advent_of_code_day1

December 3, 2021

1 Advent of Code

[ ]:
```
# set up the environment
import numpy as np
```

1.1 Day 1: Sonar Sweep

You're minding your own business on a ship at sea when the overboard alarm goes off! You rush to see if you can help. Apparently, one of the Elves tripped and accidentally sent the sleigh keys flying into the ocean!

Before you know it, you're inside a submarine the Elves keep ready for situations like this. It's covered in Christmas lights (because of course it is), and it even has an experimental antenna that should be able to track the keys if you can boost its signal strength high enough; there's a little meter that indicates the antenna's signal strength by displaying 0-50 stars.

[ ]:
```
# load the data from the source file 'data/depth.dat' and store it in an array.
depths = np.loadtxt("data/depths.dat", dtype="int", delimiter=",", unpack=False)
print(depths)
print(len(depths))
```

```
[ 173  175  171 … 7118 7115 7121]
2000
```

1.1.1 Problem 1

As the submarine drops below the surface of the ocean, it automatically performs a sonar sweep of the nearby sea floor. On a small screen, the sonar sweep report (your puzzle input) appears: each line is a measurement of the sea floor depth as the sweep looks further and further away from the submarine.

[ ]:
```
depth_increase_count = 0
previous_depth = depths[0]
for x in depths:
    if x > previous_depth:
        depth_increase_count += 1
    previous_depth = x

print("Number of depth increase: ", depth_increase_count)
```

```
Number of depth increase:  1521
```

1.1.2 Part 2

Considering every single measurement isn't as useful as you expected: there's just too much noise in the data.

Instead, consider sums of a three-measurement sliding window.

[ ]:
```
# create a new array of the sliding data
three_measurement_sums = []
for x in range(len(depths)-2):
    three_measurement_sums.append(depths[x]+depths[x+1]+depths[x+2])

# repeat the calculation with the above data
depth_increase_count = 0
previous_depth = three_measurement_sums[0]
for x in three_measurement_sums:
    if x > previous_depth:
        depth_increase_count += 1
    previous_depth = x

print("Number of depth increase: ", depth_increase_count)
```

```
Number of depth increase:  1543
```

1.2 Day 2: Dive!

1.2.1 Part One

It seems like the submarine can take a series of commands like forward 1, down 2, or up 3:

```
forward X increases the horizontal position by X units.
down X increases the depth by X units.
up X decreases the depth by X units.
```

Note that since you're on a submarine, down and up affect your depth, and so they have the opposite result of what you might expect.

The submarine seems to already have a planned course (your puzzle input). You should probably figure out where it's going. For example:

forward 5 down 5 forward 8 up 3 down 8 forward 2

Your horizontal position and depth both start at 0. The steps above would then modify them as follows:

```
forward 5 adds 5 to your horizontal position, a total of 5.
down 5 adds 5 to your depth, resulting in a value of 5.
forward 8 adds 8 to your horizontal position, a total of 13.
up 3 decreases your depth by 3, resulting in a value of 2.
down 8 adds 8 to your depth, resulting in a value of 10.
forward 2 adds 2 to your horizontal position, a total of 15.
```

After following these instructions, you would have a horizontal position of 15 and a depth of 10. (Multiplying these together produces 150.)

Calculate the horizontal position and depth you would have after following the planned course. What do you get if you multiply your final horizontal position by your final depth?

[ ]:
```
# Import the manoeuvreing data into an array fo tuples
manoeuvres = depths = np.genfromtxt("data/manoeuvre.dat", dtype="U7, i4",␣
 ↪delimiter=" ")
print(manoeuvres[:10], '...', manoeuvres[-10:])
print(len(manoeuvres))
```

```
[('forward', 8) ('forward', 9) ('forward', 9) ('down', 3) ('forward', 9)
 ('down', 1) ('down', 7) ('down', 7) ('down', 4) ('down', 2)] … [('down', 8)
('forward', 6) ('forward', 7) ('forward', 9) ('forward', 4)
 ('down', 3) ('up', 5) ('down', 7) ('down', 7) ('forward', 9)]
1000
```

[ ]:
```
horizontal_pos = 0
depth = 0
for x in manoeuvres:
    direction = x[0]
    steps = x[1]
    if direction == "forward":
        horizontal_pos += steps
    if direction == "up":
        depth -= steps
    if direction == "down":
        depth += steps

print("Displacement: ", horizontal_pos, ", Depth: ", depth)
print("Product of displacement and dept: ", horizontal_pos * depth)
```

```
Displacement:  2003 , Depth:  872
Product of displacement and dept:  1746616
```

1.2.2 Part Two

Based on your calculations, the planned course doesn't seem to make any sense. You find the submarine manual and discover that the process is actually slightly more complicated.

In addition to horizontal position and depth, you'll also need to track a third value, aim, which also starts at 0. The commands also mean something entirely different than you first thought:

```
down X increases your aim by X units.
up X decreases your aim by X units.
forward X does two things:
    It increases your horizontal position by X units.
    It increases your depth by your aim multiplied by X.
```

Again note that since you're on a submarine, down and up do the opposite of what you might expect: "down" means aiming in the positive direction.

Now, the above example does something different:

```
forward 5 adds 5 to your horizontal position, a total of 5. Because your aim is 0, your depth d
down 5 adds 5 to your aim, resulting in a value of 5.
forward 8 adds 8 to your horizontal position, a total of 13. Because your aim is 5, your depth
up 3 decreases your aim by 3, resulting in a value of 2.
down 8 adds 8 to your aim, resulting in a value of 10.
forward 2 adds 2 to your horizontal position, a total of 15. Because your aim is 10, your deptl
```

After following these new instructions, you would have a horizontal position of 15 and a depth of 60. (Multiplying these produces 900.)

Using this new interpretation of the commands, calculate the horizontal position and depth you would have after following the planned course.

[ ]:
```
horizontal_pos = 0
depth = 0
aim = 0
for x in manoeuvres:
    direction = x[0]
    steps = x[1]
    if direction == "up":
        aim -= steps
    if direction == "down":
        aim += steps
    if direction == "forward":
        horizontal_pos += steps
        depth += aim * steps


print("Displacement: ", horizontal_pos, ", Depth: ", depth)
print("Product of displacement and dept: ", horizontal_pos * depth)
```

```
Displacement:  2003 , Depth:  869681
Product of displacement and dept:  1741971043
```

[ ]:
```

```