



# Python Intermedio

## Día 3

Curso Proteco 2020-2

Instructores:

- Alicia Carballido García
- Samuel Arturo Garrido Sánchez
- Mario Álvarez Salmerón



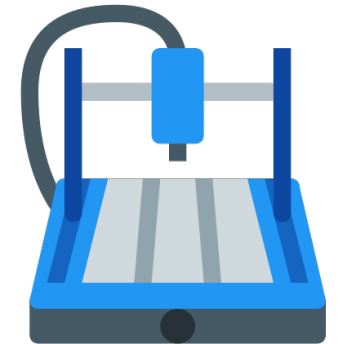
# Tarea:

- Las siguientes preguntas puede ser su próxima tarea del curso, es a libre elección :
  - ¿Cuántos nombres propios de la Biblia inician con vocales y tienen al menos 3 vocales.
  - ¿Cuántas preguntas hay en todos los libros de Harry Potter? (Diálogos, retórica, etc. que tengan signos ¿?)
  - ¿Cuántas palabras hay en el libro del señor de los anillos que tengan al menos 4 vocales?



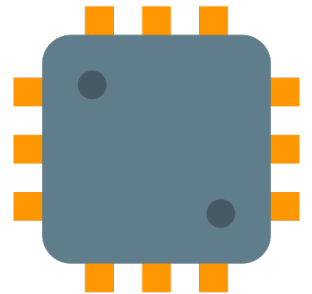
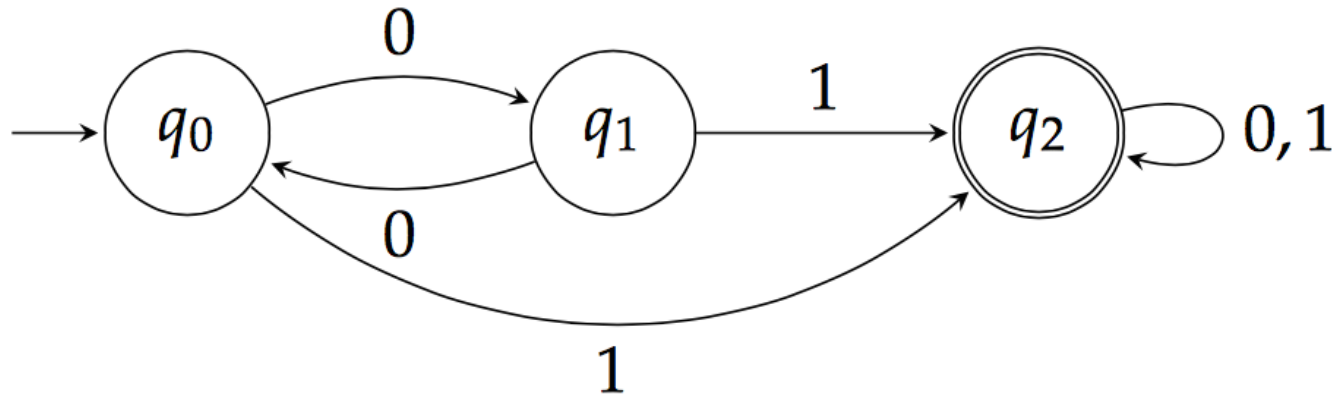
# ¿Y ahora qué? ¿A llorar? 🥲

- Pues no. Las matemáticas y la computación nos han dado herramientas muy poderosas para tratar con estos problemas
- Existe algo llamado **Lenguajes Formales y Autómatas**(❤️, hermosa asignatura), dentro de este se encuentran los lenguajes regulares, la definición de esto es:
  - Los lenguajes regulares son los más sencillos, es decir, los que se pueden generar a partir de los lenguajes básicos, con la aplicación de las operaciones de unión, concatenación y \* de Kleene un número finito de veces. Se puede expresar de muchas maneras:
    - Un autómata finito
    - Un autómata de pila
    - Una máquina de Turing de solo lectura
    - Una gramática regular
    - Una expression regular



# Como dato curioso: ¿Qué rashos es un autómatata?

- Pues básicamente esto, **NO** le tomen mucha importancia, solo es ilustrativo.



- Sin embargo esto se puede convertir en una computadora, Alan Turing y la película de su vida (**El Código Enigma**) les podrán decir por qué algo parecido a esto lo convirtió en el padre de la computación moderna.

---

# Volviendo al tema

- Resulta que nuestro idioma y muchos idiomas o incluso los lenguajes de programación los podemos considerar lenguajes formales, matemáticamente hablando, que poseen patrones que podemos identificar.
- Las expresiones regulares, a menudo llamada también regex, son unas secuencias de caracteres que forma un patrón de búsqueda, las cuales son formalizadas por medio de una sintaxis específica.



