Introduction au Traitement Automatique des Langues

3 - Les niveaux de traitement - de «bas niveau»

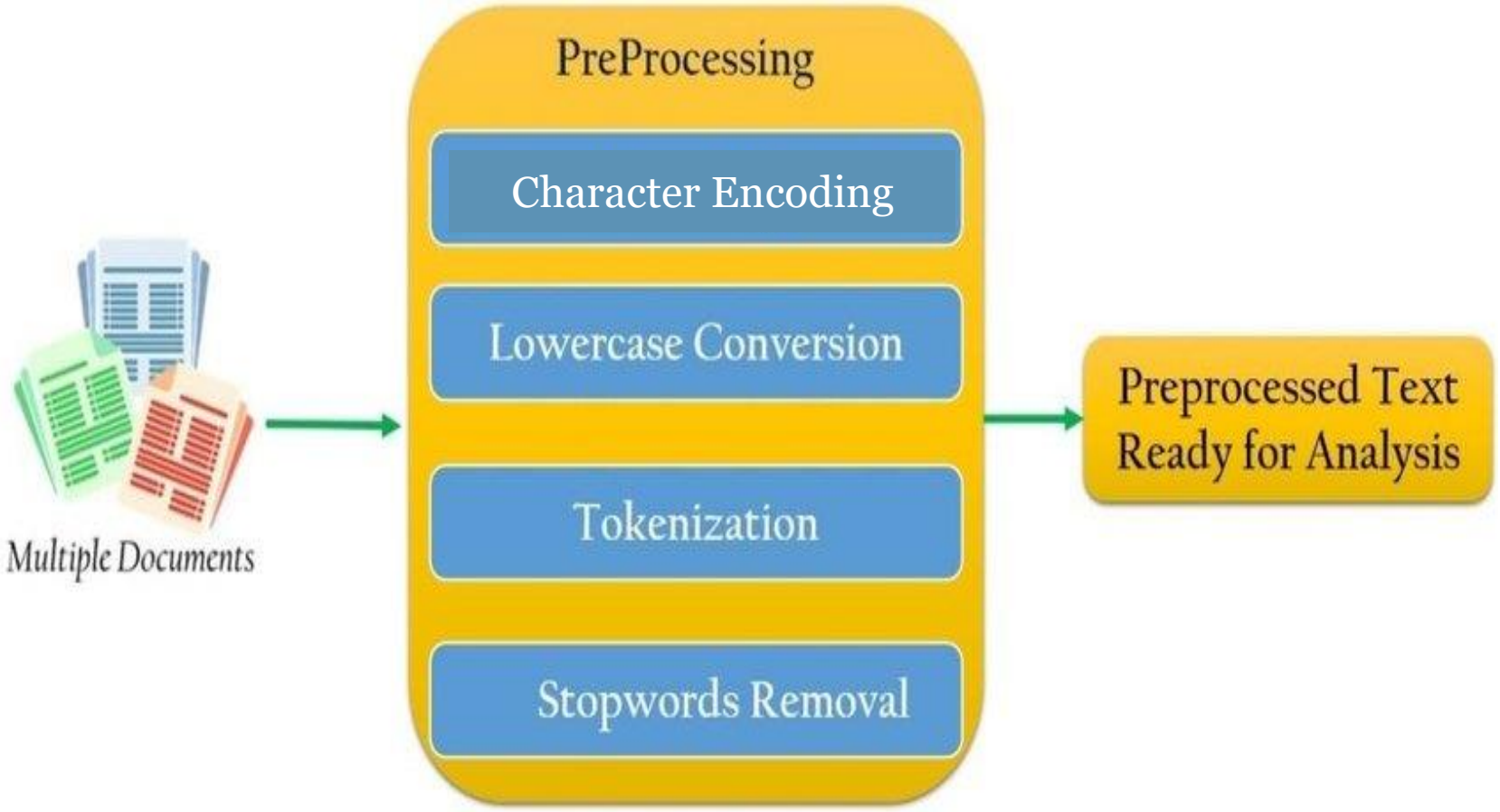
Introduction au traitement automatique des langues

Contenu de la matière :

1) Introduction Générale
2) Les applications du TAL
3) **Les niveaux de traitement - Traitements de «bas niveau»**
4) Les niveaux de traitement - Le niveau lexical
5) Les niveaux de traitement - Le niveau syntaxique
6) Les niveaux de traitement - Le niveau sémantique
7) Les niveaux de traitement - Le niveau pragmatique

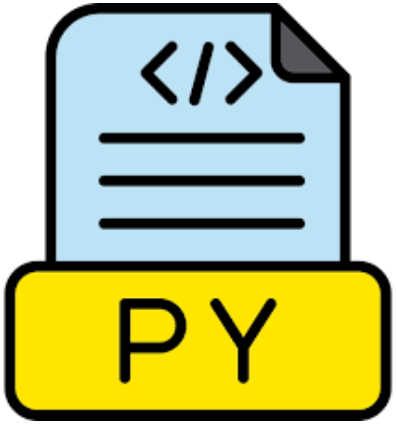
Traitement bas niveau

- Prétraitement (PreProcessing) d'un énoncé textuel :

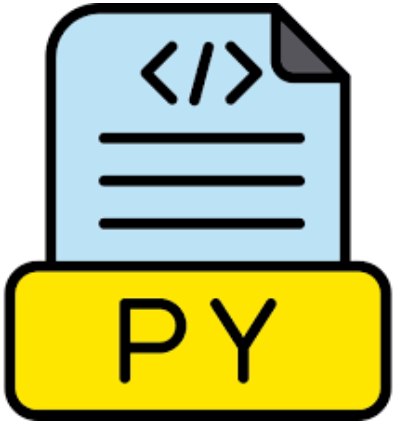


Traitement bas niveau

- **Série TP 1 - Python**



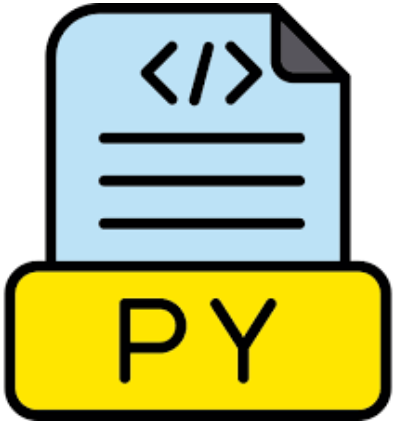
Partie 1 - Découverte



Partie 2 - Exercices

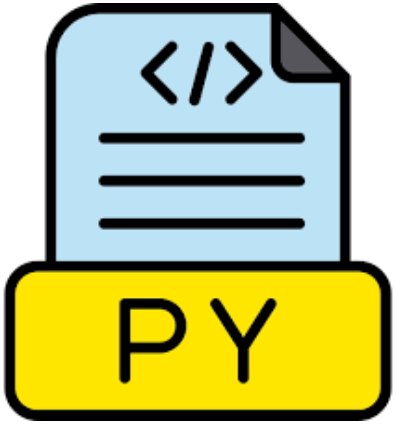
Traitement bas niveau

- **Série TP 2 – Traitement de Bas Niveau**



Partie 1 – Découverte

- Character Encoding
- Tokenization: whitespace & regex



Partie 2 - Exercices

- **Série TP 1 - Python**

- Variables & types
- Type casting
- Assignation multiple de variables
- Affichage : print & string formatting
- Lecture : input
- Data structures : List, Dictionnary, Set, Tuple
  - List : Declaration, Indexes (pos et neg), Slicing, Unpacking, Methods/Functions : append, insert, pop, len, sort.
  - Dict : Declaration, Element Access/Modify, methods (keys, values, items).
  - Sets & Tuples
- if .. elif .. else
- Les boucles : for & while
- Les fonctions

