İSTANBUL TECHNICAL UNIVERSITY
Department of Computer Engineering
BIL103E – Introduction to Information Systems – Fall 2017
Assignment 1

ASSIGNMENT 1

High-level Instructions

✻ See page 2 for step-by-step instructions. ✻

- STEP 0 (Week 3: 25 September approximately):
  - Create yourself a *GitHub* "repository" within our *GitHub* "classroom".
  - Share your username with us via *Ninova*.
- STEP 1 (Week 4: 2 October approximately):
  - Collect information for your website. The website will be a fan-website where you give information about a topic you love. Do not collect text verbatim. If your text is verbatim you must quote it and give a link or citation.
  - Clone your *GitHub* repository to your computer using *Git* and play with editing the files and *committing* them.
- STEP 2 (Week 5: 9 October approximately):
  - Create the (multiple, linked) *HTML* files for your website, so that you can view them on your local computer.
  - Using *Git*, *commit* these files and *push* this website to *GitHub* so that it can be viewed on the web.
- STEP 3 (Week 6: 16 October approximately):
  - Add *CSS* styles to your website to give it an advanced look.
  - Using *Git*, *commit* the website files and *push* them to *GitHub* so that it can be viewed on the web, and validate it with a validator.
- STEP 4 (Week 7: 23 **October 9:30am**):
  - Create a logo for your website from scratch using *Inkscape*. Incorporate it into the website so that it can be viewed from a browser.
  - Take an existing photo related to your topic and process it in *GIMP* to show off your *GIMP* skills. Put on the website **both** the original photo and the photo processed with *GIMP* for comparison.
  - *Push* this all to *GitHub* **by the due date and time** so that it can be viewed from the web.
    - Ensure the files are in the correct place so that your submission can be marked.

Submission Notes

- All steps of this assignment will be evaluated only after the final submission time (23 October 9:30am) has passed. However, it is recommended that you get the earlier steps done by their respective due dates.
- **By the final due date and time, you need to push your files to your GitHub classroom repository so that they can be downloaded by our markers.**
  - Feel free to check your fellow students' work and comment on them (GitHub provides facilities for interacting with owners of repositories).
- **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 of the assignment.
- Check the separate evaluation form to see on what basis your markers will be grading you.
  - Use all of the document & image processing techniques shown in the evaluation form.
  - If the lecture did not cover a particular technique listed on the evaluation form, it is your responsibility to learn how to do it.
- Have fun.

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

Step 0

Finish by approximately 25 September.

Create your GitHub Repository

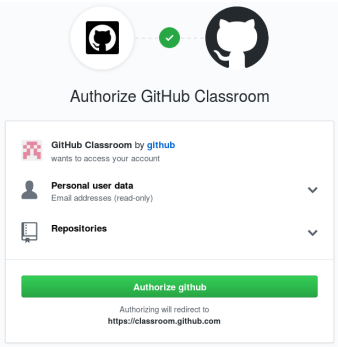
Click on this link:
https://classroom.github.com/a/fk8tNWGC

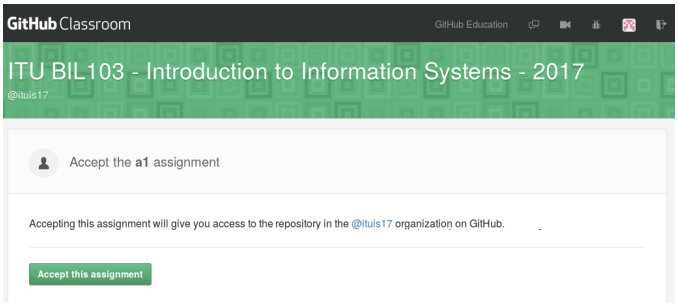
A GitHub "repository" (a place to keep your web and programming files) will be created on the GitHub website under the "ituis17" organisation. You will have access to the repository.

ⓘ

You may have to create a new GitHub account to do this. Even if you already have a GitHub account it is advised that you create a new one for this course so that you can later choose whether to show your assignments in your online portfolio.

In addition to needing to create a new GitHub account, you may need to authorise GitHub Classroom to access your account and accept the assignment – the dialogues may be similar to the screenshots below:





Inform us of your username

Create a text file containing **only the GitHub username for the account in which you created this repository.**

✻ The text file should contain only this information and nothing else. ✻

Upload it to Ninova. A Ninova assignment ("ödev") will be created for this purpose.

Step 1

Finish by approximately 2 October.

Collect information

Collect information for your website. The website will be a fan-website where you give information about a topic you love. The information will be in the form of text and pictures.

☞

DO NOT COLLECT TEXT VERBATIM. IF YOUR TEXT IS VERBATIM YOU MUST QUOTE IT AND GIVE A LINK OR CITATION.