# Ejemplo: MAG



- Tengo estas oraciones (algunas de una sola palabra):
  - Hacernos diestros en el arte de la magia
  - Ella es mágica
  - Qué música tan magical
  - Maggie Simpson es adoptada
  - Es necesario imaginar un mundo mejor
- Si queremos buscar cuántas palabras tienen "mag", solo es necesario escribir en el buscador del PDF: mag
- Pero además de mag, que tengan al menos 4 letras adelante: mag[a-z][a-z][a-z][a-z] -> imaginar

# Tabla de patrones 🤯

Metacaracter	Descripción
^	inicio de línea.
\$	fin de línea.
\A	inicio de texto.
\Z	fin de texto.
.	cualquier caracter en la línea.
\b	encuentra límite de palabra.
\B	encuentra distinto a límite de palabra.

Metacaracter	Descripción
\w	un caracter alfanumérico (incluye "_").
\W	un caracter no alfanumérico.
\d	un caracter numérico.
\D	un caracter no numérico.
\s	cualquier espacio (lo mismo que [\t\n\r\f]).
\S	un no espacio.

Metacaracter	Descripción
*	cero o más, similar a {0,}.
+	una o más, similar a {1,}.
?	cero o una, similar a {0,1}.
{n}	exactamente n veces.
{n,}	por lo menos n veces.
{n,m}	por lo menos n pero no más de m veces.
*?	cero o más, similar a {0,}?
+?	una o más, similar a {1,}?
??	cero o una, similar a {0,1}?
{n}?	exactamente n veces.
{n,}?	por lo menos n veces.
{n,m}?	por lo menos n pero no más de m veces.

- Dada las siguientes tablas, una expression regular suele ser algo como lo siguiente:

"(^[a-zA-Z0-9\_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\$.)"

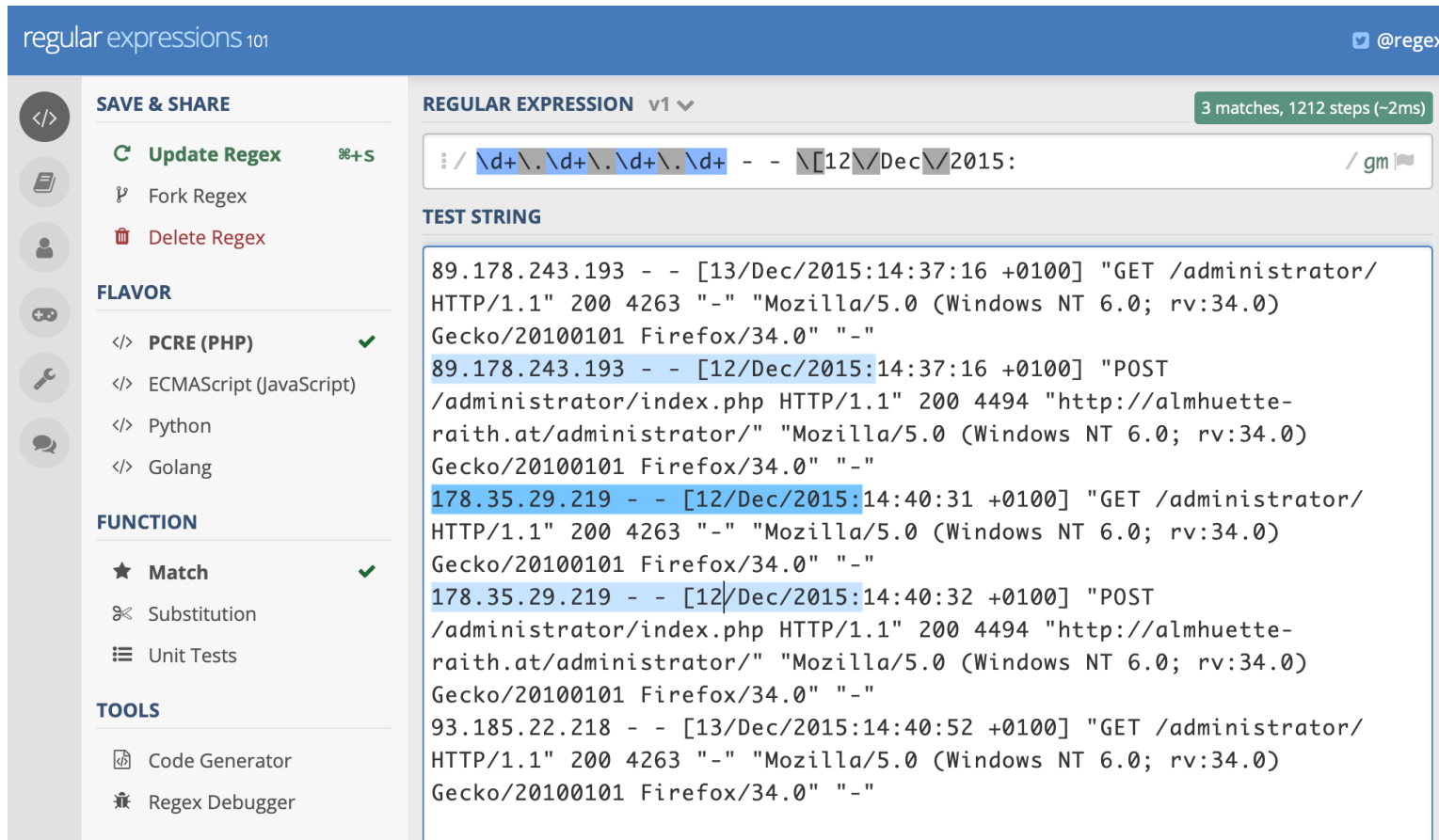
- Y aunque usted no lo crea, eso sirve para decir **si un correo electrónico es válido o no.** 🙄

