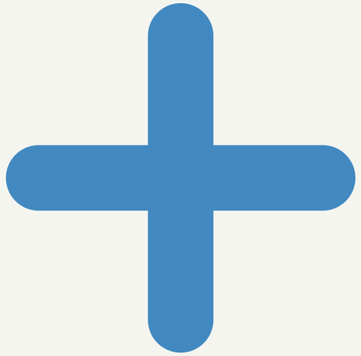
# OpenRefine

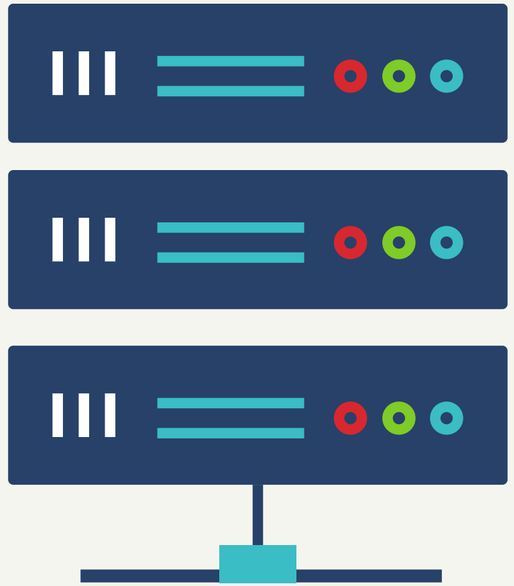
# La aplicación de escritorio que nos ayuda a limpiar nuestros datos



# Expresiones Regulares (RegEx)

Secuencias de caracteres compuestas por un patrón empleadas para buscar o buscar y reemplazar palabras en un texto

OpenRefine funciona mediante un servidor en local (localhost) escrito en Java y ejecutando por la terminal de nuestro SO



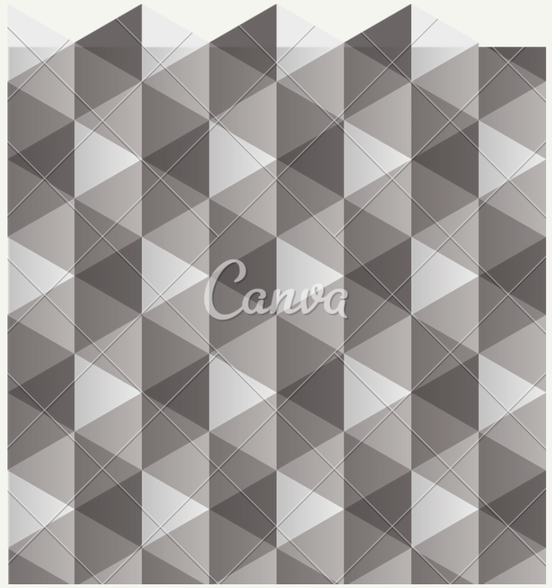

# OpenRefine permite importar y exportar ficheros en diferentes formatos

# Data Types

| | |
|---|---|
| string | "I'm a string" |
| number | 40 |
| boolean | true / false |
| date | YYYY-MM-DDTHH:MM:SSZ |

