Matthew Fitzgerald
Econ 204B
January 10, 2020

Problem Set # 1

Problem 1: Referencing, shallow copying, and deep copying

(a) Create an array named '`v`' that contains the following numbers: `10, 11, 12`.

(b) Create an array named '`w`' that contains the array '`v`' and also the numbers `1, 2, 3, 4, 5`.

(c) Create a new variable '`x`', and set it equal to '`w`'. Print '`x`'.

(d) Create a variable '`y`' with the same elements as '`x`' by copying '`x`', using the '`copy`' function. Print '`x`'.

(e) Create a variable '`z`' with the same elements as '`x`' by copying '`x`', this time using the '`deepcopy`' function.

(f) Add the number '`6`' to the array '`w`' (hint: use '`append!()`').

(g) Print '`x`', '`y`', and '`z`'. Which of the arrays have changed since their creation, and which stayed the same?

(h) Change the first element of '`v`' to `100`.

(i) Print '`x`', '`y`', and '`z`'. Which of the arrays have changed since part g?

(j) Explain the changes you saw in parts g and i. (hint: if you don't understand the changes you see, look up 'shallow vs deep copying').

Problem 2: Functions

(a) Write a function in Julia called '`my_average`' that takes the average of two user specified numbers (i.e. a function that computes $(x + y)/2$).

(b) Write a function called "`dumb_function`" that:

i Takes as arguments a function and two **integers** (call the integers '`x`' and '`y`')

ii Creates a new integer '`z`' which is equal to 10 plus '`x`', the first integer supplied by the user

iii Calls the function supplied by the user and uses '`z`' and '`y`' as arguments

iv Returns the result of the previous step

(c) Now test your function by running '`dumb_function(my_average, 10.5, 30)`' (this should produce an error since 10.5 is not an integer and `dumb_function` requires two integers as arguments).

(d) Now test your function by running '`dumb_function(my_average, 10, 30)`' (this should print out `25`).

Problem 3: Loops

(a) Write a for loop that takes the square root of each of the integers from 1 to 20. The loop should therefore print out 1, 1.41421, 1.73205, 2, etc. (note that Julia uses 1-based indexing as opposed to zero-based indexing).

(b) Write a for loop that takes each number in 1 to 20, squares it, and divides the result by 2 (i.e. compute $i^2/2$ for each number in 1 to 20). If the evaluated expression (i.e. $i^2/2$) is greater than 150, the loop should stop and print out "`BREAK: i too large`" (where '`i`' is the value of '`i`' for which $i^2/2 > 150$).

(c) Write a function called '`intersection`' that finds intersection points between $y = x$ and a user specified function. The function should have four arguments: a function '`f(x)`', a starting value '`x_0`' (this can be any number), a variable controlling the maximum number of iterations called '`maxiters`' (this is an integer), and a variable controlling the minimum tolerance level called '`tol`' (this can be any number). Create default values for '`maxiters`' and '`tol`'. Namely, set the default for '`maxiters`' to '`1000`' and the default for '`tol`' to 1×10^{-10}. The function should:

i Create a variable called '`diff`' and set it equal to the sum of '`tol`' and `1000`

ii Create a variable called '`counter`' and set it equal to 1

iii Create a variable called '`x`' and set it equal to the user specified value '`x_0`'

iv Begin a while loop that continues as long as `diff` is greater than `tol`

(1) Print out the iteration number (use '`counter`' for this)

(2) Evaluate the user specified function at the user specified starting value (i.e. '`f(x_0)`'). Call the result '`newval`'

(3) Set '`diff`' equal to the absolute value of the difference between '`x`' and '`newval`' (why do we need the absolute value?)

(4) Set '`x`' equal to '`newval`'

(5) Print out '`x`' (you might want to use '`print(" x = ", x)`')

(6) Add one to '`counter`'

(7) Write an if statement that breaks the loop and prints out "Maximum number of iterations exceeded" if '`counter`' is greater than '`maxiters`'

(8) Write an if statement that breaks the loop and prints out "Iteration Stopped: Diverging" if '`x`' is greater than `1000000000` or if '`x`' is less than `-1000000000` (to see why, compute $1,000,000,000^2$ and then $10,000,000,000^2$)

v Now create a function called '`testfun`' that is equal to x^2 (i.e. create $f(x) = x^2$)

vi Test '`intersection`' using '`testfun`'. Run the function using the following starting values: `-10`, `-1.1`, `-0.4`, `-0`, `0.5`, `1`, `1.5`, `20`.

vii Which of the starting values resulted in a correct answer? Which did not?

Problem 4: Plotting

In this problem you will do some basic plotting. In order to do this you will need the '`Plots`' package. To install a package in Julia, use '`Pkg.add("Package Name")`'. Therefore '`Pkg.add("Plots")`' will install the '`Plots`' package. Once installed you can use the '`Plots`' package by typing "`using Plots`". Once you've run these commands, check that you're using the GR backend by running '`Plots.backend()`'. This should print out '`Plots.GRBackend()`'. If it doesn't, change your backend to GR using '`Plots.gr()`'.

(a) Generate an array '`x`' that takes the values 1950 to 2010 (here we're generating our own data, so '`x`' will act as our year variable). Now generate an array '`y`' that takes the values from 1 up to the length of '`x`' (hint: look up the '`size`' function). Now plot '`x`' and '`y`' (this should just be a straight line). Title the plot "Part (a)" and put the legend in the bottom right corner. Label the x-axis "Year" and label the y-axis "Value".

(b) Again generate an array '`x`' that takes the values 1950 to 2010. Now generate an array of length '`x`' whose components are random variables drawn from a normal distribution with mean 2 and variance 10. To generate random variables, install and use the '`Distributions`' and '`Random`' packages. Before creating '`y`', set the seed using '`Random.seed!(1234)`'. Plot '`x`' and '`y`'. Title the plot "Part (b)" and put the legend in the bottom right corner. Label the x-axis "Year" and label the y-axis "Value".

(c) Set the seed using '`Random.seed!(4321)`'. Now create a new array of length '`x`' whose components are random variables drawn from a normal distribution with mean 4 and variance 10. Create a plot with two subplots next to each other (i.e. a left plot and a right plot as opposed to a top plot and a bottom plot). The first plot should be a plot of '`x`' and '`y`' and the second plot should be a plot of '`x`' and '`z`'. Label the axes of both plots the same as before, but title the first plot "X and Y" and title the second plot "X and Z".

(d) Combine the two plots from the previous part onto the same plot (i.e. make a plot with two lines, one for the data in '`y`' and one for the data for '`z`', both plotted against '`x`'). Keep the same axis labels as before, but make sure your legend includes both '`y`' and '`z`'. Title this graph "All In One".