



## Concept

This assignment is about making a web application – in other words, a *dynamic* (changing) *website* using Python 3 and the Bottle web application framework.

You will add to your fan website a capability to allow users to add comments to the bottom of pages as long as they also enter the correct password that you choose.

The assignment will be done by **individuals**.

## High-level Instructions

\* See page 2 for step-by-step instructions. \*

- **STEP 1** (Week 12: 2 December approximately):
  - Create your Assignment 2 GitHub repository from the link supplied below.
  - Sign up to Heroku & join the ITUIS18 GitHub organization. .
  - Create a Heroku web application from your repository & connect it to your GitHub repository.
  - Add your Heroku web-app address to your GitHub repository.
- **STEP 2** (Week 12: 2 December approximately):
  - Write a REPL **console** program that reads user input, saves and prints all previously entered text.
- **STEP 3** (Week 13: 9 December approximately):
  - Check the code that does “password hashing” and password checking from the provided link below.
  - Run the program with a password of your choice to generate a password hash.
  - Change the REPL program that you wrote in step 1 above so that it only allows text to be saved if the matching password is entered.
- **STEP 4** (Week 14: 16 December approximately):
  - Adapt and publish your fan web page from Assignment 1 via Bottle and Heroku.
- **STEP 5** (Week 15: 23 December approximately):
  - Adapt your bottle app from step 4 so that users can enter comments in a text field.
  - Make previously entered comments show up above or below the comment entry field.
  - Add another text field for password and only allow users who enter the password correctly enter comments. *Use a hashed password so that the plain text password is not visible via your source-code on Github.*
- **STEP 6** (Week 16: **30 December 23:59am**):
  - Polish your website, fix bugs, and ensure that it is working well.
  - Double-check that the Heroku site is linked from your Github repository.
  - Double-check the web site works on Heroku and when run on your computer.
  - Add any extensions you planned to your website.
  - Push this all to your GitHub repository **by the due date and time** so that it gets downloaded and marked.
  - Sign up for your demonstration session.

## Submission Notes

- All steps of this assignment will be evaluated only after the final submission time has passed. However, it is recommended that you get the earlier steps done by their respective due dates.
- You need to upload your Heroku web application address to your GitHub website so that your website is accessible by markers from the internet. Instructions will be given below.
- Your assignment will be automatically downloaded for marking from GitHub at the due date and time.
- There is a limit on the maximum size of your website, set by Heroku.
- **To have your assignment counted against your grades, attend your demonstration session**, which will be announced in a separate schedule and will be after the due date and time of the assignment. *Every member of your group will attend a demonstration session.*
- Check the separate evaluation form to see on what basis your markers will be grading you.
  - Use all of the advanced HTML and HTTP techniques shown in the evaluation form.
  - Use all of the Python techniques shown in the evaluation form.
- Have fun.

\* **Keep your eye on the separate evaluation form, for the marks.** \*

## Step 1

### Sign up to Heroku

Go to <https://www.heroku.com/> and create an account for yourself if you don't already have one.



The following steps will be easier if you are logged in to both Heroku and GitHub but if you are not you may be prompted to log in anyway.

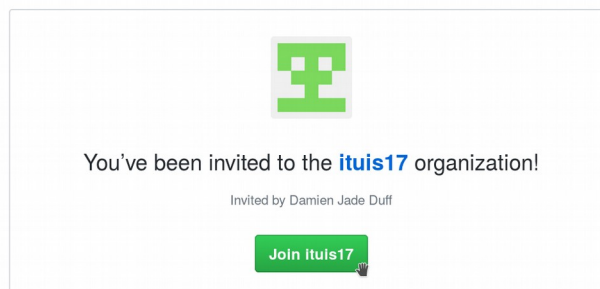
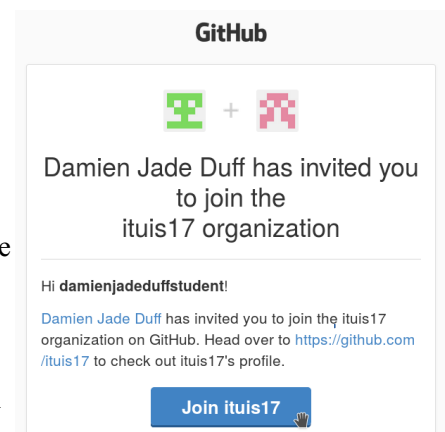
### Accept your invitation to our GitHub “organisation”

You should receive an invitation, sent to the email address that you used to join GitHub, to join our GitHub organisation “ituis18”.