- **Série TP 2 – Traitement Bas Niveau**

```python
# Afficher tous les encodages qui existent

import sys
import encodings

# Récuperer le nom des encodages
encodages = encodings.aliases.aliases.values()

# Eliminer les doublants
encodages = set(encodages)

# Trier les éléments de la liste
encodages = sorted(encodages)

# join permet de mettre un séparateur entre les éléments pour les afficher
print("\n".join(encodages))
```

Série TP 2 – Traitement Bas Niveau

```python
# Recuperer le codepoint en décimal d'un caractère : ord()

ord('A')
```

```
65
```

```python
print(ord('é'))
print(ord('€'))
```

```
233
8364
```

```python
# Recuperer le caractère par son codepoint en décimal : chr()

chr(65)
```

```
'A'
```

```python
print(chr(97))
print(chr(237))
print(chr(8364))
```

```
a
í
€
```

- **Série TP 2 – Traitement Bas Niveau**

```python
# Afficher tous les caractères (le Charset) de l'encodage ASCII 7 bits - 128 caractères
for i in range(128):
    print(chr(i))
```

```python
# Afficher les caractères arabes
for i in range(1536, 1791):
    print(chr(i))
```

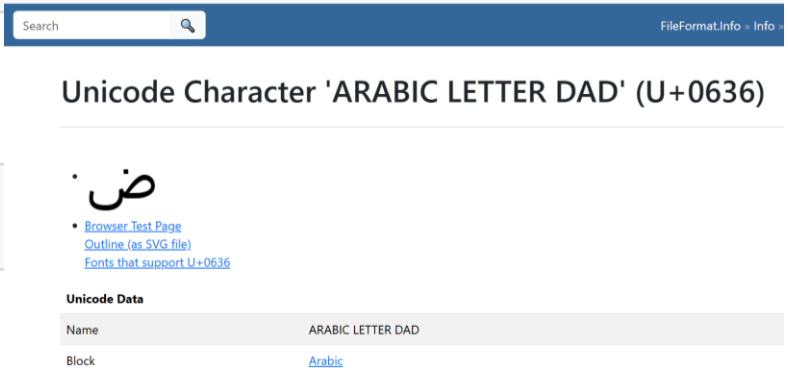
...

```python
# Afficher les caractères tifinagh
for i in range(11568 , 11631):
    print(chr(i))
```

- **Série TP 2 – Traitement Bas Niveau**

```python
# Afficher un caractère par son code point Unicode : \u + codepoint en 4 chiffres
print('\u0636')
```

ض




```python
print('\u1F68')
```

Ὠ

```python
# Afficher des formules chimiques avec codepoint unicode : \u

print("The chemical formula of water is H\u20820.Water dissociates into H\u207A and OH\u207B")
```

The chemical formula of water is H_2O.Water dissociates into H^+ and OH^-

- **Série TP 2 – Traitement Bas Niveau**

```python
# En Python3, il existe deux types de chaine de caractère: str (character string) et bytes (byte string

ch = 'Bonjour'
print(ch, type(ch))

bt = b'Bonsoir'
print(bt, type(bt))
```

```
Bonjour <class 'str'>
b'Bonsoir' <class 'bytes'>
```

- Use **str** for textual data, as it supports Unicode.
- Use **bytes** for binary data, such as file I/O, networking, or encryption.
- Convert between them using .encode() and .decode().

```python
while len(sensed_values) < 100 :
    sensor = arduino_serial.readline().decode('ascii').strip()
```

Série TP 2 – Traitement Bas Niveau

```
# En Python3, il existe deux types de chaine de caractère: str (character string) et bytes (byte string

ch = 'Bonjour'
print(ch, type(ch))

bt = b'Bonsoir'
print(bt, type(bt))
```

```
Bonjour <class 'str'>
b'Bonsoir' <class 'bytes'>
```

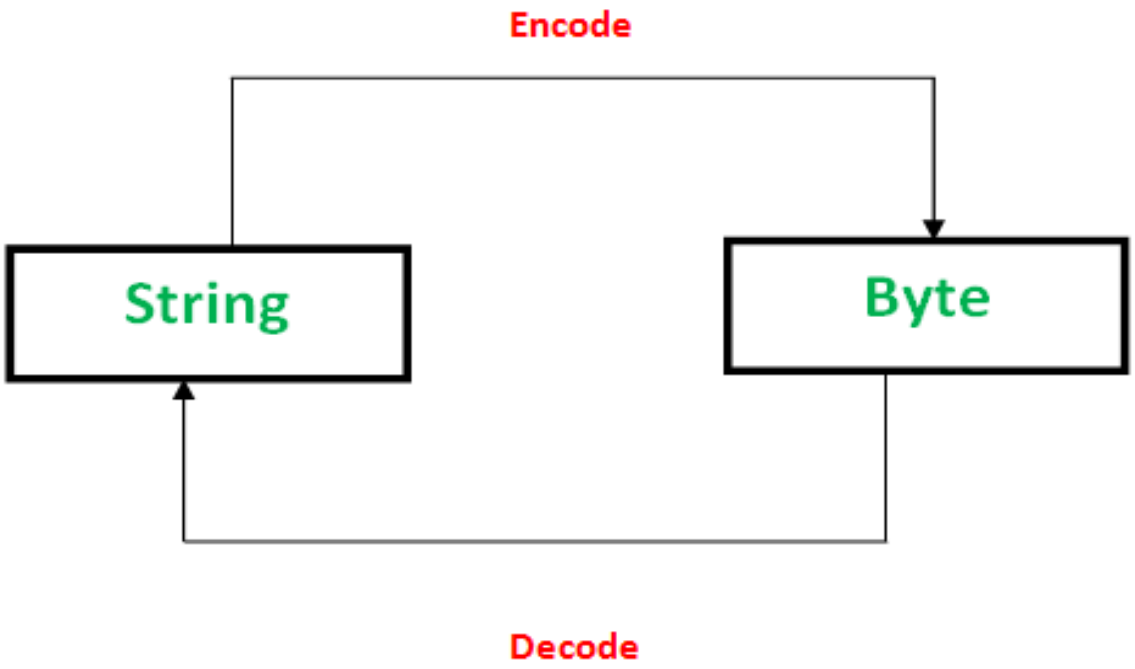
```
# A noter que : bytes ne peut contenir que des caractères ASCII (1 octet)

bt_2 = b'مرحبا'
```

```
Input In [11]
  bt_2 = b'مرحبا'
         ^
SyntaxError: bytes can only contain ASCII literal characters.
```

- **Série TP 2 – Traitement Bas Niveau**

Encoding schema:

