5.2.10. Yield expressions

```
yield_atom       ::=  "(" yield_expression ")"
yield_expression ::=  "yield" [expression_list]
```

New in version 2.5.

The `yield` expression is only used when defining a generator function, and can only be used in the body of a function definition. Using a `yield`expression in a function definition is sufficient to cause that definition to create a generator function instead of a normal function.

When a generator function is called, it returns an iterator known as a generator. That generator then controls the execution of a generator function. The execution starts when one of the generator's methods is called. At that time, the execution proceeds to the first `yield` expression, where it is suspended again, returning the value of `expression_list`to generator's caller. By suspended we mean that all local state is retained, including the current bindings of local variables, the instruction pointer, and the internal evaluation stack. When the execution is resumed by calling one of the generator's methods, the function can proceed exactly as if the `yield`expression was just another external call. The value of the `yield`expression after resuming depends on the method which resumed the execution.

All of this makes generator functions quite similar to coroutines; they yield multiple times, they have more than one entry point and their execution can be suspended. The only difference is that a generator function cannot control where should the execution continue after it yields; the control is always transferred to the generator's caller.