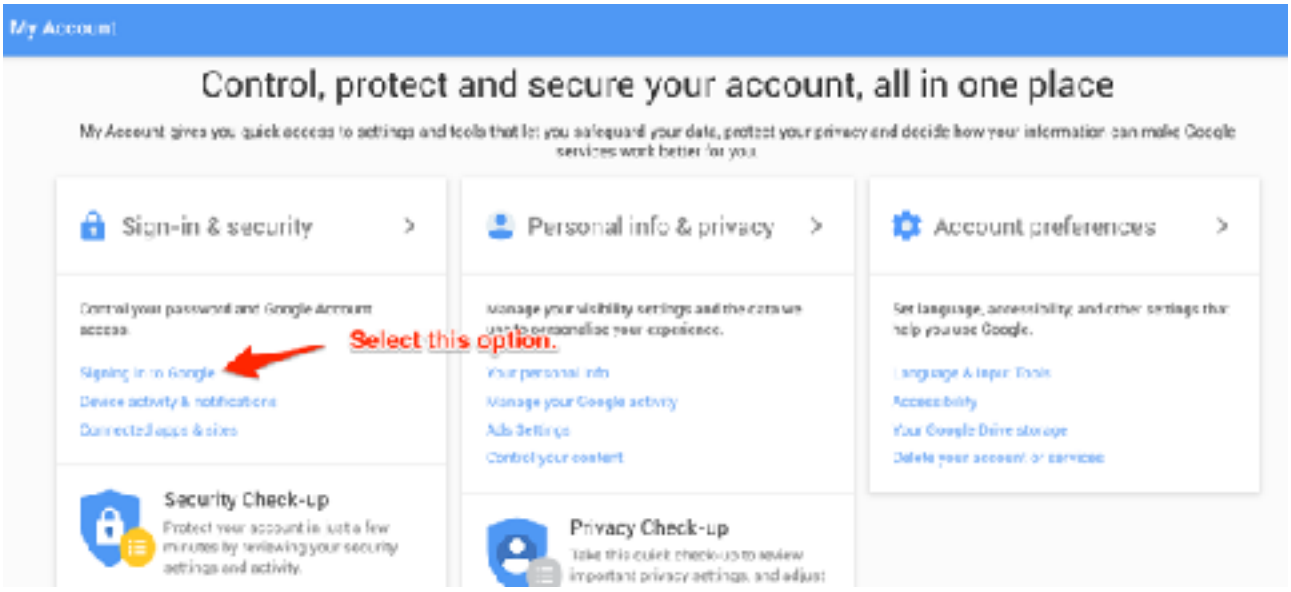
INFO3180 Lab 3

Due Date: **12 February, 2017**

Exercise 1 - Sending Mail with Gmail

Step 1 - Prepare an app password

Create a Gmail account if you don't already have one and login to your account. Once you are logged in go to https://myaccount.google.com/ to create an app password which will be used exclusively for sending email.



My Account

Control, protect and secure your account, all in one place

My Account gives you quick access to settings and tools that let you safeguard your data, protect your privacy and decide how your information can make Google services work better for you.

Control your password and Google Account access.

Manage your visibility settings and the data we use to personalise your experience.

Set language, accessibility and other settings that help you use Google.

