Advent of Code 2022

tails618

December 2022

1 Day 1: Calorie Counting

1.1 Puzzle 1

The Elves take turns writing down the number of Calories contained by the various meals, snacks, rations, etc. that they've brought with them, one item per line. Each Elf separates their own inventory from the previous Elf's inventory (if any) by a blank line.

In case the Elves get hungry and need extra snacks, they need to know which Elf to ask: they'd like to know how many Calories are being carried by the Elf carrying the most Calories. In the example above, this is 24000 (carried by the fourth Elf).

Find the Elf carrying the most Calories. How many total Calories is that Elf carrying?

1.1.1 Full Code

```
# open data file
input = open("1-data.txt", "r")
data = input.readlines()

# remove '\n' from each item except the final one
for i in range(len(data) - 1):
    data[i] = data[i][:-1]

# maybe add an extra blank at the end because jank
if data[-1] != '':
    data.append('')

# sum the items between each blank in the list
lastBlank = 0
sums = []
for i, x in enumerate(data):
    if x == '':
        sums.append(sum(int(i) for i in data[lastBlank:i]))
        lastBlank = i + 1

# find the biggest sum
sums.sort()
print(sums[-1])
```

1.2 Puzzle 2

By the time you calculate the answer to the Elves' question, they've already realized that the Elf carrying the most Calories of food might eventually run out of snacks.

To avoid this unacceptable situation, the Elves would instead like to know the total Calories carried by the top three Elves carrying the most Calories. That way, even if one of those Elves runs out of snacks, they still have two backups.

Find the top three Elves carrying the most Calories. How many Calories are those Elves carrying in total?

1.2.1 Full Code

```
print(sums[-1] + sums[-2] + sums[-3])
```

2 Day 2: Rock Paper Scissors

2.1 Puzzle 1

Appreciative of your help yesterday, one Elf gives you an encrypted strategy guide (your puzzle input) that they say will be sure to help you win. "The first column is what your opponent is going to play: A for Rock, B for Paper, and C for Scissors. The second column–" Suddenly, the Elf is called away to help with someone's tent.

The second column, you reason, must be what you should play in response: X for Rock, Y for Paper, and Z for Scissors. Winning every time would be suspicious, so the responses must have been carefully chosen.

The winner of the whole tournament is the player with the highest score. Your total score is the sum of your scores for each round. The score for a single round is the score for the shape you selected (1 for Rock, 2 for Paper, and 3 for Scissors) plus the score for the outcome of the round (0 if you lost, 3 if the round was a draw, and 6 if you won).

Since you can't be sure if the Elf is trying to help you or trick you, you should calculate the score you would get if you were to follow the strategy guide.

What would your total score be if everything goes exactly according to your strategy guide?

2.1.1 Full Code

```
# open data file
input = open("2-data.txt", "r")
data = input.readlines()

# create empty lists for the opponent's moves and your moves, as well as your scores
oppMoves, yourMoves, scores = [], [], []

# split the data into two lists for the opponent's moves and your moves
for i in range(len(data)):
    oppMoves.append(data[i][0])
    yourMoves.append(data[i][2])

# function to check if you win
# returns 0 for lose, 3 for draw, 6 for win (as per the challenge)
def checkWin(oppMove, yourMove):
    # shifts yourMove to the left 23, bringing XYZ to ABC, making it easier to check if it's the sa
    if oppMove == chr(ord(yourMove) - 23):
        return 3
    elif oppMove == 'A' and yourMove == 'Z':
        return 0
    elif oppMove == 'C' and yourMove == 'Y':
        return 0
    elif oppMove == 'B' and yourMove == 'X':
        return 0
```

```
    else:
        return 6

# calculate score for each round
for i, x in enumerate(yourMoves):
    n = checkWin(oppMoves[i], yourMoves[i])
    if x == 'X':
        n += 1
    elif x == 'Y':
        n += 2
    elif x == 'Z':
        n += 3
    scores.append(n)

# calculate total score
print(sum(scores))
```

2.2 Puzzle 2

The Elf finishes helping with the tent and sneaks back over to you. "Anyway, the second column says how the round needs to end: X means you need to lose, Y means you need to end the round in a draw, and Z means you need to win. Good luck!"