Ejemplo:

samuelgarrido.proteco@gmail.com ✓

samuelgarrido\$proteco#@\$gmail ✗

# Para que practiquen (Sin usar Python)



The screenshot shows the regex101.com interface. The top bar has the text 'regular expressions 101' and a Twitter icon with '@regex101'. The left sidebar contains navigation options: 'SAVE & SHARE' (Update Regex, Fork Regex, Delete Regex), 'FLAVOR' (PCRE (PHP), ECMAScript (JavaScript), Python, Golang), 'FUNCTION' (Match, Substitution, Unit Tests), and 'TOOLS' (Code Generator, Regex Debugger). The main area displays a 'REGULAR EXPRESSION' v1 with the pattern `/\d+\.\d+\.\d+\.\d+ - - \[12/Dec/2015:` and a status of '3 matches, 1212 steps (~2ms)'. Below the expression is a 'TEST STRING' containing several log entries. The matches are highlighted in blue: the first match is `89.178.243.193 - - [12/Dec/2015:14:37:16 +0100]`, the second is `89.178.243.193 - - [12/Dec/2015:14:37:16 +0100]`, and the third is `178.35.29.219 - - [12/Dec/2015:14:40:31 +0100]`.

- Si desean practicar qué tan buenos son escribiendo expresiones regulares vayan al sitio:
- <https://regex101.com>
- Y comiencen a practicar



# En Python



- Para usar expresiones regulares en Python se pone la siguiente estructura:
- Importamos re
- Compilamos nuestra ExpReg
- Buscamos las coincidencias
- Podemos decir si hay coincidencias con if

```
In [2]: import re

p = re.compile(r'\d+')

x = p.findall('11 es un número y 12 también')
print(x)

if x:
    print("Si, hay al menos una coincidencia!")
else:
    print("No hay coincidencias")

['11', '12']
Si, hay al menos una coincidencia!
```

# Expresiones regulares en la vida real



¡LO SENTIMOS!

Su contraseña debe contener al menos  
una mayúscula,  
un número,  
un símbolo,  
un jeroglífico,  
un sudoku  
y la sangre de una virgen.

- ¿Alguna vez se ha topado con alguna de estas bellezas?
- Pues la manera en que verifican esto o que tu email sea válido o que tu número telefónico o tarjeta de crédito o muchas muchas otras cosas sean válidas es justo:

## Expresiones Regulares

# Algunas expresiones regulares útiles

- `^((25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)\.){3}(25[0-5] | 2[0-4][0-9] | [01]?[0-9][0-9]?)$`
  - Para saber si una dirección **IP** es válida

- `^\s*(?:\+?(\d{1,3}))?[-. (]*(\d{3})[-. ]*(\d{4})(?: *x(\d+))?\s*$`
  - Para encontrar todos los números telefónicos como los siguientes:

- `^\d{1,2}\d{1,2}\d{4}$`
  - Para todas las fechas con formato DD/MM/AAAA todos números.
  - Ojo: esto no contempla a los casos como 55/55/9999 los cuáles no son fechas válidas. Habría que hacer una mejor expresión regular.

18005551234  
1 800 555 1234  
+1 800 555-1234  
+86 800 555 1234  
1-800-555-1234  
1 (800) 555-1234  
(800)555-1234  
(800) 555-1234  
(800)5551234  
800-555-1234  
800.555.1234  
800 555 1234x5678