i][0] = "X"
and tab[i][1] = "X" et tab[i][2] = "X" alors
            return(1)
        Finsi
        si tab[i][0] = tab[i][1] = tab[i][2] et tab[i][0] = "O"
and tab[i][1] = "O" and tab[i][2] = "O" alors
            return(2)
        Finsi
    Finfaire
    Pour i de 0 à longueur(tab) faire
        si tab[0][i] = tab[1][i] = tab[2][i] et tab[0][i] = "X" et
tab[1][i] = "X" et tab[2][i] = "X" alors
```

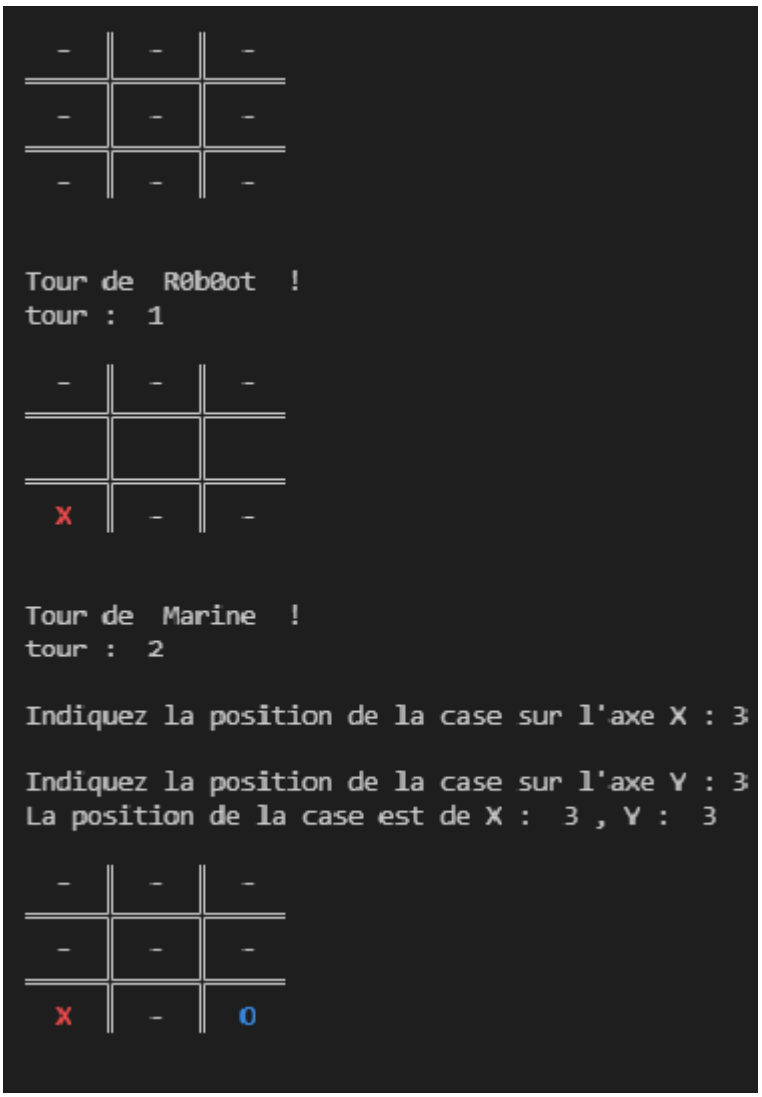
```
                return(1)
           Finsi
           Si tab[0][i] = tab[1][i] = tab[2][i] et tab[0][i] = "O" et
tab[1][i] = "O" et tab[2][i] = "O" alors
                return(2)

