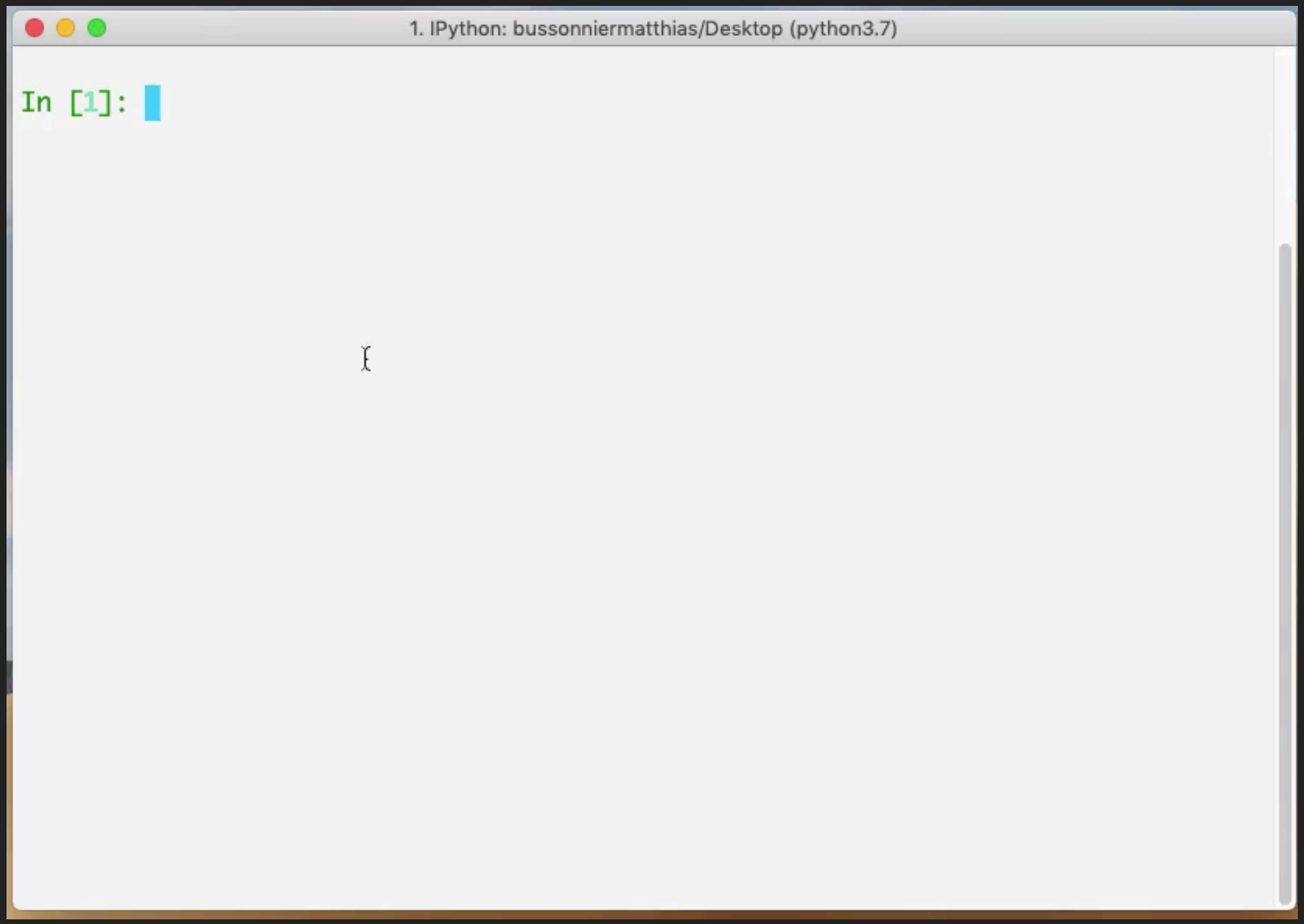
ASYNC/AWAIT IN THE REPL

