POWERSHELL - KOH LANTA

Objectif

Ce tp a pour but d'adapter le jeu télévisé "Koh lanta" en jeu d'élimination le tout codé en script powershell

Script

```
$joueur = Get-Content "C:\Users\jpallara\Downloads\tp koh-lanta
powershell\joueur.txt"
$totaljoueur = $joueur.Count
$tailleEquipe1 = [math]::Ceiling($totaljoueur / 2) #[math]::Ceiling sert à
arrondir "vers le haut" pour toujours faire en sorte que la taille des équipes
soit un nombre entier même si la nombre de joueurs est impaire
$tailleEquipe2 = [math]::Floor($totaljoueur / 2)#[math]::Floor sert à arrondir
"vers le bas" pour que dans le cas où le nombre de joueur est impaire, cette
équipe a un joueur de moins que l'autre
$joueurHasard = $joueur | Get-Random -Count $totaljoueur
$Equipe1 = $joueurHasard[0..($tailleEquipe1 - 1)]
$Equipe2 = $joueurHasard[$tailleEquipe1..($totaljoueur - 1)]

$tailleinitialEquipe1 = $Equipe1.Count
$tailleinitialEquipe2 = $Equipe2.Count

while ($Equipe1.Count -gt $initialEquipe1 / 2 -and $Equipe2.Count -gt
$initialEquipe2 / 2) {
    $gagnant = Get-Random -InputObject @("Equipe1", "Equipe2")

    if ($gagnant -eq "Equipe1") {
        $joueurElimine = $Equipe2 | Get-Random
        $Equipe2 = $Equipe2 - $joueurElimine
    } else {
        $joueurElimine = $Equipe1 | Get-Random
        $Equipe1 = $Equipe1 - $joueurElimine
    }

    if ($Equipe1.Count -le $initialEquipe1 / 2){
        Write-Host "L'équipe 1 va au conseil"
    } elseif ($Equipe2.Count -le $initialEquipe2 / 2) {
        Write-Host "L'équipe 2 va au conseil"
    } else {
        Write-Host "Les équipes sont à égalité"
    }

}

function Gérer-Consel {
    param (
        [Parameter(Mandatory=$true)] #paramètre obligatoire à la fonction
        [ref]$Equipe1,
```

```
        [Parameter(Mandatory=$true)]#paramètre obligatoire à la fonction
        [ref]$Equipe2,

        [Parameter(Mandatory=$true)]+- #paramètre obligatoire à la fonction
        [string]$joueurImmunise
    )

    # Mélanger les équipes pour déterminer les joueurs en danger
    $joueursEnDanger = @($Equipe1.Value + $Equipe2.Value) | Where-Object { $_ -ne
$joueurImmunise } #excluera le joueur immunisé pour ne garder que ceux en danger

    # Afficher les joueurs en danger et celui qui est immunisé
    Write-Host "$joueurImmunise est immunisé. Les joueurs suivants sont en danger
:"
    Write-Host ($joueursEnDanger -join ", ")

    # Tableau pour les votes
    $votes = @{} #création du table de hachage, qui fonctionne comme un
dictionnaire en python (clé/valeur), en la stockant dans la varible "vote"

    # Demander à chaque joueur de voter
    foreach ($joueur in $joueursEnDanger) {
        $voteValide = $false #initialisation des votes en statu "pas voté"
        while (-not $voteValide) { #tant que le joueur n'as pas voté
            Write-Host "$joueur, tu dois voter pour un joueur parmi les suivants :
$($joueursEnDanger -join ', ')"
            $vote = Read-Host "Pour qui veux-tu voter ?"

            # Vérifier que le vote est valide : il ne peut pas voter pour lui-même
ou pour un joueur immunisé
            if ($joueursEnDanger -contains $vote -and $vote -ne $joueur -and $vote
-ne $joueurImmunise) { #exclue le joueur immunisé et le joueur entrain de voté, du
vote
                $votes[$vote] = ($votes[$vote] + 1)
                $voteValide = $true
            } else {
                Write-Host "Vote invalide. Soit le nom ne correspond pas à un
joueur, soit tu as voté pour toi-même ou pour un joueur immunisé."
            }
        }
    }

    # Afficher les résultats des votes
    Write-Host "Résultats du vote :"
    $votes.GetEnumerator() | Sort-Object Value -Descending | ForEach-Object {
Write-Host "$($_.Key): $($_.Value) vote(s)" }

    # Trouver le joueur avec le plus de votes
    $maxVotes = ($votes.Values | Sort-Object -Descending | Select-Object -First 1)

    # Liste des joueurs ayant le plus de votes (en cas d'égalité)
    $candidatsElimination = $votes.GetEnumerator() | Where-Object { $_.Value -eq
$maxVotes }
```

```
    if ($candidatsElimination.Count -eq 1) {
        $elimine = $candidatsElimination.Key
        Write-Host "$elimine a été éliminé."
        # Éliminer le joueur de l'équipe
        if ($Equipe1.Value -contains $elimine) {
            $Equipe1.Value = $Equipe1.Value - $elimine
        } else {
            $Equipe2.Value = $Equipe2.Value - $elimine
        }
    } else {
        # Tirage au sort en cas d'égalité
        $tirage = Get-Random -InputObject ($candidatsElimination | ForEach-Object
{ $_.Key })
        Write-Host "Il y a égalité, tirage au sort. Le joueur éliminé est :
$tirage"
        # Éliminer le joueur tiré au sort
        if ($Equipe1.Value -contains $tirage) {
            $Equipe1.Value = $Equipe1.Value - $tirage
        } else {
            $Equipe2.Value = $Equipe2.Value - $tirage
        }
    }

    # Retourner les nouvelles équipes après élimination
    return $Equipe1 $Equipe2
}

function Gérer-Jeu {
    param (
        [Parameter(Mandatory=$true)]
        [ref]$Equipe1,

        [Parameter(Mandatory=$true)]
        [ref]$Equipe2,

    )
    $joueursRestants = $Equipe1.Value + $Equipe2.Value
    while ($joueursRestants.Count -gt 2) {
            $immunise = $joueursRestants | Get-Random
            Write-Host "$immunise est immunisé, les autres joueurs sont dans la
sauce."
        $joueursConseil = $joueursRestants | Where-Object { $_ -ne $immunise }
#Where-Object { $_ -ne $immunise } sert à filtrer une collection d'objet pour
sortir celui qui est imunisé en utilisant une comparaison, $_ représente le
joueur, -ne c'est "not equal"
        $nouveauxEquipes = Gérer-Consel -Equipe1 ([ref]$joueursConseil) -
joueurImmunise $immunise #Appel de la fonction pour effectuer le conseil
        $joueursRestants = $nouveauxEquipes
    }
    $grandGagnant = $joueursRestants | Get-Random

}
```