           si tab[0][0] = tab[1][1] = tab[2][2] et tab[0][0] = "X" et
     tab[1][1] = "X" et tab[2][2] = "X" alors
                                return(1)
           Finsi
           Si tab[2][0] = tab[1][1] = tab[0][2] et tab[2][0] = "X" et
tab[1][1] = "X" et tab[0][2] = "X" alors
                return(1)
           si tab[0][0] = tab[1][1] = tab[2][2] et tab[0][0] = "O" et
tab[1][1] = "O" et tab[2][2] = "O" alors
                return(2)
           Finsi
           Si tab[2][0] = tab[1][1] = tab[0][2] et tab[2][0] = "O" et
tab[1][1] = "O" et tab[0][2] = "O" alors
                return(2)
           Finsi
           Si tab[0][0] != "-" et tab[0][1] != "-"  et tab[0][2] !=
"-"  et tab[1][0] != "-"  et tab[1][1] != "-"  et tab[1][2] != "-"
et tab[2][0] != "-"  et tab[2][1] != "-"  et tab[2][2] != "-" alors
                return(3)
           Finsi
           return(0)
     Fin procédure
```

Toutes ces procédures sont utilisées par une procédure principale qui leur sert donc de lien basique. Cette procédure n'utilise donc aucune spécificité propre.

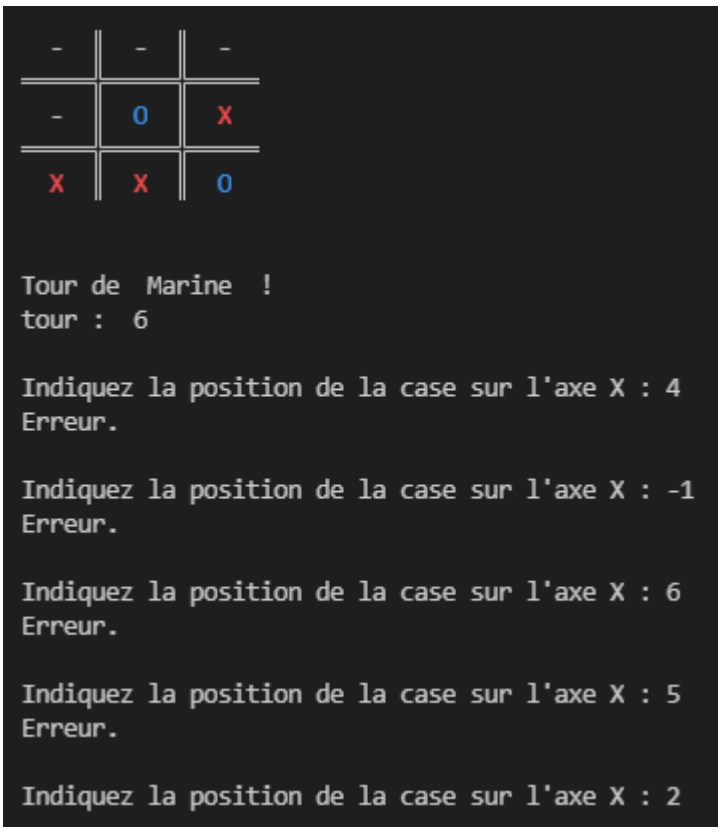
## Programme :

Le code Python sera lui entièrement envoyé dans un dossier zip à part.

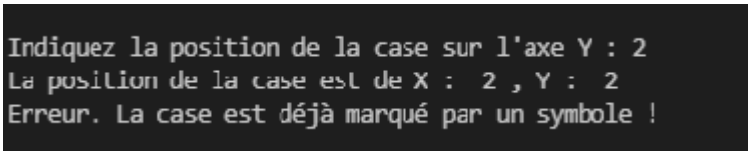
## Jeux d'essais :



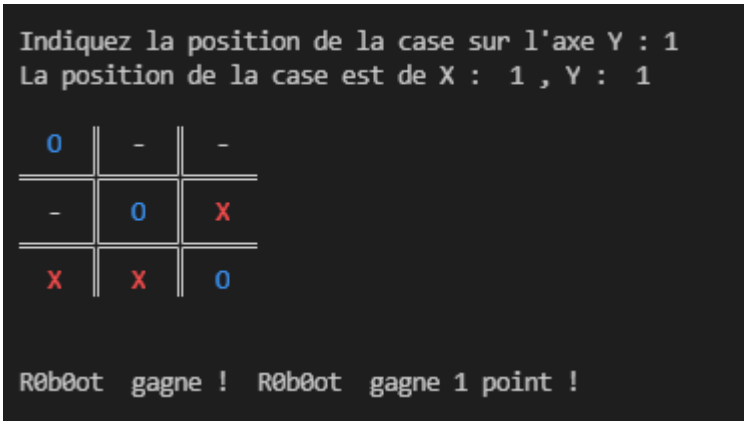

Le marquage se place correctement, le robot qu'on affronte répond. Dû à son niveau de difficulté, il choisit des cases aléatoires.




Il y a une erreur si on choisit un nombre non valide.




On ne peut pas marquer un endroit déjà marqué.




Lorsque 3 symboles sont alignés, c'est la personne ayant posé la dernière marque qui remporte le point.

# ***Devinette :***

## Rappel de l'énoncé :

Deux joueurs s'affrontent dans un duel où l'objectif est de deviner le nombre auquel pense le joueur adverse. Le joueur le plus rapide remporte le jeu et gagne 1 point.

## Analyse du travail à faire :

Dans ce jeu, le joueur le plus "rapide" remporte le point, alors s'offrait à nous deux possibilités :

- Compter le nombre de tour et donner le point à celui qui aura deviné le nombre avec le moins de tour.
