CSE 160 Section 7 Problems

1. For each line of code, determine if the code is reassigning a variable or mutating an object.

Code	Reassigning/Mutating
`my_variable = 2`	reassigning
`favorite_characters = ['Aether', 'Karma', 'Ukyo']`	reassigning
`least_favorite = favorite_characters`	reassigning
`best_friends.append('Ash')`	mutating
`lst1 = mystery1()`	reassigning
`lst1[2] = 'Xiao'`	mutating
`classes = {'cse160': 'Data Programming',` `'cse154': 'Web Programming'}`	reassigning
`favorite_numbers['Chao'] = 43`	mutating

2. Given the following code, figure out what would be printed out at the numbered lines.

```
students = ['Jakob', 'Marty', 'Karl', 'Xander']
class2 = students
print(students is class2) # 1 True
print(students == class2) # 2 True
class2.append('Joe')
print(students == class2) # 3 True
print(students[-1]) # 4 Joe
class3 = class2[:]
print(class2 is class3) # 5 False
print(class2[:] == class3[:]) # 6 True
sorted_students = sorted(students)
print(students == sorted_students) # 7 False
print(class2 is sorted_students) # 8 False
students.sort()
print(class2 is students) # 9 True
```

3.

a. Given the following functions and code, figure out what would be printed.

```
def increment_dict(dict, key):
    dict[key] = dict[key] + 1

def increment_list(list, idx):
    list[idx] = list[idx] + 1

def increment_number(number):
    number = number + 1

coins = {'Sam': 1}
increment_dict(coins, 'Sam')
print(coins) {'Sam': 2}

numbers = [1, 2, 3]
increment_list(numbers, 1)
print(numbers) [1, 3, 3]

my_num = 9
increment_number(my_num)
print(my_num) 9
```

b. The intent of these three functions is to increment the value by one. If any of the examples did not do this successfully, change the code such that it actually works.

```
def increment_number(number):
    return number + 1
# ...
my_num = increment_number(my_num)
```

4. Consider the following function `average`, which when given a list of numbers computes the average. For example, `average([1, 2, 3, 4, 5])` would return `3.0`

```
def average(nums):
    total = 0
    for n in nums:
        total = total + n
    return total / len(nums)
```

a. Are there any potential problems with this function? What exceptions could be raised, if any?

i. Hint: what happens if you call `average([])`?

If an empty list is passed into the function, we divide by the length, which is 0, causing a ZeroDivisionError.

b. If there are any exceptions, how could you fix the function to avoid this problem? There may be multiple ways to do this.

```
# solution 1
def average(nums):
    if len(nums) == 0:
        return 0.0
    else:
        total = 0
        for n in nums:
            total = total + n
        return total / len(nums)

# solution 2
def average(nums):
    total = 0
    for n in nums:
        total = total + n
    try:
        return total / len(nums)
    except ZeroDivisionError:
        return 0.0
```

5. Write code that will successfully modify the given variable, a list of tuples. After your code executes, each tuple in the list should have the field at index 1 incremented by 1.

```
likes = [('Chewbarka', 163), ('Mary Puppins', 15),
         ('Barkley', 4)]

def increment(likes):
    result = []
    for item in likes:
        result.append((item[0], item[1] + 1))
    return result
likes = increment(likes)
```

6. Write a function `favorite_number` that is given a dictionary of names mapped to their favorite number, and a name. The function should return the favorite number of the given name. If the name is not in the dictionary, a ValueError should be raised. For example:

```
people = {
    'Chongyun': 7,
    'Razor': 1,
    'Amber': 4
}
print(favorite_number(people, 'Amber')) # 4
print(favorite_number(people, 'Xiao')) # ValueError

def favorite_number(people, name):
    if name not in people:
        raise ValueError
    else:
        return people[name]
```

7. You are given a function `do_cool_thing()` that will *usually* do a cool thing, but randomly will throw an exception. Write code that calls this function, and in the event of an exception, prints some text to the console instead of crashing.

```
try:
    do_cool_thing()
except:
    print('uh oh something went wrong')
```

8. Write a function `least_coins` that takes a dictionary as a parameter, and returns the name of the person with the fewest coins. For example, the following code should print “Alex”:

```
coin_totals = {
    'Sam': 12,
    'Ash': 68,
    'Alex': 5,
    'Erin': 10
}
print(least_coins(coin_totals))

def least_coins(totals):
    min_name = ''
    min_coins = 9999
    for name in totals:
        if totals[name] < min_coins:
            min_name = name
            min_coins = totals[name]
    return min_name
```

9. The social media company Yipper has asked you to help debug some code they’ve been having issues with. They’ve given you the following exception:

```
Traceback (most recent call last):
  File "yipper.py", line 21, in <module>
    likes = likes_per_user(yips)
  File "yipper.py", line 14, in likes_per_user
    result[i][1] = result[i][1] + yip['likes']
TypeError: 'tuple' object does not support item assignment
```

a. What does this exception mean? What has gone wrong?

The code tries to mutate a tuple, but tuples are immutable - the values stored in a tuple cannot be changed.

b. The company has given you the following code. How could it be fixed to avoid this error?

One possible solution:

```
# ...
def likes_per_user(yips):
    likes = {}
    for yip in yips:
        name = yip['name']
        if yip['name'] not in likes:
            likes[name] = 0
        likes[name] = likes[name] + yip['likes']
    result = []
    for name in likes:
        result.append((name, likes[name]))
    return result

yips = get_yips()
likes = likes_per_user(yips)
print(likes)
```