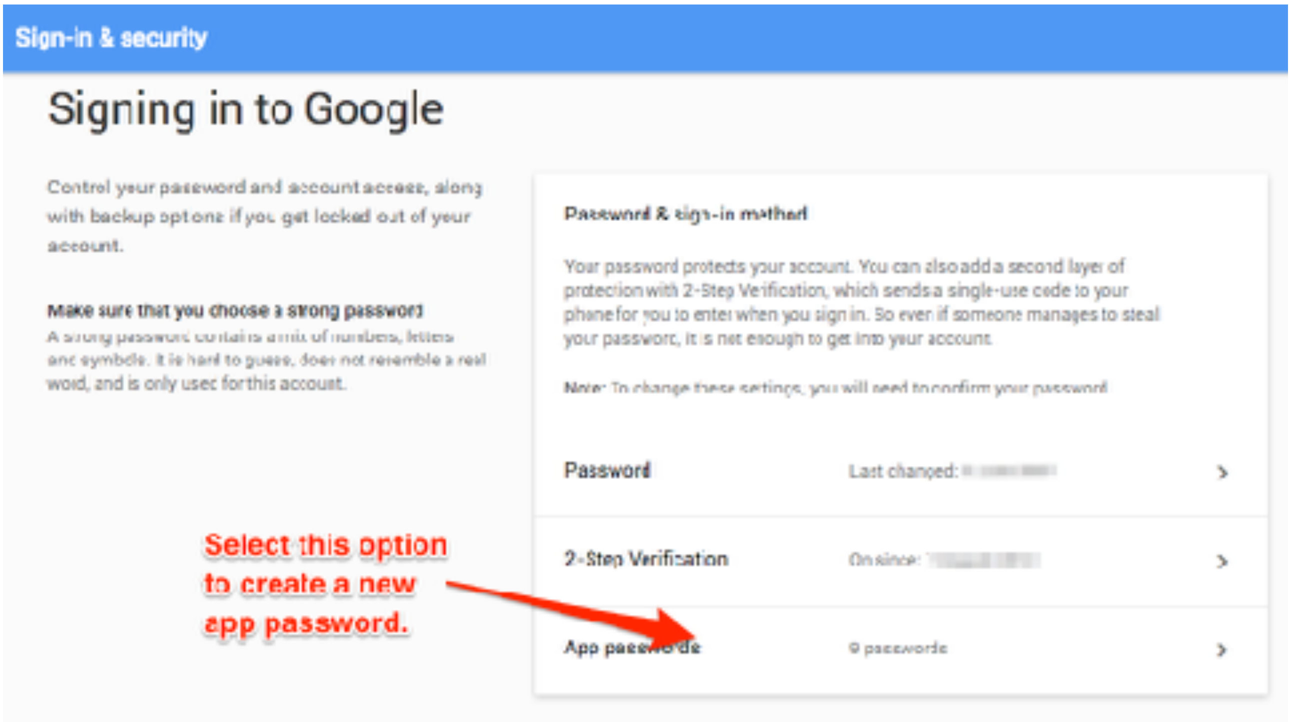
Select this option.

Security Check-up

Protect your account in just a few minutes by reviewing your security settings and activity.

Privacy Check-up

Take this quick check-up to review important privacy settings and adjust



Sign-in & security

Signing in to Google

Control your password and account access, along with backup options if you get locked out of your account.

Make sure that you choose a strong password

A strong password contains a mix of numbers, letters and symbols. It is hard to guess, does not resemble a real word, and is only used for this account.

Password & sign-in method

Your password protects your account. You can also add a second layer of protection with 2-Step Verification, which sends a single-use code to your phone for you to enter when you sign in. So even if someone manages to steal your password, it is not enough to get into your account.

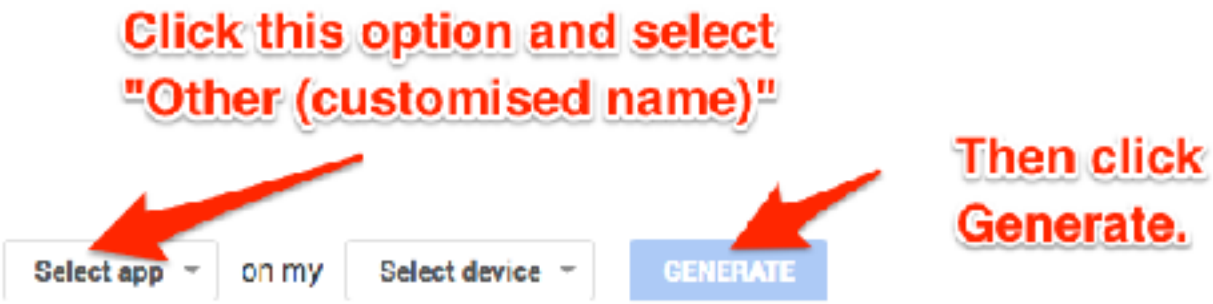
Note: To change these settings, you will need to confirm your password.