- Utiliser le temps, c'est-à-dire, donner le point à celui qui met le moins de temps, donc le moins de secondes.

Les deux choix comportent des avantages comme des inconvénients, mais nous avons opté pour utiliser un système de temps en secondes. Alors, grâce au module time, nous chronométrons le temps que met chaque joueur à trouver le nombre adverse.

Notre programme se sépare donc en 3 cas :
Si les 2 joueurs sont des humains :

- Alors l'un des deux joueurs, au hasard, choisit le nombre limite. (Il s'agit du nombre à ne pas dépasser)
- Les deux joueurs choisissent leur nombre "secret" et le disent au programme (qui les cache)
- Le joueur qui doit deviner le nombre de son adversaire peut alors faire une série de propositions. Le programme lui dit si son nombre est trop grand ou trop petit. Quand il trouve le bon nombre, le chronomètre s'arrête.
- Celui qui réalise le plus petit temps gagne !

Si il y a 1 joueur humain et 1 robot :

- Le joueur humain choisit alors le nombre limite.
- Il entre le nombre qu'il a choisi au programme. Le robot essaie alors de le deviner.

- Puis c'est au tour du joueur humain de deviner le nombre du robot.
- Celui qui réalise le plus petit temps gagne !

Si les 2 joueurs sont des robots :

- Le nombre limite est choisi au hasard entre 100 et 200 (arbitrairement il s'agit d'une fourchette pas trop grande ni trop petite)
- Les deux robots essaient alors de deviner le nombre du robot inverse.
- Celui qui réalise le plus petit temps gagne !

L'intelligence du robot lui joue sur le temps que met le robot à répondre et sur son choix, en effet, il peut alors proposer des nombres plus rapidement et en plus, il propose des nombres intelligents. Par exemple, il choisit non pas un nombre au hasard mais le milieu entre la limite haute et la limite basse. Il met aussi moins de temps à répondre.

Il faut spécialement faire attention à ce que les robots puissent trouver le nombre même si celui-ci vaut 0 ou la valeur de la limite.

# Algorithme :

*Importation des modules* (cela n'est pas traduisible en pseudo-code) :

