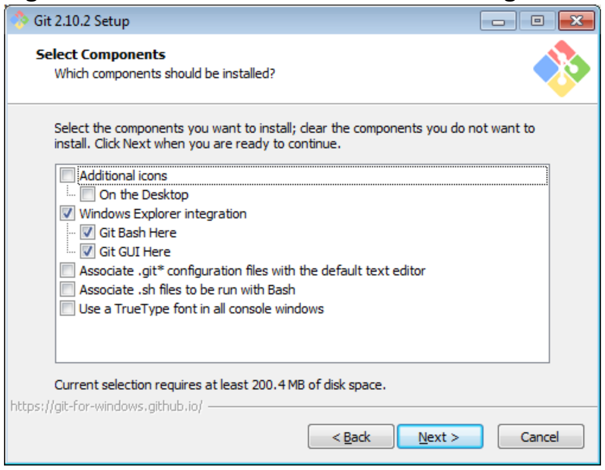
Software Install Instructions

Before we begin, there are some pieces of software you will need installed on your computer. The detailed instructions below should help you with that. After you install the tools, you might want to play with your new tools a bit, there are some example programs at https://wiki.python.org/moin/SimplePrograms.

1. Install Anaconda from https://www.continuum.io/downloads. This includes python as well as various libraries and tools we will use this semester. This package is made for scientific and data intensive computing.
    a. You will need the full Anaconda package, not just Miniconda.
    b. Please use the Python 3.5 version.
    c. I have been using the 64 bit version.
    d. In the install dialogs,
        i. You may install for 'Just Me' or All users. If you install for all users, it will place the files in c:\Program Files. If you install for 'Just Me' it will install in your user profile. If you choose for all users, when you run updates to the install, you will need to run the program as an administrator.
        ii. I recommend you 'Add Anaconda to my PATH environment variable.' This will allow you do stuff from a command line without having to browse to the install folder.
        iii. I also recommend checking 'Register Anaconda as my default Python 3.5' This will probably be your only python 3.5 interpreter.
2. Update Anaconda. Some of the stuff we will use does not work unless an update is run.
    a. If you installed Anaconda for all users, you must open Windows Power Shell as an administrator in the next step.
    b. Open a Windows PowerShell by going to the start menu and searching for 'PowerShell.' Click on the icon to open it. If you installed Anaconda for all users, you will need to right click on the icon and choose 'Run as Administrator.'
    c. If you choose to 'Add Anaconda to my PATH environment variable,' you may skip the next step.
    d. Change directory the Anaconda install folder. (type 'cd ..' to move up a directory, and 'cd' DirectoryName to navigate into a directory.
    e. Run conda update by typing 'conda update –all' If you did not add Anaconda to the path, you will need to type '.\conda update –all' from the anaconda install directory.
    f. If you get an error, you will need to first download and install C++ build tools from Microsoft. These are at http://landinghub.visualstudio.com/visual-cpp-build-tools.
3. Install PyCharm. This is an integrated development environment (IDE) for python. This is a full featured IDE with debugging tools, editing tools, environment management and much more. It also integrates Jupyter which is a great tool for plotting and working with data and mathematics. We can run python code from a Jupyter notebook so we can keep all of our work organized and ready to present.
    a. Browse to https://www.jetbrains.com/pycharm/download/ for the download.
    b. Download and install the package.
    c. I would create the desktop shortcut and create associations to ".py"
4. Install Git on your computer. Git is a software version control tool. It keeps track of software development history and helps enable quality control and good workflows. It works well for small or large projects as well as in a single developer or multi developer environment. We will use in this course to enable good collaboration, manage assignments, and test your work.
    a. Git Go to https://git-scm.com/downloads and choose the appropriate operating system.
    b. For Windows, click on the windows icon. Your download should begin automatically.

i. In the dialog, I would select 'Windows Explorer integration.' This will allow you to open a git terminal in a folder rather than having to do a bunch of 'cd' commands.


Git 2.10.2 Setup

Select Components
Which components should be installed?

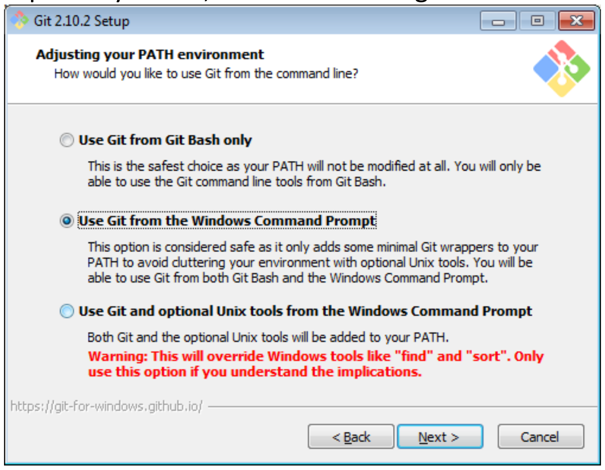
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Additional icons
  - [ ] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [ ] Associate .git* configuration files with the default text editor
- [ ] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows

Current selection requires at least 200.4 MB of disk space.

https://git-for-windows.github.io/

ii. Adding git to your path really helps as your terminal directory can be the code or repository folder, but still execute git commands.


