

# PYTHON PAR LA PRATIQUE

---

Présenté par :  
**Xavier TABUTEAU**

## 09. Regex



### Expressions régulières

Pour faire des tests, utilisez le site <https://regex101.com>

Les expressions régulières représentent les motifs qu'on recherche dans une chaîne de caractères. Par exemple, on peut chercher dans un fichier :

Les lignes qui contiennent un numéro de téléphone

Les lignes qui débutent par une adresse IP

On peut chercher dans une chaîne de caractère :

- Une séquence qui commence par « ex » et se termine par « le »
- Les mots de quatre caractères de long
- Les mots sans majuscules

On utilise les symboles suivants qui ont une signification spécifique chacune !

. ^\$\*+?{}[]\()

### Les Metacaractères

^ : précise que le résultat doit commencer par la chaîne de caractères qui le suit.

\$ : précise que le résultat doit se terminer par la chaîne de caractère qui le précède.

. : correspond à n'importe quel caractère sauf le saut de ligne.

## 09. Regex



### Les quantificateurs

- `+` : le résultat peut contenir une ou plusieurs occurrences du caractère qui le précède.
  - `*` : le résultat peut contenir zéro ou plusieurs occurrences du caractère qui le précède.
  - `?` : le résultat peut inclure zéro ou une occurrence du caractère qui le précède.
  - `{x}` : précise le nombre x de répétitions des chaînes de caractères qui le précède.
  - `[]` : obtient le résultat sur plusieurs occurrences.
  - `()` : créer des groupes.
- 
- `|` : correspond à l'expression OU. On peut s'en servir pour trouver une correspondance entre plusieurs expressions.
  - `-` : sert à regrouper des caractères dans un intervalle donné.

```
>>> re.findall("[0-9]+", "Bonjour 111 Aurevoir 222")  
>>> bool(re.match("^S", "XAVIER"))
```

## 09. Regex



*regex.py*  
**Exercice 12**

### Tokens

- `\d` : [0-9]
- `\w` : [a-zA-Z0-9\_]
- `\s` : correspond à tous les espaces dans la chaîne de caractères
- `\.` : juste les points
- `\b` : frontière des mots

### Fonctions usuelles

`re.compile(pattern)`  
`re.search(pattern, string)`

: Compile un motif vers une expression rationnelle compilée.  
: Analyse le string à la recherche du premier emplacement où l'expression rationnelle `pattern` trouve une correspondance, et renvoie l'objet de correspondance trouvé.

`re.findall(pattern, string)`

: Renvoie toutes les correspondances du motif dans la chaîne, sous forme de liste de chaînes ou de tuples. La chaîne est parcourue de gauche à droite, et les correspondances sont retournées dans l'ordre où elles ont été trouvées.

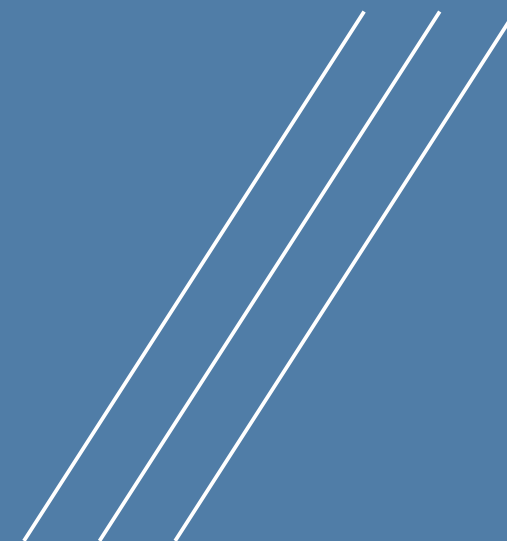
`re.match(pattern, string)`

: Si zéro ou plus caractères au début de `string` correspondent à l'expression rationnelle `pattern`, renvoie l'objet de correspondance trouvé.

`re.sub(pattern, repl, string)`

: Renvoie la chaîne obtenue en remplaçant les occurrences (sans chevauchement) les plus à gauche de `pattern` dans `string` par le remplacement `repl`. Si le motif n'est pas trouvé, `string` est renvoyée inchangée.

# PYTHON INITIATION



Présenté par  
**Xavier TABUTEAU**