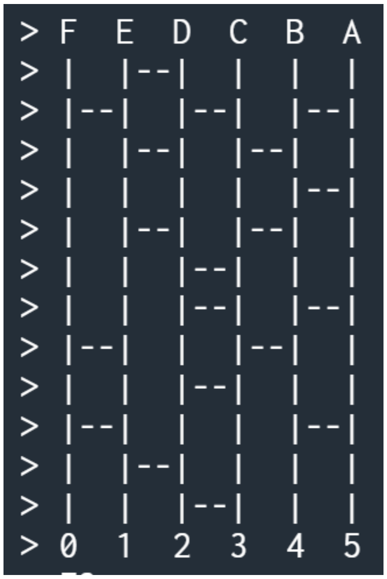
I chose the “Ghost Legs” problem & wrote my solution in Python 3

Step 4: First 15 mins of approach

I spent the first 15 minutes feeling out the input for the program / test cases–particularly the spacing between bones & bone labels. Instead of drawing the inputs, I simply told the program to print them out instead of solving the problem (an example is included below).



```
> F  E  D  C  B  A
> |  |--|  |  |  |
> |--|  |--|  |--|
> |  |--|  |--|  |
> |  |  |  |  |--|
> |  |--|  |--|  |
> |  |  |--|  |  |
> |  |  |--|  |--|
> |--|  |  |--|  |
> |  |  |--|  |  |
> |--|  |  |  |--|
> |  |--|  |  |  |
> |  |  |--|  |  |
> 0  1  2  3  4  5
```

For the remainder of the first 15 minutes, after fleshing out the inputs–as well as the expected outputs (one being ‘C2’ for the above example), I coded the scaffolding for my solution: this included isolating the bone labels into an operable data structure.

Step 6: How I solved the problem pre & post reflection

Before the reflection, I naturally gravitated towards an more correct approach–as opposed to simply coding away at it–because either the website’s python interpreter was wrong or I somehow consistently hallucinated wrong outputs; I took a more thorough, correct approach after I could not fathom how their python interpreter returned a non-zero value for any_string.strip().count(‘ ‘). I was flabbergasted. To double check myself, I even verified that the strip() string method did indeed work when I used it earlier. After the 15 minutes had elapsed & this bug I must have hallucinated vanished, I had coincidently fleshed out the problem as detailed in Step 4’s section.

After reflecting & realizing I shanghaied myself into taking the correct approach to approaching the problem, I began coding the logic needed to solve the problem (the reflection didn’t change my approach going forward because I had already done what I would have realized I needed to do from reflecting). I first accounted for the two edge cases (checking the outermost bones & not unintentionally indexing the string at [0 -3] or [len(str) +3]). After thinking my work over & affirming to myself that no more edge cases existed, I wrote the decision-making logic (i.e., if ‘-’ on a side of a bone, move to the bone in that direction for every bone trace on that looked-at bone). When my code inevitably didn’t work first try, I dusted off all my debug print statements, and quickly discovered a small logic error that ignored the bottom label of the last bone. And finally, I had solved the problem in my delirious, 1:21am state.