# REG7
# Cours des expressions régulières

Ayoub Bouchama
Ewen Le Bihan
Gauthier Rancoule
Florent Puy
Clément Cognard
Clément Safer
Ilyasse Alioui
Raphaël Giudice

April 22, 2023

## Contents

# 1 Introduction aux expressions régulières

Les expressions régulières sont des modèles de caractères qui permettent d'effectuer des recherches dans un texte en spécifiant un schéma à suivre. Par exemple, en utilisant l'expression régulière "the", on peut rechercher toutes les occurrences de la lettre t suivie de la lettre h suivie de la lettre e dans un texte.

Voici un autre exemple d'expression régulière : supposons que vous vouliez trouver toutes les adresses e-mail dans un texte. Vous pouvez utiliser l'expression régulière suivante :

**[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]2,**

Cette expression régulière correspond à l'adresse e-mail standard et recherche tous les caractères alphanumériques, les points, les tirets, les tirets bas et les signes de pourcentage dans la partie avant l'arobase, puis tous les caractères alphanumériques, les tirets et les points dans la partie après l'arobase, et enfin deux caractères alphabétiques pour le domaine de niveau supérieur (par exemple, com, org, net).

# 2 Métacaractères

Les expressions régulières utilisent les meta-caractères comme éléments de base. Les meta-caractères sont interprétés de manière spécifique et certains d'entre eux ont des significations particulières qui sont indiquées en les plaçant entre crochets. En d'autres termes, les meta-caractères constituent une sorte de vocabulaire spécialisé utilisé pour la création de motifs de recherche. Ils sont cruciaux pour le fonctionnement des expressions régulières et permettent aux programmeurs de rechercher et de manipuler des chaînes de caractères de manière très précise.

| Meta-caractère | Description |
|---|---|
| . | Un point coïncide avec n'importe quel caractère unique à part le retour à la ligne. |
| [ ] | Classe de caractères. Coïncide avec n'importe quel caractère entre crochets. |
| [ˆ] | Négation de classe de caractère. Coïncide avec n'importe quel caractère qui n'est pas entre les crochets. |
| * | Coïncide avec 0 ou plus répétitions du caractère précédent. |
| + | Coïncide avec 1 ou plus répétitions du caractère précédent. |
| ? | Rend le caractère précédent optionnel. |
| {n,m} | Accolades. Coïncide avec au moins 'n' mais pas plus que 'm' répétition(s) du caractère précédent. |
| (xyz) | Groupe de caractères. Coïncide avec les caractères 'xyz' dans l'ordre exact. |
| — | Alternation (ou). Coïncide soit avec le caractère avant ou après le symbole. |
| \ | Échappe le prochain caractère. Cela permet de faire coïncider des caractères réservés tels que [ ] ( ) { } . * + ? ˆ $ \— |
| ˆ | Coïncide avec le début de la chaîne de caractères (string). |
| $ | Coïncide avec la fin de la chaîne de caractères (string). |

## 2.1 Full Stop

Le point est un caractère spécial en expression régulière qui représente n'importe quel caractère unique, à l'exception des caractères de retour à la ligne. Par exemple, l'expression régulière ".ar" peut correspondre à des chaînes de caractères telles que "car", "bar", "tar", etc., où le point représente n'importe quel caractère unique.

Un autre exemple d'utilisation du point en expression régulière est l'expression régulière "c.t", qui peut correspondre à des chaînes de caractères telles que "cat", "cot", "cut", etc. Le point dans cette expression régulière correspond à n'importe quel caractère unique, ce qui permet de trouver toutes les chaînes de caractères qui ont la même structure que "c.t".

## 2.2 Inclusion des caractères

Les classes de caractères en expression régulière sont également appelées inclusions de caractères et sont définies à l'aide de crochets. Pour définir une plage de caractères, un trait d'union est utilisé à l'intérieur des crochets, et l'ordre des caractères dans la plage n'a pas d'importance. Par exemple, l'expression régulière [Tt]he signifie qu'il peut y avoir un T majuscule ou un t minuscule, suivi de la lettre h, puis de la lettre e.

Un autre exemple d'utilisation d'inclusions de caractères en expression régulière est l'utilisation de l'expression [0-9], qui correspond à tous les chiffres compris entre 0 et 9. Cette expression peut être utilisée pour trouver toutes les occurrences de chiffres dans une chaîne de caractères. Par exemple, l'expression régulière "ˆ[0-9]+" permet de trouver la première série de chiffres au début d'une chaîne de caractères. Si la chaîne de caractères est "123ABC456DEF", l'expression régulière trouvera "123".

