CS 1: Introduction to Computer Programming

Recitation 3: VPython, Dictionaries, and Loop Applications Solutions

In this third "problem solving session", we will take a closer look at dictionaries, as well as review loops through the lens of real-life math applications! Furthermore, to help prepare you for the upcoming VPython Physics assignment, we have included some material and practice regarding VPython.

Common Errors

VPython Syntax Errors

Input/Output Example

```
1 my_box = box(pos=vec(0, 0, 0) length=5, height=6, width=2, color=color.yellow) # Sometimes,
      it's easy to miss something small with such a long list of arguments
2 my_box = box(pos=vec(0, 0, 0), length=5, height=6, width=2, color=color.yellow) # Does not
      give an error
3 # When you see a SyntaxError message, it's good to check if you missed something small like
      a comma
```

```
>> SyntaxError
>>
```

Treating Dictionaries Like Lists

Input/Output Example

```
1 d = {1: 2, 3: 4, 5: 6}
2 print(d[0]) # The student may write this thinking it will print out (1, 2), or something
      similar
3 # Line 2 will actually give an error. A dictionary is not a list and does not have indices
      for its entries
```

```
>> KeyError
```

Problem Solutions

Introduction to VPython

Worked Example

Box 1

Create a green box centered at (0, 0, 0) with length 4, height 5, and width 3.
Store this box in a variable, and print out its height.

```
b = box(pos=vec(0, 0, 0), length=4, height=5, width=3, color=color.green)
print(b.height)
```

Sphere 1

Create a red sphere centered at (0, 0, 0) with radius 2.
Store this sphere in a variable, and print out its radius. Then, change its center's position to (4, 3, 2).

```
sph = sphere(pos=vec(0, 0, 0), radius=2, color=color.red)
print(sph.radius)
sph.pos = vec(4, 3, 2)
```

Faded Example

Box 2

Create a cyan box centered at (2, 1, 0) with length 3, height 3, and width 7.
Store this box in a variable, and print out its position (as a vector).

```
b = box(pos=vec(2,1,0), length=3, height=3, width=7, color=color.cyan)
print(b.pos)
```

Sphere 2

Create an orange sphere centered at (0, 1, 5) with radius 3.
Store this sphere in a variable, and print out its position (as a vector). Then, double its radius.

```
sph = sphere(pos=vec(0,1,5), radius=3, color=color.orange)
print(sph.pos)
sph.rad *= 2
```

Your Turn

Sphere 3

Create a sphere with a color of your choice, a center of your choice, and a radius of your choice.
Store this sphere in a variable, print out its radius, and change its center's position.

Solution:

```
my_sphere = sphere(pos=vec(1, 2, 3), radius=4, color=color.yellow),
print(sph.radius)
sph.pos = vec(5, 6, 7)
```

Nested Loops With Dictionaries

Worked Example

Simple Thesaurus

Consider the simple thesaurus below, which maps words to lists of synonyms.
Print out all the synonym pairs in the thesaurus (e.g., "happy: jolly").

```
thesaurus = {"happy": ["cheerful", "merry", "jolly", "gleeful"], "sad": ["downcast", "melancholy"],
    "angry": ["indignant", "mad", "furious"]}
for word in thesaurus.keys():
    for synonym in thesaurus[word]: # what does thesaurus[word] retrieve?
        print(word + ": " + synonym)
```

Faded Example

Simple Grocery Store

Consider the nested dictionary below, which maps sections of a grocery store to products and their prices.
Print out the sum a person would have to pay if they bought one of every item in this store.
Then, change the price of an apple to 0.98.

```
groc_prices = {"bakery": {"bagels": 3.99, "bread loaf": 4.49}, "produce": {"apple": 0.67, "
    strawberries": 3.85}, "dairy": {"milk": 3.47}}
total_sum = [0]
for section in [groc_prices.keys()] :
    for product in [groc_prices[section].keys()] :
        [total_sum += groc_prices[section][product]] # update the total sum
print(total_sum)
[groc_prices["produce"]["apple"]] = 0.98 # update the price of an apple
```

Sentence Similarity

The function `sentence_similarity(s1, s2)` takes in two sentences, `s1` and `s2`, as strings and returns a similarity score describing how similar they are.
The dictionary `student_sentences` maps the names of students in a class to a sentence that that student has written. An example of a entry in this dictionary would be `"Hopper": "I am a labradoodle."`.
Print out all the similarity scores between all the different pairs of students in this class.
Order does not matter, and no two students have the same name.

```
visited = [set()]
for student in [student_sentences.keys()] :
    for other_student in [student_sentences.keys()] :
        if [student != other_student] and [student not in visited] and
            [other_student not in visited] :
```

```
5            print(sentence_similarity(  student, other_student  ))
6            visited.add(student)
```

