# finder Documentation

*Release 1.0.2*

Bharadwaj Yarlagadda

Sep 13, 2017

# Contents

Command Line Interface tool which helps in finding a given text/pattern in a given folder/file path(s).

# CHAPTER 1

# Links

- Project: https://github.com/bharadwajyarlagadda/finder
- Documentation: http://finder.readthedocs.io
- Pypi: https://pypi.python.org/pypi/finder
- TravisCI: https://travis-ci.org/bharadwajyarlagadda/finder

# CHAPTER 2

# Quickstart

Install using pip:

```
pip install finder
```

# CHAPTER 3

# Features

- Finds given text/pattern in the given file path(s).
- Iterates through all the non-executable files in a given directory path.
- Avoids all the non-readable files in a given directory path.
- Supported on Python 3.3+.

CHAPTER 4

# Command Line

Search for a pattern in a given file path

```
$ finder search --path='~/finder/setup.py' --pattern='port'
/home/aj/Projects/finder/setup.py:4: import os
/home/aj/Projects/finder/setup.py:5: from setuptools import setup, find_packages
```

Search for a pattern in given directory path

```
$ finder search --path='~/finder' --pattern='port'
/home/aj/Projects/finder/.gitignore:37: # Unit test / coverage reports
/home/aj/Projects/finder/LICENSE.rst:16: copies or substantial portions of the␣
↪Software.
/home/aj/Projects/finder/setup.cfg:2: addopts = --doctest-modules -v -s --color=yes --
↪cov-config=setup.cfg --cov-report=term-missing
/home/aj/Projects/finder/setup.py:4: import os
...
```

Check the error occurred only if there is an error occurred while searching the file

```
$ finder search --path='~/finder' --pattern='port' --verbose
/home/aj/Projects/finder/.gitignore:37: # Unit test / coverage reports
/home/aj/Projects/finder/LICENSE.rst:16: copies or substantial portions of the␣
↪Software.
/home/aj/Projects/finder/setup.cfg:2: addopts = --doctest-modules -v -s --color=yes --
↪cov-config=setup.cfg --cov-report=term-missing
/home/aj/Projects/finder/setup.py:4: import os
...
...
UnicodeDecodeError:/home/aj/Projects/finder/tests/__pycache__/test_find.cpython-35-
↪PYTEST.pyc:'utf-8' codec can't decode byte 0xf5 in position 5: invalid start byte␣
↪None
UnicodeDecodeError:/home/aj/Projects/finder/tests/__pycache__/test_cli.cpython-35-
↪PYTEST.pyc:'utf-8' codec can't decode byte 0xf5 in position 5: invalid start byte
...
```

CHAPTER 5

# Guide

## Command Line

```
$ finder --help
  Command line interface for searching a given pattern/text in the given
  directory/file path.

  Options:
    -v, --version  Finder tool version.
    --help         Show this message and exit.

  Commands:
    search  Searches for the given pattern in the...
```

Checkout finder tool's version

```
$ finder --version
1.0.2
```

Sub-command `search` helps in searching the pattern in a given path

```
$ finder search --help
  Searches for the given pattern in the directory/file path provided.

Options:
  --path path1[,path2,...,pathN]  Searches for the pattern in the
                                  directory/file path provided. If a
                                  directory/file path is not provided, the
                                  tool searches for the pattern in the current
                                  working directory.
  -P, --pattern, --text <pattern>
                                  Text to be searched.  [required]
  -v, --verbose                   Some files cannot be opened and searched for
                                  the given pattern. For example, kernel files
```

```
                                    which generate content on go, files which
                                    are not utf-8 encoded, etc. You can use this
                                    flag if you need a detailed output of which
                                    file has an error.
  --help                            Show this message and exit.
```

Search for a pattern in a given file path

```
$ finder search --path='~/finder/setup.py' --pattern='port'
/home/aj/Projects/finder/setup.py:4: import os
/home/aj/Projects/finder/setup.py:5: from setuptools import setup, find_packages
```

Search for a pattern in given directory path

```
$ finder search --path='~/finder' --pattern='port'
/home/aj/Projects/finder/.gitignore:37: # Unit test / coverage reports
/home/aj/Projects/finder/LICENSE.rst:16: copies or substantial portions of the␣
↪Software.
/home/aj/Projects/finder/setup.cfg:2: addopts = --doctest-modules -v -s --color=yes --
↪cov-config=setup.cfg --cov-report=term-missing
/home/aj/Projects/finder/setup.py:4: import os
...
```

Check the error occurred only if there is an error occurred while searching the file

```
$ finder search --path='~/finder' --pattern='port' --verbose
/home/aj/Projects/finder/.gitignore:37: # Unit test / coverage reports
/home/aj/Projects/finder/LICENSE.rst:16: copies or substantial portions of the␣
↪Software.
/home/aj/Projects/finder/setup.cfg:2: addopts = --doctest-modules -v -s --color=yes --
↪cov-config=setup.cfg --cov-report=term-missing
/home/aj/Projects/finder/setup.py:4: import os
...
...
UnicodeDecodeError:/home/aj/Projects/finder/tests/__pycache__/test_find.cpython-35-
↪PYTEST.pyc:'utf-8' codec can't decode byte 0xf5 in position 5: invalid start byte␣
↪None
UnicodeDecodeError:/home/aj/Projects/finder/tests/__pycache__/test_cli.cpython-35-
↪PYTEST.pyc:'utf-8' codec can't decode byte 0xf5 in position 5: invalid start byte
...
```