1. ASCII
2. UTF-8
3. UTF-16
4. UTF-32



- **Série TP 2 – Traitement Bas Niveau**

```python
# Encoder (convertir) une chaine de caractères str vers bytes en utilisant
my_str = 'I am a string'
my_bt = my_str.encode('utf-8')
print(my_bt)
```

```
b'I am a string'
```

```python
# Decoder (reconvertir) le bytes vers str : decode()
my_bt.decode('utf-8')
```

```
'I am a string'
```

- **Série TP 2 – Traitement Bas Niveau**

```
# Encoder (convertir) une chaine de caractères str vers bytes en
my_str = 'I am a string'
my_bt = my_str.encode('ascii')
```

```
# Rappel: les chaînes bytes ne sont pas lisibles par l'humain, P
print(my_bt)
```

```
b'I am a string'
```

```
# Decoder (reconvertir) le bytes vers str : decode()
my_bt.decode('ascii')
```

```
'I am a string'
```

- **Série TP 2 – Traitement Bas Niveau**

```python
# Attention aux erreurs - caractères non pris en compte par l'encodage : UnicodeDecodeError
utf_str = "Euro symbol is a non ASCII character: €"
ascii_bt = utf_str.encode('ascii')
print(ascii_bt)
```

```
---------------------------------------------------------------------------
UnicodeEncodeError                        Traceback (most recent call last)
Input In [17], in <cell line: 3>()
      1 # Attention aux erreurs - caractères non pris en compte par l'encodage : UnicodeDecodeError
      2 utf_str = "Euro symbol is a non ASCII character: €"
----> 3 ascii_bt = utf_str.encode('ascii')
      4 print(ascii_bt)

UnicodeEncodeError: 'ascii' codec can't encode character '\u20ac' in position 38: ordinal not in rang
e(128)
```

- **Série TP 2 – Traitement Bas Niveau**

\\u20ac \\N{EURO SIGN}

```
# Solution : le paramètre errors ('replace', 'backslashreplace', 'ignore', 'namereplace')
utf_str = "Euro symbol is a non ASCII character: €"
ascii_bt = utf_str.encode('ascii', errors="ignore")
print(ascii_bt)
```

```
b'Euro symbol is a non ASCII character: '
```

```
# Solution : le paramètre errors ('replace', 'backslashreplace', 'ignore', 'namereplace')
utf_str = "Euro symbol is a non ASCII character: €"
ascii_bt = utf_str.encode('ascii', errors="replace")
print(ascii_bt)
```

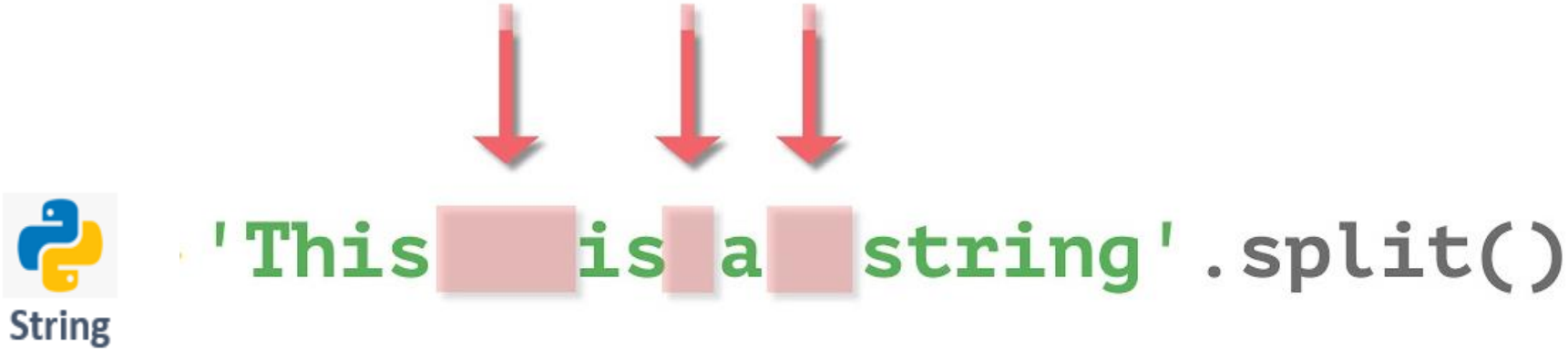
```
b'Euro symbol is a non ASCII character: ?'
```

```
# Reconvertir/décoder vers le str
ascii_bt.decode("ascii") # perte du caractère € dans la chaine originale, remplacé par le ?, car non re
```

```
'Euro symbol is a non ASCII character: ?'
```

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization



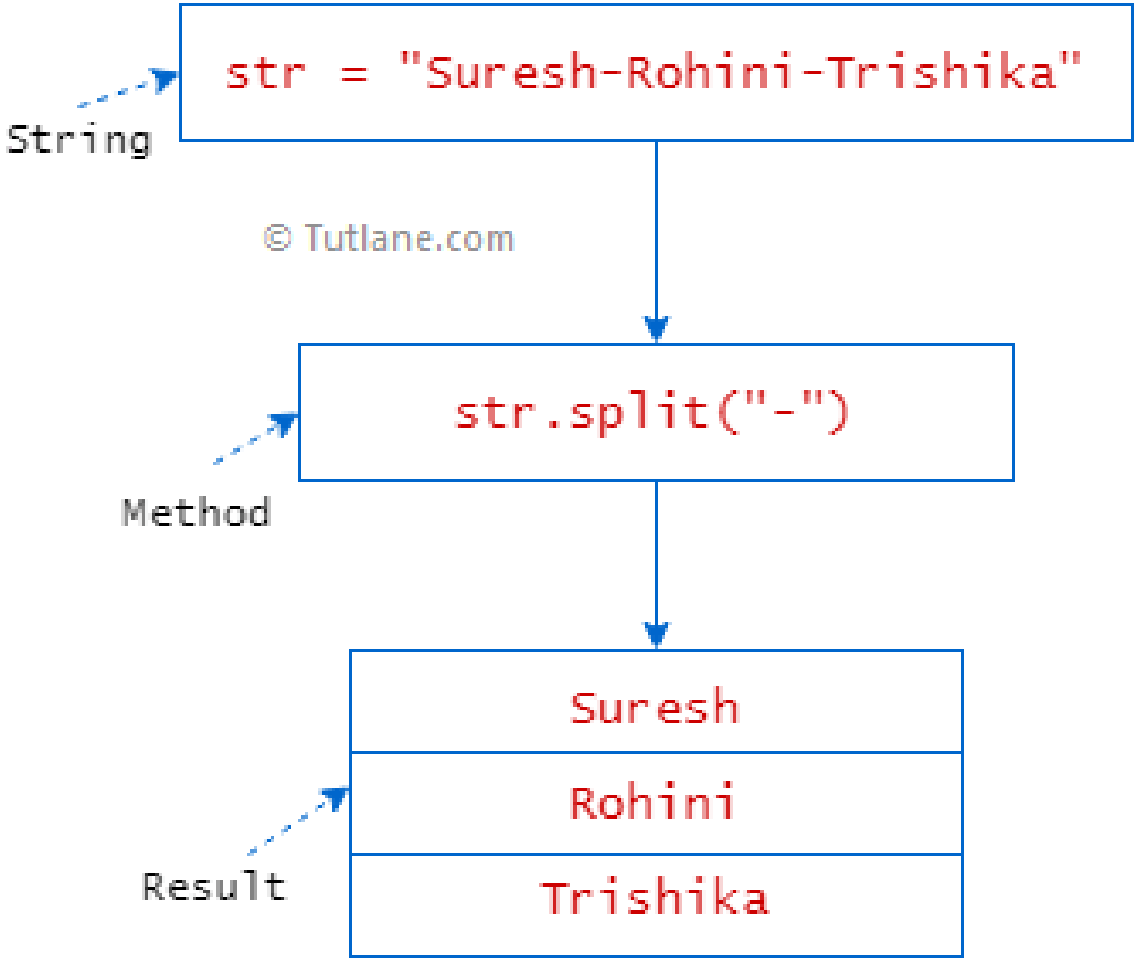
split(" ")

