Track_Calls Decorating the \texttt{factorial} Function

First \texttt{factorial} is defined, which binds its name to a function object.

Writing
\begin{verbatim}
factorial = Track_Calls(factorial)  # or using @Track_Calls
\end{verbatim}

(or using \texttt{@Track\_Calls}) rebinds \texttt{factorial} to refer to a \texttt{Track\_Calls} object, whose \texttt{f} attribute refers to \texttt{factorial}’s original function object (and \texttt{calls} is bound to 0, which \texttt{\_call\_} will examine and update).

Any direct call to \texttt{factorial} (or recursive call to \texttt{factorial} inside the function object) will look up the \texttt{Track\_Calls} object bound to \texttt{factorial} and execute the \texttt{\_call\_} method defined in that class.

\begin{verbatim}
(n):
  if n == 0
    return 1
  else:
    return n*factorial(n-1)
\end{verbatim}
Memoize Decorating the `fib` Function

First `fib` is defined, which binds its name to a function object.

Writing `fib = Memoize(fib)` (or using `@Memoize`) rebinds `fib` to refer to a `Memoize` object, whose `f` attribute refers to `fib`'s original function object (and `cache` is bound to an empty `dict`, which `__call__` will examine and update).

Any direct call to `fib` (or recursive call to `fib` inside the function object) will look up the `Memoize` object bound to `fib` and execute the `__call__` method defined in that class.