Your Turn

Simple School Roster

Consider the simple school roster below, which is represented by a nested dictionary.
There are students in each class, and each student is assigned an ID by the school.
Create a list and store in it all the student IDs.
Some students appear in multiple classes. Make sure that there are no duplicates in the list.

```
1    roster = {"math": {"Adam": 24123, "Philippe": 82312, "Shallon": 62348, "Maya": 32478}, "CS": {"
         Ellie": 58726, "Reiden": 90823, "Shrishti": 23467, "Aiden": 83921, "Shallon": 62348, "Ying":
          10928}, "physics": {"Neev": 43267, "Ellie": 58726, "Jin": 32498}, "history": {"Bisrat":
         10298, "Felipe": 78234, "Shrishti": 23467}}
```

Solution:

```
1  id_list = []
2  for class in roster.keys():
3     for student in class.keys():
4        curr_id = roster[class][student]
5        if curr_id not in id_list:
6           id_list.append(curr_id)
```

Mathematical Applications of Loops: Sequences and Series

Worked Example

Triangle Sum

Write a function that returns the nth triangle sum (the sum of the integers from 1 to n).
Print out the fifth triangle sum.

```
1     def triangle_sum(n):
2        sum = 0
3        for i in range(n):
4           sum += i+1 # Why is this i+1 and not i? How else could we have written this loop?
5        return sum
6     print(triangle_sum(5)) # What should this print out?
```

Arithmetic Series

Write a function that returns the sum of the first n terms of the arithmetic sequence $2, 6.5, 11, 15.5, ...$
Print out the sum of the first five terms.

```
1     def arith_series(n):
2        sum_n = 0
3        for i in range(n):
4           curr_term = 2 + 4.5 * i # How is each term calculated?
5           sum_n += curr_term
6        return sum_n
7     print(arith_series(5))
```

Faded Example

Factorial

Write a function that returns $n!$ (n-factorial, the product of the integers from 1 to n).
Print out $4!$.

```
def factorial(n):
    product = 1
    for i in n:
        product *= (i+1)  # update the product
    return product
print(factorial(4))
```

Geometric Series

Write a function that returns the sum of the first n terms of the geometric sequence $1, 3/2, 9/4, 27/8, ...$
Print out the sum of the first six terms.

```
def geo_series(n):
    sum_n = 0
    for i in range(n):
        curr_term = 1 * ((3/2) ** i)
        sum_n += curr_term
    return sum_n
print(geo_series(6))
```

Your Turn

Generalized Geometric Series

Write a function that, given a first term and a common ratio, returns the sum of the first n terms of the geometric sequence that results.
Hint: think about what arguments you want your function to take in!

Solution:

```
def gen_geo_series(a, r, n):
    sum_n = 0
    for i in range(n):
        curr_term = a * (r ** i)
        sum_n += curr_term
    return sum_n
```

Applications of Dictionaries

Worked Example

Data Set

The dictionary `data_set` maps values in a data set to the frequency it appears in the set.
For instance, the entry `2:4` means that the value 2 appears 4 times in the set.
Print the sum of all the values in the data set.

```
1  data_set = {1: 3, 2: 4, 3: 1, 4: 1, 6: 2}
2  sum = 0
3  for entry in data_set.keys():
4      sum += entry * data_set[entry] # What is data_set[entry]?
5  print(sum)
```

Faded Example

Weighted Average

Usually, when calculating the average of a data set, you assume every value is "weighted" equally.
Sometimes, however, some values can have a higher or lower weight than others based on their significance.
A weighted average thus considers each weight when calculating the total, and divides the total by the sum of the weights.
The dictionary `weighted_set` maps values in a data set to their weight.
For instance, the entry `7:0.82` means that the value 7 has a weight, or significance, of `0.82`.
Print the weighted average of the data set.

```
1  weighted_set = {5: 1.0, 7: 0.82, 10: 0.47, 13: 1.91, 14: 0.94}
2  total_sum = 0
3  weights_sum = 0
4  for entry in weighted_set.keys():
5      total_sum += entry * weighted_set[entry]  # update total_sum
6      weights_sum += weighted_set[entry]  # update weights_sum
7  print( total_sum / weights_sum ) # print the weighted average
```

Your Turn

Student Grades

A professor uses the `grades` dictionary below to store the scores their students got on the last assessment.
It maps every score to the number of students who received that score.
Print the average grade of the class.
`grades = {64:1, 67:2, 75:1, 77:4, 80:1, 82:1, 85:5, 88:2, 90:5, 92:3, 95:1, 98:1}`
Note: If there are any Python dictionary methods that you feel might help, feel free to use them, but they are not required.

Solution:

```
total = 0
num_students = 0
for score in grades.keys():
    total += score * grades[score]
    num_students += grades[score]
print(total / num_students)
```