This	is	a	string

['This', 'is', 'a', 'string']

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization



- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

Syntax

```
string.split(separator, maxsplit)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator
maxsplit	Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

```
example = "apple#banana#cherry#orange"
x = example.split("#")
print(x)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
example.split("#", maxsplit=1)
```

```
['apple', 'banana#cherry#orange']
```

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

```
text = "I'm Very Hungry. I want to eat something, maybe an apple."
```

```
# Sentence tokenization : split(separator, maxsplit)
text.split(".")
```

```
["I'm Very Hungry", ' I want to eat something, maybe an apple', '']
```

```
# Word tokenization - White space : split(separator, maxsplit)
text.split(" ")
```

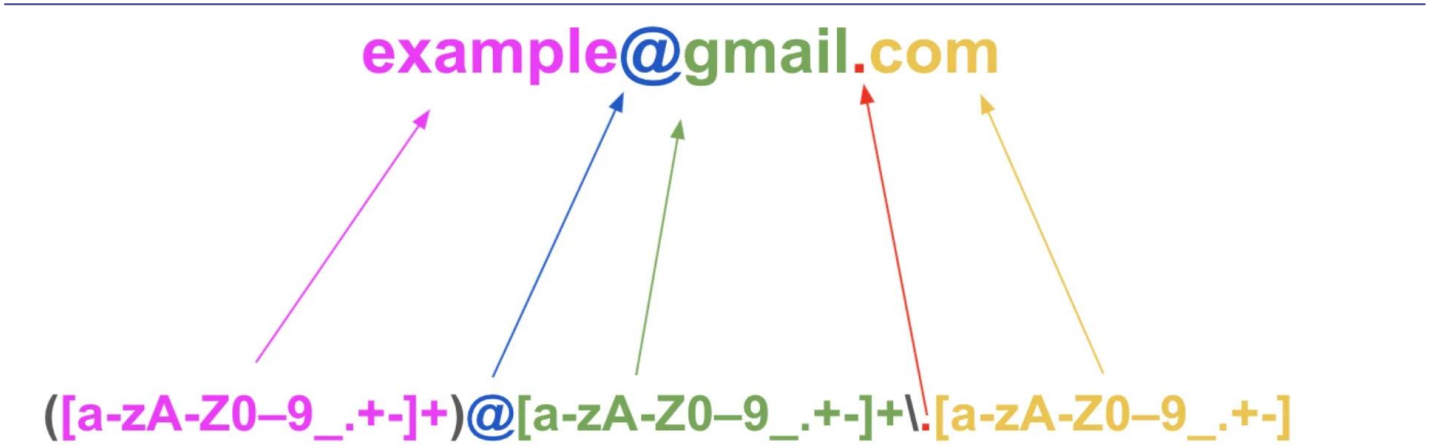
```
["I'm",
 'Very',
 'Hungry.',
 'I',
 'want',
 'to',
 'eat',
 'something,',
 'maybe',
```

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

Regex

[^]*?@[^]*?\.[^]*



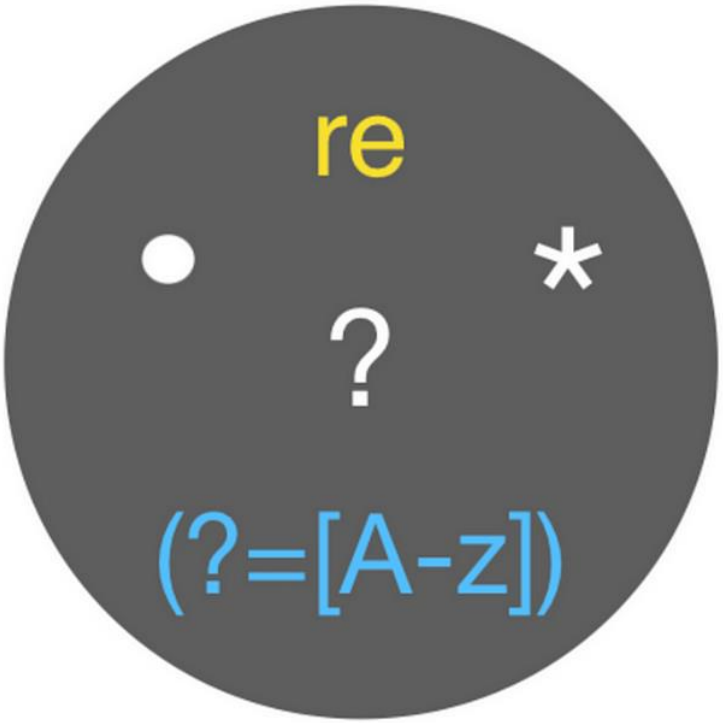

```
/^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$/g
```

- **Série TP 2 – Traitement Bas Niveau**

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

re module ➔ import re



- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

RegEx Functions

The `re` module offers a set of functions that allows us to search a string for a match:

Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

- . veut dire n'importe quel caractère
- ^ veut dire commence par
- $ veut dire se termine par
- * Indicateur de répétition de car - correspond à zéro ou plusieurs instances de ce caractère
- + Indicateur de répétitionde car - correspond à un ou plusieurs instances de ce caractère
- ? Indicateur de répétition de car - correspond à zéro ou une instance de ce caractère
- *? correspond zéro fois ou plus, mais le moins de fois possible.
- [] un ensemble de caractères
- {} exactement le nombe spécifié de caractères
- [a-z] n'importe quelle lettre minuscule
- [0-9] N'importe quel chiffre

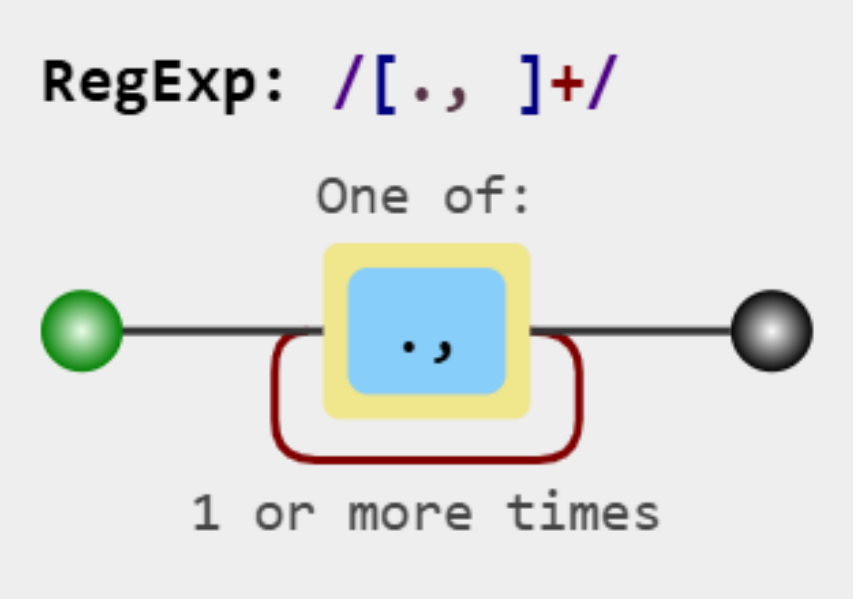
https://www.w3schools.com/python/python_regex.asp

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

