day12

December 15, 2020

1 Day 12 Part 1 - Navigation for fools

So this feels quite easy compared to the previous few days. We basically need to step through the directions keeping track of two variables, our current location, and our heading.
When we get an absolute direction, we just update our location. When we get a L/R we update our heading and when we get a Forward direction, we update our location based on our heading.

Having a glance down the data and the examples, it looks highly likely that we'll only have to handle headings on 90 degree chunks, which means we can probably keep our heading as a simple N,E,S,W, or as 0,90,180,270 depending on how I'm feeling.

We'll also want to calculate the manhatten distance between two points, x1,y1 and x2,y2, which is simple enough, but is probably worth turning into a function. I've used tuples a lot so far for coordinates, but I'm going to use the more advanced namedtuple this time, which is a little like a Scala case class. It allows us to treat the tuple object a bit like an object, with an x and y attribute, without all the overhead of classes. This is because they are read-only, but if our changes return copies with updated locations, we can simply assign those around if we want (or if we need it part 2, keep a list of all the places we went too…).

[1]:
```
import ipytest
ipytest.autoconfig()
from collections import namedtuple

Ship = namedtuple('Ship', ['x', 'y', 'heading'], defaults=(90,))
NORTH=0
EAST=90
SOUTH=180
WEST=270

def distance(ship1, ship2=Ship(0,0)):
    return abs(ship1.x-ship2.x)+abs(ship1.y-ship2.y)

assert distance(Ship(0,0), Ship(10,3)) == 13
assert distance(Ship(5,0), Ship(15,3)) == 13
assert distance(Ship(0,0), Ship(10,-3)) == 13
assert distance(Ship(10,3), Ship(0,0)) == 13
```

[2]:
```
def rotate(ship, amount):
    return Ship(ship.x, ship.y, (ship.heading + amount) % 360)
```

```
assert rotate(Ship(0,0), 90) == Ship(0,0,SOUTH)
assert rotate(Ship(0,0, SOUTH), 90) == Ship(0,0,WEST)
assert rotate(Ship(0,0, WEST), 90) == Ship(0,0,NORTH)
assert rotate(Ship(0,0, NORTH), 90) == Ship(0,0,EAST)
assert rotate(Ship(0,0), -90) == Ship(0,0,NORTH)
assert rotate(Ship(0,0, NORTH), -90) == Ship(0,0,WEST)
```

[3]:

```
def process(ship, instruction):
    dist = int(instruction[1:])
    if instruction[0]== "N":
        return Ship(ship.x, ship.y+dist, ship.heading)
    if instruction[0]== "E":
        return Ship(ship.x+dist, ship.y, ship.heading)
    if instruction[0]== "S":
        return Ship(ship.x, ship.y-dist, ship.heading)
    if instruction[0]== "W":
        return Ship(ship.x-dist, ship.y, ship.heading)
    if instruction[0]=="R":
        return rotate(ship, dist)
    if instruction[0]=="L":
        return rotate(ship, -dist)
    if instruction[0]=="F":
        if ship.heading == NORTH:
            return Ship(ship.x, ship.y+dist, ship.heading)
        if ship.heading == EAST:
            return Ship(ship.x+dist, ship.y, ship.heading)
        if ship.heading == SOUTH:
            return Ship(ship.x, ship.y-dist, ship.heading)
        if ship.heading == WEST:
            return Ship(ship.x-dist, ship.y, ship.heading)

ship = Ship(0,0)
ship = process(ship, "F10")
assert ship == Ship(10,0,EAST)
ship = process(ship, "N3")
assert ship == Ship(10,3,EAST)
ship = process(ship, "F7")
assert ship == Ship(17,3,EAST)
ship = process(ship, "R90")
assert ship == Ship(17,3,SOUTH)
ship = process(ship, "F11")
assert ship == Ship(17,-8,SOUTH)

assert distance(ship) == 25
```

Ok, let's try that on production data

[4]:
```
directions = [line.strip() for line in open("day12.txt").readlines()]
ship = Ship(0,0)
for direction in directions:
    ship = process(ship, direction)
print(distance(ship))
```

```
1687
```

1.1 Vexatious vectors

Oooh, this is interesting. I've done lots of standard cartesian directions before, but in this case we are talking about using vectors.

In this case, there is a point that is X distance east of the boat, and y distance north of the. boat. This vector will form a right angled triangle with the length of the vector being the length of the hypotonus.

Luckily, we don't actually need to handle the hypotonus, so no pythagorus for us this time, all we need to do is recognise that we are going to track a vector around the ship, and then when we do the F, we move by multiplying the vector east and north values by the value of the instruction.

