

# Q1. On crée une chaîne de caractères : chaine = "j'aime les caulleresiens".  
Quel est le type  
# de cette variable ?

```
chaine1 = "j'aime les caulleresiens"
```

```
println(typeof(chaine1))  
#Q2. Récupérer les caractères de 3 à 6  
stringq2 = chaine1[3:6]  
println(stringq2)
```

# Q3. Récupérer les caractères de 20 jusqu'à la fin de la chaîne

```
stringq3 = chaine1[20:end]  
println(stringq3)
```

#Q4. Idem avec des sauts de 3 caractères

```
stringq4 = chaine1[20:3:end]  
println(stringq4)
```

#Q5 Q5. Ecrire la chaîne de caractère à l'envers

```
stringq5 = chaine1[end:-1:1]  
println(stringq5)
```

#Q6. Extraire la chaîne de caractères de 1 à 6 et programmer un message constitué 3 répétitions de

# cette chaîne, chaque répétition étant séparée par "..."

```
stringq6 = chaine1[1:6]  
result = join([stringq6, "..."])  
global n = 10  
global result2 = ""
```

```
for i in 1:3  
    global result2 = result2 * result
```

```
end  
println(result2)
```

#Q6. Extraire la chaîne de caractères de 1 à 6 et programmer un message constitué de N répétitions de

# cette chaîne, chaque répétition étant séparée par "...". Faire une boucle (N=10)

```
stringq7 = chaine1[1:6]  
resultbis = join([stringq7, "..."])
```

```

global nbis = 10
global result2bis = ""

for i in 1:nbis
    global result2bis = result2bis * resultbis

end
println(result2bis)

#Q7 Q7. Décomposer la nouvelle chaîne de caractères avec la fonction split et
le séparateur "...".

v = split(result2bis, "...")
println(v)

# Q8. Entrer la chaîne de caractères : episode = "Alias-S03E13-
Phase_one.mkv". Utiliser les
# expressions régulières avec findfirst. r"expression_reguliere". Test : trouver
# un caractère donné. Test : trouver un chiffre entre 0 et 9.

episode = "Alias-S03E13-Phase_one.mkv"
inde = findfirst(r"0", episode)
println(inde)

index = findfirst(r"[0-9]", episode)
println(index)

#Déterminer de cette manière le numéro de l'épisode et faire une impression
écran
#regex and findnext and other find.. fonction to find one that find an occurence
and return the next value
numero_episode = findfirst(r"E\d+", episode)

println(numero_episode[2], numero_episode[3])

#how to print the value and not only the position of it ? :

println(episode[numero_episode[2]], episode[numero_episode[3]])

# Q10. Tester la commande findnext("as",episode,8) et commenter
findnext("as",episode,8)

# the fonction findnext in the line 82 have 3 parameters findnext("the thing to
search", the array where we search the thing, where we start to search the
thing)

```

#Q11. Tester la commande occursin(r"-([0-9|A-Z])\*-",episode) et commenter

```
occursin(r"-([0-9|A-Z])*-",episode)
```

*#occursin is a fonction that cerify if there is an occurence of a certain parameters , giving back ether true or false*

*#here we ask if there is numbers or/and letter between two '-' in the array episode*

#Q12. Remplacer l'extension du fichier .mkv par .mp4

```
episode_mp4 = episode[1:end-4]  
episode_mp4 = episode_mp4 * ".mp4"
```

```
println(episode_mp4)
```

RESULTAT

String

aime

siens

sn

sneiserelluac sel emia'j

j'aime...j'aime...j'aime...

j'aime...j'aime...j'aime...j'aime...j'aime...j'aime...j'aime...j'aime...j'aime...

SubString{String}["j'aime", "j'aime", "j'aime", "j'aime", "j'aime", "j'aime",  
"j'aime", "j'aime", "j'aime", "j'aime", ""]

8:8

8:8

1112

13

Alias-S03E13-Phase\_one.mp4

EXERCICE 6

##### EXERCICE

6#####

```
print("EXERCICE 6\n")
```

# Q1. Créer une fonction `hello_world` qui affiche un message de bienvenue.  
Lancer la fonction sur l'invite julia :  
`# Julia > hello_world()`

```
function hello_world()
    print("hello_world\n")
    return 0
end
```

#Q2. Créer une fonction `name_length()` qui affiche une question (on demande le nom), qui lit le nom entré par l'utilisateur et qui affiche un message du type : « Hello, Mister Smith. Your name has 5 letters »  
`#Julia > name_length`

```
function name_length()
    print("Quel est votre nom? : ")
    nom = readline()
    longueur_nom = length(nom)
    println("Hello, $nom your name have $longueur_nom letter")
end
```

#####EXERCICE 7 #####

#Q1. Créer la chaîne de caractères Julia : `chaine = ". Faire une boucle pour afficher chaque # caractère de la chaîne avec son indice`

```
chaine = "Langage Julia"
longueur_chaine = length(chaine)

for i in 1:longueur_chaine
    print(chaine[i])
    print(i)
end
```

#Q2 Q2. Créer une fonction `decoration_string` qui décore la chaîne précédente avec des ". " entre chaque lettre du type :  
`#"L.a.n.g.a.g.e .J.u.l.i.a."`

```

function decoration_string()
    chaine = "Langage Julia"
    longueur_chaine = length(chaine)
    for i in 1:longueur_chaine
        print(chaine[i])
        print(".")
    end
end

```

# qQ3. Appeler la fonction avec une autre chaîne donnée en entrée. Exemple :  
#Julia> decoration\_string("je ne sais pas")

```

function decoration_string_bis(chaine)
    longueur_chaine = length(chaine)
    for i in 1:longueur_chaine
        print(chaine[i])
        print(".")
    end
end

```

##### exercice 7 #####

#Q1. Créer une chaîne chaine = "To be or not to be, that is the question".  
Trouver le premier caractère  
#"e" avec la fonction findnext('e',chaine,indice\_de\_depart)

```

chaine_be = "To be or not to be, that is the question"
findnext("e", chaine_be, 1)

```

#q2 Q2. Utiliser cette fonction avec un caractère qui n'est pas présent dans la chaîne. Exemple :

```

#Julia>
#i got "nothing"
#Tester la condition d==nothing. Quel est le résultat ?
# i got True

```

```

testt = findnext('z',chaine_be,1)
println(testt)
println(testt==nothing)

```

# Q3. Créer un vecteur vide.  
# Julia> locations = []  
# Puis, en s'appuyant sur les tests précédents, faire une boucle avec while avec la condition d != nothing  
#Afficher le nombre de caractères 'e' trouvés dans la chaîne avec leur position dans le chaine.

```
location = []
chaine_be_bis = "To be or not to be, that is the question"
length_be_bis = length(chaine_be_bis)
global d = 1

while d != nothing && d <= length_be_bis

    if chaine_be_bis[d] == 'e'
        push!(location, d)
    end
    global d = d + 1
end

println(location)
```

## RESULTAT

EXERCICE 6  
L1a2n3g4a5g6e7 8J9u10l11i12a13nothing  
true  
Any[5, 18, 31, 35]