Code	Done	Objective
CL-D1	Done	I can open a new terminal window
CL-D2	Done	I can list the contents of the current directory
CL-D3	Done	I can print the absolute path of the current directory to the terminal
CL-D4	Done	I can change directories by providing an absolute path
CL-D5	Done	I can run a program in the current directory
CL-C1	Done	I can list the contents of a directory without entering it
CL-C2	Done	I can change directories by providing a relative path
CL-C3	Done	I can print the first 10 lines of a text file to the terminal
CL-C4	Done	I can print the last 10 lines of a text file to the terminal
CL-C5	Done	I can create an empty text file from the command line
CL-C6	Done	I can print the full contents of a text file to the terminal
CL-C7	Done	I can create a copy of a file without destroying the original
CL-C8	Done	I can rename a file
CL-C9	Done	I can move a file to a different directory
CL-C10	Done	I can delete a file
CL-C11	Done	I can create a new directory
CL-C12	Done	I can delete a directory
CL-C13	Done	I can copy a directory and all of its contents
CL-C14	Done	I can read the manual
CL-C15	Done	I can search for a particular word in the manual
CL-C16		I can see the permissions on a file
CL-C17		I can change ownership of a file
CL-C18		I can change the permissions on a file
CL-C19		I can list all the environment variables that are active in my terminal session
CL-B1		I can add multiple lines of text to a file without leaving the terminal
CL-B2		I can list (or copy, or delete) all the files in a directory that end in .dat (and only the files that end in .dat)
CL-B3		I can list all the files in a directory in chronological order, by modification date (oldest first)
CL-B4		I can list the contents of the current directory (including hidden files and directories which start with .)
CL-B5		I can add a new environment variable to ~/.bashrc and make that change take effect in the current session
CL-B6		I can add a directory to my PATH environment variable (keeping the directories that were in the old PATH)
CL-B7		I can create an alias for a command and make that alias active every time I open a new terminal
CL-B8		I can write and execute a bash script which demonstrates any 5 skills from the C or D levels, with comments showing which skills are being demonstrated
CL-A1		I can write a single command to combine the contents of every .dat file in a directory into a new .dat file in a different directory
CL-A2		I can write a single command to create a new .txt file that contains the filenames of all the .dat files in a directory, with each filename on a new line, in chronological order (oldest first)

Code	Done	Objective
CWP-D1	Done	I can install the latest Python on MacOS
CWP-D2	Done	I can launch the Python 2 interpreter from the command line
CWP-D3	Done	I can launch the Python 3 interpreter from the command line
CWP-D4	Done	I can assign a value to a variable
CWP-D5	Done	I can print the value of a variable
CWP-C1	Done	I can write a python statement that uses 2 different arithmetic binary operators
CWP-C2	Done	I can convert an integer to a floating-point number
CWP-C3	Done	I can divide 4 by 5 and get 0.8 in Python 3
CWP-C4	Done	I can divide 4 by 5 and get 0.8 in Python 2
CWP-C5	Done	I can import the math module
CWP-C6	Done	I can write a python statement that calculates the cosine of 60 degrees
CWP-C7	Done	I can create a list of numbers
CWP-C8	Done	I can append a number to the end of a list
CWP-C9	Done	I can extend a list with a list
CWP-C10	Done	I can print the first item in a list using an index
CWP-C12	Done	I can print the last item in a list using an index
CWP-B1	Done	I can write python statements that assign a value to the variable T_c, convert that to Fahrenheit, assign the result to T_f, and print T_f
CWP-B2	Done	I can import just one function or variable from the math module and use it
CWP-B3	Done	I can import every function from the math module
CWP-B4	Done	I can import the math module and rename it mathematics (and wonder why I would ever do that)
CWP-B5	Done	I can write multiple python statements in a file and execute them from the command line
CWP-B6	Done	I can use a slice to extract every other item from a list, ignoring the first two and last two items in the original list
CWP-B7	Done	I can use the string format method to print the values of variables as part of a string
CWP-B8	Done	I can prompt the user of a python program for input, and store that input in a variable
CWP-A1		I can write a python program to complete Exercise 2.6 in *CP*

Code	Done	Objective
SWP-D1	Done	I can write an if statement
SWP-D2	Done	I can write a for loop
SWP-D3	Done	I can write a while loop
SWP-D4	Done	I can define a new function
SWP-D5	Done	I can import the numpy package and rename it np
SWP-C1	Done	I can write a function that converts a celsius temperature to fahrenheit
SWP-C2	Done	I can write a function that calculates the distance from the origin to a point given in cylindrical coordinates (r, θ, z)
SWP-C3	Done	I can create an empty numpy array
SWP-C4	Done	I can create a numpy array full of zeroes
SWP-C5	Done	I can create a two-dimensional numpy array
SWP-C6	Done	I can use numpy to load an array from a plain text file
SWP-C7	Done	I can add a value to all the elements in a numpy array
SWP-C8	Done	I can multiply all the elements in a numpy array by an integer
SWP-C9	Done	I can add two arrays together
SWP-C10	Done	I can multiply two arrays together element-by-element
SWP-C11	Done	I can find the vector dot product of two arrays
SWP-C12	Done	I can write an if-elif-else statement
SWP-C13	Done	I can use a try-except block to catch an exception
SWP-C14	Done	I can raise an exception
SWP-C15	Done	I can return a tuple from a function
SWP-B1	Done	I can use a list comprehension to square all the elements of a list
SWP-B2		I can write a python program to complete Exercise 2.10 in *CP*
SWP-A1		I can complete Exercise 2.12 in *CP*
SWP-A2		I can write a python program to classify supernovae (see handout.)

