This assignment describes setting up a Github repository, the place in which all of your assignments will be turned in.

Github

If you have had me before, and this seems **TL;DR** make sure you:

- Scroll past the Git intro and view:
  - Make sure you create your repo correctly with the proper `README.md`
  - Remember how to use Markdown
  - Rules for organizing your repo (pretty much the same)
  - README files in general
  - General expectations for all assignments turned in

In addition to our team chat, we will be using `version control` software to organize all of our code. Don't panic. Please trust me when I say you will be better for using some kind of version control. In this case we will use `git` (the version control software) in conjunction with `Github` a web site that makes your programs publicly viewable. There is a bit of a learning curve, but you will thank me for it.

What is Git?

Don't let the following definitions scare you. We will use `git` and `Github` at a very basic (simple) level.

However lets define it:

> `Git` is a distributed revision control and source code management (SCM) system with an emphasis on speed,data integrity,and support for distributed, non-linear workflows. Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.
>
> As with most other distributed revision control systems, and unlike most client–server systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server. Like the Linux kernel, Git is free software distributed under the terms of the GNU General Public License [1].

So, `Git` is **distributed revision control system** (version control). This is just a fancy way of letting us manage our code. In fact, I really only want to use git so you can `push` our code to a repository that I can access so I can download and run your code. `Github` is such a place.

> `Github` is a social site that allows programmers to share code with other programmers. It's also a great place to discover projects to work on, discover code to use in your own projects, and a great

> place to start a portfolio. Whereas `Git` is simply the revision control system that can be installed anywhere, and only used locally if that's the users choice.

Remember though: `Git != Github`, Git does version control, but has commands built in to `push` your code to a repository on another computer. This other computer that we will use is `Github`.

Why Github for this course?

I use Github as a home-base for all assignments, lectures, and resources. It's basically our courses homepage. That is why I use it. When you push your code to Github, you are placing your code in a central location irregardless of what computer your working on. If you do some work in the lab, you can `push` your changes to Github. If you go home, you can `pull` your changes to your home machine and continue working. If you only work on a single machine, like a laptop, then Github can be your "backup" copy.

Create a Github account.

- You shouldn't need this tutorial, but just in case:
    - https://www.wikihow.com/Create-an-Account-on-Github
- When you create a Github account, you must choose a `username`.
- This is very important to remember, because you will use this as part of completing the form in the next assignment so I know where to find all your assignments.
- Create a repository with a name that starts with your course number. For example:
    - `2143-OOP`
    - `3013-Algorithms`
    - `5243-Advanced-Algorithms`
- Check the box that says: "Add a README.md file"
- Providing me with the repo name is important (next assignment) so I can pull all of your assignments to my computer for grading.

Edit the README.md

- A readme file is a file that gets displayed when you are viewing a folder on Github. So we place a README.md in the repo's folder so we can display information about the repo. Likewise, any README.md in a folder is used to display information about the folder it is in.
- Edit the readme file on Github and place your contact information inside along with a picture of YOU. NOT an avatar. NOT a thumbnail. But an easily identifiable picture of you.
- Your readme should include:
    - Your first and last name
    - Your email address
    - Your website (if you have one)
    - Your image
    - Slack / Github avatar

Example README.md

If you're not familiar with `markdown` you can go here to get an idea of what it can do for you. It's basically a simple syntax that you can use to format files. Use the template below to get your `README` started. The example might not reflect your exact course, but just replace relevant parts with your own course info.

Markdown Input

```
## Albert Einstein

#### Repo: 4883 Programming Techniques

#### Email: albert.einstein@yahoo.com

#### Website: https://en.wikipedia.org/wiki/Albert_Einstein

#### Image:

<img src="https://images2.imgbox.com/b3/d5/09sLKh1N_o.png" width="200">

#### Avatar:

<img src="https://images2.imgbox.com/ae/bd/tERTAGTL_o.png" width="75">
```

