Previous Next

Build the App Skeleton

Estimated Time: 1 hour

In this assignment you'll set up a skeleton for the snippets application which you'll build throughout this lesson.

First, create a new folder for this project called 'snippets-app' and from the command line. `cd` into it. Do a `git init` to start version control, and go ahead and create a remote repository on Github to link this local repo to.

Next you are going to introduce a new module: `logging`. This module will allow you to track what happens in the application, and will help you identify any problems in the code. Create a file called *snippets.py* and add the following code:

```
import logging

# Set the log output file, and the log level
logging.basicConfig(filename="snippets.log", level=logging.DEBUG)
```

The `filename` argument describes where you want the log to be saved. In this case, it will write to a file called "snippets.log" in the current directory. The `level` argument sets the log level. The logging module splits log messages into one of five severity levels:

- **DEBUG**: Detailed information, typically of interest only when diagnosing problems.
- **INFO**: Confirmation that things are working as expected.
- **WARNING**: An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
- **ERROR**: Due to a more serious problem, the software has not been able to perform some function.
- **CRITICAL**: A serious error, indicating that the program itself may be unable to continue running.

When you set the log level to `DEBUG` all of the messages will be logged. If it is set to `WARNING` only log messages with a severity of `WARNING` or higher will be logged.

Creating a program skeleton

Let's figure out what operations you'll need, and create *stubs* for each main function. A stub is a function that is defined but does nothing useful. The most important feature is storing a snippet. In your *snippets.py* file add the following stub:

```
def put(name, snippet):
    """
    Store a snippet with an associated name.

    Returns the name and the snippet
    """
    logging.error("FIXME: Unimplemented - put({!r}, {!r})".format(name
    return name, snippet
```

When called, this function will report in the log exactly that the put feature doesn't exist yet. Using the well-recognized tag `FIXME` identifies the problem both in the source and the log. Formatting the strings with the `!r` modifier means that the `repr()` function will be run over the data to provide the output. (`repr()` returns a string containing a printable representation of an object.) This ensures that the log is clean and readable, no matter what string is provided. Add another stub:

```
def get(name):
    """Retrieve the snippet with a given name.

    If there is no such snippet, return '404: Snippet Not Found'.

    Returns the snippet.
    """
    logging.error("FIXME: Unimplemented - get({!r})".format(name))
    return ""
```

This is a good time to think about error conditions. What would be the right action for get() in the case where the snippet doesn't exist? In this example, we return a fixed string explaining the problem, but you might choose instead to return `None` , or raise an exception, or create some sort of auto-generated snippet, or anything else you like.

Before moving on, add and commit your work so far.

Trying it out

You can try this out at the interactive interpreter, even though it's not very useful at the moment. Open up a terminal in the same directory as your snippets.py folder, and then open up the Python interpreter by typing `python3` with no script name. You'll be greeted with interactive Python's distinctive prompt: `>>>` . Now try using your function to store and retrieve a snippet:

```
>>> import snippets
>>> snippets.put("list", "A sequence of things - created using []")
>>> snippets.get("list")
```

It may appear that nothing has happened, but have a look at the log file (snippets.log) and you should see something like this:

```
ERROR:root:FIXME: Unimplemented - put('list', 'A sequence of things - 
ERROR:root:FIXME: Unimplemented - get('list')
```

Add stubs for whatever other functions you think will be necessary, and again discuss this with your mentor.

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

Mark as completed

Previous Next