Code	Done	Objective
AVD-D1		I can load data from a csv file with NumPy
AVD-D2		I can import pylab
AVD-C1		I can change the shape of a NumPy array
AVD-C2		I can print the data type of a NumPy array
AVD-C3		I can add a 1x3 NumPy array to a 4x3 NumPy array to get a new 4x3 NumPy array
AVD-C4		I can use fancy indexing to pull ou the fourth, last, and second elements of a NumPy array
AVD-C5		I can apply a mask to a NumPy array
AVD-C6		I can create the structured NumPy array dtype "fluid" on *ECP*, page 221.
AVD-C7		I can use a NumPy function to calculate the mean of an array
AVD-C8		I can use a NumPy function to append an array to an array
AVD-C9		I can add axis labels to a plot
AVD-C10		I can make a 2D scatter plot by reading data from a file
AVD-B1		I can use a mask to pull out elements of an array that are less than 5, or greather than or equal to 7
AVD-B2		I can index a NumPy structured array with dtype "fluid" (*ECP*, page 221) using a string matching the name of a column
AVD-B3		I can plot a sine function using functions from the numpy and pylab packages
AVD-B4		I can plot a 2D density map of wave interference by following Example 3.1 in *CP*
AVD-A1		I can complate Exercise 3.3 in *CP* (data files available on Blackboard)
AVD-A2		I can complete Exercise 3.8 in *CP* (data files available on Blackboard)

Code	Done	Objective
LVG-D1	Done	I can install git
LVG-D2	Done	I can ask git for help
LVG-D3	Done	I can configure git to remember my user name, email, and editor
LVG-C1	Done	I can initialize a local git repository
LVG-C2	Done	I can add a new file to a local repository
LVG-C3	Done	I can check the status of a local git repository
LVG-C4	Done	I can commit a change to a local git repository with a commit message
LVG-C5	Done	I can view a log of changes to a local git repository
LVG-C6	Done	I can view the differences between two versions of a file in a local git repository
LVG-C7		I can list the branches that exist in a local git repository
LVG-C8		I can create a new branch in a local git repository
LVG-B1		I can view the differences between the current and staged version of a file in a local git repository
LVG-B2		I can unstage a file in my local git repository
LVG-B3		I can reset to a previous version of my local git repository
LVG-B4	Done	I can revert to a previous version of my local git repository
LVG-B5		I can merge a branch back into the master branch of my local git repository
LVG-A1		I can merge a branch back into the master branch of my local git repository and resolve a conflict (see *ECP*, p. 381)

Code	Done	Objective
PWL-D1		I can download and install MacTeX
PWL-D2		I can write a LaTeX document that contains the text "Hello World!"
PWL-D3		I can declare the author name in the preamble of my LaTeX document
PWL-D4		I can declare the title of the document in the preamble of my LaTeX document
PWL-D5		I can declare a date in the preamble of my LaTeX document
PWL-C1		I can use pdflatex to produce a PDF from my "Hello World!" document
PWL-C2		I can use a command to format the title, aithor, and date in my pdf
PWL-C3		I can use sections in my LaTeX document
PWL-C4		I can typeset a mathematical formula inline with my text
PWL-C5		I can typeset a mathematical equation separately from my text
PWL-C6		I can include a figure with a caption, label, and graphics
PWL-B1		I can include text between equations, without leaving the equation environment
PWL-B2		I can align equations using an align environment
PWL-B3		I can center the graphics within a figure
PWL-B4		I can use internal references in my text to refer to a specific figure
PWL-B5		I can write a .bib file with an entry for Effective Computation in Physics
PWL-B6		I can cite "Effective Computation in Physics" in my text using a cite command
PWL-B7		I can generate a bibliography in my document that includes "Effective Computation in Physics"
PWL-A1		I can use a command to include the contents of one .tex file in another .tex file
PWL-A2		I can use the package hyperref to make my references and citations

Code	Done	Objective
TYC-D1	Done	I can import the unittest package
TYC-D2	Done	I can import a module that I have written into my test module
TYC-C1	Done	I can write a test case that inherits from unittest.TestCase
TYC-C2	Done	I can write a unit test using assertEqual
TYC-C3	Done	I can write a unit test using assertTrue or assertFalse
TYC-C4	Done	I can write a unit test using assertAlmostEqual
TYC-C5		I can write a unit test using assertRaises
TYC-B1	Done	I can disover and run unit tests from the command line
TYC-B2		I can write code that passes each of the C-level unit tests
TYC-A1		I can demonstrate test-driven development in python, documenting my progress with git commits