En règle générale, le caractère circonflexe représente le début d'une chaîne de caractères. Cependant, lorsqu'il est utilisé immédiatement après le crochet ouvrant, il permet d'exclure la plage de caractères définie à l'intérieur des crochets. Par exemple, l'expression régulière [ˆc]ar signifie : n'importe quel caractère sauf c, suivi par la lettre a, suivie par la lettre r.

## 2.3 Répetition

En expression régulière, les symboles spéciaux +, *, ou ? sont utilisés pour indiquer la répétition d'un sous-modèle dans une expression régulière. Le comportement de ces symboles varie selon le contexte d'utilisation. L'astérisque est l'un de ces symboles qui permettent de spécifier la présence d'un sous-modèle zéro fois ou plusieurs fois.

### 2.3.1 Astérisque

Le symbole * en expression régulière permet de spécifier zéro ou plusieurs occurrences du modèle précédent. Par exemple, l'expression régulière `a*` correspond à zéro ou plusieurs occurrences de la lettre “a”. Si le symbole * est utilisé après une liste de caractères, cela signifie que la liste entière peut se répéter zéro ou plusieurs fois. Par exemple, l'expression régulière `[a-z]*` signifie que la chaîne peut contenir n'importe quel nombre de lettres minuscules.

En utilisant le méta-caractère `.`, le symbole * peut être utilisé pour correspondre à n'importe quelle chaîne de caractères en utilisant l'expression `.*` . De même, le caractère espace vide `\s` peut être utilisé avec le symbole * pour correspondre à une chaîne d'espaces vides avec l'expression `\scat\s` signifiant qu'il y a zéro ou plusieurs espaces, suivis par le mot cat", suivi à nouveau par zéro ou plusieurs espaces. Par exemple, l'expression régulière `\scat\s` correspondrait à la phrase The fat cat sat on the concatenation".

### 2.3.2 Le plus

Le meta-caractère `+` en expression régulière signifie "une ou plusieurs occurrences" du caractère ou modèle précédent. Par exemple, l'expression régulière `c.+t` signifie qu'il doit y avoir la lettre "c" minuscule, suivie d'au moins un caractère quelconque, suivi de la lettre "t" minuscule. Le `+` exige donc qu'il y ait au moins un caractère entre la lettre "c" et la lettre "t". Le "t" qui correspond est le dernier "t" de la chaîne.
Voici un autre exemple : l'expression régulière `\d+` correspondra à une ou plusieurs occurrences de chiffres. Par exemple, l'expression régulière `\d+apple` signifie qu'il doit y avoir un ou plusieurs chiffres, suivi du mot "apple". Donc, "123apple" correspondra à cette expression régulière mais pas "apple123".

### 2.3.3 Le point d'interrogation

Le symbole ? en expression régulière signifie "zéro ou une occurrence" du caractère ou modèle précédent. Par exemple, l'expression régulière c.?t signifie qu'il doit y avoir la lettre "c" minuscule, suivi d'un caractère quelconque (ou aucun), suivi de la lettre "t" minuscule. Le "t" qui correspond est le dernier "t" de la chaîne.
Voici un autre exemple : l'expression régulière ?apple signifie qu'il doit y avoir zéro ou un chiffre, suivi du mot "apple". Donc, "apple" correspondra à cette expression régulière mais pas "1apple".

## 2.4 Les accolades

Les accolades permettent de spécifier le nombre exact de répétitions d'un caractère ou d'un modèle. Par exemple, l'expression régulière c2t signifie qu'il doit y avoir la lettre "c" minuscule, suivie de deux fois la lettre "c" minuscule, suivi de la lettre "t" minuscule. Le "t" qui correspond est le dernier "t" de la chaîne.
Voici un autre exemple : l'expression régulière 3apple signifie qu'il doit y avoir trois chiffres, suivi du mot "apple". Donc, "123apple" correspondra à cette expression régulière mais pas "12apple".

## 2.5 Groupement de caractère

Les parenthèses sont utilisées pour grouper des sous-modèles dans une expression régulière.
Par exemple, l'expression régulière (cat—dog) signifie qu'il doit y avoir soit "cat" soit "dog". Les parenthèses sont également utilisées pour extraires des sous-chaînes de caractères. Par exemple, l'expression régulière (\d3)-(\d3)-(\d4) permet d'extraire les trois groupes de chiffres séparés par des tirets.
Un autre exemple simple est l'expression régulière (\d+). Elle permet d'extraire un nombre quelconque de chiffres.