The email should resemble the image on the right (except the organisation name ituis17 may change).

Start the procedure to accept the invitation by clicking “Join ituis18” in the email.

You should find yourself on a web-page where you should accept the invitation by again selecting “Join ituis18”, as in the image below.

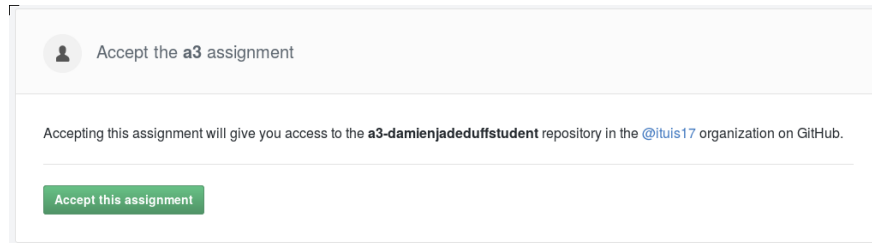


## Make your Assignment GitHub repository

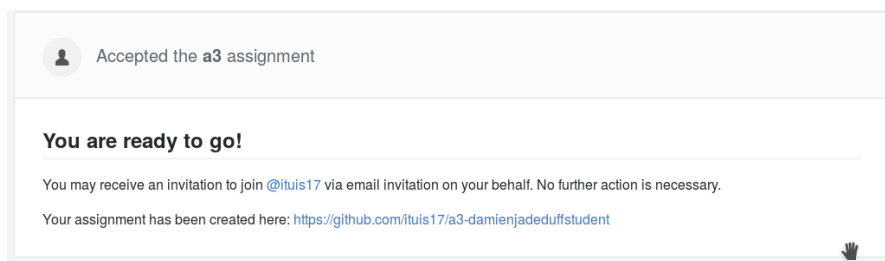
While logged in to GitHub (or by logging in after you click), navigate to the below link:

<https://classroom.github.com/a/oFREegFl>

On the resulting web page accept the assignment to create your new assignment repository, similar to the below image (though the assignment name and organisation name may differ):



Your assignment GitHub repository will be available at the link provided on the subsequent web page:



If you navigate to the repository you will find a **bottle** web application along with a README file and files necessary for deploying your bottle web application to the Heroku cloud service. The function of the files in this repository are as follows:

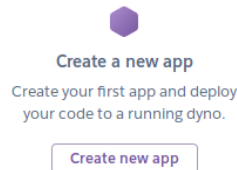
- **bottle\_app.py**: this is the Python source file in which you will write your web application (this will come in later steps - for now you do not need to understand how it works).
- **README.md**: A text file that should contain any information you might like to supply with other developers using your code. You can change this as you like.
- **app.json**: A configuration file that tells Heroku how to “deploy” (run) your web application. You do not need to edit this file.
- **Procfile**: A configuration file that tells Heroku what to run in your web application. You do not need to edit this file.
- **requirements.txt**: A configuration file that specifies what extra modules need to be installed by Heroku in order to run your web application. You usually do not need to edit this file.

You can add as many other files to the repository as you like, but it is not recommended you edit the `app.json`, `Procfile` or `requirements.txt`.

## Create a Heroku web application from the resulting repository

Ensure you are logged into your Heroku account.

Assuming you have not yet created any apps with Heroku, the Heroku homepage (<https://dashboard.heroku.com/>) should give you the opportunity to create a new web app:



You will reach a web page prompting you to make a new web application on Heroku from your repository, similar to below:

The image shows a Heroku form for creating a new app. It has a section 'App name' with a text input field containing 'mybig101assignment' and a green checkmark icon to its right. Below this is a green message: 'mybig101assignment is available'. The next section is 'Choose a region' with a dropdown menu showing 'Europe'. Below that is a link 'Add to pipeline...' with a small icon. At the bottom is a purple 'Create app' button.

Call the application what you want. You will later upload add its URL to your GitHub repository README as a link so that your markers can find it.