```
Depuis objets import joueur
import random
import time
import getpass
```

*Procédure principale :*

```
Procedure Devinette(j1:joueur,j2:joueur) :
  """
  Procédure Devinette qui regroupe tout l'algorithme pour le mini-jeu.
  Il prend et modifie les variables j1 et j2 qui sont les deux joueurs.
  La procédure utilise le nom et le mode des joueurs et modifie leur score
  """
```

*déclaration des variables :*

```
 Avec
  nbre_max:entier
  nbre_j1:entier
  nbre_j2:entier
  limiteh:entier
  limiteb:entier
  tic:réel
  tac:réel
  tps1:réel
  tps2:réel
  nbre_propose:entier
  hasard:entier
  dif1:réel
  dif2:réel
```

## *Robot contre robot :*

*(choix de la limite entre 100 et 200) le module random n'est pas traduisible en pseudo-code :*

```
    nbre_max<-random.randint(100,200)
```

choix des nombres secrets au hasard :

```
    nbre_j1<-random.randint(0,nbre_max)
    nbre_j2<-random.randint(0,nbre_max)
```

proposition au hasard :

```
nbre_propose<-random.randint(0,nbre_max)
```

début du chrono (module time toujours pas traduisible en pseudo-code)

```
tic<-time.time()
```

premières limites : (celle du bas est 0 et celle du haut est la limite que l'on a défini précédemment)

```
limiteh<-nbre_max
limiteb<-0
```

## Proposition des nombres en fonction de l'intelligence :

Si le robot est normal (pas intelligent) :

```
si j1.mode="Normal" alors
  Tant que nbre_propose != nbre_j2 faire
    Afficher j1.nom,"propose :",nbre_propose
    si nbre_propose>nbre_j2 alors
      Afficher "Trop grand !"
      limiteh<-nbre_propose
      nbre_propose<-random.randint(limiteb,limiteh-1)
    Sinon
       si nbre_propose<nbre_j2 alors
      Afficher "Trop petit !"
      limiteb<-nbre_propose
      nbre_propose<-random.randint(limiteb+1,limiteh)
     Finsi
    Finsi
Finfaire
```

Si le robot est intelligent :

```
Sinon
  Tant que nbre_propose != nbre_j2 faire
    Afficher j1.nom,"propose :",nbre_propose
    Si nbre_propose>nbre_j2 alors
      Afficher "Trop grand !"
      limiteh<-nbre_propose
      nbre_propose<-(limiteb+limiteh)//2
    Sinon
        Si nbre_propose<nbre_j2 alors
      Afficher "Trop petit !"
      Si limiteb=nbre_propose alors
        nbre_propose<-nbre_propose+1   #Si le nombre choisi vaut la limite haute
```

```
            Sinon
              limiteb<-nbre_propose
            Finsi
          Finsi
nbre_propose<-(limiteb+limiteh)//2
        time.sleep(dif1)
    Afficher j1.nom,"propose :",nbre_propose
Finfaire
Finsi
```

*Fin du chrono :*

```
tac<-time.time()
time.sleep(1.0)
tps1<-tac-tic   #calcul du temps
```

*Le second robot suit absolument le même algorithme, à la seule différence qu'il donne un temps tps2 qui lui est propre. On ne recopie donc pas cette partie !*

On peut dès alors, comparer les deux temps :

```
Si tps2-tps1>5 ou tps1-tps2>5 alors
  Afficher "Le score est écrasant !!"
Sinon
  Afficher "La partie a été serrée !!"
time.sleep(3.0)
Finsi
Si tps2=tps1 allors
  Afficher "Il y a égalité, personne ne gagne de point !"
Finsi
Si tps2>tps1 alors
   Afficher j1.nom,"a été plus rapide de",tps2-tps1,"secondes ! Il gagne donc 1 point !"
  j1.score<-j1.score+1
Finsi
Si tps2<tps1 alors
  Afficher j2.nom,"a été plus rapide de",tps1-tps2,"secondes ! Il gagne donc 1 point !"
  j2.score<-j2.score+1
time.sleep(1.0)
Finsi
```

## Robot contre joueur :

*choix du nombre limite par le joueur :*

```
Afficher j2.nom," a l'honneur de choisir le nombre limite :"
time.sleep(1.0)
Afficher "Veuillez entrer le nombre limite"
Saisir nbre_max
time.sleep(1.0)
```

```
Afficher "Le nombre limite a été choisi :",nbre_max
```

choix du nombre par la machine :

```
nbre_j1<-random.randint(0,nbre_max)
time.sleep(1.0)
Afficher j1.nom,"a choisi son nombre !"
time.sleep(1.0)
```

choix du nombre par la machine :

```
Afficher "Veuillez entrer votre nombre :"
Saisie nbre_j2
Tant que nbre_j2>nbre_max ou nbre_j2<0 faire
  Afficher "Erreur ! Veuillez entrer un nombre fonctionel :"
  Saisir nbre_j2#vérification d'un nombre
Finfaire
```

*La machine commence d'abord par deviner le nombre du joueur :*
*Il y a donc une proposition des nombres en fonction de l'intelligence comme précédemment, on peut alors retrouver cette étape sur la page précédente.*

*Puis c'est au tour du joueur de deviner le nombre de la machine :*

Le joueur peut alors proposer des nombres :

