**ANALYSE DE L’ALGORITHME DE LA METHODE (Fonction) VerifLigne, incluse dans la classe TableauJeu**

VerifLigne(TabTest : Tableau(NbCol), TabSolution :Tableau(NbCol)) :TabResult Tableau(NbCol)

|  |
| --- |
| **Lexique** :  TabTest Tableau de la ligne proposée par l’utilisateur de dimension nbcol à tester  TabSolution Tableau de la ligne résultat de dimension nbcol  TabSauve Tableau de la ligne résultat de dimension nbcol  i : entier indice de parcours du tableau TabTest  j : entier indice de parcours du tableau TabSolution  r : entier indice de parcours du tableau TabResult |
| TabResult  TabTest    TabSolution  **Algorithme :**  Début  ***// Sauvegarde du tableau Solution***  TabSauve=TabSolution  Si TabTest = TabSolution alors  ***// Fonction qui rempli TabResult avec des points Rouge***  Rempli(TabResult,R,1,NbCol)  Trouve = Vrai  FinSi  r = 0  Si non Trouve alors  Pour i de 0 à NbCol-1  Trouve=Faux  j = 0  Tant Que j<= NbCol-1 et Non Trouve Faire  Si TabTest(i)=TabSolution(j)  alors  trouve = Vrai  si j=i Alors  TabResult(r)=Rouge  Sinon  TabResult(r)=Blanc  FinSi  TabSolution(j)= ‘’  r = r+1  Sinon  j = j+1  Finsi  FinTantQue  FinPour  FinSi  ***// Réinitialisation de TabSolution***  TabSolution = TabSauve  Retourner TabResult  Fin  def mastermind (solution, test):  #initialisation d'un tableau vide  resultat=[]  #ajoute les pions rouges au tebleau resultat  for i in range(len(solution)):  if solution[i]==test[i]:  resultat.append('r')  #ajoute les pions blanc au tableau résultat  for i in range(len(solution)):  for j in range(len(test)):  if solution[i] == test[j] and solution[i] != test[i]:  resultat.append('b')  #tant que le tableau ne fait pas la longueur désiré, complète le tableau avec 'v'  while len(resultat)<len(solution):  resultat.append('v')  return resultat |