Click “Create app”.

*You will need to wait a small amount of time.*

## Connect your Heroku web application to your GitHub repository

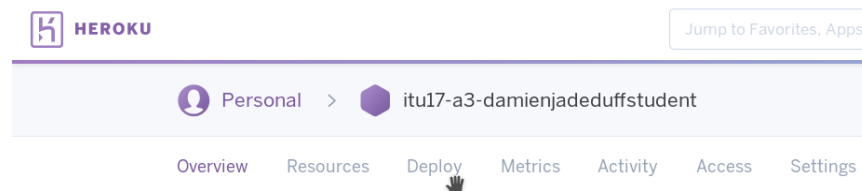


The following step should be attempted only after you have accepted your invitation to the ITUIS18 GitHub organisation (you should have been automatically invited to join the ITUIS18 organisation). If you cannot find the invitation email, the invitation should still be visible on the organisation homepage: <https://github.com/ituis18>

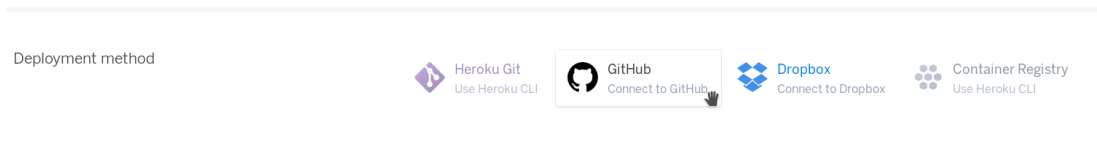
So far you have made a web app on Heroku. Now you need to deploy your code in your new GitHub repository to Heroku – and ensure that it gets redeployed whenever changes come into your GitHub repository! In order to see changes to your code reflected in your Heroku web application as further changes are made, you need to “connect” your Heroku application to GitHub.

Once you have done this, any changes made to your repository on GitHub (for example, by “pushing” new “commits”) will be deployed in a matter of seconds or minutes to Heroku.

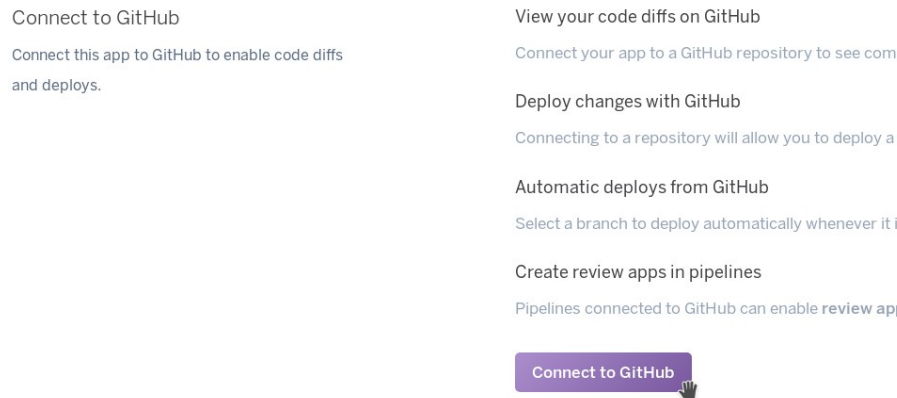
To connect, navigate to your repository’s dashboard (you can get there from the [Heroku home page](#)). Choose the “Deploy” tab as below (it should also open directly after you first create your web app):



Choose “GitHub” as your “deployment method” as below:



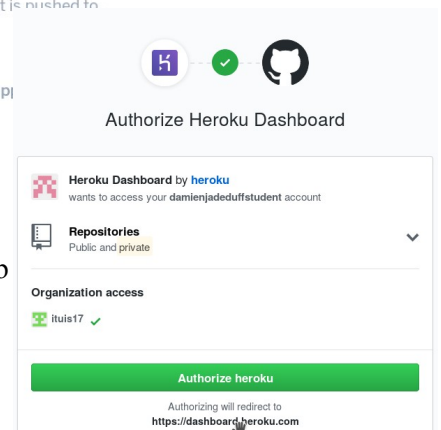
Click on “Connect to GitHub” as below:



A pop-up window should appear asking to authorize Heroku to access your GitHub account as shown on the right.