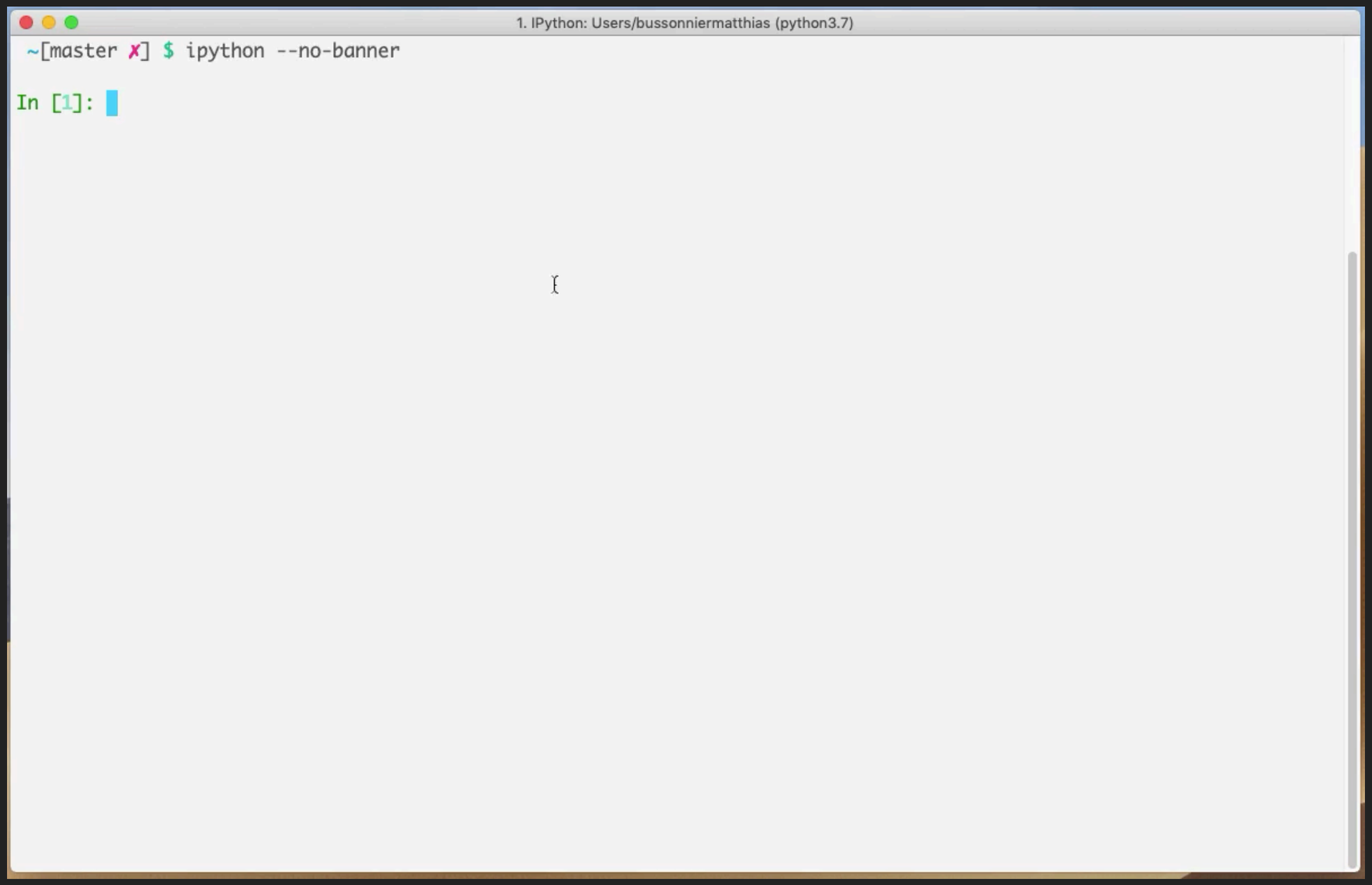
MATTHIAS BUSSONNIER (@MBUSSONN)

