SUBSCRIPTING

A[:,c]	Selects the **c^{th} column** of the array **A**.
A[r,:]	Selects the **r^{th} row** of the array **A**.
A[:,:]	Selects all elements of **A**; this is equivalent to the entire array.
A[:,j:k]	Selects all rows in columns from **j** to **k-1** (exclusive).
A[:,c]	Equivalent to selecting a specific column
A[l,r,:]	In a three-dimensional array **A**, this selects all elements in the **l^{th}** layer and **r^{th}** row, retrieving all columns along the third dimension.
A[j:k] or A[j:k,] or A[j:k,:]	Selects rows from index **j** to **k-1** (exclusive).
A[:]	Selects all elements of **A**, treating it as a single vector. On assignment, **A[:]** fills **A**, preserving its shape.

PRECEDENCE

Operator	Description
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}`	Binding or parenthesized expression, list display, dictionary display, set display
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`await x`	Await expression
`**`	Exponentiation [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`*`, `@`, `/`, `//`, `%`	Multiplication, matrix multiplication, division, floor division, remainder [6]
`+`, `-`	Addition and subtraction
`<<`, `>>`	Shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`in`, `not in`, `is`, `is not`, `<`, `<=`, `>`, `>=`, `!=`, `==`	Comparisons, including membership tests and identity tests
`not x`	Boolean NOT
`and`	Boolean AND
`or`	Boolean OR
`if` – `else`	Conditional expression
`lambda`	Lambda expression
`:=`	Assignment expression