```
Afficher j2.nom,"propose :"
```

La variable vaut alors une valeur qu'on lui donne :

```
Saisir nbre_propose
tic<-time.time()
limiteh<-nbre_max
limiteb<-0
```

Et tant que le choix qu'on lui donne n'est pas bon, le programme continue de nous dire si le nombre qu'on indique est trop grand ou trop petit :

```
Tant que nbre_propose != nbre_j1 faire
  Afficher j2.nom,"propose :",nbre_propose
  Si nbre_propose>nbre_j1 alors
    Afficher "Trop grand !"
    Afficher j2.nom,"propose :"
    Saisir nbre_propose
  Sinon
    Si nbre_propose<nbre_j1 alors
    Afficher "Trop petit !"
    Afficher j2.nom,"propose :"
    Saisir nbre_propose
   Finsi
```

```
        Finsi
    Finfaire
    Afficher j2.nom,"propose :",nbre_propose
    Afficher "C'est gagné !"
fin du chrono
    tac<-time.time()
    tps2<-tac-tic
    Afficher j2.nom,"a trouvé le nombre en",tps2,"secondes !"
    time.sleep(1.0)
```

*Comparaison des temps :*

Il y a finalement une comparaison des temps, il s'agit de la même comparaison que précédemment. Le joueur ayant été le plus rapide remporte 1 point !

## La partie joueur contre joueur :

*Choix de la personne qui choisit le nombre limite :*
*Un des deux joueurs doit donc donner le nombre limite, il est choisi au hasard.*

```
hasard<-random.randint(1,2)
Si hasard=1 alors
  Afficher j1.nom,"a l'honneur de choisir le nombre limite !"
Finsi
Si hasard=2 alors
  Afficher j2.nom,"a l'honneur de choisir le nombre limite !"
Finsi
Afficher "Veuillez entrer le nombre limite :"
Saisir nbre_max
time.sleep(1.0)
Afficher "Le nombre limite a été choisi :",nbre_max
time.sleep(1.0)
```

*Choix des nombres :*
*Le module getpass n'est pas traduisible en pseudo-code mais il permet de cacher le nombre que l'on entre pour pas que l'autre joueur le voit.*

```
Afficher j2.nom,"peut indiquer son nombre : (Ne t'inquiète pas, ça reste entre nous !)"
Saisir nbre_j2 (avec getpass.getpass pour pas que ça s'affiche)
```

*On vérifie que le nombre est bien entre les deux limites :*

```
Tant que nbre_j2>nbre_max ou nbre_j2<0 faire
  Afficher "Erreur ! Veuillez entrer un nombre fonctionel :"
  Saisir nbre_j2 (avec getpass.getpass pour pas que ça s'affiche)
 Finfaire
```

```
    Afficher j2.nom,"a choisi son nombre !"
```

*On fait la même chose une deuxième fois pour le deuxième joueur, nous n'allons donc pas écrire l'algorithme.*

Les joueurs peuvent alors proposer des nombres pour deviner celui du joueur adverse :

```
    Afficher j2.nom,"commence par proposer :"
    Saisir nbre_propose
début du chrono
    tic<-time.time()
    limiteh<-nbre_max
    limiteb<-0
    Tant que nbre_propose != nbre_j1 faire
      Si nbre_propose>nbre_j1 alors
        Afficher "Trop grand !"
        Afficher j2.nom,"propose alors :"
        Saisir nbre_propose
      Sinon
       Si nbre_propose<nbre_j1 alors
        Afficher "Trop petit !"
        Afficher j2.nom,"propose alors :"
        Saisir nbre_propose
       Finsi
      Finsi
    Finfaire
    Afficher "C'est gagné !"
    tac<-time.time()
fin chrono
```

*Calcul du temps :*