The trick here is the rotation command. It feels like we might need to do triganometry, but then I looked at the examples, and thought through the rotations. Since we are always rotating by 90 degrees, it'll never be a partial rotation. So something at 7,3 when rotated 90 degrees will become 3, -7. Rotated another 90 degrees, it'll become -7, -3, and one more time, to -3, 7.

What we need to do is work out those translations for both 90 degrees right and 90 degrees left as well.

We're also going to need a slightly different model, because instead of a heading we'll need a location and a waypoint (which is a location itself).

[5]:
```
Position = namedtuple('Position', ['x', 'y'], defaults=(0,0))
Ship = namedtuple('Ship', ['loc', 'waypoint'], defaults={Position(),␣
 ↪Position()})

def distance(p1, p2=Position(0,0)):
    return abs(p1.x-p2.x)+abs(p1.y-p2.y)

def rotate_right(p1):
    return Position(p1.y,-1*p1.x)

assert rotate_right(Position(7,3)) == Position(3, -7)
assert rotate_right(Position(3,-7)) == Position(-7, -3)
assert rotate_right(Position(-7, -3)) == Position(-3, 7)
assert rotate_right(Position(-3, 7)) == Position(7, 3)

def rotate_left(p1):
    return Position(-1*p1.y,p1.x)
```

```
assert rotate_left(Position( 7,  3)) == Position(-3,  7)
assert rotate_left(Position(-3,  7)) == Position(-7, -3)
assert rotate_left(Position(-7, -3)) == Position( 3, -7)
assert rotate_left(Position( 3, -7)) == Position( 7,  3)

def rotate(ship, angle):
    wp = ship.waypoint
    while angle > 0:
        wp = rotate_right(wp)
        angle -= 90
    while angle < 0:
        wp = rotate_left(wp)
        angle += 90
    return Ship(ship.loc, wp)

assert rotate(Ship(Position(0,0), Position(7,3)),90) ==␣
↪Ship(Position(0,0),Position(3, -7))
assert rotate(Ship(Position(0,0), Position(7,3)),180) ==␣
↪Ship(Position(0,0),Position(-7, -3))
assert rotate(Ship(Position(0,0), Position(7,3)),270) ==␣
↪Ship(Position(0,0),Position(-3, 7))
assert rotate(Ship(Position(0,0), Position(7,3)),-90) ==␣
↪Ship(Position(0,0),Position(-3, 7))
assert rotate(Ship(Position(0,0), Position(7,3)),-180) ==␣
↪Ship(Position(0,0),Position(-7, -3))
assert rotate(Ship(Position(0,0), Position(7,3)),-270) ==␣
↪Ship(Position(0,0),Position(3, -7))


def process(ship, instruction):
    dist = int(instruction[1:])
    if instruction[0]== "N":
        return Ship(ship.loc, Position(ship.waypoint.x, ship.waypoint.y+dist))
    if instruction[0]== "E":
        return Ship(ship.loc, Position(ship.waypoint.x+dist, ship.waypoint.y))
    if instruction[0]== "S":
        return Ship(ship.loc, Position(ship.waypoint.x, ship.waypoint.y-dist))
    if instruction[0]== "W":
        return Ship(ship.loc, Position(ship.waypoint.x-dist, ship.waypoint.y))
    if instruction[0]=="R":
        return rotate(ship, dist)
    if instruction[0]=="L":
        return rotate(ship, -dist)
    if instruction[0]=="F":
```

```
        return Ship(Position(ship.loc.x+(ship.waypoint.x*dist), ship.loc.
↪y+(ship.waypoint.y*dist)), ship.waypoint)

ship = Ship(Position(0,0), Position(10, 1))
ship = process(ship, "F10")
assert ship == Ship(Position(100,10),Position(10, 1))
ship = process(ship, "N3")
assert ship == Ship(Position(100,10),Position(10, 4))
ship = process(ship, "F7")
assert ship == Ship(Position(170,38),Position(10, 4))
ship = process(ship, "R90")
assert ship == Ship(Position(170,38),Position(4, -10))
ship = process(ship, "F11")
assert ship == Ship(Position(214,-72),Position(4, -10))

assert distance(ship.loc) == 286
```

Let's try that on the real data and see what works

[6]:
```
directions = [line.strip() for line in open("day12.txt").readlines()]
ship = Ship(Position(0,0),Position(10, 1))
for direction in directions:
    ship = process(ship, direction)
print(distance(ship.loc))
```

```
20873
```