Markdown Output

Markdown Output

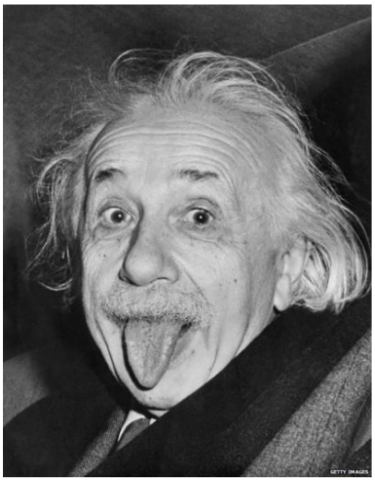
Albert Einstein

Repo: **2143 Object Oriented Programming**

Email: albert.einstein@yahoo.com

Website: https://en.wikipedia.org/wiki/Albert_Einstein

Image:



Avatar:

Organizing Your Repo

- Create an `Assignments` folder inside your Github repository
- In the `Assignments` folder, create a `README.md` file that has the following in it:

```
## Assignments

|  #  | Folder Link | Assignment Description |
| :-: | ----------- | ---------------------- |
|  0  | link 0      | description 0          |
```

- This will be a table in the root of your folder that will eventually hold a link to each assignment uploaded to Github.
- Your table will initially look very worthless with this example filler data for a single row:

#	Folder Link	Assignment Description
0	link 0	description 0

But then it will improve as you add rows to the table which link to each of your projects. I will link to some of the resources in this repo as an example:

Markdown

```
|  #  | Folder Link                              | Assignment Description
|
| :-: | ---------------------------------------- | ----------------------
-------------------------- |
|  1  | [A01](./FakeAssignments/A01/README.md) | [Show up to class ]
(./FakeAssignments/A01/README.md) |
|  2  | [A02](./FakeAssignments/A02/README.md) | [Listen in class ]
(./FakeAssignments/A02/README.md)  |
|  3  | [P01](./FakeAssignments/P01/README.md) | [Hello World ]
(./FakeAssignments/P01/README.md)      |
|  1  | [P02](./FakeAssignments/P02/README.md) | [Hello Mars ]
(./FakeAssignments/P02/README.md)       |
```

Resulting Table

#	Folder Link	Assignment Description
1	A01	Show up to class
2	A02	Listen in class
3	P01	Hello World
1	P02	Hello Mars

- Eventually you will create a directory structure that should look structurally similar to the example below (names will be different).
- But this is the structure that I expect your `Assignments` folder to follow. Any files outside of the `Assignments` folder are not any of my concern and you can organize as you see fit.

Example Repo Directory Structure

```
.
└── 5143-Opsys-xxx                (main repo folder)
    ├── Assignments                  (assignments folder)
    │   ├── A01                      (single assignment folder)
    │   │   ├── README.md            (assignment readme)
    │   │   ├── code_file_1.cpp      (assignment files)
```

```
      |    |    ├── code_file_2.cpp
      |    |    └── code_file_x.hpp
      |    ├── A02
      |    |    ├── README.md
      |    |    ├── code_file_1.cpp
      |    |    ├── code_file_2.h
      |    |    └── code_file_x.cpp
      |    ├── AXX
      |    |    ├── README.md
      |    |    ├── code_file_1.cpp
      |    |    ├── code_file_2.hpp
      |    |    └── code_file_x.cpp
      |    ├── P01
      |    |    ├── README.md
      |    |    ├── code_file_1.cpp
      |    |    ├── code_file_2.cpp
      |    |    └── code_file_x.cpp
      |    ├── P02
      |    |    ├── README.md
      |    |    ├── code_file_1.cpp
      |    |    ├── code_file_2.cpp
      |    |    └── code_file_x.cpp
      |    ├── PXX
      |    |    ├── README.md
      |    |    ├── code_file_1.cpp
      |    |    ├── code_file_2.cpp
      |    |    └── code_file_x.cpp
      |    └── README.md               (this is the assignments folder
readme)
      └── README.md                    (this is your main repo readme)
```

README's In General

Every single folder on your repo should contain a `README.md`. This is because when I or someone else looks at your repo, and clicks on a folder, they will see not just a directory listing, but an explanation and description of what that folder contains. I'm not making you create README's to be an ass ... I promise. It simply adds credibility to your repo, and makes you look like you know what your doing (and care and visitors to your site). I understand that a lot (if not most) probably don't want anyone looking at your repo, but you should still put your best foot forward. When I browse repos on Github, I not only rely on the README files for information, but find myself judging the software or library based on how organized and descriptive the README's are. And after talking to other developers, I found that I'm not the only one.