TL;DR: B.P.O 34616

MAKE TOP LEVEL AWAIT VALID; AND DO THE RIGHT THING

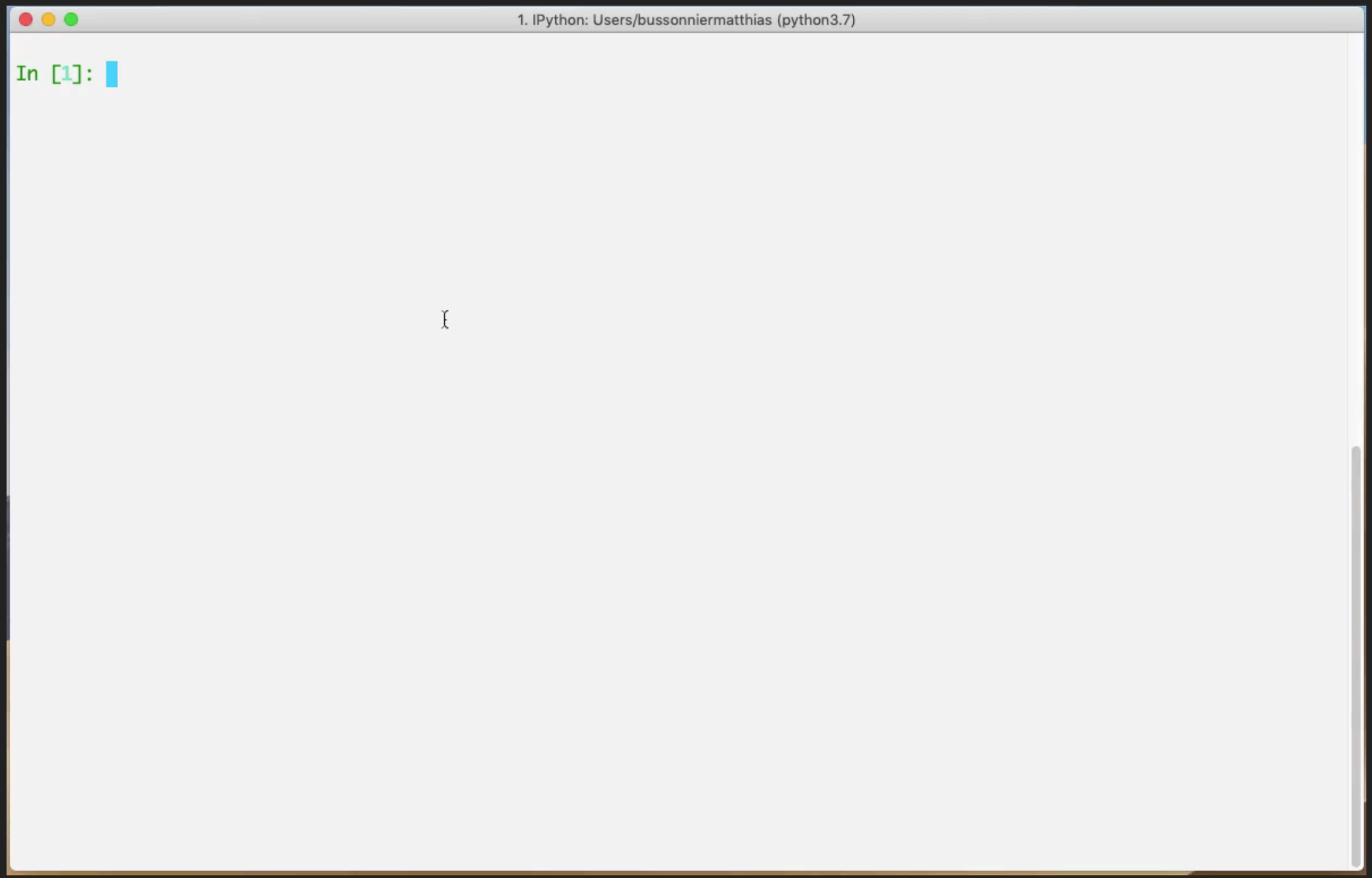


IS ACTUALLY USEFUL AND USED* WHEN WORKING AT THE REPL



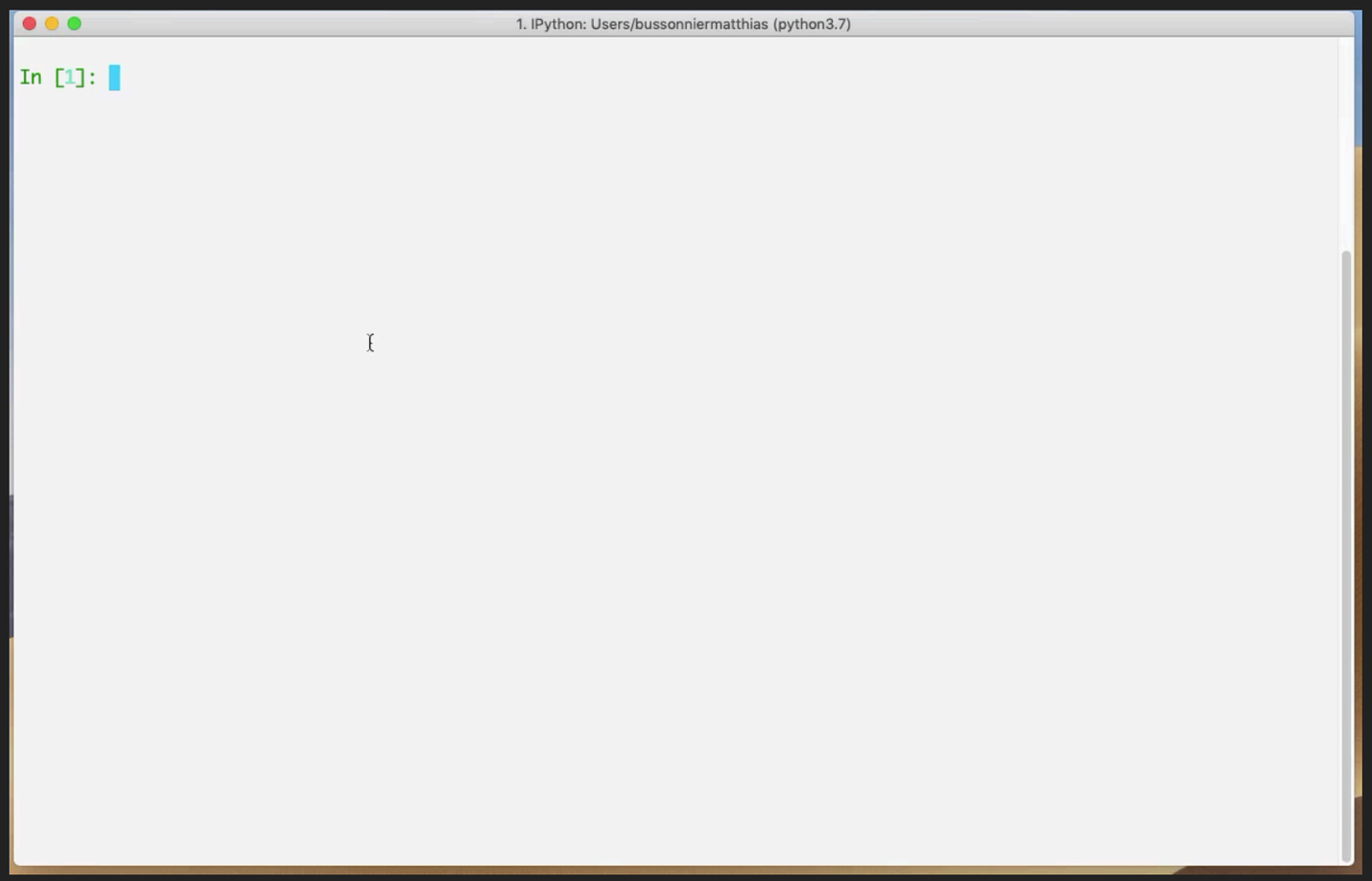
*WE GOT BUG REPORTS REALLY FAST AFTER RELEASE

WORKS WITH NON-ASYNCIO



. . . SOME ISSUES DUES TO WORKAROUND PYTHON LIMITATIONS

BACKGROUND TASKS DON'T ALWAYS RUN



ALSO ISSUES WITH. . . .

DOCSTRINGS – LOCALS/GLOBALS – MULTIPLE STATEMENTS – INCORRECT TRACEBACKS. . .

HOW ?

- Compile, and if SyntaxError:
  - String templating In async-def,
  - Exec that extract ast function body and Re-validate ast + modifs, (custom)
  - Copy co_code.flags|no_new_locals
    - Exec that -> coroutine
    - Await coroutine
- Else:
  - “World dumbest coroutine runner” for non-async code (can’t nest event loops)

WISHES

- top-level await valid **in a limited opt-in context**
  - **(**Still **forbidden in module context)**
  - Known Semantic for alternate implementations.
  - REPL-mode for “Compile(…, ‘REPL’)” & co
    - ‘REPL’ could also support multi-statements input
- Utility to detect if code is top-level-async
  - code_object.is_top_level_async
- await exec(…..)
- codeop.compile_command, ast… (some work already done so THANKS !)

WHY

- 3rd party tools like linters could use Python Stdlib*
- Allow to copy-past any piece of code in the REPL.
- Useful for education (and non-developers)
  - Remove some misconception that ASYNC is hard.
- (Do it before Javascript Can ? – not sure if this is pro or con)

TL;DR: B.P.O 34616