## 2.6 L'alternation

L'alternation est utilisée pour spécifier plusieurs modèles possibles pour une expression régulière. Par exemple, l'expression régulière (cat—dog) signifie qu'il doit y avoir soit cat" soit dog". L'alternation est également utilisée pour extraire des sous-chaînes de caractères. Par exemple, l'expression régulière (\d3)-(\d3)-(\d4) permet d'extraire les trois groupes de chiffres séparés par des tirets.
Un autre exemple simple est l'expression régulière (\d+). Elle permet d'extraire un nombre quelconque de chiffres.

## 2.7 Caractère d'échappement

Les caractères d'échappement sont utilisés pour spécifier des caractères spéciaux dans une expression régulière. Par exemple, l'expression régulière \s signifie qu'il doit y avoir un espace vide. L'expression régulière \s+ signifie qu'il doit y avoir un ou plusieurs espaces vides. L'expression régulière \s* signifie qu'il doit y avoir zéro ou plusieurs espaces vides. L'expression régulière \s? signifie qu'il doit y avoir zéro ou un espace vide. L'expression régulière \s{2} signifie qu'il doit y avoir deux espaces vides. L'expression régulière \s2, signifie qu'il doit y avoir deux ou plusieurs espaces vides.
L'expression régulière \s2,4 signifie qu'il doit y avoir deux, trois ou quatre espaces vides.

Voici une liste des caractères d'échappement les plus courants : \. * \+ \? \[ \] ^\$ \( \) \ \ |\s \S \d \D \w \W \b \B \t \n \r \f \v \cX \xhh \uhhhh \uhhhhhhh \Q \E \A \Z \z \G \p \P \N \k \K \1 \2 \3 \4 \5 \6 etc...

### 2.8 Ancres

Les ancres sont utilisées pour spécifier des positions spécifiques dans une chaîne de caractères. Par exemple, l'expression régulière `^\d{3}-\d{3}-\d{4}$` signifie qu'il doit y avoir trois chiffres, suivi d'un tiret, suivi de trois chiffres, suivi d'un tiret, suivi de quatre chiffres, et que la chaîne de caractères doit commencer et se terminer par ces caractères.
- Le circumflexe (`^`) est utilisé pour spécifier le début de la chaîne de caractères.
- Le dollar (`$`) est utilisé pour spécifier la fin de la chaîne de caractères.

## 3 Liste des caractères abrégés

| Abréviation | Description |
|---|---|
| . | N'importe quel caractère à part le retour de ligne |
| \w | Caractères alphanumériques : [a-zA-Z0-9_] |
| \W | Caractères non-alphanumériques : [^\w] |
| \d | Chiffres : [0-9] |
| \D | Non-numériques : [^\d] |
| \s | Espace vide : [\t\n\f\r\pZ] |
| \S | Tout sauf espace vide : [^\s] |

## 4 Recherche

Les groupes non-capturants de recherche en avant et en arrière sont utilisés pour identifier un schéma sans le capturer dans la liste de correspondance. Les recherches en avant positives sont utiles lorsque l'on veut identifier un schéma précédé ou suivi par un autre schéma. Par exemple, si nous voulons trouver tous les chiffres qui sont précédés par le caractère $ dans la chaîne "$4.44 and $10.88", nous pouvons utiliser l'expression régulière suivante : "(?¡=$)[0-9]*". Cette expression signifie : trouver tous les nombres qui contiennent le caractère . et qui sont précédés par le caractère $. Les expressions régulières contiennent plusieurs types de recherches, telles que :

| Symbole | Description |
|---|---|
| ?= | Recherche en avant positive |
| ?! | Recherche en avant négative |
| ?<= | Recherche en arrière positive |
| ?<! | Recherche en arrière négative |

### 4.1 Recherche avant positive

La recherche en avant positive dans les expressions régulières permet de s'assurer que la première partie de l'expression est suivie par une expression recherchée spécifique. Seul le texte qui correspond à la première partie de l'expression est retourné. Pour définir une recherche en avant positive, on utilise des parenthèses contenant un point d'interrogation et un signe égal : `(?=...)`. L'expression de recherche est écrite après le signe égal dans les parenthèses. Par exemple, l'expression régulière suivante `a-z(?=oo)` recherche la lettre minuscule suivie de la séquence "oo". Cela trouvera toutes les lettres minuscules qui sont suivies de la séquence "oo" sans inclure "oo" dans la liste de correspondance.

### 4.2 Recherche en avant négative

La recherche en avant négative dans les expressions régulières permet de s'assurer que la première partie de l'expression n'est pas suivie par une expression recherchée spécifique. Seul le texte qui correspond à la première partie de l'expression est retourné. Pour définir une recherche en avant négative, on utilise des parenthèses contenant un point d'interrogation et un signe exclamé : `(?!...)`. L'expression de

recherche est écrite après le signe exclamé dans les parenthèses. Par exemple, l'expression régulière suivante `a-z(?!oo)` recherche la lettre minuscule qui n'est pas suivie de la séquence "oo". Cela trouvera toutes les lettres minuscules qui ne sont pas suivies de la séquence "oo" sans inclure "oo" dans la liste de correspondance.

