The word `void` in parentheses indicates that `print_pun` has no arguments. (This time, we're using `void` as a placeholder that means "nothing goes here.")

To call a function with no arguments, we write the function's name, followed by parentheses:

```
print_pun();
```

The parentheses *must* be present, even though there are no arguments.

Here's a tiny program that tests the `print_pun` function:

pun2.c

```
/* Prints a bad pun */

#include <stdio.h>

void print_pun(void)
{
  printf("To C, or not to C: that is the question.\n");
}

int main(void)
{
  print_pun();
  return 0;
}
```

The execution of this program begins with the first statement in `main`, which happens to be a call of `print_pun`. When `print_pun` begins to execute, it in turn calls `printf` to display a string. When `printf` returns, `print_pun` returns to `main`.

Function Definitions

Now that we've seen several examples, let's look at the general form of a *function definition:*

function definition

return-type function-name (*parameters*)
{
  declarations
  statements
}

The return type of a function is the type of value that the function returns. The following rules govern the return type:

- Functions may not return arrays, but there are no other restrictions on the return type.
- Specifying that the return type is `void` indicates that the function doesn't return a value.