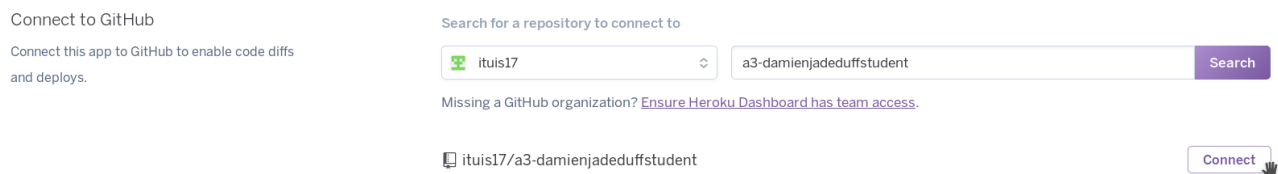
**Ensure that “Organization Access” is selected.**

Authorize Heroku to access your GitHub account by clicking “Authorize heroku”.



⚠ If you do not grant organisation access to the *ituis18* organisation, the following steps may not work. ⚠


Now choose “ituis18” from the dropdown list next to “Connect to GitHub”, type your assignment repository name, click “Search” to find it and click “Connect” next to the repository in the resulting search results list, similar to below:



Next, ensuring the branch selected for deployment is set to “master”, choose “Enable Automatic Deploys”:

Enable automatic deploys from GitHub


Every push to the branch you specify here will deploy a new version of this app. Deploys happen as soon as the branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

 master

☐ Wait for CI to pass before deploy  
Only enable this option if you have a Continuous Integration service configured on your repo.

**Enable Automatic Deploys**

Whenever you come to this Deploy tab from now on you should see a message that automatic deploys are enabled, similar to:

 Automatic deploys from  master are enabled

Now whenever you make a change to your repository on GitHub, the changes should be reflected in your Heroku application (use the “Activity” tab in Heroku if you want to see when Heroku brought changes from GitHub).

This one time, you will need to manually deploy your app from this dashboard (because you have connected your app to your GitHub repository any changes later pushed to your GitHub repository will automatically be deployed). So click on “Deploy Branch” next to “Manual Deploy”:


**Manual deploy**

Deploy the current state of a branch to this app.

**Deploy a GitHub branch**

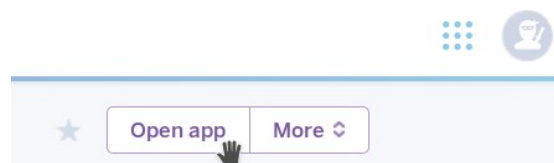
This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

 master

**Deploy Branch**

This could take up to a minute to complete and should display a progress indicator. Now, at any time you can open the Heroku running web application from the top right of the Heroku dashboard:



Or you can bookmark the resulting link. The link would be something like `https://YOURAPPNAME.herokuapp.com/`

**\* Deployments to Heroku from GitHub can take some time. Use the “Activity” tab to follow the process.**

### Link to your Heroku web application from your GitHub repository

Edit the README.md file in your GitHub repository and add the URL of your active Heroku web-site (i.e. `https://YOURAPPNAME.herokuapp.com/` or similar) (that is, the resulting website, not the dashboard) so that when someone clicks on it they will be able to see and interact with your website (note that users should be able to do this while not logged in to Heroku).

## Step 2

---

### Clone the repository to your computer

In order to run all of the rest of the following steps you need *Python 3* and *Bottle* installed on your computer. To install bottle on a Linux Mint or Ubuntu system, the simplest way is:

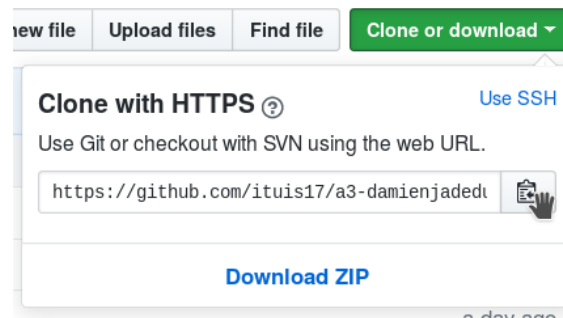
```
sudo apt-get install python3-bottle
```

Now you can run the assignment repository code on your own computer (that will come later).

