Pattern Language

If you are familiar with regex engines (i.e. regular expression engines), this 'pattern language' is going to seem simple. Which is good!

If you are not familiar with this type of tool, it is ideal to look at the examples more than once.

Types:

- **Token** → The pattern matcher has no concept of 'character', instead it breaks down strings (i.e. sequence of characters) into tokens. These can be either words, numbers, or symbols.
    - The following symbols are considered tokens themselves:
        - **? . , + - / \ * ^ & ( ) = > < : [ ] { }**
    - Any other single character or set of characters is considered a token if they are isolated by spaces.
        - **Hello world** → 2 tokens i.e. **hello** and **world**
- **Slot** → Slots are used to declare/define patterns. A slot is essentially a **set of** series of tokens. It is not a set of tokens, it is not a series of tokens. It is a set of series of tokens. This means it contains series and this series contain tokens. When declaring a pattern, a slot is delimited by **<** and **>**, open and close clauses, respectively.
- **Content** → A content is always found in slot declaration, it tells the slot what tokens it has to match with, when provided with a query.
    - **String content** → It tells the slot to match exactly with the tokens contained within itself.
        - **For example <hello world>** is telling the slot to only look for **hello world**.
        - It is possible to define more than one content per slot, for example **<hello world, goodbye world>** is telling the slot to match either with **hello world** or **goodbye world**.
    - **Integer content** → In cases where we know what kind of token we want to match with, but it is not possible to list all the variations, for instance integer numbers - then we use **param:int**. This tells the slot to only match with **any integer**.
        - **For example <param:int>** is telling the slot to match with any integer in the query.
        - It is also possible to mix content types: **<param:int, hello world>** will match with either **any integer** or **hello world**.
    - **Day content** → It tells the slot to only match with tokens that correspond to a day of the week. It is declared as **param:day**.
- **Blank slot** → For cases where matching is irrelevant (i.e. there is a gap in the pattern), we use **<...>**. The behavior of this slot is a bit more complex:
    - **Declaration: <...>** is equivalent to **<#:0>** or **<#:-1>** … or **<#:-infinity>** - Essentially, any of those means **<...>**. However, if the parameter is positive, then the behavior of the blank slot changes (you will see how). For any **<#:1>** or **<#:2>** … or **<#:+infinity>**, the blank slot limits the number of tokens to match.
        - **For example**
            - **<...>** will match with anything

        - **<#:3>** will only match up to 3 tokens (i.e. from 0 to 3 tokens)
    - **Matching behavior:**
        - **Case (a) [The blank slot is the last slot in the pattern]:** The blank slot will match with any tokens up to the specified limit. If the limit is less or equal to zero, then this limit is interpreted as infinite (i.e. no limit)
        - **Case (b) [The blank slot is followed by a non-blank slot (i.e. a solid slot)]:** The blank slot will match any tokens up to the specified limit (if limit <= 0 then limit = +infinity) or until there is a match for the following non-blank slot. This means that the blank slot will not override/interfere with the matching of solid slots.
            - **For example:**
                - **<...>** will match with any query
                - **<#:3>** will only match with the first 3 tokens of any query
                - **<...> <param:int>** will only match with any tokens up until an integer is found

Notes:

- Slots must match consecutively (i.e. one after the other) in the query. So, if a slot does not match with any tokens right after the previous one, the pattern does not match.
- All slots must match for a pattern to match a query.
- Not all contents must match for a slot to match, it is enough with just one. **In fact, only one content per slot can match.**
- Consecutive blank slots are not allowed (i.e. <...> <...>).
- Tokens are all matched as lowercase.
- When a content in a slot is a substring of another content in the same slot and both match in the query, the superstring will be prioritized over the substring.

Example:

<My name is, I am, I am called> <...> <and I am> <param:int> <years old>
Hello, my name is Andres Errera and I am 35 years old.

The colors show what part of the query matches with each slot. As you can see, they are all consecutive.

Once the matching is done, we obtain a sequence of pairs (slot, matched tokens). This is useful because we know the slots we defined, so we can easily extract the segment of the query that matched with a slot simply by the index of the slot. However, this is not in the scope of this small doc.

Tagging System: Now it is possible to add tags to the slots in a pattern.

<My name is, I am, I am called> <@name_slot, ...> <and I am> <@years_old, param:int> <years old>

This way we can then refer to these slots by tag instead of by index.