**Names:** Karthik Bhargav and Vivian Feng

**Date:** 1/24/2020

**Period:** 7

**Problem:** Lab 18 Plan

**Purpose**

Automatically calculate a player’s final score in a round of bowling given the number of pins knocked down in each frame inputted to the algorithm in a space-separated text file with 10 lines, one line for each frame.

**Examples of possible scores**

**normal case: no extra roll at the end, spares in middle**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Frame** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Pins knocked down** | 8 | 4 | 5 | 6 | 7 | 9 | 6 | 5 | 3 | 2 |
| 1 | 3 | 3 | 3 | 2 | / | 1 | 3 | 2 | 5 |
| **Cumulative score** | 9 | 16 | 33 | 42 | 51 | 67 | 74 | 82 | 87 | 94 |

**special case: all strikes with strike at end**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Frame** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Pins knocked down** | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| X | X | X | X | X | X | X | X | X | 10 |
| 10 |
| **Cumulative score** | 30 | 60 | 90 | 120 | 150 | 180 | 210 | 240 | 260 | 300 |

**spare at end**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Frame** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Pins knocked down** | 8 | 4 | 5 | 6 | 7 | 9 | 6 | 5 | 3 | 2 |
| 1 | 3 | 3 | 3 | 2 | 1 | 1 | 3 | 2 | 8 |
| 7 |
| **Cumulative score** | 9 | 16 | 33 | 42 | 51 | 67 | 74 | 82 | 87 | 94 |

**Class Headers, Method Headers, and Private Fields**

**Method Headers:**

* open() - open file with scores
* read() - read opened files
* split() - convert read scores into a list of scores
* int() - change type of score into an integer

**Class Headers:**

* No classes used

**Private Fields:**

* No private fields

**Algorithm**

**Natural Language:**

To calculate the score, first, we must read the score from the file its in. Then, we can place the pins knocked down in each roll into one list. We can use a variable to keep track of the total. To account for special cases, we will use a counter variable to keep track of the index and a while loop to stop iterating. Iterating through each element on the list, we add them to the total score. However, before adding an element to the list, we have to check the elements after the current element to ensure that there isn’t a strike or spare. If it is spare, the values of the next two elements are added along with the current element. We have to skip the element directly after the current element though, to prevent double counting. If an element has a value of 10(a strike), we add the values of the next two roles but do not skip any elements as we did with a spare because a frame with a strike only has one roll. Also, when we get to the last three rolls, we will no longer look for strikes or spares because there are only 10 frames in bowling, and the extra roll(s) in the last frame is solely for the purpose of giving the bonuses in case of a strike or spare on the last frame.

**Pseudocode:**

* Read scores to a list, scoreList
* Set total = 0
* Set index counter(index) to 0
* While the index is less than the length of the list:
  + Check to make sure it is not the last three elements. If not:
    - Check if the roll’s value(scoreList[index]) is 10:
      * If so, increase the total by current score and the values of the next two rolls
      * increase index by 1
    - Else check if scoreList[index] + next element == 10:
      * This is a spare, so total is increased by scoreList[index] + next two elements
      * increment index by 2(skipping next element to prevent overcount)
    - if the first two conditions do not apply
      * add roll value to score
      * increment index by 1
  + If the index is at last three elements
    - if first of the last three elements is 10(strike):
      * increase the total by the values of the last three elements
      * increase index by 3
    - else if the first two of last three elements add to 10(spare):
      * increase the total by value of the sum of the last three elements
      * increase index by three
    - if none of the above conditions apply(normal roll):
      * increase the total by values of the last two roll
      * increase index by two
      * end loop

**Code:**

#read file from txt and coverts to list, can be replaced with other methods

scorefile = open(filename, 'r').read()

strscorelist = scorefile.split()

scorelist = list(map(int, strscorelist))

# as long as the score is not 10, go through elements in pairs

total = 0

index = 0

while index < len(scorelist):

# as long as the index if less than len(scorelist)-3, do this

if index < len(scorelist) - 3:

# if the score is 10, add the next three elements of the list to the total score and move on to next score

if scorelist[index] == 10:

total += (scorelist[index] + scorelist[index + 1] + scorelist[index + 2])

index += 1 # increment counter by 1

# if a pair sums to 10, take the roll after the second score in the pair and add it to the total along with pair

elif scorelist[index] + scorelist[index + 1] == 10:

total += (scorelist[index] + scorelist[index + 1] + scorelist[index + 2])

index += 2

# else, add pair to final score

else:

total += (scorelist[index] + scorelist[index + 1])

index += 2

# when scorelist gets to last three elements, check if scorelist[-3] == 10

else:

# if so, total += all of the last three

if scorelist[index] == 10:

total += scorelist[index] + scorelist[index + 1] + scorelist[index + 2]

index += 3

# else if -3 and -2 pair add to 10, do same

elif scorelist[-3] + scorelist[-2] == 10:

total += scorelist[index] + scorelist[index + 1] + scorelist[index + 2]

index += 3

# else, just add pair to total score

else:

total += scorelist