Password — Last changed: [redacted]

2-Step Verification — On since: [redacted]

App passwords — 9 passwords

Select this option to create a new app password.



Click this option and select "Other (customised name)"

Then click Generate.

You will then see a modal with a generated password that you can use in your application.

Step 2 - Create an email sending script

Edit the script below to successfully send an email. ***Hint:*** *There will be some errors that you need to fix.* 😉

```
import smtplib

from_addr = 'yourusername@gmail.com'

to_addr  = 'david@someemail.com'

message = """From: {} <{}>

To: {} <{}>
```

```
Subject: {}

{}
"""
message_to_send = message.format(from_name, from_addr, to_name,
                to_addr, subject, msg)

# Credentials (if needed)
username = 'yourusername@gmail.com'
password = '{youremailapppassword}'

# The actual mail send
server = smtplib.SMTP('smtp.gmail.com:587')
server.starttls()
server.login(username, password)
server.sendmail(from_addr, to_addr, message_to_send)
server.quit()
```

Call the script "`sendemail.py`". You can run the script by running the following at the command line:

```
python sendemail.py
```

Once you have it working add the script to your github account as a new repository. Name the repository "`info3180-lab3-ex1`". It should contain the working script.

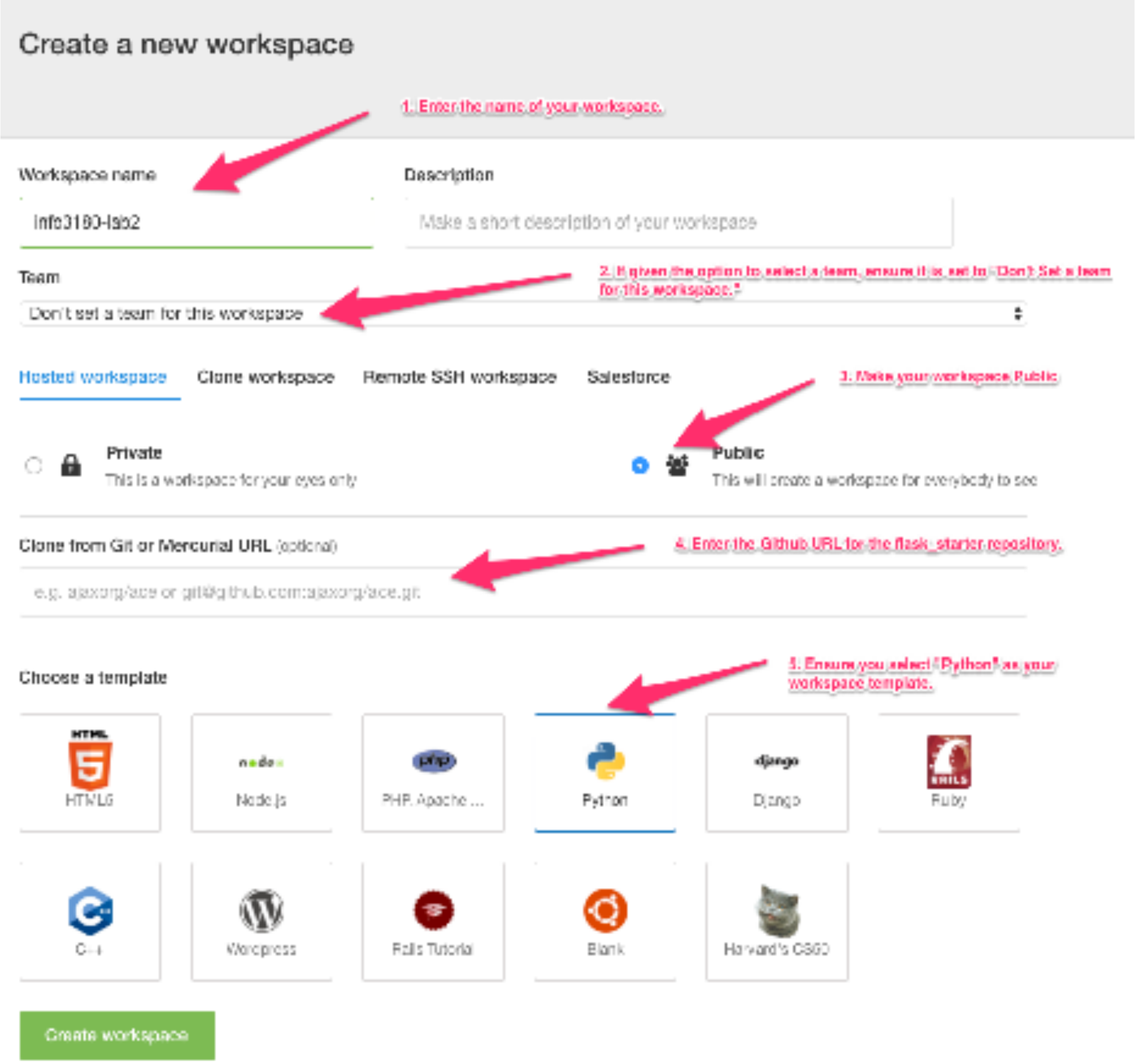
Note: Please remove your Gmail username and password before committing and pushing to your repository.

