```
}
```

What we can see from listing 12.2 is that bytecode for a generator function code object is never executed when the generator function is called - the execution of bytecode only happens when the returned generator object is executing and we look at this next.

12.2 Running a generator

We can run a generator object by passing it as argument to the `next` builtin function. This will cause the generator to execute till it hits a `yield` expression then it suspends execution. The questions of importance to us here is how the generators are able to capture execution state and update those at will.

Looking back at the generator object definition from listing 12.1, we see that generators have a field that references a frame object and this is filled in when the generator is created as shown in listing 12.2. The frame object as we recall has all the state that is required to execute a code object so by having a reference to that execution frame, the generator object can capture all the state required for its execution.

Now that we know how a generator object captures execution state, we move to the question of how the execution of a suspended generator obect is resumed and this is not too hard to figure out given the information that we have already. When the `next` builtin function is called with a generator as an argument, the `next` function dereferences the `tp_iternext` field of the generator type and invokes whatever function that field references. In the case of a generator object, that field references a function, `gen_iternext` , that simply invoke another function, `gen_send_ex` , that does the actual work of resuming the execution of the generator object. Recall that before the generator object was created, all the intial setup was already carried out by the `_PyEval_EvalCodeWithName` function - frame object was intialised and variables initialised correctly, so the execution of the generator object involves calling the `PyEval_EvalFrameEx` with the frame object contained within the generator object as the frame argument. The execution of the code object contained within the frame then proceeds as explained the chapter on the evaluation loop