**Steps to Deploy on Google Cloud Platform**

This document describes steps to follow while deploying the Python code (Code for Intents & detecting biases) on the GCP.

## How does the Interaction works between UI & GCP

* When the user clicks “compute” the UI converts the user's intent to a json & calls the GCP via a http request. The [json structure](https://docs.google.com/document/u/0/d/1oA5-r6-I1FbaUUyqJygPbXholVTn4T7FQXWmMcRsxM0/edit) includes keys like ‘intent’ etc.
* After computing the results the Intents functions returns another json to the UI

# Steps

## Step1 - Getting Json of query

In the user interface of the Add-On after clicking the “compute query” button, the json gets generated & is shown in the browser console.

Copy json from then browse console.

To open the console

1. Right click
2. inspect element

## Step2 - Paste the copied json in intents/data/obj.json locally

The json will be used to execute the query “locally”

Note : Further steps assume that python3 & related libraries are installed. If not, run the command -

pip install requirements.txt

## Step3 - Run debug.py

What debug.py does

1. It reads the json as string
2. Converts string to json using json.loads()
3. Passes the json to main.py as a dummy http request
   1. This step **stimulates** what actually happens when the UI sends a Http request to the GCP endpoint.
4. Main.py behaves the same way as it would behave when the UI calls it through an actual Http request over the server
5. By this way main.py extracts the parameters involved in the user query and main.py calls the respective intent

Command to run main.py using debug.py-

python3 debug.py

## Step4 - Debug the intents/oversights code locally

If any error occurs while querying, the python traceback error will be visible in the terminal.

After fixing the changes again run debug.py.

Doing the same thing on GCP is very difficult & time consuming.

## Step5 - Uploading code to GCP

Select all the files/folders and zip them. Ignore these files while zipping

1. Database folder
2. All the test files including test\_all.sh
   1. test\_all.sh is a utility file that runs all the python test files and displays the results of each test. Running all tests using test\_all.sh is much easier.
   2. It is recommended to run test\_all.sh before deploying to GCP or creating a PR.
      1. It ensures that after making the change, all tests pass or not.

Upload the zip file to GCP endpoint. There is an option to “Zip upload”

**Steps to Deploy on AppsScript**

## Step1 - Clone the Google spreadsheet

## This [Sheet](https://docs.google.com/spreadsheets/d/1a_s6LRuxoTcdVmcjyc2ijrV_7dVs8g0EiT8t2LZYZDA/edit#gid=1546712586) contains the updated version of the Add-On.

Make a copy of the sheet.

In the copy, you would be able to edit the AppsScript code in it.

## Step2 - Make changes in the AppsScript

Open script editor & it would contain the latest version of the Add-On.

## Step3 [optional]- Download code from AppsScript & make a PR

# Appendix

## Issues faced while deploying code directly to GCP

* When the function crashes it takes some time (around 1 minute) to display this in the logs
* In GCP logs, the Python Traceback is not displayed, so it was difficult to figure out exactly where/why the error occurred
* The only way was to make changes in the code, then deploy and check if the GCP endpoint crashes or not.
  + But this also took a lot of time as a function takes around 2 minutes to deploy & then Querying from the UI & checking logs.