Exam 3 Review

COSC 101: Intro to Computing I
November 13, 2017
The following statements have already been executed:

\[
a = 'lace'
\]
\[
b = [18, 'alan', 20, 'love']
\]
\[
c = { 'don':38, 'ada':15, 'alan':12}
\]

What is the output of:

\[
b[1] \text{ in } c
\]
The following statements have already been executed:

\[
\begin{align*}
a &= 'lace' \\
b &= [18, 'alan', 20, 'love'] \\
c &= \{ 'don':38, 'ada':15, 'alan':12}\end{align*}
\]

What is the output of:

\[
b[-4] - \text{len}(c)
\]
The following statements have already been executed:

```python
a = 'lace'
b = [18, 'alan', 20, 'love']
c = { 'don':38, 'ada':15, 'alan':12}
```

What is the output of:

```python
12 in c or 'a' in c
```
The following statements have already been executed:

```python
a = 'lace'
b = [18, 'alan', 20, 'love']
c = {'don':38, 'ada':15, 'alan':12}
```

What is the output of:

```python
'ada' in c.values()
```
The following statements have already been executed:

```python
a = 'lace'
b = [18, 'alan', 20, 'love']
c = {'don':38, 'ada':15, 'alan':12}
```

What is the output of:

```python
b.index(c['don']-b[2])
```
The following statements have already been executed:

```python
a = 'lace'
b = [18, 'alan', 20, 'love']
c = {'don':38, 'ada':15, 'alan':12}
```

What is the output of:

```python
sum(b[::2])
```
The following statements have already been executed:

```python
a = 'lace'
b = [18, 'alan', 20, 'love']
c = {'don':38, 'ada':15, 'alan':12}
```

What is the output of:

```python
a[c[b[1]]-10] = 'd'
```
What does this print?

def energy(tmp):
    print("t1 =", tmp)
    res = "tea"
    tmp = res
    tmp = "latte"
    print("t2 =", tmp)
    return res

tmp = "coffee"
res = energy(tmp)
print("Either =", res, "or", tmp)
What does this print?

def mess(x, y):
    print("X=", x)
    y["bar"] = x[:]
    y["foo"].append(3)
    tmp = y["foo"]
    del y["foo"]
    y = tmp
Z = [1,2]
y = {"foo":z, 'bar':[0,1]}
mess(z, y)
print("z=", z, "\ny=", y)
What does this print?

```python
def loops(lists):
    new_list = []
    i = len(lists)-1
    while i > 0:
        new_list.append([])
        for j in range(len(lists[i])):
            k = lists[i][j]
            if k % 2 == 0:
                l = 2
                new_list[len(lists)-i-1].append(k)
            else:
                l = 1
                new_list[len(lists)-i-1].append(lists[i][j])
        i -= l
    return new_list

a = [ 1, 2, 3 ]
b = [ 4, 5, 6 ]
c = [ 7, 8, 9 ]
d = [ a, b, c ]
print( loops(d) )
print( d )
```

```
Suppose there is a file named `attendances.txt` which contains information about the participants at each event of a conference:

- #Beyond the Imposter Syndrome
  32 students
  4 faculty
  10 industries

- #Diversifying Computing Panel
  15 students
  20 faculty

- #Movie Night
  52 students

Write a function called `most_attended` that takes as a parameter the string of a filename (which has to follow the format above). The function reads the file and returns the name of the event that has the highest number of participants.

For the above file, for instance, `most_attended` should return 'Movie Night' (52 attendees vs. 46 and 35).
Write a function that accepts a list of strings that contain a comma-separated pair of "City, Country Abbreviation" names like:

```plaintext
```

and returns a list of country abbreviations for those countries that have at least two cities in the list. The order of the list can be arbitrary. Given the above list the function should return:

```plaintext
[ 'US', 'UK' ] or [ 'UK', 'US' ]
```

For each string in the parameter list, you can assume that the country abbreviation is the last two characters in the string.
Since you walk or bike most everywhere, you’re a bit of a weather forecast junkie. Someone gives you a Python list with the forecast for the next several days, as in the following:

['Monday: Sunny, high of 35', 'Tuesday: 20 degrees and snow', 'Wednesday: Partly cloudy, high of 37', 'Thursday: High of 25. Sunny in the morning, changing to snow']

Write a function `average_high` that takes a list of forecasts (list of strings), and returns the average forecast high temperature as a floating point number. For the above list, the function should return 29.25 (\( \frac{35 + 20 + 37 + 25}{4} \)).

You can assume that there is only one number in each string, and it refers to the forecast high. Note that the number can appear anywhere within the string. *Hint:* the `isdigit` method might come in handy.

Your solution must consist of at least two functions, the `average_high` function described above, plus at least one “helper” function.