```
["alice", "bob"]
|> Enum.reduce fn x, a ->
   a <> x end
```
2

```
false || 11
```
2

```
1 == 1.0
```
2

```
1 === 1.0
```
2

```
1 < :atom
```
2

```
Regexp.replace ~r{
  [aeiou]},
  "catepillar",
  "*"
```
2

```
Regexp.scan ~r{
  [aeiou]},
  "catepillar"
```
2

```
What does make_ref ?
```
2

```
m = %{a: 1, b: 2, c: 3}
m1 = %{m | b: "two",
       c: "foo"}
m1
```
3

```
m = %{a: 1, b: 2, c: 3}
m1 = %{m | b: "two",
       c: "foo"}
m
```
3

```
%{item => :ok} = %{
  1 => :ok,
  2=> :error
}
```
3

```
%{2 => state} = %{
  1 => :ok,
  2 => :error
}
state
```
3

```
%{f: 4, i: 5, s: 6}
|> Dict.values
|> Enum.sum
```
3

```
[one: 1, two: 2, three: 3]
|> Enum.into(HashDict.new)
|> Dict.values
|> Enum.sum
```
3

```
inspect &(&1 + 2)
```
3

```
defmodule User do
  defstruct name: "Joe"
end
joe = %User{}
joe[:name]
```
3

```
(1..10)
|> Enum.map(&(&1 * &1))
|> Enum.filter(&(&1 < 40))
```
3

```
defmodule Boo do
  def hey([a,b | tail]) do
    [b,a | hey(tail)]
  end
  def hey([_]) do
    raise "Hello"
  end
end
Boo.hey([1, 2, 3])
```
3