```
# Tokenize par des exp-reg simples
re.split("[., ]+", text)
```

```
["I'm",
 'Very',
 'Hungry',
 'I',
 'want',
 'to',
 'eat',
 'something',
 'maybe',
 'an',
 'apple',
 '']
```



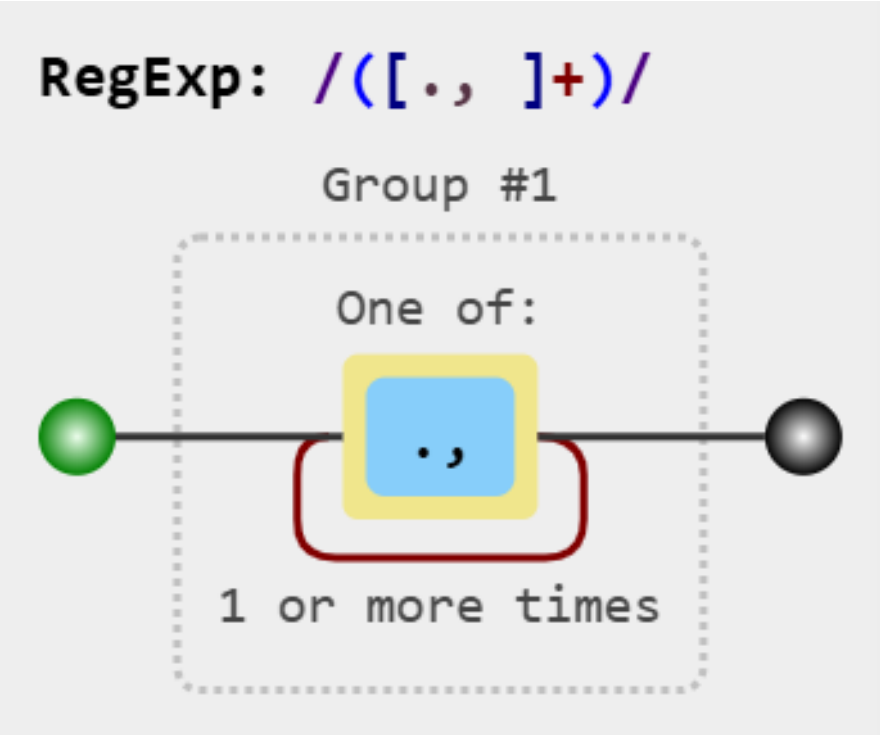
https://jex.im/regulex

- **Série TP 2 – Traitement Bas Niveau**

2 - Segementation - Tokenization

```
# Tokenize par des exp-reg simples, en gardant la ponctuation
re.split("([., ]+)", text)
```

```
["I'm",
 ' ',
 'Very',
 ' ',
 'Hungry',
 '. ',
 'I',
 ' ',
 'want',
 ' ',
 'to',
 ' ',
 'eat',
 ' ',
 'something',
 ', ',
 'maybe',
 ' ',
 'an',
```



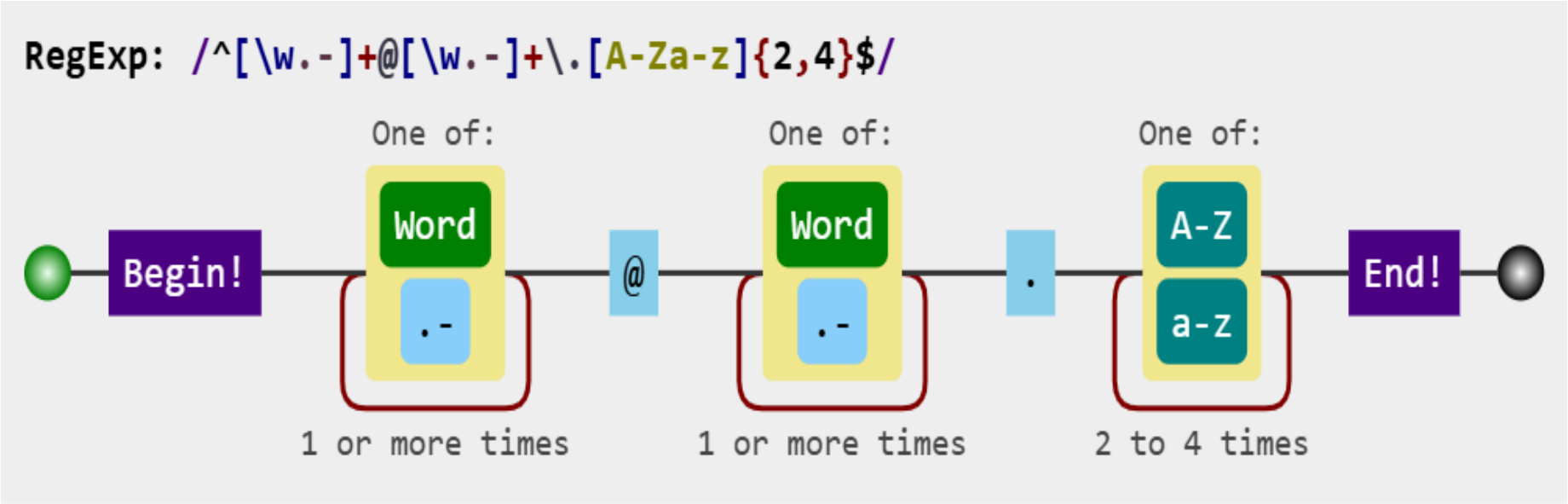
- **Série TP 2 – Traitement Bas Niveau**

https://regexr.com/3e480

https://jex.im/regulex

Expression ^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}$