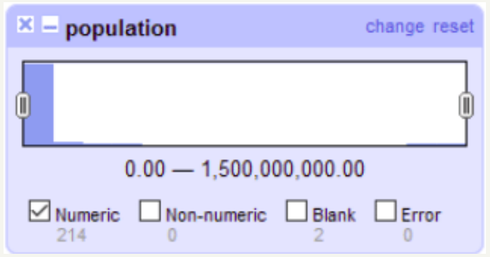
# Facets

Permiten buscar patrones y tendencias en diferentes tipos de datos

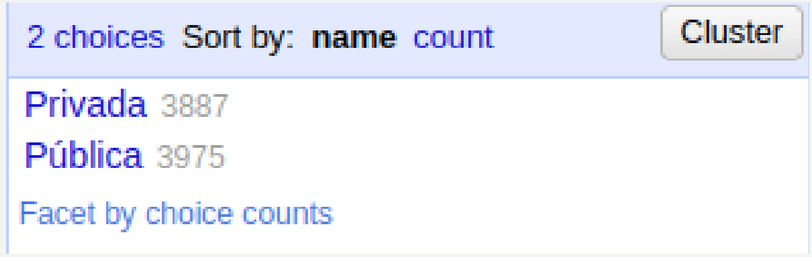


# Facets

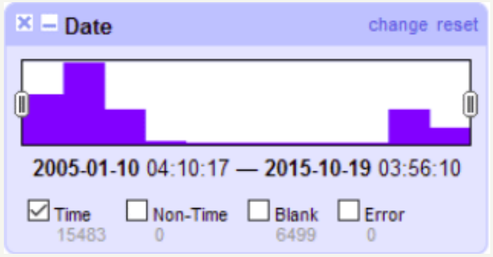
## Numeric




## Text

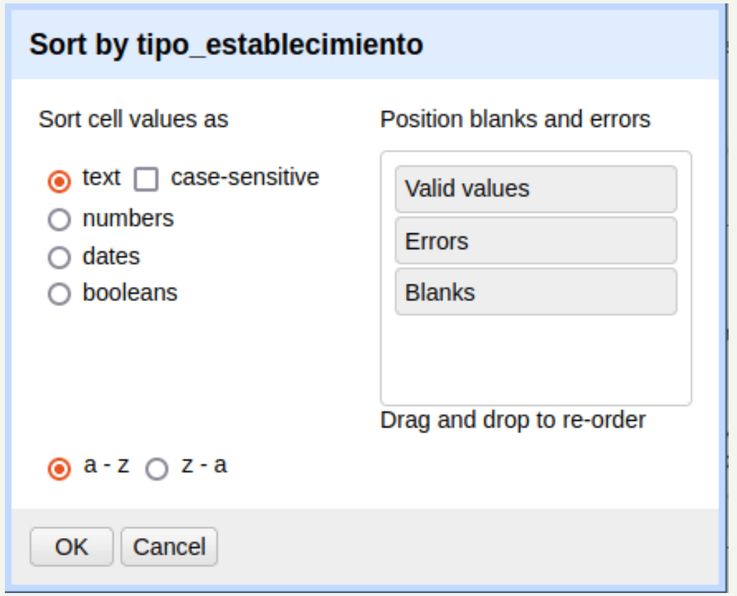



## Date



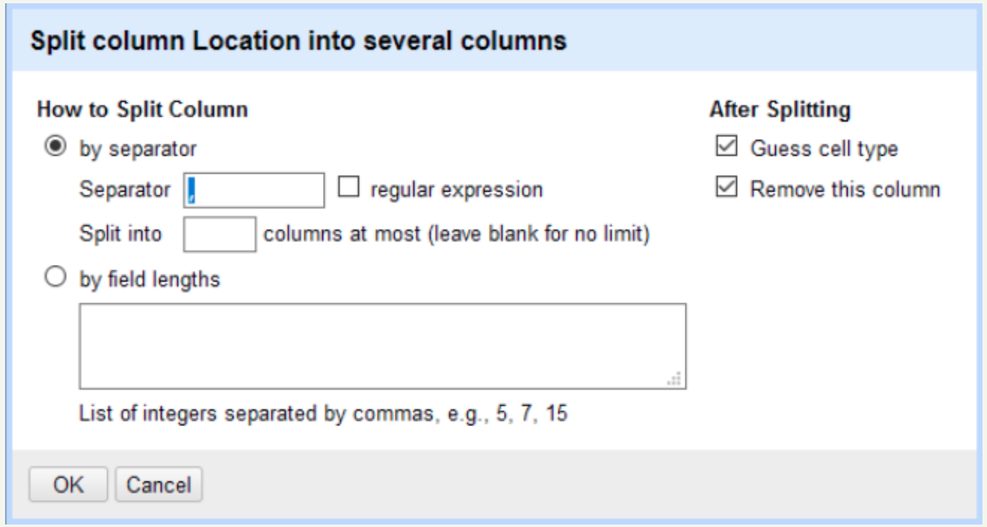
# Sort

Se puede ordenar en base a cualquier tipo de dato ASC/DESC

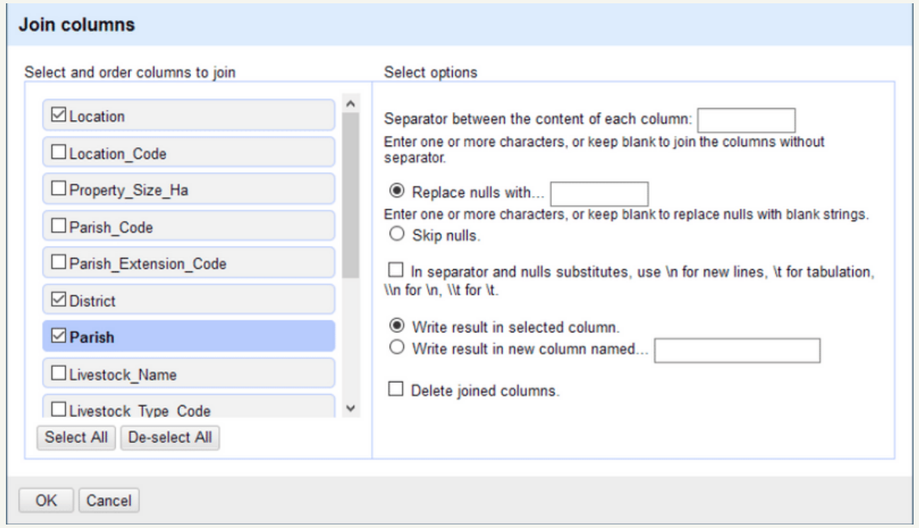


# Transform data

## split




## join



# Variables

| Variable | Meaning |
| --- | --- |
| `value` | The value of the cell in the current column of the current row (can be null) |
| `row` | The current row |
| `row.record` | One or more rows grouped together to form a record |
| `cells` | The cells of the current row, with fields that correspond to the column names (or row.cells) |
| `cell` | The cell in the current column of the current row, containing value and other attributes |