(it is recommended not to use copy-paste at all and if you do, clearly state in your notes the text's origin so that you can quote it if you include it in your website)

Clone your repository

You need to "clone" (copy) your repository from GitHub to your own computer so that you can edit the files in it there. When you clone a repository, you copy all the files in it and you also have the ability to later "push" the files that you "add" and "commit" back to the repository you cloned it from.

In order to do this on Linux (for other systems, the approach may differ), the procedure is as follows:

1. You need to go to your repository page. So for example, if my username is **damienjadeduff**, I would go to:

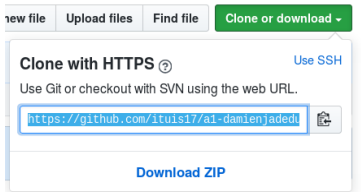
   `https://github.com/ituis17/a1-damienjadeduff`

   Conversely, if my username were **uyar**, I would go to:

   `https://github.com/ituis17/a1-uyar`

2. There you will be able to find a link to give to Git by looking for the "Clone or download" button:

3. Clicking that button will reveal a link which you can copy (the icon on the right will copy it your clipboard):



✻ Don't click "Download ZIP" - you need to "clone" your repository so you can later "push" to it. ✻

4. You will now use the "git clone" command in a terminal to clone the repository from that link. Open up a terminal console (for example, by clicking on the programs menu and searching for "terminal" and clicking to run the program that comes up – e.g. *xfce4-terminal*). If the link that you retrieved were **https://github.com/ituis17/a1-damienjadeduff.git** then you would type into the terminal the following command:

   ```
   git clone https://github.com/ituis17/a1-damienjadeduff.git
   ```

   Similarly, if the link that you retrieved were **https://github.com/ituis17/a1-uyar.git** then you would type into the terminal the following command:

   ```
   git clone https://github.com/ituis17/a1-uyar.git
   ```

   Git will now create a directory called **a1-yourusername** where you "yourusername" would be replaced with your GitHub username. For example, in the above case, it would create a directory called **a1-damienjadeduff** or **a1-uyar**. In that directory is just one file called "index.html".

✯

*You can move this directory into other directories on your computer's file system without breaking anything. All the configuration files relating to the repository are stored in a hidden folder called "***.git***".*

Editing a file and committing it with Git

Now you will make some edits to files in your repository on your computer and save them using Git.

✯

In this section, you will be saving ("committing") your files using Git only on your computer.

In a later section we will look at "pushing" your changes to GitHub.

Git is a program for storing files related to programs, websites, etc. It is mostly for storing text files and "code" files. It has lots of features that make it useful for this purpose (for keeping a full history, helping to collaborate with others, etc.). In this assignment we will not be using most of these features. But we still need to know how to save files with git ("commit" them in the terminology used by Git). This goes beyond simply saving a file to disk but we can think of it as a way of saving the file to a Git repository. You need to "commit" files to the git repository so Git can work with them. To do this it is best we have some theory.

In Git, a files can be copied to three places:

1. In the "**working directory**" - this is the normal file system – a file as you know it in a directory – nothing special.

2. In the Git "**staging area**" - the file is ready to be saved to the Git history. Git also knows which subdirectory the file was in when it was a normal file. To get a normal file from the working directory into the staging area you need to use the "git add" command. Git add will not change the working directory; it is internal to Git.
3. In a Git "**commit**" - the file has been saved to the Git history. Git also knows which subdirectory the file was in when it was a normal file. To get a file from the staging area to a commit you need the "git commit" command.

✻ If you make a change to a file in the working directory the version in the staging area or commit will not change. You need to go through the process of adding and committing the file again to save the changes. ✻

The normal process when saving a file to Git is to "add" a file (or multiple files) to the staging area and then to "commit" it (or them). Then it would be saved in the Git repository.

Let's do this:

1. Open up the file **index.html** with a text editor (e.g. *Geany*). Change some things (e.g. add some information that will eventually go on your website). Save it back to disk using your text editor. The file has now been changed in your working directory. We want to save it to the Git repository so:
2. Open a terminal and change the current directory to the directory of your repository. E.g. if my repository were in my home directory and my GitHub username were **damienjadeduff**, it would be something like:

   ```
   cd ~/a1-damienjadeduff
   ```

   Similarly if my repository were in the directory **~/Desktop** and my GitHub username were **uyar**, it would be something like:

   ```
   cd ~/Desktop/a1-uyar
   ```
3. Add the file to the repository's staging area by typing:

   ```
   git add index.html
   ```
4. Commit the file by typing:

   ```
   git commit -m "More topic information added to index.html"
   ```

Note "`-m`" tells Git that the next thing on the command line is a message for keeping track of the reason the file changed. The message needs to be surrounded by double quote-marks (`"`). Making meaningful commit messages is useful when you use advanced Git facilities like viewing file history.