README's For Assignmentss

More HERE

We will use the README's that you will create in every assignment folder to do the following:

- Top of the README:
    - Project Title
    - Your name
    - Project Description (It needs to be "descriptive". You can take text straight from the assignment / requirements with a little additions here and there.)
- After description:
    - List all the files and or folders in that current project in a tabular format with a small description of each file.
    - The filename in that table should link to the actual code file or folder.
- After table:
    - Instructions
        - Any and all information needed to run your code.
        - These instructions will be used for others in the class to run your project on their machine. I PROMISE: you will learn very fast how not to assume things just work. Because they DON'T. When your code doesn't run on anyone else's system and you get a zero, I bet you start to let "assumptions" go out the window.

Common Errors

- Common Errors / Reasons for projects not to run (we can grow a similar list as the class goes on to help everyone get past common pitfalls):
    - Local libraries that you installed and not everyone else did the same.
    - Accessing files or fonts using local paths that will not be the same on running on someone else's machine (this one is annoying).
        - Example:
            - Your path: `C:\my_documents\pythonProjects\PyCharm\P01\Helvetica.ttf`
            - My Path: `/User/griffin/Projects/P01/Helvetica.ttf`
            - We both have P01 folders looking for `Helvetica.ttf`, but the whole path is different. We will also discuss this in class (relative vs absolute paths).
    - Hard-coding names of external resources that you do not provide:
        - Example: You called your data file **"data.txt"** and use: `open("data.txt")` I called my file **"data.dat"** and now your program crashes.
        - Solution: Read in the filename when the user runs your program, so you get the proper local name (I will discuss in class argv or command line params)

Example Assignment README

```
## P02 - Bouncy Balls
### Sally Smith
### Description:

Bacon ipsum dolor amet pork loin kielbasa pork, drumstick leberkas shankle
strip steak fatback beef ribs ham hock.
Pork short ribs doner andouille cupim pastrami picanha landjaeger pig.
Salami swine capicola spare ribs boudin
leberkas. Capicola ground round pork meatloaf, ham jowl swine prosciutto
```

```
bacon alcatra pancetta burgdoggen pig
spare ribs leberkas. Shankle fatback ground round, porchetta frankfurter
cupim venison strip steak pig meatball
biltong filet mignon. Shoulder chislic ground round pork burgdoggen
hamburger.

### Files

|   #   | File             | Description
|
| :---: | ---------------- | ----------------------------------------------
----- |
|   1   | Main.cpp         | Main driver of my project that launches game.
|
|   2   | HelperClass.cpp  | Helper class that holds movement functions
|
|   3   | TextureClass.cpp | Helper class that assists with textures and
images |

### Instructions

- Make sure you install library `blahblah.cpp`
- My program expects two parameters to be placed on the command line when
you run the program.
- Parameters `<input file> <number of players>`
- The input file should be formatted with a players name and age on a
seperate line:

+------------+
| name1 age1 |
| name2 age2 |
| etc...     |
+------------+

- Example Command:
    - `python <code> <input file> <num players>`
    - `python main.cpp input.txt 3`
```

The above markdown will produce a README that looks like:

P02 - Bouncy Balls

Sally Smith

Description:

Bacon ipsum dolor amet pork loin kielbasa pork, drumstick leberkas shankle strip steak fatback beef ribs ham hock. Pork short ribs doner andouille cupim pastrami picanha landjaeger pig. Salami swine capicola spare ribs boudin leberkas. Capicola ground round pork meatloaf, ham jowl swine prosciutto bacon alcatra pancetta burgdoggen pig spare ribs leberkas. Shankle fatback ground round, porchetta frankfurter cupim venison strip steak pig meatball biltong filet mignon. Shoulder chislic ground round pork burgdoggen hamburger.

