Macros
The Cairo language has some plugins that allow developers to simplify their code. They are called `inline_macros` and are a way of writing code that generates other code.
`consteval_int!` Macro
In some situations, a developer might need to declare a constant that is the result of a computation of integers. To compute a constant expression and use its result at compile time, it is required to use the `consteval_int!` macro.
Here is an example of `consteval_int!`:
```cairo,noplayground
const a: felt252 = consteval_int!(2 * 2 * 2);
```
This will be interpreted as `const a: felt252 = 8;` by the compiler.
`selector!` Macro
`selector!("function_name")` macro generates the entry point selector for the given function name.
`print!` and `println!` Macros
Please refer to the [Printing](./ch11-08-printing.md) page.
`array!` Macro
Please refer to the [Arrays](./ch03-01-arrays.md) page.
`panic!` Macro
See [Unrecoverable Errors with panic](./ch09-01-unrecoverable-errors-with-panic.md#panic-macro) page.
`assert!` and `assert_xx!` Macros
See [How to Write Tests](./ch10-01-how-to-write-tests.md) page.
`format!` Macro
See [Printing](./ch11-08-printing.md#formatting) page.
`write!` and `writeln!` Macros
See [Printing](./ch11-08-printing.md#printing-custom-data-types) page.
`get_dep_component!`, `get_dep_component_mut` and `component!` Macros
Please refer to the [Composability and Components](./ch16-02-00-composability-and-components.md) chapter.