The total score is still calculated in the same way, but now you need to figure out what shape to choose so the round ends as indicated.

Following the Elf's instructions for the second column, what would your total score be if everything goes exactly according to your strategy guide?

2.2.1 Full Code

```
# given the opponent's move and how the game needs to end, decide your move
def chooseMove(oppMove, yourMove):
    if oppMove == 'A':
        if yourMove == 'X':
            return 'Z'
        elif yourMove == 'Y':
            return 'X'
        elif yourMove == 'Z':
            return 'Y'
    elif oppMove == 'B':
        if yourMove == 'X':
            return 'X'
        elif yourMove == 'Y':
            return 'Y'
        elif yourMove == 'Z':
            return 'Z'
    elif oppMove == 'C':
        if yourMove == 'X':
            return 'Y'
        elif yourMove == 'Y':
            return 'Z'
        elif yourMove == 'Z':
            return 'X'

# replace yourMove with the results of chooseMove
```

```
for i, x in enumerate(yourMoves):
    yourMoves[i] = chooseMove(oppMoves[i], yourMoves[i])

# calculate score for each round
for i, x in enumerate(yourMoves):
    n = checkWin(oppMoves[i], yourMoves[i])
    if x == 'X':
        n += 1
    elif x == 'Y':
        n += 2
    elif x == 'Z':
        n += 3
    scores[i] = n

# calculate total score
print(sum(scores))
```

3 Day 3: Rucksack Reorganization

3.1 Puzzle 1

Each rucksack has two large compartments. All items of a given type are meant to go into exactly one of the two compartments. The Elf that did the packing failed to follow this rule for exactly one item type per rucksack.

The Elves have made a list of all of the items currently in each rucksack (your puzzle input), but they need your help finding the errors. Every item type is identified by a single lowercase or uppercase letter (that is, a and A refer to different types of items).

The list of items for each rucksack is given as characters all on a single line. A given rucksack always has the same number of items in each of its two compartments, so the first half of the characters represent items in the first compartment, while the second half of the characters represent items in the second compartment. To help prioritize item rearrangement, every item type can be converted to a priority:

- Lowercase item types `a` through `z` have priorities 1 through 26.
- Uppercase item types `A` through `Z` have priorities 27 through 52.

Find the item type that appears in both compartments of each rucksack. What is the sum of the priorities of those item types?

3.1.1 Full Code

```
# open data file
data = open("3-data.txt", "r").readlines()

# make a list of the element that appears in both halves of each line
matches = []
for x in data:
    half = int(len(x)/2)
    for char in x[:half]:
        if char in x[half:]:
            matches.append(char)
            break

# convert each item in matches to a score
def toScore(l):
```

```
    scores = []
    for i in l:
        scores.append(ord(i) - 96 if i.islower() else ord(i) - 38)
    return scores

print(sum(toScore(matches)))
```

3.2 Puzzle 2

For safety, the Elves are divided into groups of three. Every Elf carries a badge that identifies their group. For efficiency, within each group of three Elves, the badge is the only item type carried by all three Elves. That is, if a group's badge is item type B, then all three Elves will have item type B somewhere in their rucksack, and at most two of the Elves will be carrying any other item type.

The problem is that someone forgot to put this year's updated authenticity sticker on the badges. All of the badges need to be pulled out of the rucksacks so the new authenticity stickers can be attached.

Additionally, nobody wrote down which item type corresponds to each group's badges. The only way to tell which item type is the right one is by finding the one item type that is common between all three Elves in each group.

Priorities for these items must still be found to organize the sticker attachment efforts: here, they are 18 (r) for the first group and 52 (Z) for the second group. The sum of these is 70.

Find the item type that corresponds to the badges of each three-Elf group. What is the sum of the priorities of those item types?

3.2.1 Full Code

```
# make a list of the element that appears in all three items
badgeMatches = []
for i in range(0, len(data), 3):
    for x in data[i]:
        if x in data[i+1] and x in data[i+2]:
            badgeMatches.append(x)
            break

print(sum(toScore(badgeMatches)))
```