Files

#	File	Description
1	Main.py	Main driver of my project that launches game.
2	HelperClass.py	Helper class that holds movement functions
3	TextureClass.py	Helper class that assists with textures and images

Instructions

- Make sure you install library `blahblah.py`
- My program expects two parameters to be placed on the command line when you run the program.
- Parameters `<input file> <number of players>`
- The input file should be formatted with a players name and age on a seperate line:

```
+------------+
| name1 age1 |
| name2 age2 |
| etc...     |
+------------+
```

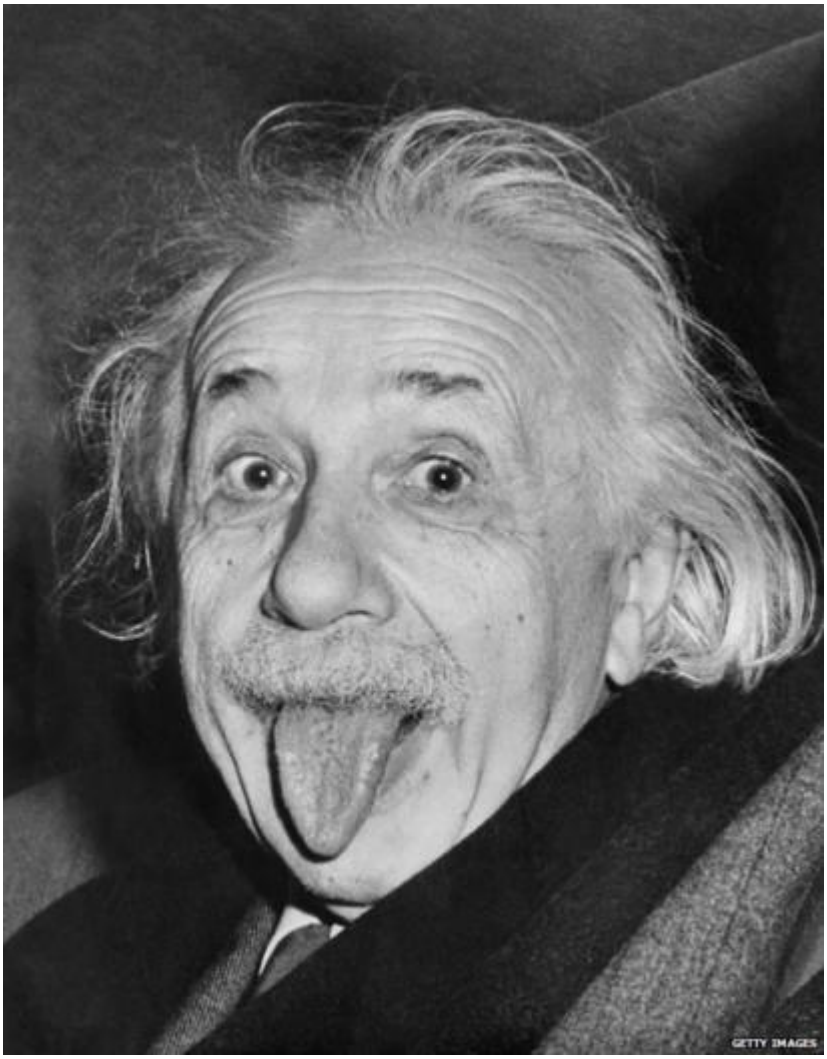
- Example Command:
  - `python <code> <input file> <num players>`
  - `python main.py input.txt 3`

Displaying an Image

The simplest way to add an image to your `README.md` file is to place said image in the same place as the README.md file. If you do this, then you can simply link to it in one of two ways. Let's assume my image is called: `MugShot.png` then I can do the following:

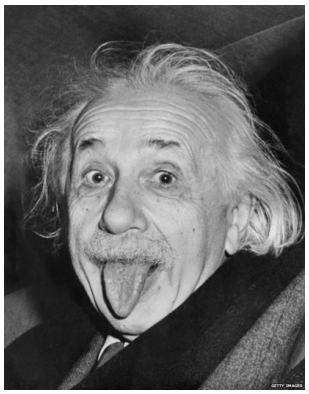
1. `![Alt Txt](https://images2.imgbox.com/8d/7c/e6DsilWD_o.png)`

1. `![Alt Txt](https://images2.imgbox.com/8d/7c/e6DsilWD_o.png)`



Original Size

2. `<img src="https://images2.imgbox.com/8d/7c/e6DsilWD_o.png" width="150">`



Width Set to 150px

These both work, with the second option giving you a little more control on sizing the image.

The image below was uploaded to a free service called img box (https://imgbox.com/) which allows me to keep all my images in one place and not have to place them in my repository's. It's really just ones preference in how or where they keep their images.

3. `<img src="https://images2.imgbox.com/69/ed/bUCkBWpD_o.png" width=200>`