Référence rapide Expressions régulières

Motifs, quantificateurs, groupes, assertions, drapeaux

Motifs de base

Métacaractères

`.`	N'importe quel caractère (sauf saut de ligne)
`^`	Début de chaîne / ligne
`$`	Fin de chaîne / ligne
`*`	0 fois ou plus ce qui précède
`+`	1 fois ou plus ce qui précède
`?`	0 ou 1 fois ce qui précède (optionnel)
`\`	Échapper un métacaractère

Correspondance littérale

```
hello     # matches "hello" exactly
a.c       # matches "abc", "alc", "a-c", etc.
\.txt     # matches literal ".txt"
```

Classes de caractères

Expressions entre crochets

`[abc]`	Correspond à a, b ou c
`[^abc]`	Tout sauf a, b, c
`[a-z]`	Lettre minuscule
`[A-Z]`	Lettre majuscule
`[0-9]`	Chiffre
`[a-zA-Z0-9]`	Alphanumérique

Classes abrégées

`\d`	Chiffre `[0-9]`
`\D`	Non-chiffre `[^0-9]`
`\w`	Caractère de mot `[a-zA-Z0-9_]`
`\W`	Caractère non-mot
`\s`	Espace blanc `[ \t\n\r\f]`
`\S`	Non-espace blanc

Quantificateurs

Quantificateurs gourmands

`*`	0 fois ou plus (gourmand)
`+`	1 fois ou plus (gourmand)
`?`	0 ou 1 fois (gourmand)
`{n}`	Exactement n fois
`{n,}`	n fois ou plus
`{n,m}`	Entre n et m fois

Quantificateurs paresseux

`*?`	0 fois ou plus (paresseux / non gourmand)
`+?`	1 fois ou plus (paresseux)
`??`	0 ou 1 fois (paresseux)
`{n,m}?`	Entre n et m fois (paresseux)

Les quantificateurs paresseux correspondent au minimum de caractères possible

Gourmand vs Paresseux

```
<.+>      # greedy: "<b>bold</b>"
<.+?>     # lazy:   "<b>"
```

Ancres

`^`	Début de chaîne (ou de ligne avec le drapeau `m`)
`$`	Fin de chaîne (ou de ligne avec le drapeau `m`)
`\b`	Limite de mot
`\B`	Non-limite de mot
`\A`	Début de chaîne (non affecté par `m`)
`\Z`	Fin de chaîne (non affecté par `m`)

Exemples d'ancres

```
^Hello        # starts with "Hello"
world$        # ends with "world"
\bword\b      # "word" as whole word
\Bword\B      # "word" inside another word
```

Groupes & Alternance

Groupes capturants

```
(abc)         # capture group: match "abc"
(a|b|c)       # alternation: a or b or c
(cat|dog)     # match "cat" or "dog"
(\d{3})-(\d{4})  # groups: "123-4567"
```

Types de groupe

`(pattern)`	Groupe capturant
`(?:pattern)`	Groupe non capturant
`(?P<name>pat)`	Groupe nommé (Python)
`(?<name>pat)`	Groupe nommé (JS, .NET)
`\1` `\2`	Rétroréférence aux groupes 1, 2
`a|b`	Alternance : a ou b

Assertion avant & arrière

`(?=pattern)`	Assertion avant positive
`(?!pattern)`	Assertion avant négative
`(?<=pattern)`	Assertion arrière positive
`(?<!pattern)`	Assertion arrière négative

Exemples d'assertions

```
\d+(?= USD)     # digits followed by " USD"
\d+(?! USD)     # digits NOT followed by " USD"
(?<=\$)\d+      # digits preceded by "$"
(?<!\$)\d+      # digits NOT preceded by "$"
```

Les assertions correspondent à une position sans consommer de caractères

Motifs courants

`\d{1,3}(\.\d{1,3}){3}`	Adresse IPv4 (basique)
`[\w.+-]+@[\w-]+\.[\w.]+`	Email (basique)
`https?://[\w./\-?&#=]+`	URL (basique)
`\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}`	Numéro de téléphone US
`\d{4}-\d{2}-\d{2}`	Date (AAAA-MM-JJ)
`#?[0-9a-fA-F]{6}`	Code couleur hexadécimal

Ces motifs sont simplifiés ; la production peut nécessiter une validation plus stricte

Drapeaux

`g`	Global : trouver toutes les correspondances
`i`	Insensible à la casse
`m`	Multiligne : `^` / `$` correspondent aux limites de ligne
`s`	Dotall : `.` correspond aussi aux sauts de ligne
`x`	Verbeux : ignorer les espaces, autoriser les commentaires
`u`	Unicode : support Unicode complet

Utilisation des drapeaux par langage

```
/pattern/gi           # JavaScript
re.compile(r"pat", re.I | re.M)  # Python
grep -iE "pattern"    # grep (extended)
```