```
/^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}$/g
```



- **Série TP 2 – Traitement Bas Niveau**

https://regexr.com/3e480

https://jex.im/regulex

Expression ^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}$

/^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}$/g

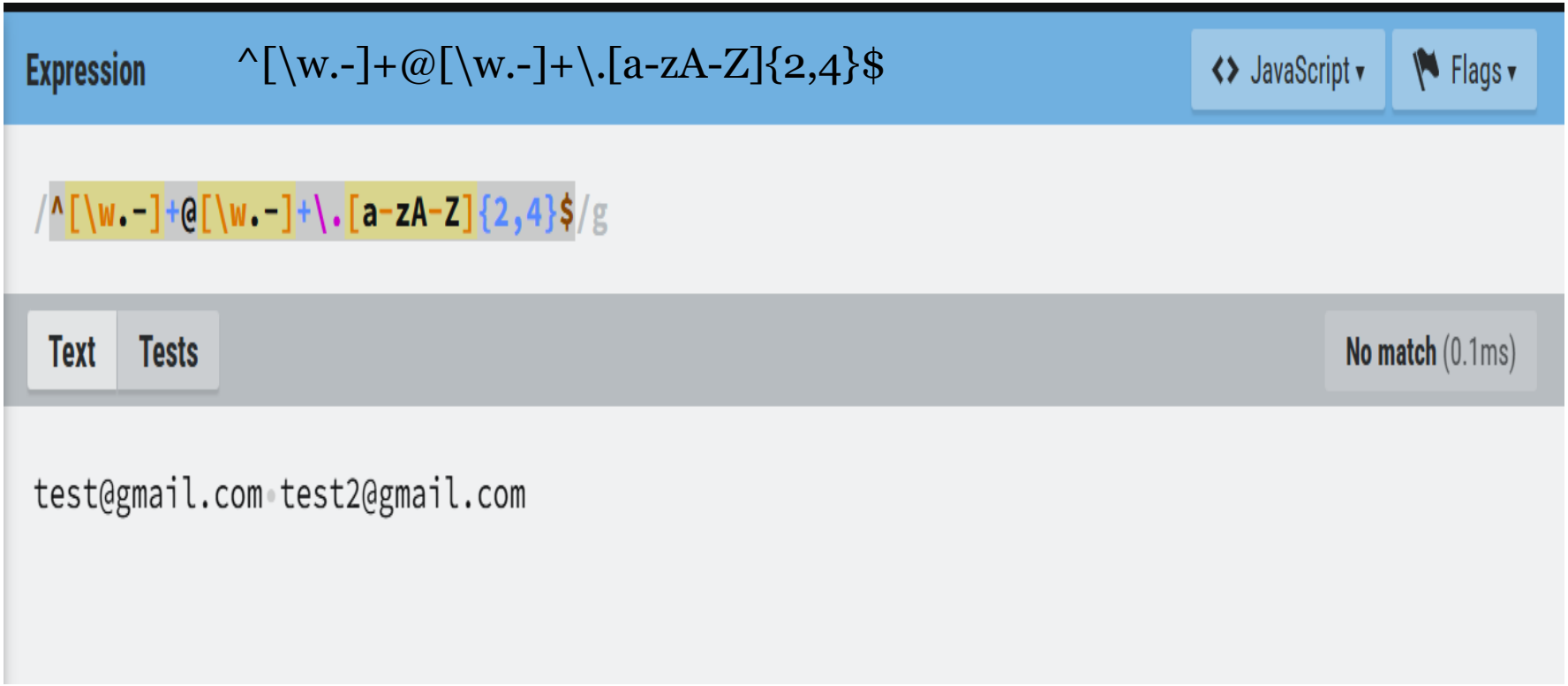
Text Tests

test@gmail.com

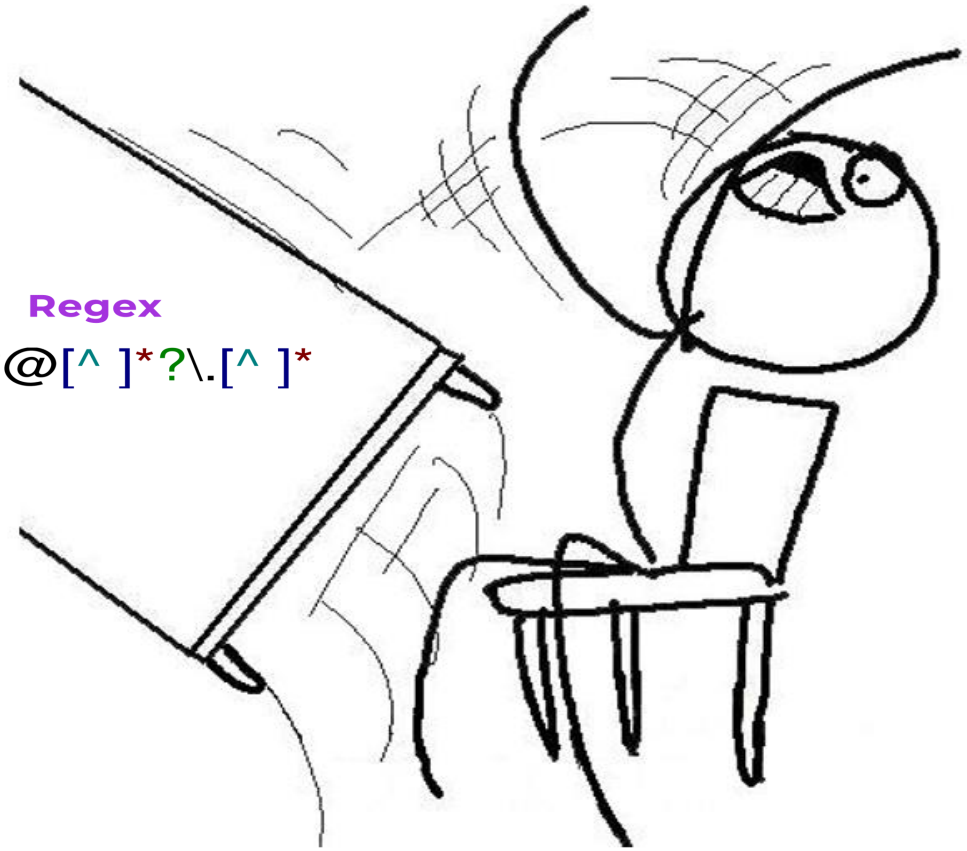
- **Série TP 2 – Traitement Bas Niveau**

https://regexr.com/3e480

https://jex.im/regulex



- **Série TP 2 – Traitement Bas Niveau**



- **Série TP 2 – Traitement Bas Niveau**

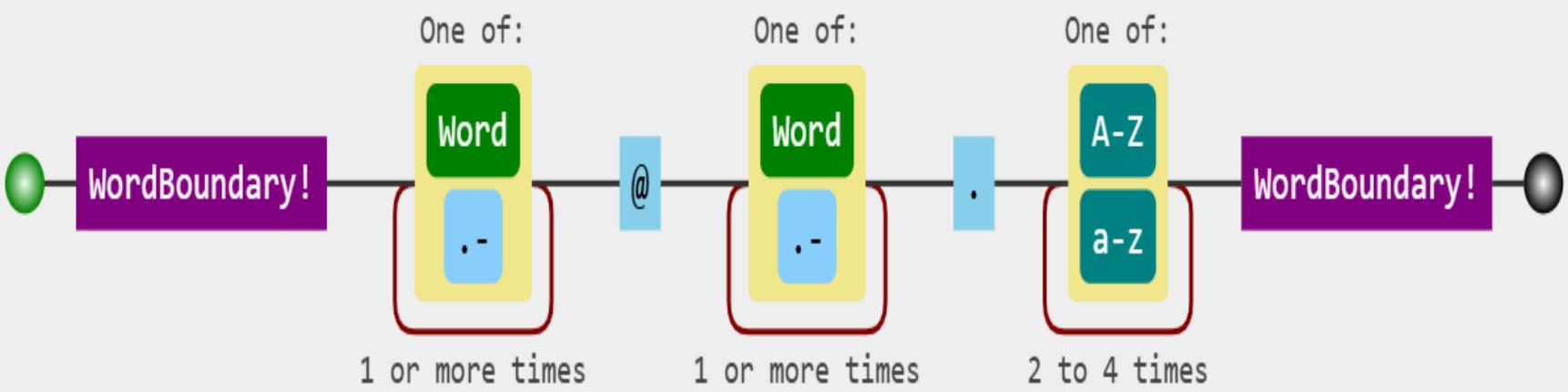
https://regexr.com/3e480

https://jex.im/regulex

Expression \b[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}\b

/\b[\w.-]+@[\w.-]+\.[a-zA-Z]{2,4}\b/g

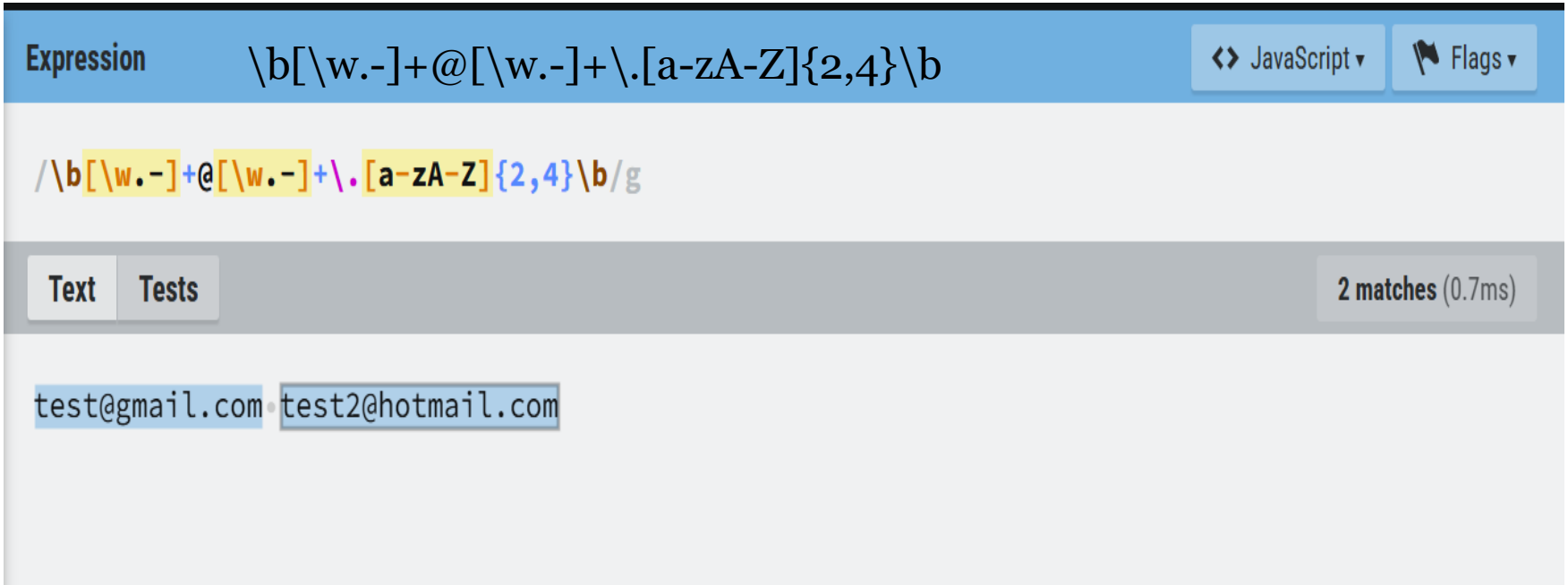
RegExp: /\b[\w.-]+@[\w.-]+\.[A-Za-z]{2,4}\b/



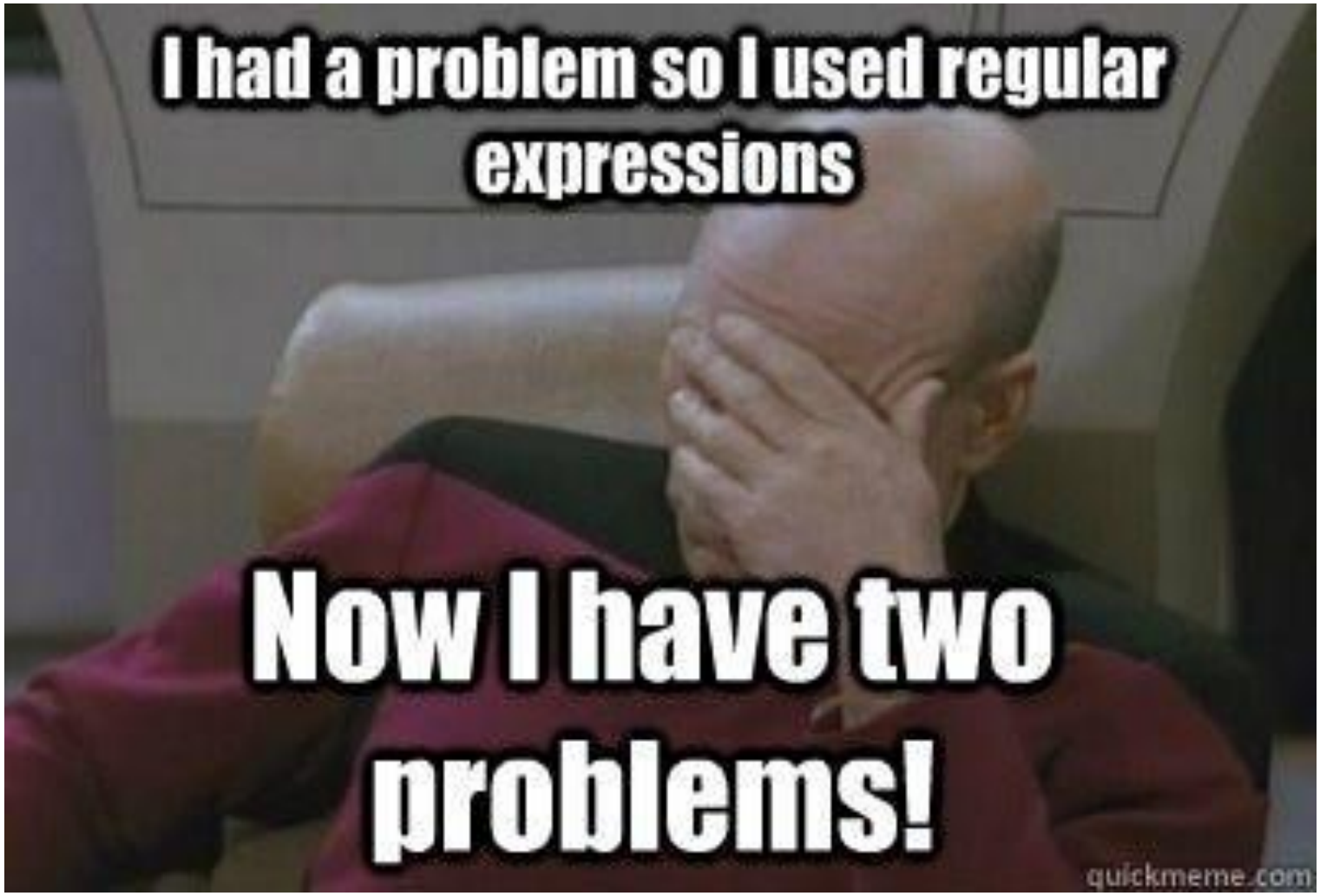
- **Série TP 2 – Traitement Bas Niveau**

https://regexr.com/3e480

https://jex.im/regulex



- **Série TP 2 – Traitement Bas Niveau**



- **Série TP 2 – Traitement Bas Niveau**

text = "I'm Very Hungry. I want to eat something, maybe an apple."