Git 2.10.2 Setup

Adjusting your PATH environment
How would you like to use Git from the command line?

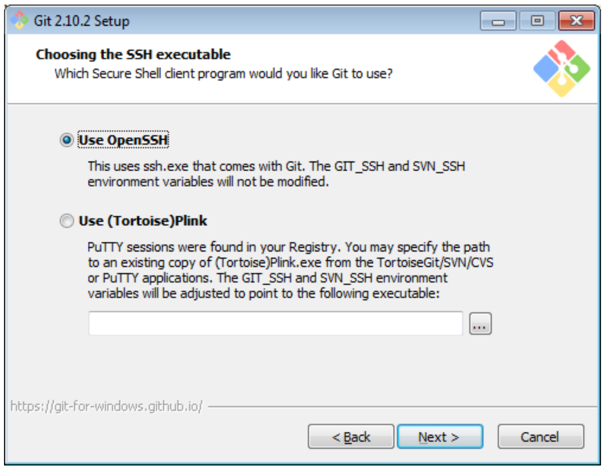
○ **Use Git from Git Bash only**
This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ **Use Git from the Windows Command Prompt**]
This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

○ **Use Git and optional Unix tools from the Windows Command Prompt**
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://git-for-windows.github.io/

iii. I would choose the ssh that is included with Git.


Git 2.10.2 Setup

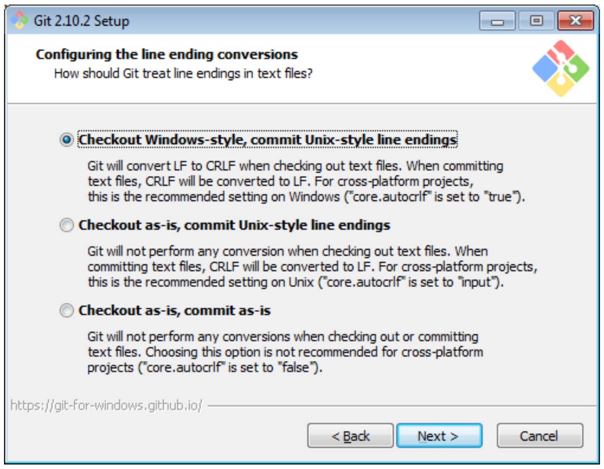
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?

[✓ **Use OpenSSH**]
This uses ssh.exe that comes with Git. The GIT_SSH and SVN_SSH environment variables will not be modified.

○ **Use (Tortoise)Plink**
PuTTY sessions were found in your Registry. You may specify the path to an existing copy of (Tortoise)Plink.exe from the TortoiseGit/SVN/CVS or PuTTY applications. The GIT_SSH and SVN_SSH environment variables will be adjusted to point to the following executable:

https://git-for-windows.github.io/

iv. Windows and UNIX use different line endings for end of lines. UNIX systems just use line feed character. Windows places a carriage return and line feed at the end of lines. The first option in the 'Configuring the line ending conversions' allows seamless work across UNIX and Windows environments.


Git 2.10.2 Setup

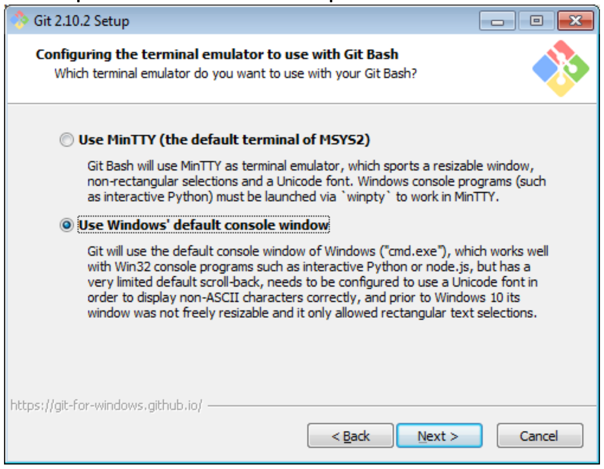
Configuring the line ending conversions
How should Git treat line endings in text files?

(●) **Checkout Windows-style, commit Unix-style line endings**
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

( ) **Checkout as-is, commit Unix-style line endings**
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

( ) **Checkout as-is, commit as-is**
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

v. I choose to use the default Window's console. (Note: It seems whomever wrote these descriptions is somewhat opinionated about windows consoles.)

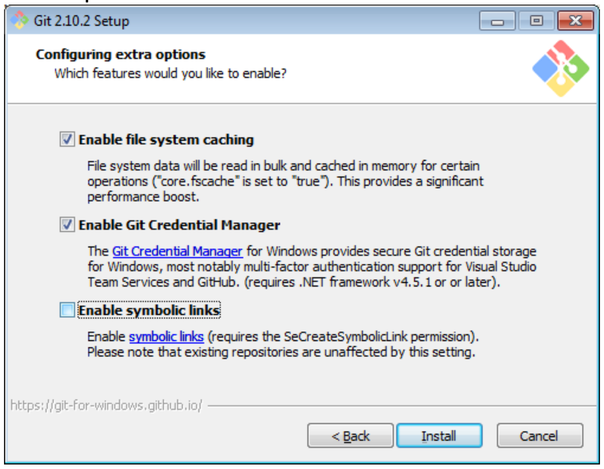

Git 2.10.2 Setup

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

( ) **Use MinTTY (the default terminal of MSYS2)**
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

(●) **Use Windows' default console window**
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

vi. I would certainly enable to first two options on the following dialog. I might enable the third option as well.


