Got it!
Your Cloud Computing task is:

Install and configure Google App Engine (GAE), then **create a Hello World app** and **other simple web apps using Python** on **Windows**.

I'll give you:

- **Part 1:** Step-by-Step Instructions
- **Part 2:** Detailed Explanation of each step

Part 1: Steps to Install, Configure GAE, and Create Python Apps on Windows

Step 1: Install Python

- Download and install Python 3.x from the official website.
- During installation, **check the box** that says **"Add Python to PATH"**.

Step 2: Install Google Cloud SDK

- Download the **Google Cloud SDK installer** for Windows from Google Cloud SDK Download.
- Install it by running the installer.
- During setup, select the option to **install the App Engine extension for Python**.

Step 3: Initialize Google Cloud SDK

- Open **Google Cloud SDK Shell** (installed with SDK).
- Run:
- `gcloud init`
-
- Sign in to your Google account.
- Select or create a **Google Cloud Project**.

Step 4: Install App Engine Component for Python

- Run in the SDK Shell:
- `gcloud components install app-engine-python`
-
- (For newer Python 3 versions: install `app-engine-python-extras` if needed.)

Step 5: Create a Project Folder for Your App

- Create a folder on your computer, e.g., `hello-world-app`.

Step 6: Create a Simple Hello World Python App

Inside the folder, create these two files:

1. **main.py** (your app code):

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World from Google App Engine!'

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True)
```

2. **app.yaml** (app configuration):

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app

handlers:
- url: /.*
  script: auto
```

Step 7: Install Required Python Packages

Inside your app folder, open command prompt and run:

```
pip install Flask gunicorn
```

```
pip freeze > requirements.txt
```

(This saves the required libraries.)

Step 8: Test Locally

Run your app locally:

```
python main.py
```

Visit http://localhost:8080 and see **Hello, World!**

Step 9: Deploy to Google App Engine

From inside your app folder, run:

```
gcloud app deploy
```

- Confirm deployment if asked.

Step 10: View Your Live App

After deployment finishes:

- Run:
- `gcloud app browse`
-
- Or manually open the URL shown (something like `https://your-project-id.uc.r.appspot.com`).

Your Hello World app is now live on Google App Engine!

📚 Part 2: Detailed Explanation of Each Step

Step 1: Install Python

Google App Engine supports Python apps. Installing it ensures you can code and run Python scripts on your PC.

Step 2: Install Google Cloud SDK

The **Google Cloud SDK** is the toolkit you need to:

- Interact with Google Cloud from your computer.
- Deploy apps to App Engine.

Step 3: Initialize Google Cloud SDK

`gcloud init` connects your machine to your Google Cloud account. It sets up **authentication** and **project selection** so you can deploy apps.

Step 4: Install App Engine Component

Although you installed SDK, App Engine for Python is a separate part. Installing it enables deploying Python-based apps.

Step 5: Create a Project Folder

Organizing your app into a folder makes it easier to manage. You'll store your Python scripts and config files there.

Step 6: Write Python App

- **main.py** contains your web application logic.
- **app.yaml** tells App Engine:
- Which Python version you're using.
- How to start your app (entrypoint).

Step 7: Install Flask and Gunicorn

- **Flask**: A lightweight Python web framework.
- **Gunicorn**: A production server recommended by Google Cloud.
- `requirements.txt` tells GAE which libraries your app needs.

Step 8: Test Locally

Running it on `localhost` allows you to catch errors **before** uploading to cloud.

Step 9: Deploy to GAE

`gcloud app deploy` uploads your project to Google's servers and hosts it live.

Step 10: View App

You can open the app directly or let SDK open it in your browser.

Bonus: Examples of Other Simple Web Apps

Once Hello World works, you can create variations like:

- A page showing current date and time.
- A form that takes user input and displays it back.
- A simple website with multiple pages (routes).

Would you also like me to give you **examples** of a few more simple apps you can quickly deploy after Hello World?
(Example: Date-Time app, Calculator app, Form submission app?)