Exercise 2 - Creating a Email Sending Form

For this tutorial use the flask_starter repository as your starting point, you can get it here: https://github.com/uwi-info3180/flask_starter

Step 1 - clone the repository

If working with Cloud9 then do the following to clone the repository to the C9 workspace:



Create a new workspace

1. Enter the name of your workspace.

Workspace name: info3180-lab2

Description: Make a short description of your workspace

2. If given the option to select a team, ensure it is set to "Don't Set a team for this workspace."

Team: Don't set a team for this workspace

3. Make your workspace Public

○ Private – This is a workspace for your eyes only

● Public – This will create a workspace for everybody to see

4. Enter the Github URL for the flask_starter repository.

Clone from Git or Mercurial URL (optional): e.g. ajaxorg/ace or git@github.com:ajaxorg/ace.git

5. Ensure you select "Python" as your workspace template.

Choose a template

or if working on your own local computer then do the following:

```
git clone https://github.com/uwi-info3180/flask_starter info3180-lab3-ex2
```

Step 2 - Create your own info3180-lab3-ex2 repository in github

You can do this by removing the **`.git`** folder and initialising git again:

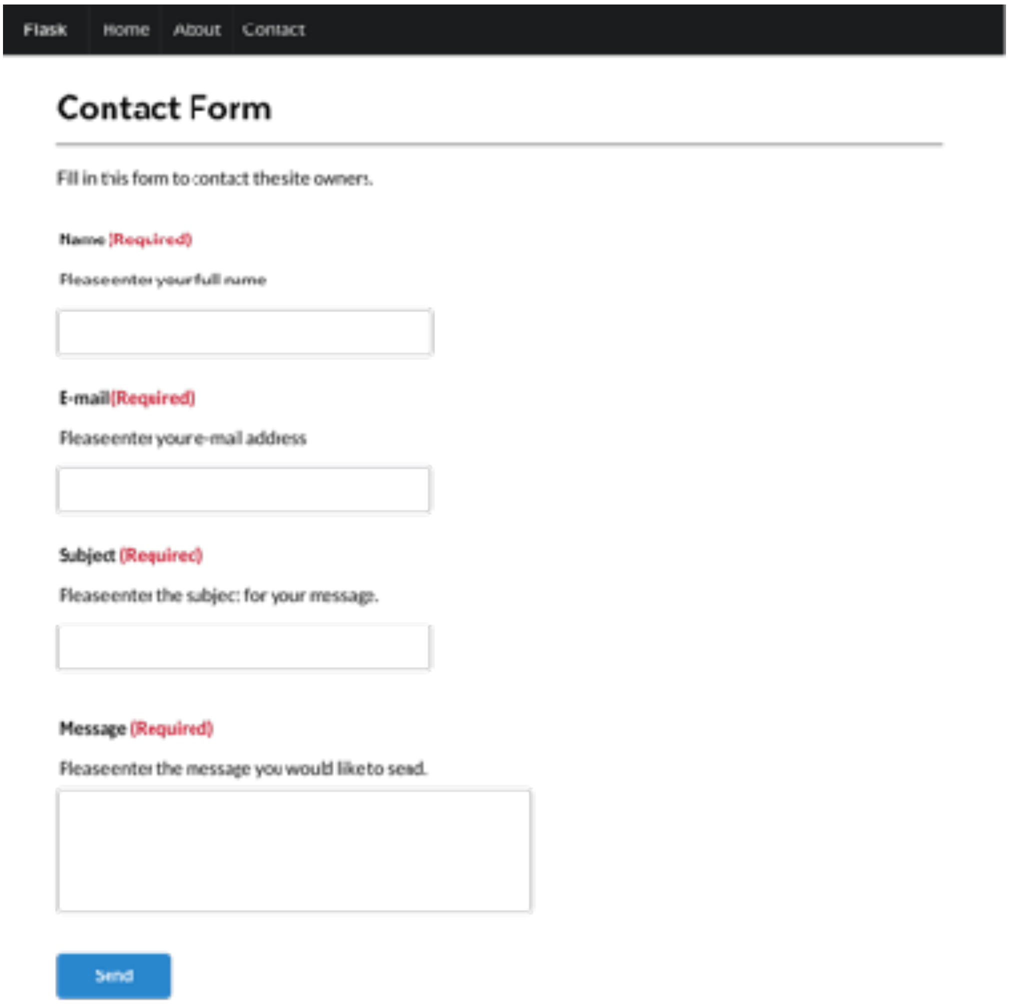
```
cd info3180-lab3-ex2 #not needed if you are working on Cloud9
rm -rf .git/
git init
git add .
git commit -am 'initial commit'
```

Then login to your Github account and visit http://github.com/new . Create a repository with the name "**`info3180-lab3-ex2`**". Ensure you don't create this repository with a **README** or any other files initially. Next add this new repository as a remote repository to your workspace and then push your code to it.

```
git remote add origin https://github.com/{yourusername}/info3180-
lab3-ex2.git
git push -u origin master
```

Step 3 - Create Contact form