As with previous assignments, you will clone the GitHub repository to your computer to edit it. With the command line, you would run:

```
git clone ADDRESS
```

Where ADDRESS can be found from your GitHub repository home page as shown below.



You can subsequently use *git* commands to commit to your repository and push the changes to GitHub (from where they will be automatically deployed to Heroku if you have successfully completed the previous steps above).

### Write a REPL program for saving and printing comments from the console

Now we are going to move away from web apps temporarily because we probably have not learnt the relevant technology in class to help us with writing them. Instead, we are going to work on the infrastructure that will enable us to create the web app we need.

In this step you will use your knowledge of functions like **input()** and **print()** as well as a **while** loops and **lists** to retrieve input from the user, append that input to a list, and print out all previously entered input. This means that your program will have a REPL (Read-Eval-Print-Loop) structure which, in *pseudocode*, is like this:


Do until quit:

1. get input from user
2. process that input
3. print the result

Sample output of the program might look like this (user input is in blue, the rest is generated by the program):

```
Enter your comment: I like this program.
Previously entered comments:
1. I like this program.
Enter your comment: I really do.
Previously entered comments:
1. I like this program.
2. I really do.
```

Add and commit the program to your assignment repository with the name **repl\_comments.py** and push to your GitHub repository.

 Ace programming tip: If you are getting an error after changing the code and you don't know how it happened, try reverting to the original code and introducing your changes one at a time till you find the problem. This is an essential “debugging” technique.

### Step 3

---

#### Hash your password

Check the computer program at the following link:

[https://bitbucket.org/damienjadeduff/hashing\\_example/raw/master/hash\\_password.py](https://bitbucket.org/damienjadeduff/hashing_example/raw/master/hash_password.py)

You are going to borrow that code for use in your own website, so do take a look and try and understand it. But first, you will use it. Download the *hash\_password.py* file, and run it using Python 3. The running program will ask you to enter a password and it will provide you the hash of that password. Enter that hash into your assignment *repl\_comments.py* program as a literal assigned to a variable. For example, if I want the hash of the password “TerriblePassword” I might run the program given and get the following output.

```
Please enter your password:TerriblePassword
The hash of that is 401f83e57ce50aaa8d46d44f0303e10133188f618d616f25a36d8d90e7b50ece
```

So I will enter somewhere into my *repl\_comments.py* program:

```
password_hash = '401f83e57ce50aaa8d46d44f0303e10133188f618d616f25a36d8d90e7b50ece'
```

#### Password protect your REPL program

Change the REPL program that you wrote in step 1 above so that it only allows text to be saved if the matching password is entered.

Sample output of the program might look like this (user input is in blue, the rest is generated by the program):

```
Enter your comment: I like this program.
Enter your password: GoodPassword
I a sorry I can't let you do that.
Enter your comment: I like this program.
Enter your password: TerriblePassword
Previously entered comments:
1. I like this program.
```

You will need to “borrow” some code from the example code provided in *hash\_password.py* so ensure to provide a link to the [original code](#) as a comment next to where the borrowed code is in your file.

Add and commit the program to your local assignment repository and push to your GitHub repository.

### Step 4

---

#### Run the web app from your computer.

In the provided repository was a template web application file, *bottle\_app.py*.

Using a terminal set the terminal's current directory to the repository directory and run it like this:



```
python3 bottle_app.py
```

It will print something a little like this:

```
Bottle v0.12.0 server starting up (using WSGIRefServer())...
Listening on http://127.0.0.1:8080/
Hit Ctrl-C to quit.
```

Open the address (e.g. `http://127.0.0.1:8080/`) in your browser.


You should see the following web-paged being served by Bottle:

```
This is going to be an awesome website, when it is finished.
```

To quit Bottle go back to the terminal in which you ran `python3` and press `<Ctrl-C>` (the Ctrl key together with the C key).

Try making a change to the source file `bottle_app.py` (e.g. change some of the text) then rerun the program and then go back to your browser and reload the address. The change should be visible.

**Note:** you can at any time see this same web application running in Heroku by using `git commit` and `git push` to send the files to GitHub, from where they will be sent to Heroku. However, it is usually more efficient to develop the code locally before worrying too much about internet deployment.