| `cell.recon` | The cell's reconciliation information returned from a reconciliation service or provider |
| `rowIndex` | The index value of the current row (the first row is 0) |
| `columnName` | The name of the current cell's column, as a string |

# GREL

El lenguaje de expresiones que nos permite trabajar con todos los valores de OpenRefine

# Anatomía de una expresión

```
value.expresion(<arguments>, [options])
```

```
if(<statement>, true, false)
```

# forEach

```
forEach(<iterable>, <element>, <condition>)
```

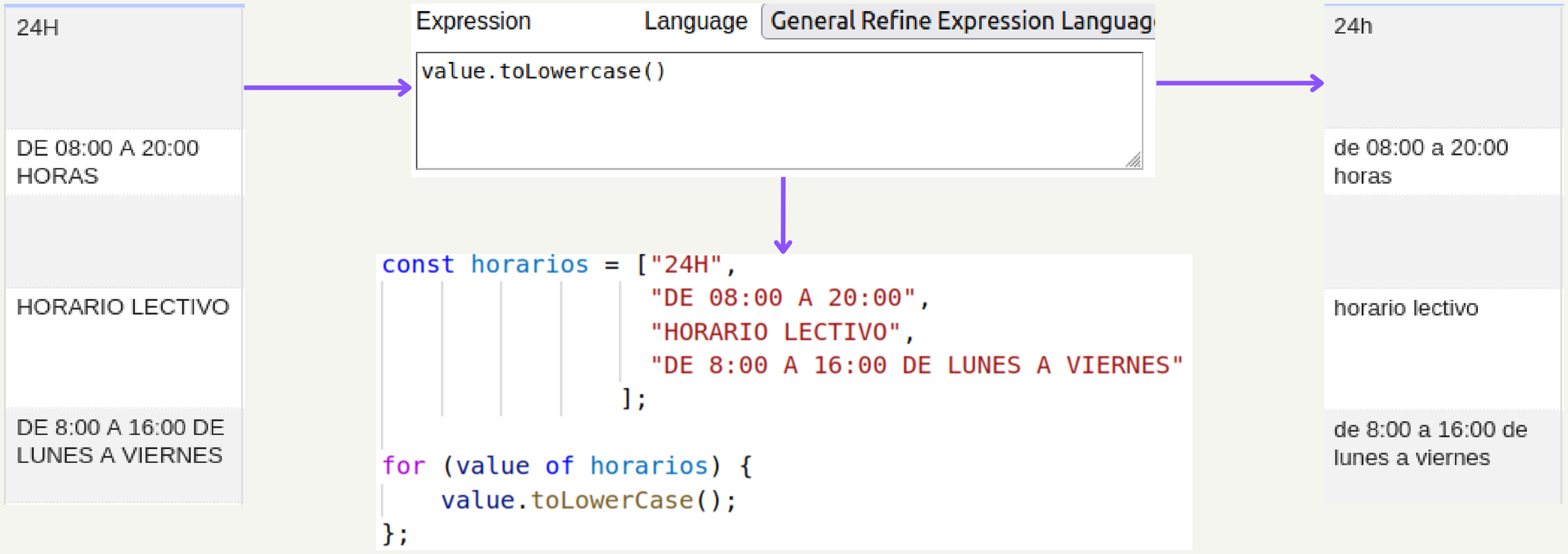
```
forEach([1,2,3,4], v, mod(v,2) == 0)
// [false, true, false, true]
```

# forEach

```
filter(<iterable>, <element>, <condition>)
```

```
filter([1,2,3,4], v, mod(v,2) == 0)
                    // [2, 4]
```

# Lowercase transformation



# Operators

| | |
|---|---|
| Arithmetic | +,-,*,/,pow |
| Comparison | <,>,>=,==,!= |
| Logical | and, or, not |