```
# Word tokenization - regex : split() from re module
import re
```

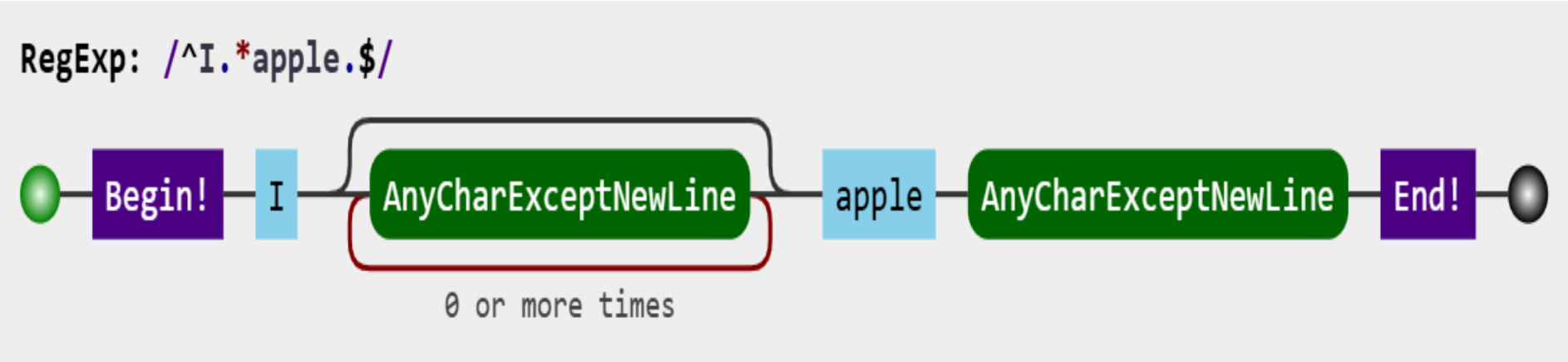
```
# Vérifie si text commence par I et se termine par apple.
bl = re.search("^I.*apple.$", text)
if bl:
  print("YES! We have a match!")
else:
  print("No match")
```