 **Bottle debugging tip:** When running your app, leave the log window open to see all output. There may be earlier errors that you are not seeing. If working with Heroku, redeploy if necessary to start from scratch (the app may have died at some earlier time and the error that you are seeing may just be because the app is already dead).

## Adapt your assignment 1 website to bottle

In the current assignment you are going to be making a *dynamic* website, one that changes the contents of pages over time. In particular, it will receive input from users and use that to build its content. Ultimately you will be able to obtain data from your website users in ways similar to that illustrated below.

Dear user, please input your data:

- ☐ Option 1
- ☐ Option 2

However, at this point you are going to taking the first step of adapting the web site you made in assignment 1 as a *bottle web application*. So, edit the file `bottle_app.py`, so that the web app behaves roughly the same as the pure HTML web-site you made for assignment 1. You do this by creating functions that generate HTML and assigning them to resources (routes) by using the `route()` function. Information about this is given in lectures or you can use the provided template as a starting point.

**Note:** to serve images, and to make it easier to serve plain HTML and CSS, it will be a lot easier for you if you use the bottle “static files” functionality. Entering the following code would mean that visits to addresses like <https://YOURAPPNAME.herokuapp.com/static/myfile.txt> would result in the browser displaying the file `myfile.txt` in the subdirectory `static`:

```
from bottle import static_file, route
def static_file_callback(filename):
    return static_file(filename, root='static')
route('/static/<filename>', 'GET', static_file_callback)
```

Ensure that the pages produced are valid HTML 5. You would not be expected to use HTML forms at this point; so skip those bits until next week.

Add and commit the program to your local assignment repository and push to your GitHub repository.

**FOR THIS ASSIGNMENT WE ARE USING PYTHON VERSION 3. Do not develop your code for any version less than 3.3.**

## Step 5

---

### Add comments to your website

Adapt your bottle app from step 4 so that users can enter comments in a text field. Make previously entered comments show up above or below the comment entry field. Add another text field for password and only allow users who enter the password correctly enter comments. *Use a hashed password so that the plain text password is not visible via your source-code on Github.*

You will need to add HTML forms to your web-pages to get the necessary input from the user.

You have already written code for saving and displaying comments, and checking for a password in step 3, but in this website version you will abandon the REPL approach and comments will be saved when users enter them into a text form and submit that to your website (so that the route and function that handles that submission is responsible for saving this information and also checking the password), and they will be displayed on the relevant web page (so that the route and function that handles displaying the webpage is responsible for also displaying the comments).

You do not need to worry about saving any data to file in this assignment. Everything can be saved to memory.

Add and commit the program to your local assignment repository and push to your GitHub repository.

**Note:** To ensure that your website is compliant with the HTML 5 standard, put the Heroku-served page addresses into the HTML validator at <https://html5.validator.nu/>.

## Step 6

---

⚠ **Finish by the due date and ensure your files are in the GitHub repository that was created when you accepted the assignment - your files will be automatically retrieved from GitHub.**

### Finalise information display & design

In this step, you will finalise the design of your website so that it matches your intentions as closely as possible. For example, maybe you would like to add a navigation bar.



*If you have any other features of your own invention that you would like to incorporate into your website, do so! Learning works best when you are working towards something you care about. For example, multiple users, comment rating and auto-hiding, login sessions so that users do not have to enter their password every time (difficult).*

Take particular attention also to the **evaluation criteria** to ensure you at least include functionality relevant to each criterion.

### Ensure it works on Heroku and on your computer with Python 3

The template application you got when you cloned the assignment repository containing `bottle_app.py`

should have worked with both Heroku and when run on your own computer. Ensure the website that you have created still does work with both Heroku and when run on your own computer without any problem.

△ Watch out: Ensure you are running using Python 3.3+ on your local machine; behaviour may be different between versions otherwise. You may suddenly get new errors on Heroku, which uses a particular version of Python 3 by default.

### Push to GitHub

Ensure you have committed all your changes and pushed those changes to the GitHub repository repository that was created when you clicked on the assignment link. These will be extracted automatically at the due time be marked and will be the subject of your demonstration session (which you should ensure to attend to get your marks).



***If you get the code and output pushed to the correct GitHub repository  
by the due date and time, it will be marked.***