ⓘ

You can add multiple files to the staging area before making a commit.
This is usually what we do when a change is made that affects multiple files.

Adding a new a file and committing it with Git

The process for adding a new file to your Git repository is the same as changing a file. Let us say you used a text editor to create a new file called "**a_list_of_things_i_like.txt**" and saved it in your repository's working directory. Now to commit that file to Git you need to open a terminal, change the directory to your repository directory, and type:

```
git add a_list_of_things_i_like.txt
git commit -m "Added a list of things I like"
```

Try adding some more files to your git repository related to the topic on which you will build a website.

ⓘ

Try typing the command
`git status`
to see information about which files have been changed in the working directory or are in the staging area ready to be committed.

Pushing your files back to GitHub.

In order to make your files available to the instructors (and ultimately to the markers) as well as to make your website visible from the internet, you are going to have to learn to "push" the repository to GitHub. When you "push" your repository to GitHub, the whole repository is sent to GitHub. To do this, open a terminal, change the directory to your repository directory (doing this is discussed in the previous section), and type:

```
git push
```

You may need to enter your GitHub username and password when prompted.

Step 2

Finish by approximately 9 October.

Create the HTML files for your website

Using the browser, text editor and file manager of your choice, create the HTML files for your fan website and commit them to your repository.

Important points:

➤ This can be done on any operating system with any browser but the website will be viewed by the markers using *Firefox* on *Linux Mint 18 XFCE*.

➤ The work of creating the website should be done on your own computer, **not** on *GitHub*. The website should be designed so that it can be viewed both on your computer ("locally") and via *GitHub*. So use "relative links".

➤ The repository you create should contain at least a file called `index.html`, which is the home page for your website (this file was provided when you first cloned the repository).

➤ Make sure to create multiple HTML files with links between them. When you click on a link, the correct file should be loaded.

➤ You can view your website while it is on your computer ("locally") by loading it into your browser.

For this purpose you can use the "File... Open File..." capability of your browser, navigating to the `index.html` file and selecting it.

➤ Check the evaluation criteria for this assignment so that you know at least the minimum HTML capabilities that you need to use for this assignment, and be prepared to explain to the demonstrator where you used these in your website.

➤ Extra information about HTML and websites can be found from the tutorial website http://w3schools.com/.

➤ The following textbook is the suggested reference. The section "HTML Basics" is relevant to this step:

http://interactivepython.org/runestone/static/webfundamentals/index.html

➤ The website should be written in valid HTML according to the "HTML 5" specification.

Upload to GitHub and GitHub pages

Using the method described in Step 1, commit your changed and new files to your repository, and push them to your GitHub repository. We have enabled a capability of GitHub on your GitHub repository called "GitHub Pages" that will allow anyone on the internet to view the website in your repository from a browser if they know the address of the website.

If your GitHub username is **damienjadeduff** then you can find the website being served from your repository at the address:

https://ituis17.github.io/a1-damienjadeduff/

If your GitHub username is **uyar** then you can find the website being served from your repository at the address:

https://ituis17.github.io/a1-uyar/

And so on.

Validate your website

Navigate to https://html5.validator.nu/ and in the "Address" textbox section type in the address to your website as above.

Click "Check". If there are any errors, fix them on your computer, add the files that were changed to the Git staging area, commit them and push the repository.

Step 3

Finish by approximately 16 October.

Use the knowledge you have learned from lessons or from http://w3schools.com/ to style your web site to make it more visually appealing. Use *external* stylesheets (`.css` files and `<link rel ...>` tags).

As described in the previous step, push the result to GitHub and validate the HTML as HTML 5 using https://html5.validator.nu/.

Further points:

➤ In the Fundamentals of Web Programming textbook, "Cascading Style Sheets" is the relevant section:

http://interactivepython.org/runestone/static/webfundamentals/index.html

➤ The website should be written in valid HTML according to the "HTML 5" specification (so do ensure you validate it).

➤ Check the evaluation criteria for this assignment so that you know at least the minimum CSS capabilities that you need to use for this assignment, and be prepared to explain to the demonstrator where you used these in your website.

Step 4

Finish by 23 October 9:30am.

1. Use the *Inkscape* program to create a logo for your website. Ensure that you check the evaluation criteria document so that you use at least the capabilities of *Inkscape* described there.
2. Add the logo so that it is visible on the website.
3. Download a photo related to your website and process it with *GIMP*. You may choose the functions of *GIMP* that you use to process the photo – of course, try to make it relevant to the theme of the website. Ensure that you check the evaluation criteria document so that you use at least the capabilities of *GIMP* described there.
4. Add both the original photo and the processed photo so that they are visible somewhere on your website.
5. Again commit all the files, and push the website to GitHub and validate it.

If you get the website pushed to GitHub in the correct location by the due date and time, it will be automatically saved for marking.