Programmation et projet encadré - LYSET13

Expressions Régulières

Yoann Dupont yoann.dupont@sorbonne-nouvelle.fr
Pierre Magistry pierre.magistry@inalco.fr

2025-2026

Université Sorbonne-Nouvelle
INALCO
Université Paris-Nanterre

Qu'est-ce qu'une expression régulière ?

- Une **expression régulière** (regex) est une séquence de caractères qui définit un motif de recherche.
- Utilisée pour :
  - Valider des formats (url, e-mails, numéros de téléphone, etc.)
  - Extraire des informations dans un texte
  - Rechercher et remplacer du texte

Pour bien suivre cette séquence

- Rendez-vous sur `https://regex101.com`
- essayez au fur et à mesure

Caractères littéraux et métacaractères

Caractères littéraux

Recherchent une correspondance exacte (ex: `chat` correspond à "chat").

les Métacaractères devront être “échappés" pour retrouver un sens littéral

`.` : n'importe quel caractère (sauf nouvelle ligne)

`*` : zéro ou plusieurs occurrences du caractère précédent

`+` : une ou plusieurs occurrences

`?` : zéro ou une occurrence

À distinguer des Jokers dans les chemins !

Classes de caractères [ ]

Définition

Les classes de caractères permettent de spécifier un ensemble de caractères autorisés à une position donnée.

- `[abc]` : correspond à "a", "b", ou "c".
- `[a-z]` : correspond à n'importe quelle lettre minuscule de "a" à "z".
- `[A-Z]` : correspond à n'importe quelle lettre majuscule de "A" à "Z".
- `[0-9]` : correspond à n'importe quel chiffre de "0" à "9".
- `[^ ]` : correspond à n'importe quel caractère **non** présent dans la classe (ex: `[^0-9]` correspond à un caractère qui n'est pas un chiffre).

Exemples

- `[aeiou]` : correspond à une voyelle.
- `[a-zA-Z]` : correspond à une lettre majuscule ou minuscule.
- `[0-9][0-9]` : correspond à un nombre à deux chiffres.

Métacaractères de frontières (ancres)

Définition

Les ancres permettent de spécifier la position d'un motif dans une chaîne de caractères.

- `^` : début de la chaîne (ex: `^Bonjour` correspond à "Bonjour" uniquement en début de chaîne).
- `$` : fin de la chaîne (ex: `au revoir$` correspond à "au revoir" uniquement en fin de chaîne).
- `\b` : frontière de mot (ex: `\bchat\b` correspond à "chat" comme mot entier).
- `\B` : absence de frontière de mot (ex: `\Bachat` correspond à "achat" dans "purchase").

Quantificateurs

- `*` : zéro ou plusieurs occurrences
- `+` : une ou plusieurs occurrences
- `?` : zéro ou une occurrence
- `{n}` : exactement `n` occurrences
- `{n,m}` : entre `n` et `m` occurrences

Exemple

`a{2,4}` correspond à "aa", "aaa", ou "aaaa".

Groupes et captures

- Les parenthèses ( ) permettent de regrouper des motifs. quantificateurs s'appliquent alors à tout le groupe
- Les captures permettent de récupérer des parties du texte :
  - `(motif)` : capture le texte correspondant au motif
  - Référence arrière : `\1`, `\2`, etc.
- les groupes sont souvent utiliser avec l'opérateur de disjonction | ("ou")
- Voir aussi :
  - les groupes non capturants (?: ...) qui ne créent pas de référence \n
  - les *look ahead* et *look behind* qui sont non capturants et peuvent être positifs ou négatifs (*pla: ...) (*plb: ...) (*nla: ...) (*nlb: ...).
    Ils permettent de vérifier le contexte d'une expression sans "consommer" de caractère (comme le font les \b)

Quelques exemples à construire

En utilisant le site regex101.com:

Essayez de repérer:

- un numéro de téléphone (avec ou sans l'indicatif du pays ?)
- une adresse email
- toutes les formes du verber "manger" dans un texte en français

Utilisation des regex en bash

- tester sur le site `regex101.com`
- filtrer et extraire du texte avec `grep`
- chercher-remplacer avec `sed`

À partir de regex101.com

- récupérer le texte `docs/mail-liste-hn.txt` sur notre git et copier-coller son contenu dans l'interface web.
- construire une regex pour retrouver toutes les url.
- construire une regex pour retrouver tous les numéro de téléphone.
- construire une regex pour isoler chaque mot du texte.
- tester dans le terminal avec `grep`

Remplacements avec Sed

- récupérer le texte `docs/pg16066.txt` sur notre git (le premier tome de Don Quichotte, obtenu sur le site du proejt Gutenberg.org)
- Repérage des moulins à vent
  - avec grep, isoler les lignes contenant les occurrences de moulins
  - avec sed, remplacer toutes les occurrences de “moulins à vent" par moulins-à-vent (en utilisant un argument 's/.../.../')
- Pseudo-lemmatisation de MANGER
  - avec grep, isoler les lignes contenant les occurrences du verbe "manger" quelque soit sa flexion.
  - avec sed, remplacer toutes ces occurrences par la forme MANGER (en utilisant un argument 's/.../.../')
  - toujours avec sed, effectuer la même opération mais conserver l'information sur la flexion qu'on indiquera après MANGER, en la séparant d'un caractère +. par exemple, “mangeaient" doit devenir “MANGER+eaient"

Revisitons le passage au html du miniprojet

On aurait pu faire la seconde partie du mini projet (passer du format tsv au html) sans modifier le premier script mais en en écrivant un second, pour convertir notre tableau TSV en un document html.
On pourrait appeler ce script tsv2html.sh...