Make your contact form available at the route **`"/contact"`** and name the associated method **`"contact()"`**. You should also name your template **`"contact.html"`**
It should look something like this:



Contact Form

Fill in this form to contact the site owners.

Name (Required)

Please enter your full name

E-mail (Required)

Please enter your e-mail address

Subject (Required)

Please enter the subject for your message.

Message (Required)

Please enter the message you would like to send.

Step 4 - Create a method that will send email

Create a method called **`send_email()`** that will send an email, based on what you know of the **`smtplib`** library that you used in Exercise 1. The method will use the input of the form to send messages. It will work something like this:

```
send_email(from_name, from_email, subject, msg)
```

You should use this **`send_email()`** method within your **`contact()`** method when a **POST** request is made. You can access then values your form data by using the global request object, for example **`request.form['field_name']`**. After you send the message, ensure you **redirect** the user to your home page and display a **flash** message letting them know their email was successfully sent.

Hint: Here's an example of how to use the **`request`** object to get form data and the **`flash()`** and **`redirect()`** methods.

http://flask.pocoo.org/docs/0.12/quickstart/#the-request-object
http://flask.pocoo.org/docs/0.12/patterns/flashing/

Note: Please remove your Gmail username and password before saving/ committing to your repository.

Submission

Submit your code via the "Lab 3 Submission" link on OurVLE. You are NOT required to push your code to Heroku. You should submit the following links:

1. Your Github repository URL for your Email sending script e.g. https://github.com/{yourusername}/info3180-lab3-ex1
2. Your Github repository URL for your Flask Exercise e.g. https://github.com/{yourusername}/info3180-lab3-ex2