Orginal Document: https://cloud.google.com/appengine/docs/python/tools/using-local-server#Python_Debugging_with_PDB

Using the Local Development Server

The App Engine Python SDK includes a local development server you run locally to simulate your application running in production App Engine. The simulated environment enforces some sandbox restrictions, such as restricted system functions and Python module imports, but not others, like request time-outs or quotas.

The local development server also simulates the services provided by App Engine Python SDK libraries (e.g. Datastore, Memcache, Task Queue) by performing their tasks locally. Note that when your application is running in the development server, you can still make remote API calls to the production infrastructure using Google APIs HTTP endpoints.

Running the local development server

Once you have a directory for your application and an app.yaml configuration file, you can start the local development server using the `dev_appserver.py` command.

```
dev_appserver.py myapp
```

The local server listens on port 8080 by default. You can visit the application at this URL: http://localhost:8080/.

To change which port the local server uses, use the `--port` option:

```
dev_appserver.py --port=9999 myapp
```

To stop the local server: with Mac OS X or Unix, press Control-C or with Windows, press Control-Break in your command prompt window.

Note: **`dev_appserver.py`** only runs Python 2.7 apps. For help migrating, see Migrating to Python 2.7.

Specifying application IDs

If you want to use application IDs with the local development server, don't use the App ID from the `APPLICATION_ID` environment variable. The development server differs from production App Engine service in that the local server prepends the string `dev~` to the`APPLICATION_ID` environment variable.

Instead, if you need to access your App ID in the local server, for example to spoof an email address, use the `get_application_id()` function. To get the hostname of the running app, use the `get_default_version_hostname()` function.

Detecting application runtime environment

To find out whether your code is running in production or in the local development server, check `if os.getenv('SERVER_SOFTWARE', '').startswith('Google App Engine/')`. When this is `True`, you're running in production; otherwise, you're running in the local development server.

Using the local Datastore

The local development server simulates the App Engine datastore using a local file that persists between invocations of the local server.

For more information on indexes and `index.yaml`, see the Datastore Indexes and Datastore Index Configuration pages.

Browsing the local Datastore

If your app has written data to your local Datastore using the local development server, you can browse it in the local development console.

To browse local Datastore:

1. Start the development server.
2. Access the Datastore Viewer in the local development console. (The URL is `http://localhost:8000/datastore`.)
3. View your local Datastore contents.

Specifying the ID allocation policy

For production App Engine, you can set the Datastore to automatically generate entity IDs.

Although the auto ID assignment policies for the production server are completely different than those used by the development server, you can also set the automatic ID allocation policy for the local server.

To specify the automatic ID assignment policy, use the `--auto_id_policy` option:

```
dev_appserver.py --auto_id_policy=sequential
```

where `--auto_id_policy` can be one of the following:

- `scattered`: (default) IDs are assigned from a non-repeating sequence of approximately uniformly distributed integers.
- `sequential`: IDs are assigned from the sequence of consecutive integers.

Note: Your app should make no assumptions about the sequence of automatic IDs assigned in production. .

Clearing the local Datastore

To clear the local datastore for an application, invoke the local development server as follows:

```
dev_appserver.py --clear_datastore=yes myapp
```

Changing local Datastore location

To change the location used for the datastore file, use the `--datastore_path` option:

```
dev_appserver.py --datastore_path=/tmp/myapp_datastore myapp
```

Using the Users service

App Engine provides a Users Service to simplify authentication and authorization for your application. The local development serversimulates the behavior of Google Accounts with its own sign-in and sign-out pages. While running under the local development server, the`users.create_login_url` and `users.create_logout_url` functions return URLs for `/_ah/login` and `/_ah/logout` on the local server.

Using Mail

The local development server can send email for calls to the App Engine mail service using either an SMTP server or a local installation ofSendmail.

USING SMTPUSING SENDMAIL

To enable mail support with an SMTP server, invoke `dev_appserver.py` as follows::

```
dev_appserver.py --smtp_host=smtp.example.com --smtp_port=25 \
  --smtp_user=ajohnson --smtp_password=k1tt3ns myapp
```

where you set the `--smtp_host`, `--smtp_port`, `--smtp_user` and `--smtp_password` options with your own values.

Using URL Fetch

When your application uses the URL fetch API to make an HTTP request, the local development server makes the request directly from your computer. The URL Fetch behavior on the local server may differ from production App Engine if you use a proxy server for accessing websites.

Using the Interactive Console

The **Interactive Console** allows developers to enter arbitrary Python code into a web form and execute it inside their app's environment; it provides the same access to the application's environment and services as a .py file inside the application itself.

Important: If you specified the argument **`--address`** when you start the local server instead of using the default address, the Interactive console is disabled to prevent someone remotely executing arbitrary Python code on your computer. You can bypass this and re-enable the Interactive Console by starting the development server using the **`--enable_console`**argument.

To use the Interactive Console:

1. Start the development server.
2. Access the Interactive console in the in the local development console. (The URL is `http://localhost:8000/console`.)
3. Enter any Python code you'd like to run in the text area, then submit the form to execute it. For example the following code will add a Datastore entity called `Greeting` with text content of `Hello`:

```
from google.appengine.ext import ndb
class Greeting(ndb.Model):
  content = ndb.TextProperty()

e = Greeting(content="Hello")
```

```
  e.put()
```

Debugging with PDB

To use the Python PDB debugger:

1. Add this line into your code:

   ```
   import pdb; pdb.set_trace();
   ```

   `dev_appserver` will break at this point, and drop into the PDB REPL, allowing you to debug your code from the command line.
2. If your application makes multiple simultaneous requests that invoke `pdb.set_trace()`, multiple debugging sessions will start concurrently, each of which sends output to STDOUT. To avoid this, serialize your requests by disabling the `dev_appserver` multi-threading and multi-processing support, as follows:

   a. Disable multi-threading for

      - All modules using the `--threadsafe_override=false` flag.
      - One module using the `--threadsafe_override=<MODULENAME>:false` flag
      - Multiple modules using the `--threadsafe_override=<MODULE1NAME>:false,<MODULE2NAME>:false` flag

   b. Disable multi-processing for

      - All modules using the `--max_module_instances=1` flag
      - One module using the `--max_module_instances=<MODULENAME>:1` flag;
      - Multiple modules using the `--max_module_instances=<MODULE1NAME>:1,<MODULE2NAME>:1` flag