# Concat

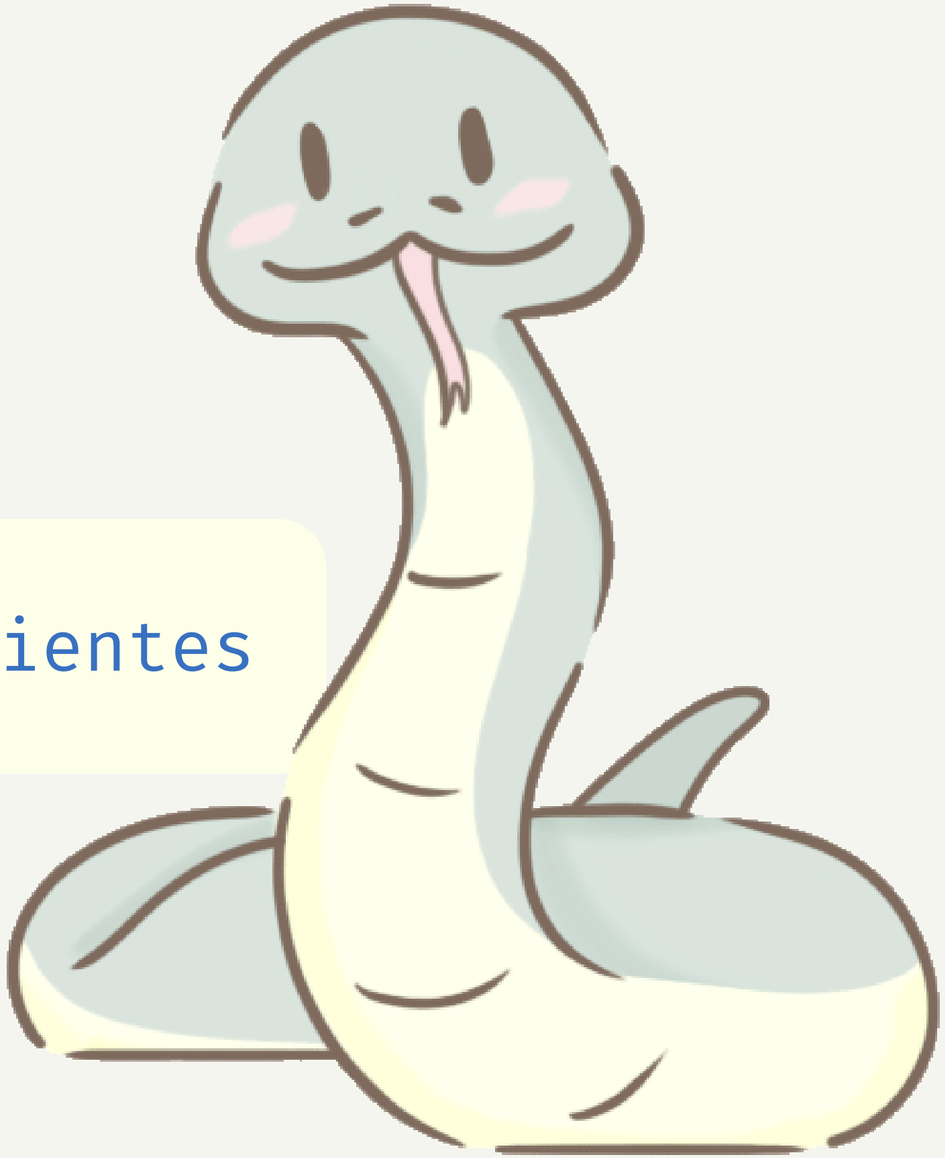
El uso del operador de suma (+) entre strings es denominado concatenación

```
value + "!"
```

```
frase!
```

# RegEx

El idioma de las serpientes



https://docs.rackspace.com/support/how-to/regex-cheat-sheet-basics/

https://download.microsoft.com/download/c/3/e/c3ef6850-d455-478a-afbe-b89e57df8569/Regular_Expressions_Cheat_Sheet.pdf

# Character classes

| | |
|---|---|
| \d | Matches a single digit character |
| \w | Matches a single word character (letters, numbers, and underscore) |
| \s | Matches a single white space character, including tabs and line breaks |
| \D | Matches a single non-digit character |
| \W | Matches a single non-word character (letters, numbers, and underscore) |
| \S | Matches a single non-white space character, including tabs and line breaks |
| . | Matches any single character |

# Quantifier

| | |
|---|---|
| * | Indicates zero or more |
| + | Indicates one ore more |
| ? | Indicates zero or one |
| x{n} | Used to specify the number of times (‘n’) the previous character (‘x’) should appear |
| x{n, } | Used to specify the minimum number of times (‘n’) the previous character (‘x’) should appear |
| x{n, m} | Used to specify the minimum (‘n’) and maximum (’m') number of times the previous character (‘x’) should appear |