## Functionality

1. The main entry point for our API is the `api.find()` method. You can pass in both file/directory paths and the pattern to be searched for.
2. If you provide a directory path, it will go ahead and do a `os.walk` and brings out all the files in that directory path.
3. **While searching for the pattern,**
    (a) **It avoids all the non-readable files:**
        i. Audio files.
        ii. Video files.

            iii. Image files.
            iv. Some of the kernel based files such as the files under `/proc` in `Ubuntu/Linux.`

      (b) It reads the file line by line so that we can avoid saving the whole file in the memory (which of course will be memory issue for huge files).

4. This whole process runs concurrently. As in, the API allots thread for each file to be searched and once the search is complete, the thread comes and joins back in the main process.
5. I personally have tested the performance and the memory usage is very low. If you face any of the performance issues, please report it at Issues.
6. The data from the API looks is explained under `Schema` section. The output fields are also explained in the same section.
7. The output data is a JSON-encoded string and it is generated only when `finder` finds tha pattern in a given file.

## Schema

API output schema

```
{
    "api_version": "<api_version>",
    "requested_on": "<datetime>",
    "path": "<file_path>",
    "total_items": "<total_items>",
    "items": [
        {"line_number": "<line_number>",
         "line": "<line>"},
        {"line_number": "<line_number>",
         "line": "<line>"},
        ...
    ],
    "error": [
        {"type": "<error_type>",
         "message": "<error_message>",
         "extra": "<extra_message>"}
    ]
}
```

Data fields:

- `api_version`: Finder tool's version
- `requested_on`: Datetime value (when the tool was requested)
- `path`: File path
- `total_items`: Total items returned.
- `items`: All the data items returned from the finder tool. The items comprise of:
  - `line_number`: Line number at which the pattern was found.
  - `line`: Actual line in which the pattern was found.
- `error`: Errors from the finder tool if any.
  - `type`: Error type (Ex. PermissionError, OSError, etc.)

  – `message`: Error message from the finder tool.
  – `extra`: Any extra error message from the finder tool.

Example

```
{
    "api_version": "1.0.0",
    "requested_on": "2017-04-14T04:17:41.204588",
    "path": "/home/aj/Projects/finder/api.py",
    "total_items": "7",
    "items": [
        {"line_number": "3",
         "line": "import os"},
        {"line_number": "4",
         "line": "import queue"},
        ...
    ]
    "error": []
}
```

When there are errors while searching the file,

```
{
    "api_version": "1.0.0",
    "requested_on": "2017-04-14T04:17:41.204588",
    "path": "/etc/init.d/apache2",
    "total_items": "0",
    "items": []
    "error": [
        {"type": "PermissionError",
         "message": "...",
         "extra": "..."}
    ]
}
```

# CHAPTER 6

# Project Info

## License

## Changelog

- Add colors to CLI output.
- Add `sleep()` call in `find()` method to reduce the CPU utilization percentage.
- Move `search` method from `finder.api` to `finder.utils`.
- Move `iterfiles` method from `finder.api` to `finder.utils`
- Add documentation.

### v1.0.2 (2017-05-07)

- FIX bug in CHANGELOG.

### v1.0.1 (2017-05-07)

- Minor updates.

### v1.0.0 (2017-05-07)

- Add `search` method to search for a given pattern in the text provided.
- Add `iterfiles` method to yield all the file paths in a given folder path.
- Add `is_executable` method to validate whether the given file is a executable or not.
- Add `read` method to read a given file line by line.
- Add wrapper method `find` to iterate through the given list of files/directories and find the given pattern in the files.
- Add `FileReader` class to searching all the files concurrently.
- Add schemas for serializing the data to a JSON-encoded string.
- Add command line wrapper around the API. User can now use the command line interface to get all the search results.

## Authors

### Lead

- Bharadwaj Yarlagadda, yarlagaddabharadwaj@gmail.com, bharadwajyarlagadda@github

## Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

### Types of Contributions

#### Report Bugs

Report bugs at https://github.com/bharadwajyarlagadda/finder.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

### Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" or "help wanted" is open to whoever wants to implement it.

### Write Documentation

finder could always use more documentation, whether as part of the official finder docs, in docstrings, or even on the web in blog posts, articles, and such.

### Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bharadwajyarlagadda/finder.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## Get Started!

Ready to contribute? Here's how to set up `finder` for local development.

1. Fork the `finder` repo on GitHub.
2. Pull your fork locally:

   ```
   $ git clone git@github.com:<username>/finder.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenv installed, this is how you set up your fork for local development:

   ```
   $ cd finder
   $ pip install -r requirements-dev.txt
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass linting and all unit tests by testing with tox across all supported Python versions:

   ```
   $ invoke tox
   ```

6. Add yourself to `AUTHORS.rst`.
7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

## Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the README.rst.
3. The pull request should work for Python 2.7, 3.4, and 3.5. Check https://travis-ci.org/bharadwajyarlagadda/finder/pull_requests and make sure that the tests pass for all supported Python versions.

# CHAPTER 7

# Indices and tables

- genindex
- modindex
- search