```
YES! We have a match!
```

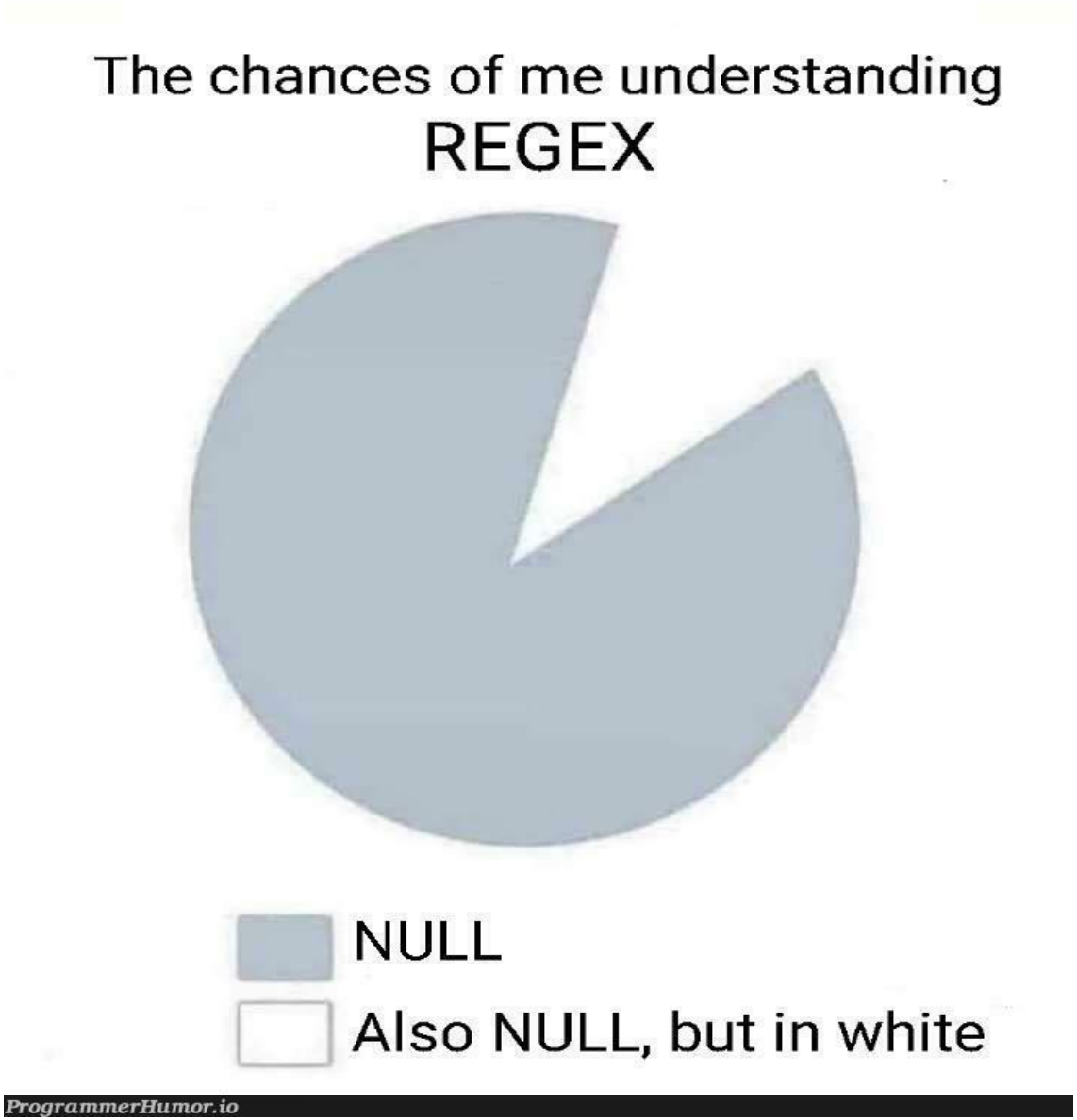
- **Série TP 2 – Traitement Bas Niveau**

https://regexr.com/3e480

https://jex.im/regulex



- **Série TP 2 – Traitement Bas Niveau**



Références

Livre - Speech and Language Processing, de Dan Jurafsky.

Cours - *François Yvon* – Une petite introduction au Traitement Automatique des Langues Naturelles,
https://perso.limsi.fr/anne/coursM2R/intro.pdf

Codage des caractères : https://www.fil.univ-lille1.fr/~wegrzyno/portail/Info/Doc/HTML/seq7_codage_caracteres.html

Text Processing with Unicode - http://nltk.sourceforge.net/doc/en/app-unicode.html

Data Cleaning Challenge: Character Encodings - https://www.kaggle.com/rtatman/data-cleaning-challenge-character-encodings

Tokenization for Natural Language Processing - https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4?gi=6b15f97fe07d

Cours - ARIES Abdelkrime - Le traitement automatique du langage naturel. https://github.com/projeduc/ESI_2CS_TALN