Interpreter Guide for replkit

This guide explains how to create, plug, and customize interpreters using the `replkit` package. It walks through minimal and advanced examples, focusing on Python-based interpreters.

✨ Introduction

`replkit` is a toolkit to build and run interactive command-line interfaces (REPLs). It allows you to plug in any interpreter that exposes an `eval()` method and optionally a `get_keywords()` method for autocompletion.

🚀 Getting Started: A Minimal Python Interpreter

Here's the simplest custom interpreter compatible with `replkit`:

```
from replkit import repl

class MyInterpreter:
    def eval(self, line: str) -> None:
        print(f"You typed: {line}")

    def get_keywords(self) -> set[str]:
        return {"help", "exit"}

if __name__ == "__main__":
    repl(interpreter=MyInterpreter(), argv=["--prompt", "Demo> "])
```

Run it:

```
$ python my_interpreter.py
```

✓ Full Example: Boolean Expression Evaluator

File: `calc_boolean_repl.py`

This interpreter evaluates boolean expressions using a custom syntax, variable assignment, and logic operators.

```
from replkit import repl

class CalcBoolInterpreter:
    def __init__(self):
```

```
        self.variables = {}
        self.operators = {
            'not': {'precedence': 4, 'assoc': 'right', 'fn': lambda x: not
x},
            'and': {'precedence': 3, 'assoc': 'left', 'fn': lambda x, y: x
and y},
            'or':  {'precedence': 2, 'assoc': 'left', 'fn': lambda x, y: x
or y},
            'xor': {'precedence': 2, 'assoc': 'left', 'fn': lambda x, y: x
!= y},
        }

    def eval(self, line: str) -> None:
        # Implementation here (see full file in the docs directory)
        pass

    def get_keywords(self) -> set[str]:
        return set(self.operators) | {"let", "vars", "clear", "True",
"False"} | set(self.variables)

def main():
    repl(
        interpreter=CalcBoolInterpreter(),
        argv=[
            "--prompt", "Bool> ",
            "--loglevel", "debug",
            "--hello", "Welcome in CalcBooleanInterpreter",
            "--file", "init.txt",
            "--run", "let D = A or B or C"
        ]
    )

if __name__ == "__main__":
    main()
```

Install and run calc_boolean_repl.py

```
$ mkdir calc_boolean && cd calc_boolean
$ python -m venv venv
$ . venv/bin/activate
$ pip install git+https://github.com/serge-pierre/replkit.git

# Copy at the root of calc_boolean directory the files
# - calc_boolean_repl.py
# - init.txt

$ python calc_boolean_repl.py
```

The files repl.log and repl_history will be at their default place : the user home directory.

Supported Commands:

- `let var = expr` → Assign a variable
- `vars` → Display all current variables
- `clear` → Reset variables
- `and`, `or`, `not`, `xor` → Boolean logic

⚙ Command-Line Options

`replkit` exposes several options via `argv`:

Option	Description
`--history`	History file
`--prompt`	Custom prompt string
`--hello`	Welcome message
`--log`	Log file
`--loglevel`	Logging verbosity (`debug`, `info`, ...)
`--run`	Command to run before interactive mode
`--file`	File with commands to run at startup

🧠 Best Practices

- Keep interpreter logic decoupled from REPL logic.
- Always implement `eval(line: str)`.
- Add `get_keywords()` to enhance user experience.
- Use a `main()` function to simplify testing and reuse.

🚫 Note on External Language Support

While `replkit` is technically capable of driving interpreters from other languages (via subprocess or protocol bridges), robust and persistent integration with non-Python interpreters (such as Guile, Node.js, etc.) introduces complexity that is not yet fully resolved.

As of now, we recommend focusing on Python-based interpreters to ensure predictable behavior and full feature support (prompt, autocompletion, logging).

Future versions of this guide may reintroduce advanced external interpreter integration once a stable architecture is validated.

📄 Resources

- GitHub: https://github.com/serge-pierre/replkit

- PyPI Package: replkit -- Not published for the moment
- Docstring & help: `python -m replkit --help`

Happy hacking!