PART 2A (3 pts)

The variable result should calculate the list of colors from the list hues that don't end in a vowel.

Using the list named hues and string named vowels below, the variable result would calculate ['cyan', 'teal', 'green'].

```
hues=['orange', 'cyan', 'purple', 'teal', 'blue', 'green']

vowels='aeiou'

result=[color for color in hues if color[-1] not in vowels]
```

PART 2B (4 pts)

The variable result should calculate the list of integers from the list nums whose square is less than 40.

Using the list nums below, the variable result would calculate [4, 1, 6, 3, 5, 0].

```
nums = [4, 1, 10, 6, 8, 7, 3, 5, 0]

result=[x for x in nums if x**2<40]
```

OR

```
result=[x for x in nums if x*x<40]
```

PART 2C (4 pts)

The variable result should calculate the sum of the integers from the list nums that are at least 4 and less than 30.

Using the list nums below, the variable result would calculate 70.

```
[7, 31, 12, 85, 18, 30, 4, 29]

result=sum([x for x in nums if x>=4 and x<30])
```

OR

```
result=sum[x for x in nums if 4<=x<30]
```

PART 2D (4 pts)

The variable result should calculate the list that contains the second letter of each word from the list words.

Using the list named words and string named vowels below, the variable result would calculate ['e', 'o', 'o', 'x', 'r', 'o'].

```
words = ['hey', 'how', 'does', 'exam', 'preparation', 'go']
```

result=[w[1] for w in words]

PART 2E (4 pts)

The variable result should calculate the list of strings from the list words where the first letter is either 'a', 'e', or 'i' of the string vowels.

Using the list words below, the variable result would calculate ['and', 'is'].

['test', 'prep', 'and', 'one', 'for', 'other', 'classes', 'is', 'hard']

vowels = 'aeiou'

result=[w for w in words if w[0] in vowels[0:3]]
OR
result=[w for w in words if w[0] in vowels[:3]]

PART 2F (4 pts)

The variable result should calculate the list of all even integers from the list nums, multiplied by 10.

Using the list nums below, the variable result would calculate [20, 80, 200, 440].

nums = [1, 2, 5, 8, 11, 20, 43, 44]

*result=[x*10 for x in nums if x%2==0]*

PART 2G (4 pts)

The variable result should calculate the list of the remainder of all integers from the list nums divided by four.

Using the list nums below, the variable result would calculate

[1, 2, 1, 0, 3, 0, 3, 2].

nums = [1, 2, 5, 8, 11, 20, 43, 42]

result=[x%4 for x in nums]

Problem 3A (3 points)

Use set operations to determine what is in names3 that is not common across names2 and names1. That is, {'Kia', 'Blake', 'Jermaine'.

names1={**'Aaron', 'Blake', 'Denice', 'Jacqueline', 'Timothy'**}
names2={**'Kim', 'Aaron', 'Erin', 'Tia', 'Jermaine'**}
names3={**'Jermaine', 'Kia', 'Blake', 'Aaron'**}

```
result=names3 - (names2 & names1)
```

Problem 3B (3 points)

Using indexing and tuple operations, update the tuple to (15, [5, 10, 45], 30). This requirement should include how the value 45 is determined.

```
value = (15, [5,10], 30)
```

ACCEPTABLE SOLUTIONS

```
value[1].append(value[0]+value[-1])
value[1].append(value[0]+value[2])
value[1].append(value[0]*3])
```

Problem 3C (3 points)

Using indexing and concatenation, create the string 'yelp'.

```
value = ('yellow', 'aqua', 'pink')
```

```
result=value[0][:3]+value[-1][0]
result=value[0][:3]+value[2][0]
```

Problem 3D (3 points)

Using indexing, create the list
['purple', 'color'].

```
value = (15, 'color', ['purple', 10])
```

SOLUTIONS ACCEPTABLE

```
result=[value[-1][0], value[1]]
result=[value[2][0], value[1]]
```

```
result=list((value[2][0], value[1]))
list((value[-1][0], value[1]))
```

Problem 4A (10 points)

