10/9/25

15-110 Recitation Week 7

Reminders

- Tues, Oct. 21 - Check3/HW3 revisions due (Tuesday after break)
- Recitation feedback form
- Have a restful and rejuvenating break!

Overview

- Big-O Exercise
- For-Iterable Loop Review
- Dictionary Code Writing
- Tree Code Writing

‘

Problems

BIG-O EXERCISE

Calculate the Big-O for the following examples:

Returning the last character in a string S[-1]	

`# input size = n`
`def powersOfTwo(n):`
`  m = 1`
`  while m <= n:`
`  print(m)`
`  m = m * 2`	

`# .index(), .pop() are O(n) worst case!`
`# input size = len(L) = n`
`def foo(L):`
`  if L == []:`
`  return 0`
`  else:`
`  L.append(L[0])`
`  n = L.index(10)`
`  L.pop(0)`
`  return n`	

`# L is a n by n 2D list`
`# input size = n`
`def tripleLoop(L):`
`  for i in range(20):`
`  for row in L:`
`  for elem in row:`
`  print(elem)`	

FOR-ITERABLE LOOP REVIEW

Notes:

Use this code to answer the following questions:

```
s = "15-110"
for i in range(len(s)):
    print(i)
for i in s:
    print(i)
```

What does each loop print?

What is the data type of i in each loop?

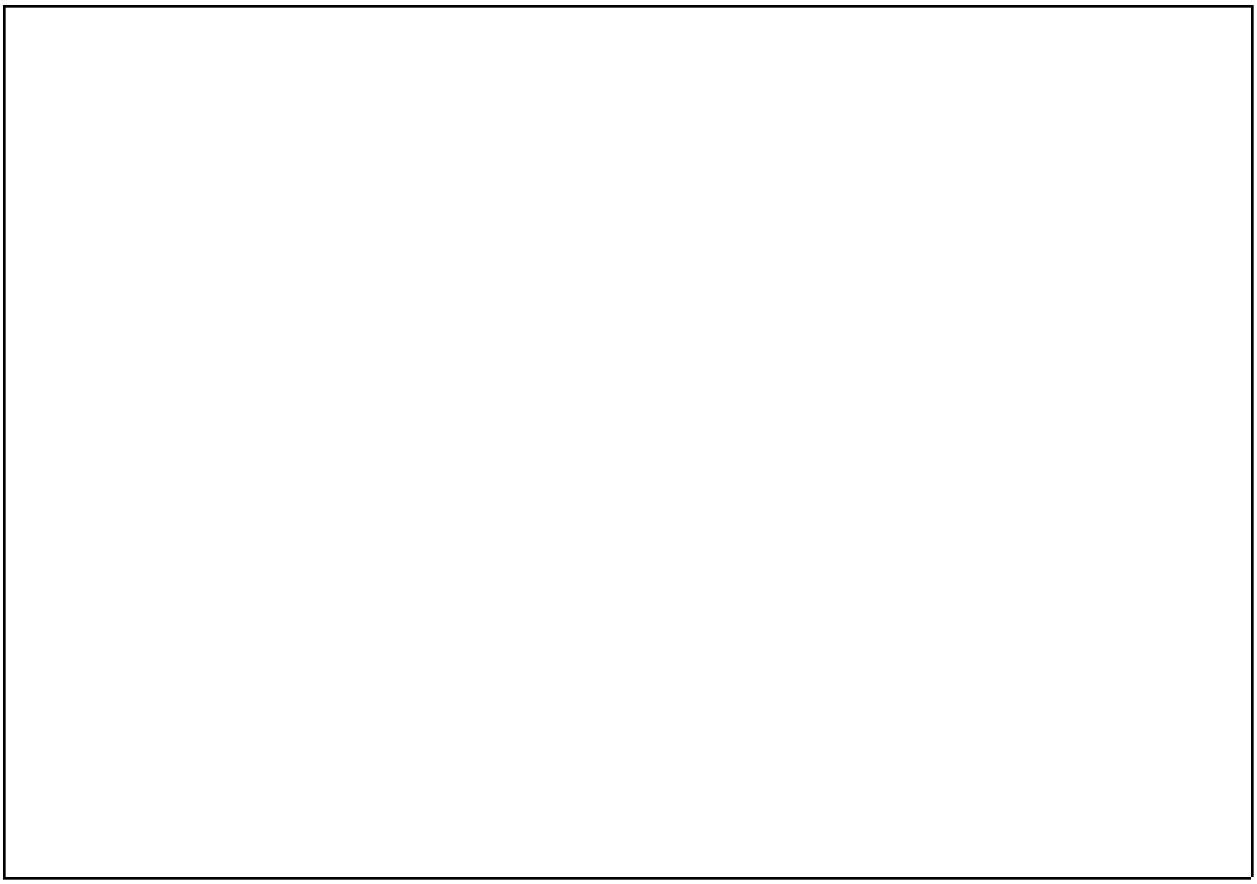
DICTIONARY CODE WRITING

We're given a dictionary that maps some number of football teams (e.g. CMU, Pitt, OSU, PennState) to the number of wins and losses they have (represented as [wins, losses]), and an integer representing the minimum number of games to be considered. We want to return the team with the highest percentage of wins and that has played the minimum number of games. There will be no ties.

E.g. bestTeam({ "CMU" : [1, 10], "Pitt" : [7, 10], "OSU" : [10, 6], "PennState" : [2, 1] }, 5) returns "OSU"

Look at the following pseudocode and try and translate it into a working Python function:

```
def bestTeam(winsLosses, minGames):

    # initialize variables: best team, best percent

    # loop through all teams

          # determine wins and losses

          # determine games played

          # check if team has played enough games

                # calculate percentage of wins

                # check if the percentage is best we have seen so far

                      # if so: reset best percent

                      # if so: reset best team

    # return best team
```



TREE CODE WRITING

Write the function addLeaves(tree) that takes in a dictionary representation of a tree and returns a **sum** of the **values** held by **leaves**. There is a space for you to try writing the pseudocode first, if you found that helpful in the previous problem.

Pseudocode	Code

Suppose now we were to write the function addEvenLeaves(tree) that takes in a dictionary representation of a tree and returns a **sum** of only the **even values** held by **leaves**. How would we modify addLeaves to do this?
