Data Structures

Beginning Examples DS + Python + Problem solving

- Programming Problem:
- In basketball, three plays score points: a three-point shot, a two-point shot, and a one-point free throw.
- You just watched a basketball game between the A's and the B's and recorded the number of successful three-point, two-point, and one-point plays for each team. Indicate whether the game was won by the A's, the game was won by the B's, or it was a tie.

- Input:
- There are six lines of input
- The first three give the scoring for A’s and latter three for B’s
- Thoughts:
  - The first line gives the number of successful three-point shots for A’s
  - The second line gives the number of successful two-point shots for A’s
  - The third line gives the number of successful one-point free throw by the A’s
  - The fourth line gives the number of successful three-point shots for the B’s
  - The fifth line gives the number of successful two-point shots for the B’s.
  - The sixth line gives the number of successful one-point free throws for the B’s
- Rule:
  - Each number is an integer between 0 and 100

- Output:
- The output is a single character
  - If the A’s scored more points than the B’s - output A
  - If the B’s scored more points than the A’s - output B
  - If the A’s and B’s scored the same - output T (for Tie)

- So - input should be an int
- Read in 6 integers
- Use variables to store value
- Logic:
  - multiply successful three-point shots by 3
  - multiply successful two-point shots by 2
  - use print to output A, B or T
  - need to think about how our program will make these decisions
  - remember a computer needs to be told everything, we can’t assume it knows anything at this point

- Let’s work through some design test cases before coding
- Data: { 5, 1, 3} - A’s
- Data: {1, 1, 1} - B’s
- A’s = 5 * 3 + 1 * 2 + 3 = 20 points
- B’s = 1 * 3+ 1 * 2 + 1 = 6 points
- Winner would be - A’s
- Next test case:
- Data: { 1,1,1} - A’s
- Data: {5,1,3} - B’s
- Winner would be - B’s

- The solution must be able to compare the total points by the A's and by the B's - the use the result to compare whether the output is A, B or T
- Of course, we will use conditionals - we need to control the flow of our code so if, elif, else or new in Python 3 - switch case statements
- Python has int, Boolean, float, str
- We can use a function called type(9.5) to give is the type like:
- print(type(9.5))

- Python has relational operators to check for > < = // != and == equivalence
- The word ‘in’ can be used on strings - it returns True if the first string occurs at least once in the second - otherwise False
- ‘ppl’ in ‘apple’
- True
- ‘ale’ in ‘apple’
- False

- So what’s the output:

```
a = 3
b = (a != 3)
print(b)
```

True / False / 3 / syntax error ??

- What’s the value of x after execution?

```
x = 5
if x > 2:
    x = -3
if x > 1:
    x = 1
else:
    x = 3
```

Is it -3 / 1 / 2 / 3 / 5

- Temperature is a number
- Do these print the same output:

```
if temperature > 0:
    print('warm')
elif temperature == 0:
    print('zero')
else:
    print('cold')
```

```
if temperature > 0:
    print('warm')
elif temperature == 0:
    print('zero')
print('cold')
```

- Back to our problem code:

```
a_three = int(input(“Enter A’s threes ”))
a_two = int(input(“Enter A’s twos “))
a_one = int(input(“Enter A’s ones “))

b_three = int(input(“Enter B’s threes ”))
b_two = int(input (“Enter B’s twos ”))
b_one = int(input (“Enter B’s ones ”))
```

- Next, we need to determine the number of points scored by the A's and the B's.
- For each team, we add the points from three-point, two-point, and one-point plays.
- We can do that as follows:

```
a_total = a_three * 3 + a_two * 2 + a_one
b_total = b_three * 3 + b_two * 2 + b_one
```

- Next, we produce the output.
- If the A's win, we output A; if the B's win, we output B; otherwise, we know that the game is a tie, so we output T.
- We use an if statement to do this, as follows:

```
if a_total > b_total:
    print('A')
elif b_total > a_total:
    print('B')
else:
    print('T')
```

- OK - simple - let’s go further
- Let’s use some common Python data structures - a list and a dictionary
- List - similar to an array
- Dictionary - key value pairs
- Add to the problem -
  - Handling multiple games - instead of just one game - let’s assume the user will handle multiple games
  - Each game has the same input structure - you just need to determine the winner or tie
  - Store results - let’s store the results of wins, loses and ties - output an overall summary
  - And let’s add some data structures - use lists and dictionaries to store and process the input data, handle multiple games and summarize the results

- So:
- Input
  - The first input line is an integer ‘n’ - number of games
  - For each game, six lines of input
- Output
  - After processing all games - output:
    - Number of games won by A’s
    - Number of games won by B’s
    - Number of games ended in a tie

```
N = int(input(“Enter the number of games: “)
results = {‘A’: 0, ‘B’, 0, ‘T’: 0}
for x in range(N):
        a_scores = [
                int(input(“A’s 3-points: “))
                int(input(“A’s 2-points: “))
                int(input(“A’s 1-points: “))
        ]
        b_scores = [
                int(input(“B’s 3-points: “))
                int(input(“B’s 2-points: “))
                int(input(“B’s 1-points: “))
        ]
```

```
# Calculating total scores
    a_score = a_scores[0] * 3 + a_scores[1] * 2 + a_scores[2] * 1
    b_score = b_scores[0] * 3 + b_scores[1] * 2 + b_scores[2] * 1

    # Determine the result for this game
    if a_score > b_score:
        results['A'] += 1  # A’s won
    elif b_score > a_score:
        results['B'] += 1  # B’s won
    else:
        results['T'] += 1  # Tie
```

```
print(f"A’s won {results['A']} game(s)")
print(f"B’s won {results['B']} game(s)")
print(f"{results['T']} game(s) ended in a tie")
```

- Now our code ->
- Handling Multiple Games:
- The first line of input indicates the number of games.
- A loop runs for each game, reading the input and calculating the scores.
- Data Structures:
- List for Scores:
- Each team's scores for three-pointers, two-pointers, and free throws are stored in a list.
- This allows easy calculation of the total score.
- Dictionary for Results:
- The results of all games are stored in a dictionary (results).
- The keys are 'A' for A’s, 'B' for B’s, and 'T' for ties, with values representing the count of each outcome.

- Score Calculation:
- The total score for each team is computed using the same logic as before.
- The list allows you to easily access and sum up the points.
- Result Processing:
- After calculating the scores for each game, the result is updated in the results dictionary.
- After all games have been processed, the summary of the results is printed, showing how many games each team won and how many were ties.

- What about making this OOP?
- We could define classes to encapsulate the functionality related to the game, teams and scoring system.
- Grab the code from canvas - GameOOPV1.txt
- Add it to a project in PyCharm
- Let’s review it