CSE 160 Spring 2024 - Midterm Exam

Instructions:
- You have **40 minutes** to complete this exam.
- The exam is **closed book**, including no calculators, computers, phones, watches or other electronics.
- You are allowed a single sheet of notes for yourself.
- We also provide a syntax reference sheet.
- Turn in all sheets of this exam, together and in the same order when you are finished.
- When time has been called, you must put down your pencil and stop writing.
  - Points will be deducted if you are still writing after time has been called.
- You may only use parts and features of Python that have been covered in class up to this point.
- You may ask questions by raising your hand, and a TA will come over to you.

Good luck!

<table>
<thead>
<tr>
<th>Question</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>Expressions</td>
</tr>
<tr>
<td>Question 2</td>
<td>Loops, Ifs</td>
</tr>
<tr>
<td>Question 3</td>
<td>Lists</td>
</tr>
<tr>
<td>Question 4</td>
<td>Files</td>
</tr>
<tr>
<td>Question 5</td>
<td>Dictionaries, Lists</td>
</tr>
</tbody>
</table>
**Question 1)** For each of the below expressions, write what the expression evaluates to and the type of that value. You should assume that variables have been declared and assigned as follows:

```python
a = "cse160 is fantastic"
b = 10
c = 3.6
d = True
e = ["you're", "going", 2, "be", "amazing"]
f = [10, 7, 3, 6, 9]
g = {"is": "python", 2.0: "fun"}
```

If evaluating the expression would result in an error, write "Error" in both the value and type columns.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value of output</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>output = 8 + b * c</td>
<td>44</td>
<td>float</td>
</tr>
<tr>
<td>output = a[3]</td>
<td>&quot;1&quot;</td>
<td>string</td>
</tr>
<tr>
<td>output = str(b) + &quot; &quot; + e[2]</td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>output = b % 2 == 0 and not d</td>
<td>False</td>
<td>boolean</td>
</tr>
<tr>
<td>output = f.sort()</td>
<td>None</td>
<td>NoneType</td>
</tr>
<tr>
<td>output = a + 'Let's do it!'</td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td>output = e[-1] + g[&quot;is&quot;] + g[b / len(f)]</td>
<td>&quot;amazingpythonfun&quot;</td>
<td>string</td>
</tr>
</tbody>
</table>
**Question 2**) For all sub-parts (a through c) of this question, assume that *words* is defined as follows:

```
words = ["Python ", "in a ", "expanse ", "dance ", "is a ", "digital "]
```

a) What is the output of the following code snippet?

```python
poem = ""
for i in [0, 4, 3, 1, 5, 2]:
    poem += words[i]
poem += "."
print(poem)
```

*Python is a dance in a digital expanse .*

b) What is the output of the following code snippet?

```python
for i in [0, 4, 3, 1, 5, 2]:
    if i % 2 == 0:
        print(words[i])
    else:
        line = ""
        for j in range(i):
            line += words[i]
        print(line)
```

*Python
is a
dance dance dance
in a
digital digital digital digital digital expanse*
c) Given the following list `words2`, write a short program that will find all the strings that occur in both lists, `words` and `words2`, and print them out in a list. To receive full credit, you **must** use `range()`

The `words` list is the same as it was on the previous page:

```
words = ["Python ", "in a ", "expanse ", "dance ", "is a ", "digital "]
```

And `words2` is defined as:

```
words2 = ["in a ", "digital ", "space ", "Python ", "has ", " a ", " place"]
```

The output should be: ['Python ', 'in a ', 'digital ']

```
shared_words = []

for i1 in range(len(words)):
    for i2 in range(len(words2)):
        if words[i1] == words2[i2]:
            shared_words.append(word1)

print(shared_words)
```
Question 3) What is the output of the following code?

```python
foods = ['beef', 'burger', 'sushi', 'fries', 'carrot', 'pizza']
result = []
for i in range(0, 6, 2):
    result.append(foods[i][-1])
result = result + ['z', 'is']

if (len(foods) % 2) == 0:
    result.extend('awesome')
else:
    result.insert(0, 'cool')

print(result)

['f', 'i', 't', 'z', 'is', 'a', 'w', 'e', 's', 'o', 'm', 'e']
```
**Question 4)** Given a list of lists of integers, where each list contains all the exam scores of a class, write code that writes to a file "max_scores.txt" that contains a line **Class <class number>'s maximum score is <max_score>** for each class. You may not use the max built-in python function. You may assume that the `class_scores` variable is already defined.

For example, given

```python
class_scores = [[66, 78, 43], [33, 12, 99], [67, 87, 2]]
```

The file `max_scores.txt` should contain the following content after running your code:

```
Class 1's maximum score is 78.
Class 2's maximum score is 99.
Class 3's maximum score is 87.
```

```python
f = open("max_scores.txt", "w")
for i in range(len(class_scores)):
    max_score = 0
    for score in class_scores[i]:
        if score > max_score:
            max_score = score
    f.write("Class " + str(i + 1) + ", maximum score is "
            + str(max_score) + "\n")
f.close()
```
Question 5) A vet clinic wants you to create a function `average_weight()` that will find the average weight of their dogs in order to track the changes in canine obesity overtime. It will take in a list of dictionaries (patient_data) and return a float. You can assume that this is the only species that we care about.

For example if given the list:

```python
[{'Patient': 'Lucky', 'Height': 23, 'Weight': 30, 'Species': 'Dog'},
{'Patient': 'Remi', 'Height': 30, 'Weight': 40, 'Species': 'Dog'},
{'Patient': 'Licorice', 'Height': 10, 'Weight': 15, 'Species': 'Cat'},
{'Patient': 'Shadow', 'Height': 15, 'Weight': 25, 'Species': 'Cat'}]
```

The function would return:

```python
def average_weight(patient_data):
    number_of_dogs = 0
    total_weight = 0
    for i in patient_data:
        if i['Species'] == 'Dog':
            number_of_dogs += 1
            total_weight += i['Weight']
    return total_weight / number_of_dogs
```

35.0