There are several ways they may have solved this. Should you have questions, you can always create a version of this program in PyCharm and run their code to ensure it works correctly. Using the list called inventory that includes the values in the examples:

```
inventory=[["Ernie Barnes", "The Sugar Shack", 1971], ["Romare Bearden",
"Out Chorus", 1980], ["Kehinde Wiley", "The Virgin Martyr St. Cecilia",
2008], ["Jean-Michel Basquiat", "Untitled", 1982]]
```

```
inventory=[["Ernie Barnes", "The Bench", 1959], ["Romare Bearden", "Out
Chorus", 1980], ["Romare Bearden", "Pepper Jelly Lady", 1980], ["Jean-
Michel Basquiat","Untitled", 1982]]
```

```
inventory=[["Jean-Michel Basquiat", "Horn Players", 1983], ["Jean-Michel
Basquiat","Out Chorus", 1980], ["Jean-Michel Basquiat", "Untitled",
1982]]
```

SOLUTION

```
def firstPiece(artists):
    year=2050  #Should be high enough to update
    for item in artists:
        if item[2] < year:
            year = item[2]
    return year
```

Problem 4B (10 points)

There are several ways they may have solved this. Should you have questions, you can always create a version of this program in PyCharm and run their code to ensure it works correctly. Using the list called inventory that includes the values in the examples:

```
inventory=[["Ernie Barnes", "The Sugar Shack", 1971], ["Romare Bearden",
"Out Chorus", 1980], ["Kehinde Wiley", "The Virgin Martyr St. Cecilia",
2008], ["Jean-Michel Basquiat", "Untitled", 1982]]

most=artistCount(inventory, "Romare Bearden")
```

```
inventory=[["Ernie Barnes", "The Bench", 1959], ["Romare Bearden", "Out
Chorus", 1980], ["Romare Bearden", "Pepper Jelly Lady", 1980], ["Jean-
Michel Basquiat","Untitled", 1982]]

most=artistCount(inventory, "Romare Bearden")
```

```
inventory=[["Jean-Michel Basquiat", "Horn Players", 1983], ["Jean-Michel
Basquiat","Out Chorus", 1980], ["Jean-Michel Basquiat", "Untitled",
1982]]

most=artistCount(inventory, "Jean-Michel Basquiat")
```

SOLUTION

```
def artistCount(artists, name):
    total=0
    for item in artists:
```

```
        if item[0]==name:
            total+=1
    return total
```

Problem 5 (10 points)

Create the function **calculateAvg** that has one parameter: **states**, which is a list of tuples that contain a string representing the two-letter state acronym and the yearly average temperature. The function returns the average temperature of all states included in the list. Ignore any formatting to determine a specific number of decimal points (do not worry about this).

call	returns
calculateAvg([("NC", 58), ("GA", 62), ("SC", 61), ("NY", 48)])	57.25
calculateAvg([("MD", 54), ("NC", 58), ("VT", 43)])	51.66
calculateAvg([("NC", 58)])	58.0

Complete the function calculateAvg below.

```
def calculateAvg(states):
```

There are several ways they may have solved this. Should you have questions, you can always create a version of this program in PyCharm and run their code to ensure it works correctly.

```
def calculateAvg(states):
    sum=0
    for i in states:
        sum+=states[1]

    avg=sum/len(states)
    return avg
```

#Part 6 (8 points)

Consider the following program, which is designed to create a file that stores the pay for each student UTA and the total amount paid to all UTAs for week 12 in a file named week12.txt:

```
1    f = open("../../lectures/data/week12.txt", "w")
2    names=["Kim", "Aaron", "Rhonda", "Leslie", "Tonisha", "James"]
3    pay=[326.50, 180, 200.25, 150, 250, 145.50]
4    f.write("Student Pay for Week 12\n")
5    for x in range(len(names)):
6         f.write(names[x]+": $"+str(pay[x])+"\n")
7         total+=pay[x]
8    f.write("Total paid to students for week 12: $"+str(total))
9    f.close()
```

This program does not compile. It produces one error.

#PART 6A (2 points)

Where is the error?

Answer: Line 7 creates the error.

#PART 6B (2 points)

What caused the error?
Answer: Line 7 uses an accumulator for total. However, total is never initialized anywhere in the program. Thus, line 7 attempts to add to the sum of a variable (total) that was never initialized to a starting value.

#PART C (4 points)

Briefly explain how you would fix this code so that it worked as intended.

Answer:
Initialize total=0 anywhere above line 5 (i.e., before the for loop begins).