# 1 Julia interfaces

`Julia` can be used in many different manners. This describes a few.

## 1.1 The `REPL`

Base `Julia` comes with a `REPL` package, which provides a means to interact with `Julia` at the command line.

```
Error: UndefVarError: ImageFile not defined
```

The `julia>` prompt is where commands are typed. The `return` key will send a command to the interpreter and the results are displayed in the REPL terminal.

The REPL has many features for editing, for interacting with the package manager, or interaction with the shell. However it is command-line based, which no support for mouse interaction. For that, other options are available.

## 1.2 `IJulia`

"Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages." The `IJulia` package allows `Julia` to be one of these programming languages. This package must be installed prior to use.

The Jupyter Project provides two web-based interfaces to `Julia`: the Jupyter notebook and the newer JupyterLab. The juliabox project and the binder project use Juptyer notebooks for their primary interface to `Julia`.

If not installed, these interfaces are available once `IJulia` is installed. The following command should do this:

```
] add IJulia
```

Should that not work, then this should as well:

```
using Pkg
Pkg.add("PyCall")
Pkg.add("IJulia")
```

---

The notebook interface has "cells" where one or more commands can be entered:

```
Error: UndefVarError: ImageFile not defined
```

The notes have blocks of commands, as though they are entered in a notebook.

In `IJulia`, a block of commands is sent to the kernel (the `Julia` interpreter) by typing "shift+return" or clicking on a "run" button. The output is printed below a cell, including graphics.

When a cell is evaluating, the leading `[]` has an asterick (`[*]`) showing the notebook is awaiting the results of the calculation.

Once a cell is evaluated, the leading `[]` has a number inserted (e.g., `[1]`, as in the figure). This number indicates the order of cell evaluation. Once a notebook is interacted with, the state of the namespace need not reflect the top-to-bottom order of the notebook, but rather reflects the order of cell evaluations.

To be specific, a variable like `x` may be redefined in a cell above where the variable is intially defined and this redefinition will hold the current value known to the interpreter. As well, a notebook, when reloaded, may have unevaluated cells with output showing. These will not influence the state of the kernel until they are evaluated.

When a cell's commands are evaluated, the last command executed is displayed. If it is desirable that multiple values be displayed, they can be packed into a tuple. This is done by using commas to separate values. `IJulia` will also display other means to print output (e.g., `@show`, `display`, `print`, ...).

To run all cells in a notebook from top to bottom, the "run all" command under the "Cell" menu is available.

If a calculation takes much longer than anticipated, the "kernel" can be interrupted through a menu item of "Kernel".

If the kernal appears unresponsive, it can be restarted through a menu item of "Kernel".

Notebooks can be saved (as `*.ipynb` files) for sharing or for reuse. Notebooks can be printed at HTML pages, and if the proper underlying software is available, as formatted pages.

---

JupyterLab has more features, commonly associated with an integrated development environment (IDE).

```
Error: UndefVarError: ImageFile not defined
```

The figure shows a notebook and a menubar in the side menu. In addition to notebooks, JupyterLab offers the change to edit source files, for example, for project development.

---

For an integrated development environment, where many features for project development are included, there are two powerful ones being developed for `Julia` that leverage various free-to-use projects.

### 1.2.1 Juno

Juno is a powerful, free environment for the Julia language. Juno is based on the cross-platform, javascript-based Atom editor and provides interfaces for the repl, graphics, help pages, debugging, project management, etc. Atom was developed by GitHub, since acquired by Microsoft.

### 1.2.2 VSCode

The VS Code extension provides support for the julia programming language for VS Code. VS Code is an open-sourced code editor supported by Microsoft. Similar to `Juno`, VS Code provides a cross-platform interface to `Julia` geared towards programming within the language.