Git 2.10.2 Setup

Configuring extra options
Which features would you like to enable?

[✓] **Enable file system caching**
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

[✓] **Enable Git Credential Manager**
The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or or later).

[ ] **Enable symbolic links**
Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

Create a GitHub Education User Account

1. If you don't already have a GitHub Account, go to https://github.com/join and create one.
   a. GitHub is used by future employers so think professionally in choosing your username.
   b. Feel free to choose a free plan. In the next step you can upgrade to a student account which gives you unlimited private repositories for free.

2. Change your GitHub account to a student account, and get the Student Developer Pack.
    a. Go to https://education.github.com/pack
    b. I don't have access to any of this, so if needed, let me know of other instructions I need to add here.

Notes

We will first introduce the following packages:

- **Python** is the language we will be using this semester. It is an open source language known for its quick development time, extensibility, and large standard library. Python will work well for us because it is easy to learn, and since it is open source and extensible, there are many scientific computing and data analysis tools for the language.
- **PyCharm** is an integrated development environment for Python. PIt has built in tools to help us write and debug our python code. It also provides a seamless integration for Jupyter.
- **Jupyter** is an open source notebook creation / viewing tool that is somewhat similar to Mathematica. The notebook allows integration of equations, figures, functions and python code. Since we can run python code in Jupyter, it provides a nice interface to our python programs and functions.
- **Anaconda** is a collection of python modules tailored for scientific computing and data analysis. In addition to the python modules, it also includes tools for managing which modules and module versions are available to a program. The python interpreter is part of the Anaconda package. Jupyter as well as other packages like MatPlotLib, NumPy, BeautifulSoup, AstroPy, and SciPy as well as many others.

Resources:

Some good resources to get started with python, PyCharm, and Jupyter are:

- ☐ The python tutorial, especially useful now are sections 1, 3, 4 and 5.
- ☐ Python's list of simple programs has some great programing examples.
- ☐ PhyCharm has a page on Creating and Running Your First Python Project. There is also a good video describing PhyCharm down a bit on the PhyCharm homepage.
- ☐ The tutorial titled Using Jupyter Notebook with PyCharm is a good introduction to Jupyter.

In – Class Introductory Assignments

- ☐ Create a project in PyCharm and write a program to:
  - ○ Display “Hello World.”
  - ○ Use a loop structure to print the numbers 1 – 10.
  - ○ Use another loop structure to print the number names, “one, two, three …” from zero to ten.
  - ○ In doing this assignment, you may want to look at The Python Tutorial, especially sections 3.2 – 4.4.
- ☐ Create a Jupyter notebook and in that notebook complete the tutorial titled Using Jupyter Notebook with PyCharm from JetBrains’ website. In doing so, you will:
  - ○ Create a Jupyter notebook file (I suggest you create this in the project you already have)
  - ○ The tutorial does not include it, but demonstrate that it can calculate that 2 + 2 is 4.
  - ○ Import a python module.
  - ○ Create a graph.
  - ○ View the notebook in a web browser.
  - ○ The tutorial does not have you do this, but edit the notebook in the browser. Also have the notebook perform calculations (press shift – enter while on a cell).
  - ○ Add some headings – View these in the web browser as well.
  - ○ Write an equation in LaTex and display it.
- ☐ Investigate solving $\frac{dP}{dt} = rP$ using numerical methods.
  - ○ Write a python function that uses finite difference methods to simulate the population growth. Like the example in your text, use $P_0 = 100$, $r = 0.1$ and a time step size of 0.005, $\Delta t = 0.005$. You may want to write out pseudo code before you write in python.
  - ○ Create a Jupyter notebook and plot your simulation results for population vs. time. Add an appropriate title and axis labels to this graph.
  - ○ In your previous graph, have the population data displayed with red hexagon markers with a transparent face. You might want to check the matplotlib documentation for plot. Also, setting the color alpha value to zero will make the marker transparent as described on this stackoverflow discussion.
  - ○ On a single graph, have plots for the population vs time for different time step sizes, Δt. Create at least four different plots with Δt values ranging from 0.005 to 1.
  - ○ Create a plot of the simulation error vs time for the simulations you created in the previous step. Note that the simulation error is the difference between the exact value and the simulation value. For this, you may want to look at the range function in python.
  - ○ As a final step in understanding the effect of the step size on simulation error. Plot the value of the population at $t = 100$ as a function of time step size, Δt.
  - ○ Document your work and turn in this problem. In doing so, you need to create appropriate headings and add text regions which discuss your work. You need to include at least one LaTex equation. For this, you should check out the linked pages on markdown formatting, and LaTex.