**Discussion 9: Dictionaries & List Comprehension in Python**

**List Comprehension Warm-Up**

1. Consider the Snap! code given below:

https://lh6.googleusercontent.com/0SFMTt0xLckCl4OMqrNOR4mIYkDA1ksBe65LIwzYfMmCRTP6FfXPfTfdDtEcGeY0k9eCI_yP4-Pa1y9eMxq_KhscuwTouuDg9f68QtDhUWjWMXbjQxBoMleSmWwTe3fdERBheT8W

Translate this expression into Python using a list comprehension.

Answer:

*def* first\_chr\_of\_letter (*list*):

    i= [i[0] for i in *list* if len(i) > 5 ]

    return(i)

2. Write a list comprehension over list\_ of \_ lists (a list of sub lists) that calculates the sum of each sub list and adds each sum to a new list. You can assume the sub lists in list \_of \_lists contain only numbers.

Answer:

*def* sum\_list\_in\_list(*list*):

    i=[sum(i) for i in *list* ]

    return(i)

3. Write a list comprehension that finds the index of an item in a list. You may assume that the item appears only once in the list. If you get stuck, it may be easier to first write this function using a for loop, then translate your code into a list comprehension.

Answer:

*def* find\_index (*item*, *list*):

    index=0

    for i in *list* :

        if i == *item* :

           index+=1

    return( index )

**Planning Your Phase 1**

1. In the table below, write Python code to execute the listed commands on class\_ dict.

Answer:

class\_dict = {'Math':'1A', 'English':'R1A'}

class\_dict['CS']=['10']  #  Add the key ‘CS’ with the value ‘10’

print(class\_dict('Math')) #Access the value of ‘Math’

class\_dict['Math']='1B'  #Change the value of ‘Math’ to ‘1B’

if 'UGBA' in class\_dict.keys ():  #Change the value of ‘Math’ to ‘1B’

    print("yes")

if '10' in class\_dict.values(): #Check if ‘10’ is a value in class\_dict

    print("YES")

print(class\_dict.keys())  #Get a list of the keys in class\_dict

1. Can you access a key, value pair in a dictionary by its index?

Answer:

No you cannot access a key, value pair in a dictionary by its index because dictionary is unordered.

1. Are keys and/or values in a dictionary returned in a predictable order?

Answer:

No because it is unordered or unindexed.

1. Can dictionaries have duplicate keys? What about duplicate values?

Answer:

No dictionary cannot have duplicate keys. The values are added if you add duplicate keys.

**Dictionary Practice**

Fav \_ numbers = {‘Dan’: 18, ‘Alonzo’: 12, ‘Oski’: 7, ‘Carol Christ’: 152} nums = [7, 12]

1. On the lines below, write Python code that increments each person’s favorite number by the length of their name.

Answer:

*def* FN (*dictionary*):

    i= {a : len (a)+b for a , b in *dictionary* . items()}

    return( i )

2. On the lines below, use a list comprehension to output a list of people whose favorite numbers are in nums.

Answer:

*def* FN (*dictionary* , *list*):

    j= {a : b for a,b in *dictionary* . items()

       if b >=(*list*[0]) and b<=(*list*[1])}

    return(j)

3. Write a function merge \_ dicts that takes two dictionaries as input, and returns a new dictionary that contains all entries from both input dictionaries. Your function should not modify the inputs. You can assume that both input dictionaries have strings as keys and numbers as values. For any keys present in both input dictionaries, the corresponding value in the output dictionary should be the sum of the values in the inputs.

Answer:

*def* merge\_dicts(*d1*, *d2*):

    new\_dict = {}

    for key in *d1*:

       new\_dict[key] = *d1*[key]

       for key in *d2*:

            if key in new\_dict:

             new\_dict[key] += *d2*[key]

            else:

              new\_dict[key] = *d2*[key]

    return new\_dict

4. Assume we have defined food \_ dict in the Python interpreter, as below. What will be

displayed after each of the following lines executes? If the result is an error message, just

write "Error." Assume that the commands are executed independently, NOT sequentially.

>>> food \_ dict = {"fruit": "apple", "veggie": "carrot", "beverage": "water",

"grain": "rice"}

>>> len (food \_ dict)

No error, the result(output) is 4

>>> list (food \_dict)

No error, the keys become the element of list

>>> food \_ dict[0]

Error is appeared, because the dictionary is unordered so we cannot abstract specific key or value by its index.

>>> (‘fruit’ in food \_dict) and (‘apple’ in food \_dict)

The output is ‘False’. No error.

>>> ("fruit" in food \_dict. keys ()) and ("apple" in food \_dict. values ())

No error. Output is ‘True’.

>>> for food in food \_dict:

... food += "s"

>>> food \_dict

Error free. The output is food \_dict without any change.

>>> def recursion \_ is \_ fun (dict1, dict2):

... if dict2 == {}:

... return dict1

... dict2.pop(list(dict2) [0])

... return recursion \_ is \_fun (dict1, dict2)

>>> copy = food \_ dict

>>> recursion \_ is \_ fun (food \_dict, copy)

{} is the output and error will appeared

>>> more \_ food = {"protein": "chicken"}

>>> food \_ dict ["more food"] = more \_ food

>>> food \_ dict

{"fruit": "apple", "veggie": "carrot", "beverage": "water", "grain": "rice", more food: {"protein": "chicken"}}