```
>
```

In Maple, *all* functions take an arbitrary number of arguments. The formal parameter list in a function definition specifies the *minimum* number of arguments that the function takes. When a function is called, all of its actual parameters are placed in a list called **`args`** that is a local variable to the function (and another local variable, **`nargs`**, holds the number of arguments in the list **`args`**). If the number of actual parameters in a function call is less than the number of formal parameters in the function definition (that is, if the number of actual parameters is less than he specified minimum number of parameters), then the interpreter raises an error. Notice that no error is raised if the number of actual parameters in a function call is greater than the number of formal paramters in the function definition (since *every* function takes an arbitrary number of paramters).

Below are a number of examples that demonstrate the above ideas.

```
> f := proc(x) x end proc;
```

$$f := \textbf{proc}(x)\ x\ \textbf{end proc}$$

```
> f();
Error, (in f) f uses a 1st argument, x, which is missing

> f(1);
```

$$1$$

```
> f(1,2);
```

$$1$$

```
> f(1,2,3);
```

$$1$$

```
>
> g := proc(x, y) x+y end proc;
```

$$g := \textbf{proc}(x, y)\ x + y\ \textbf{end proc}$$

```
> g();
Error, (in g) g uses a 1st argument, x, which is missing

> g(1);
Error, (in g) g uses a 2nd argument, y, which is missing

> g(1,2);
```

$$3$$

```
> g(1,2,3);
```

$$3$$

```
>
> h := proc()
>   local i, sum;
>   sum := 0;
>   for i from 1 to nargs
>     do
>      sum := sum + args[i]
>      od;
>    sum
> end proc;
```

$h := \textbf{proc}()\ \textbf{local}\ i, sum;\ sum := 0;\ \textbf{for}\ i\ \textbf{to}\ \mathrm{nargs}\ \textbf{do}\ sum := sum + \mathrm{args}[i]\ \textbf{end do};\ sum\ \textbf{end proc}$

```
> h();
```

$$0$$

```
> h(1);
```

$$1$$

```
> h(1,2);
```

$$3$$

```
> h(1,2,3,4,5,6);
```

$$21$$

```
>
> h := proc(start)
>    local i, sum;
>    sum := start;
>    for i from 1 to nargs
>      do
>        sum := sum + args[i]
>      od;
>    sum
> end proc;
```

$h := \textbf{proc}(start)$

$\textbf{local}\ i, sum;$

$\quad sum := start;\ \textbf{for}\ i\ \textbf{to}\ \mathrm{nargs}\ \textbf{do}\ sum := sum + \mathrm{args}[i]\ \textbf{end do};\ sum$

$\textbf{end proc}$

```
> h();
Error, (in h) h uses a 1st argument, start, which is missing
> h(3);  # What is wrong with the above definition?
```

$$6$$

```
> h(2,3);
```

$$7$$

```
>
```