```
    tps2<-tac-tic
    Afficher j2.nom,"a trouvé le nombre en",tps2,"secondes !"
```

*Le deuxième joueur devine alors le nombre du premier jour, il s'agit du même algorithme que précédemment, nous n'allons donc pas réécrire toute cette partie.*

Il suffit alors de calculer l'écart de temps :

```
    Si tps2-tps1>5 ou tps1-tps2>5 alors #écart > 5 secondes ou pas
      Afficher "Le score est écrasant !!"
    Sinon
     Afficher "La partie a été serrée !!"
    time.sleep(3.0)
   Finsi
```

Puis il faut les comparer :

```
    Si tps2=tps1 alors
      Afficher "Il y a égalité, personne ne gagne de point !"
```

```
    Finsi
    Si tps2>tps1 alors
      Afficher j1.nom,"a été plus rapide de",tps2-tps1,"secondes ! Il gagne donc 1 point !"
      j1.score<-j1.score+1 #ajout du point
    Finsi
    Si tps2<tps1 alors
      Afficher j2.nom,"a été plus rapide de",tps1-tps2,"secondes ! Il gagne donc 1 point !"
      j2.score<-j2.score+1 #ajout du point
    time.sleep(1.0)
    Finsi
```

## Programme :

Le code Python sera lui entièrement envoyé dans un dossier zip à part.

## Jeux d'essais :

*Nous commençons par regarder ce qu'il se passe entre deux joueurs robots l'un étant intelligent et l'autre basique :*

```
Choisissez J1 : 2
Choisissez J2 : 4

Bienvenue dans le mini-jeu Devinette

Vous devez dans ce mini-jeu, dans le but de gagner, deviner le nombre du joueur adverse le plus rapidement.

Le joueur qui aura mis le moins de temps remporte 1 point !

Vous êtes prêt ?

Les deux joueurs qui s'affrontent sont : R0b0ot et Robotint

Le nombre limite a été choisi : 197 . Veuillez ne pas dépasser cette limite !
```

Les règles s'affichent bien correctement. La nombre limite a été choisi, dans ce cas : 197.

```
Les deux joueur ont choisi leur nombre !

R0b0ot va deviner le nombre de Robotint

R0b0ot propose : 32
Trop petit !
R0b0ot propose : 41
Trop petit !
R0b0ot propose : 182
```

Ensuite, le premier robot basique propose des nombres :
Nous pouvons voir qu'il s'agit d'entiers proposés au hasard entre la limite basse et la limite haute. Cependant il n'y a aucune réflexion dans ce choix, (182 et 41 n'ont aucun rapport).

```
R0b0ot propose : 165
C'est gagné !
R0b0ot a trouvé le nombre en 21.072509288787842 secondes !
```

Il trouve finalement le nombre  en 21 secondes.

```
Robotint va désormais deviner le nombre de R0b0ot
Robotint propose : 96
Trop grand !
Robotint propose : 48
Trop grand !
Robotint propose : 24
Trop petit !
Robotint propose : 36
Trop petit !
Robotint propose : 42
Trop grand !
Robotint propose : 39
Trop petit !
Robotint propose : 40
C'est gagné !
Robotint a trouvé le nombre en 9.038728475570679 secondes !
```

C'est alors le tour du robot intelligent.  Celui-ci propose des nombres plus rapidement et les nombres sont plus cohérents. Il propose alors à chaque fois le nombre entre la limite maximale et celle minimale. Nous pouvons voir ici qu'il a trouvé le nombre en 9 secondes.

```
Le score est écrasant !!
Robotint a été plus rapide de 12.033780813217163 secondes ! Il gagne donc 1 point !
```

L'écart de temps est supérieur à 5 secondes ! Le score est donc bien écrasant. Le robot intelligent gagne alors 1 point !

*Nous pouvons alors voir ce qu'il se passe entre un joueur robot et un joueur humain :*

```
Jean  a l'honneur de choisir le nombre limite:
Veuillez entrer le nombre limite
```

Dans un premier temps, le joueur humain choisit la limite haute de la partie.

```
Le nombre limite a été choisi : 160
R0b0ot a choisi son nombre !
Jean à votre tour !
Veuillez entrer votre nombre :100
```

Puis, il doit entrer le nombre qu'il a choisi.