### 4.3 Recherche en arrière positive

La recherche en arrière positive dans les expressions régulières permet de s'assurer que la dernière partie de l'expression est précédée par une expression recherchée spécifique. Seul le texte qui correspond à la dernière partie de l'expression est retourné. Pour définir une recherche en arrière positive, on utilise des parenthèses contenant un point d'interrogation, un signe égal et un signe inférieur : `(?<=...)`. L'expression de recherche est écrite après le signe égal et le signe inférieur dans les parenthèses. Par exemple, l'expression régulière suivante `(?<=oo)[a-z]` recherche la lettre minuscule qui est précédée de la séquence "oo". Cela trouvera toutes les lettres minuscules qui sont précédées de la séquence "oo" sans inclure "oo" dans la liste de correspondance.

### 4.4 Recherche en arrière négative

La recherche en arrière négative dans les expressions régulières permet de s'assurer que la dernière partie de l'expression n'est pas précédée par une expression recherchée spécifique. Seul le texte qui correspond à la dernière partie de l'expression est retourné. Pour définir une recherche en arrière négative, on utilise des parenthèses contenant un point d'interrogation, un signe exclamé et un signe inférieur : `(?<!...)`. L'expression de recherche est écrite après le signe exclamé et le signe inférieur dans les parenthèses. Par exemple, l'expression régulière suivante recherche la séquence "oo" suivie de la lettre minuscule. Cela trouvera toutes les lettres minuscules qui ne sont pas précédées de la séquence "oo" sans inclure "oo" dans la liste de correspondance.

## 5 Drapeaux

Les drapeaux sont aussi appelés modifieurs car ils modifient la sortie d'une expression régulière. Ces drapeaux peuvent être utilisés dans n'importe quel ordre et combinaison et font partie intégrante de la RegExp.

| Drapeau | Description |
|---|---|
| i | Insensible à la casse : Définit que la correspondance sera insensible à la casse. |
| g | Recherche globale : Recherche la correspondance dans la chaine de caractères (string) entière. |
| m | Multiligne : Meta-caractère ancre qui agit sur toutes les lignes. |

### 5.1 Sensibilité à la casse

L'insensibilité à la casse est un drapeau qui permet de rechercher une expression régulière sans tenir compte de la casse. Par exemple, l'expression régulière suivante recherche la séquence `oo` suivie de la lettre minuscule, en étant insensible à la casse : `/oo[a-z]/i`. Cela trouvera toutes les lettres minuscules qui ne sont pas précédées de la séquence `oo` sans inclure `oo` dans la liste de correspondance.

### 5.2 Correspondance globale

La correspondance globale est un drapeau qui permet de rechercher une expression régulière dans la chaîne de caractères entière. Par exemple, l'expression régulière suivante recherche la séquence `oo` suivie de la lettre minuscule, de façon globale dans la chaîne de caractères : `/oo[a-z]/g`. Cela trouvera toutes les lettres minuscules qui ne sont pas précédées de la séquence `oo` sans inclure `oo` dans la liste de correspondance.

### 5.3 Multiligne

La correspondance multiligne est un drapeau qui permet de rechercher une expression régulière sur plusieurs lignes. Par exemple, l'expression régulière suivante recherche la séquence `oo` suivie de la lettre minuscule, de façon multiligne : `/oo[a-z]/gm`. Cela trouvera toutes les lettres minuscules qui ne sont pas précédées de la séquence `oo` sans inclure `oo` dans la liste de correspondance.

## 6 Conclusion

En conclusion, les expressions régulières sont un outil puissant pour la manipulation et la recherche de motifs dans des chaînes de caractères. Elles offrent une flexibilité et une précision incroyables pour identifier et extraire des données spécifiques à partir de grands ensembles de données textuelles. Nous avons exploré les éléments de base des expressions régulières, tels que les caractères littéraux, les classes de caractères, les quantificateurs et les métacaractères d'ancrage. Nous avons également étudié les concepts plus avancés tels que les recherches en avant et en arrière, ainsi que l'utilisation de drapeaux pour modifier la sortie de nos expressions régulières.
La maîtrise des expressions régulières est essentielle pour les développeurs et les analystes de données qui travaillent avec du texte. Bien que cela puisse sembler intimidant au début, la pratique et la compréhension de ces concepts fondamentaux vous aideront à écrire des expressions régulières plus efficaces et précises.