Name: _____

Email Address (UW Net ID): _____ @uw.edu

Section: _____

CSE 160 Spring 2023 - Final Exam

Instructions:

- You have **110 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.
- All questions assume Python version 3.7, as we have been using all quarter.
- You may ask questions by raising your hand, and a TA will come over to you.
- If we discover typos or useful clarifications during the exam, we will add them to an Errata document presented on the screens.

Good luck!

1. Read the following code and traceback. Then, in the space below, explain the cause of the error

Code Snippet

```
 1  def foo():
 2      l = list()
 3      odd_numbers = [1, 3, 5, 7, 9]
 4      even_numbers = [2, 4, 6, 8, 10]
 5      l.append(odd_numbers)
 6      l.append(even_numbers)
 7      baz(l, None)
 8      bar(l)
 9
10
11  def bar(l):
12      d = dict()
13      d["odd_numbers"] = l[0]
14      d["even_numbers"] = l[1]
15      d["numbers"] = l
16      baz(l, d)
17
18
19  def baz(l, d):
20      s = set()
21      if d is not None:
22          s.add(tuple(d.keys()))
23          s.add(d["numbers"][1])
24      else:
25          s.add(l[0][0])
26
27
28  def main():
29      foo()
30
31
32  if __name__ == "__main__":
33      print("Welcome to the Final!")
34      main()
```

Traceback

```
Welcome to the CSE 160 Final!
Traceback (most recent call last):
  File ".../error.py", line 34, in
<module>
    main()
  File ".../error.py", line 29, in main
    foo()
  File ".../error.py", line 8, in foo
    bar(l)
  File ".../error.py", line 16, in bar
    baz(l, d)
  File ".../error.py", line 23, in baz
    s.add(d["numbers"][1])
TypeError: unhashable type: 'list'
```

2. For parts 1 and 2 of this question, consider the following code:

```
moe = 5.5

def eeny(num):
    if (num - 5) > 0:
        return meeny(moe)
    else:
        return meeny(num) * 20

def meeny(moe):
    if moe % 2 == 0:
        return miny(2, 6) * 2
    else:
        return miny(moe, 3) * 20

def miny(num, moe):
    if moe * num > 12:
        return 12
    else:
        return moe * num
```

Part 1:
For each of the following, what will print out?

Code	**Output**
`print(eeny(10))`	
`print(eeny(4))`	
`print(eeny(-1))`	

Part 2:
Provide another example call to `eeny()` that would print out the number `480`. **Note**: You may not use any of the calls to eeny from the previous question. E.g., you cannot write `print(eeny(10))`, but you could write `print(eeny(9))`.

3. Write a function `divisible` that, given two lists of integers `products` and `factors`, *returns* the number of ints in `products` divisible by any one of the factors.

   Examples:
   `divisible([36, 45, 7], [3, 11])` returns `2`
   Explanation: 36 is divisible by 3 and 45 is divisible by 3
   `divisible([36, 45, 7], [3, 5])` returns `2`
   Explanation: 36 is divisible by 3 and 45 is divisible by 3 or 5, but is still only counted once
   `divisible([36, 45, 48], [2, 3, 6])` returns `3`
   Explanation: 36 is divisible by 2,3, and 6; 45 is divisible by 3; 48 is divisible by 2, 3 and 6
   `divisible([36, 45, 48], [7, 11])` returns `0`
   Explanation: None of the numbers are divisible by 7 or 11

   Even if a product is divisible by multiple factors, it should only be counted once (see examples 2 and 3).

```
def divisible(products, factors):
      # Write your code below this line
```

4. Write 3 test cases for the following function that test 3 different cases. Please specify what the cases are testing (as in, why they are different from the other two tests) as well as the value of the input variables. Remember to clearly define the input you will use to call the function!

```
def find_common(TA_food, TA1, TA2):
      """
      Parameters:
            TA_food: dictionary where the keys are TAs, and the
      values are a set of the TA's favorite food
            TA1: a string representing a TA in TA_food
            TA2: a string representing another TA in TA_food

      Returns: a set of the food items that both TAs like. If
      the two TAs do not have any common favorite foods, return
      an empty set.

      Example:
      >>> TA_food = {'Sneh': {'pizza', 'pasta'}, 'Zoe':
{'pizza', 'sushi'}, 'Sierrah': {'Apple pie', 'chicken
teriyaki'}}
      >>> find_common(TA_food, 'Sneh', 'Zoe')
      {'sushi', 'pizza'}
      """
```

5. A researcher has written the function below to be able to find how many times a specific string repeats in a list:

```
def CountTotals(list, i):
      count = 0
      for current_string in list:
            if current_string == i:
                  count += 1
      return count
```

What three main style issues should the researcher fix, and give an example of what should be changed to fix the issues.

Style Issue	**Example Fix**

6. For parts 1 and 2 of this question, you will write and use a class designed to store information about certain kinds of songs.

   Part 1: Complete the missing two class functions below. The descriptions for each function are provided in the docstrings.

```
class EurovisionSong:
    def __init__(self, name, length, country):
        """
        length is given in seconds (int)
        name and country are given as strings
        """
        # Fill in the initialization for the three provided
attributes:
```

```
    def get_length(self):
        """
        Returns the length of the song as a tuple (hours,
minutes, seconds).
        For example:
        >>> s = EurovisionSong("Heart of Steel", 154,
"Ukraine")
        >>> s.get_length()
        (0, 2, 34)
        """
        # Write your implementation here:
```

Part 2: Write a function that takes a list of EurovisionSong instances and returns a dictionary mapping country names (keys) to the name of the longest song for that country.

For example

```
songs = [
  EurovisionSong("Ein bisschen Frieden", 197, "Germany"),
  EurovisionSong("Fairytale", 183, "Norway"),
  EurovisionSong("Hard Rock Hallelujah", 204, "Finland"),
  EurovisionSong("Waterloo", 169, "Sweden"),
  EurovisionSong("Satellite", 173, "Germany"),
  EurovisionSong("La det swinge", 180, "Norway")
]
map_songs(songs)
```

would return

```
{
    "Germany": "Ein bisschen Frieden"
    "Norway": "Fairytale",
    "Finland": "Hard Rock Hallelujah"
    "Sweden": "Waterloo"
}
```

```
def map_songs(songs):
    # Write your implementation here:
```

7. We've written the following function to help us find the longest name in a text file listing the names of some CSE160 staff members:

```
def longest_name(data):
    """
    Returns the longest name in the given parameter file. If
there is a
    tie for longest name, returns the first name with the
longest length.
    """
    myfile = open(data)

    longest_name = None

    for persons in myfile:
        for person in persons.split():
            if longest_name is None or len(persons) >
len(longest_name):
                longest_name = person

    myfile.close()
    return longest_name
```

When we run

```
print(longest_name("data.txt"))
```

… where the file named "`data.txt`" has the following contents

```
Fitz Zoe
Paolo Mark Annalisa
Sneh Amanda Erica
```

… the code prints "Erica", which has 5 letters, while the longest name we are looking for is "Annalisa", which has 8 letters. Take a look at the function's implementation and circle which line of code is causing the bug. Then, below, explain how we can fix the bug so that our code produces the correct output.

Extra Credit (1pt)

Draw a pokemon. Drawing anything for this question gets you the point.

Which problem did you hate the most?