```
Jean a choisi son nombre !
R0b0ot va d'abord deviner le nombre de Jean
R0b0ot propose : 42
Trop petit !
R0b0ot propose : 102
Trop grand !
R0b0ot propose : 91
Trop petit !
```

Le robot commence alors par essayer de trouver le nombre du joueur humain. Nous avions choisi d'utiliser un robot normal dans ce cas-ci.

```
R0b0ot propose : 100
C'est gagné !
R0b0ot  a trouvé le nombre en 15.057315349578857 secondes !
Jean va désormais deviner le nombre de R0b0ot
Jean propose :
105
```

Quand celui-ci trouve le nombre. C'est au tour du joueur de trouver celui du robot. Il faut pour cela qu'il propose des nombres.

```
Jean proposez alors :
89
Jean propose : 89
Trop grand !
Jean proposez alors :
65
Jean propose : 65
Trop grand !
Jean proposez alors :
43
Jean propose : 43
Trop grand !
Jean proposez alors :
23
Jean propose : 23
Trop grand !
```

Il faut alors proposer les nombres le plus rapidement et intelligemment possible pour faire mieux que le robot.

Finalement le résultat dépend des scores !

*Finalement nous pouvons nous demander ce que donne le programme quand deux joueurs humains s'affrontent :*

```
Marine a l'honneur de choisir le nombre limite !
Veuillez entrer le nombre limite :160
```

Un des deux joueurs, au hasard, choisis le nombre limite. Ici il s'agit de Marine.

```
Marine peut indiquer son nombre : (Ne t'inquiète pas, ça reste entre nous !)
Nombre choisi :
Erreur ! Veuillez entrer un nombre fonctionel
```

Après, les joueurs choisissent leur nombre. Celui-ci est invisible grâce au getpass. De plus, il faut que celui-ci soit  entre 0 et la limite choisie précédemment sinon il y a un message d'erreur.

```
Marine va désormais deviner le nombre de Jean
Marine commence par proposer :
90
Trop petit !
Marine propose alors :
101
Trop petit !
Marine propose alors :
120
C'est gagné !
```

Finalement, chaque joueur propose des nombres, le programme lui répond si celui-ci est trop grand ou trop petit, jusqu'à ce qu'il trouve le bon nombre.

Au final c'est toujours le joueur le plus rapide qui gagne et remporte un point !

```
C'est gagné !
Jean a trouvé le nombre en 10.91187596321106 secondes !
Le score est écrasant !!
Marine a été plus rapide de 7.320566415786743 secondes ! Il gagne donc 1 point !
```

*Pour les cas spéciaux :*

Si l'on choisit comme nombre secret, le nombre limite :

```
Jean  a l'honneur de choisir le nombre limite:
Veuillez entrer le nombre limite 160
Le nombre limite a été choisi : 160
R0b0ot a choisi son nombre !
Jean à votre tour !
Veuillez entrer votre nombre :160
```

Le robot normal le trouve :

```
R0b0ot propose : 153
Trop petit !
R0b0ot propose : 160
C'est gagné !
R0b0ot  a trouvé le nombre en 15.04151725769043 secondes !
```

Le robot intelligent aussi :

```
Robotint propose : 159
Trop petit !
Robotint propose : 160
C'est gagné !
Robotint  a trouvé le nombre en 15.084606647491455 secondes !
```

Si l'on choisit comme nombre secret le nombre "0" :

```
Jean à votre tour !
Veuillez entrer votre nombre :0
Jean a choisi son nombre !
```

Le robot normal le trouve :

```
R0b0ot propose : 28
Trop grand !
R0b0ot propose : 22
Trop grand !
R0b0ot propose : 0
C'est gagné !
R0b0ot  a trouvé le nombre en 15.034308195114136 secondes !
```

Le robot intelligent aussi :

```
Robotint propose : 7
Trop grand !
Robotint propose : 3
Trop grand !
Robotint propose : 1
Trop grand !
Robotint propose : 0
C'est gagné !
Robotint  a trouvé le nombre en 10.543309688568115 secondes !
```