How do you run?

Compiled language:
- Source Code
  - Compiler
    - Machine Code
    - VM
    - Machine/OS

Interpreted language:
- Source Code
  - Interpreter
  - Machine/OS
## Compile or interpret?

<table>
<thead>
<tr>
<th>Language</th>
<th>Compile Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>COMPiled</td>
</tr>
<tr>
<td>Java</td>
<td>COMPiled (VM)</td>
</tr>
<tr>
<td>C++</td>
<td>COMPiled</td>
</tr>
<tr>
<td>Pascal</td>
<td>COMPiled</td>
</tr>
<tr>
<td>Ruby</td>
<td>INTERPREted</td>
</tr>
<tr>
<td>Elixir/Erlang</td>
<td>COMPiled (VM)</td>
</tr>
<tr>
<td>Python</td>
<td>INTERPREted</td>
</tr>
<tr>
<td>JavaScript</td>
<td>INTERPREted</td>
</tr>
<tr>
<td>Go</td>
<td>COMPiled</td>
</tr>
<tr>
<td>PHP</td>
<td>INTERPREted</td>
</tr>
<tr>
<td>Rust</td>
<td>COMPiled</td>
</tr>
</tbody>
</table>
A basic language

Sequence

\[
\begin{align*}
x &= \text{foo}; \\
y &= \text{:nil}; \\
\{z, \_\} &= \{\text{:bar}, \text{:grk}\}; \\
\{x, \{z, y\}\}
\end{align*}
\]

1. \(x = \text{foo};\)
2. \(y = \text{:nil};\)
3. \(\{z, \_\} = \{\text{:bar}, \text{:grk}\};\)
4. \(\{x, \{z, y\}\}\)

Pattern matching expressions

Single expression
A sequence

Sequence ::= Expression

Sequence ::= Match ; Sequence

Example

x = foo; y = :nil; {z, _} = {:bar, :grk}; {x, {z, y}}
A match

Match ::= Pattern = Expression

Example

x = foo

Pattern = Expression

{z, _} = {:bar, :grk}

Pattern = Expression
An expression

Expression ::= Atom
Expression ::= Variable
Expression ::= { Expression, Expression }
Terms

Expressions

Atom
Variable
\{Expression, Expression\}

Case expressions

Lambda expressions
Let's evaluate

Example

\[
x = \text{foo};
\]
\[
y = \text{nil};
\]
\[
\{z, _\} = \{\text{bar}, \text{grk}\};
\]
\[
\{x, \{z, y\}\}
\]

Sequence (terms) \quad \xrightarrow{\text{evaluation}} \quad \text{Data structure}

\[
\{\text{foo}, \{\text{bar}, \text{nil}\}\}
\]
A pattern

Pattern ::= Atom

Pattern ::= Variable

Pattern ::= Don’t care

Pattern ::= { Pattern, Pattern }
An environment

Contains variables bindings

Initially empty

Immutable: always return new copy

\[ \text{Env } \sigma_0 \]

\[
\begin{align*}
\text{x/foo} \\
\text{y/nil}
\end{align*}
\]
Evaluation: step by step

\[ x = \text{foo}; \quad y = :\text{nil}; \quad \{z, _\} = \{:\text{bar}, :\text{grk}\}; \quad \{x, \{z, y\}\} \]
Evaluation: step by step

\[
x = \text{foo}; \ y = :\text{nil}; \ {z, \ _} = {:\text{bar}, \ :\text{grk}}; \ {x, \ {z, \ y}}
\]
Evaluation: step by step

\[x = \text{foo}; \; y = :\text{nil}; \; \{z, \_\} = \{:\text{bar}, :\text{grk}\}; \; \{x, \{z, y\}\}\]
Evaluation: step by step

\[ x = \text{foo}; \ y = \text{nil}; \ \{z, _\} = \{\text{bar}, \text{grk}\}; \ \{x, \{z, y\}\} \]
Evaluation: step by step

\[
x = \text{foo};
y = \text{:nil};
\{z, \_\} = \{\text{:bar}, \text{:grk}\};
\{x, \{z, y\}\}
\]

Env \(\sigma_3\)

- x/foo
- y/:nil
- z/:bar
Evaluation: step by step

\[
x = \text{foo}; \quad y = \text{nil}; \quad \{z, \_\} = \{\text{bar}, \text{grk}\}; \quad \{x, \{z, y\}\}
\]
Good luck!