**Understanding lists and dictionaries**

**Lists** are just that – just like any kind of list, they are words or numbers listed one after another. With computers, the lists need a reference point. Lists start with the number zero. If I had a list called "**my\_list**", it would be defined by "***my\_list = []***", without the quotation marks. Or we could define it with predefined elements – ***my\_list = [52, 19, 32]***

we can then add items to the list by saying (using this example)

***my\_list.append (122)*** The list would then contain the following four elements ***[52, 19, 32, 122]***

So, the first item in the list is referenced by ***my\_list [0]***. The value here in this example would be **52**.

***my\_list [1]*** would be **19**, etc.

**Compound Lists** are exactly the same concept as above with only one exception – the items can be composed of different datatypes – strings, floating-point numbers, integers, other lists, and perhaps a dictionary. There is no difference in how you reference an item in a compound list compared to a simple list.

***my\_list = [52, "a value", 32]***

***my\_list.append (["a",3, "b"])***

***this would result in [52, "a value", 32, ["a",3, "b"]]***

here, the fourth element of “**my\_list**” is also another list. If we want to get the digit 3, then we would reference this by **my\_number = my\_list [3] [1]**

You can also have a **list of dictionaries**. See the following when you understand dictionaries:

https://www.askpython.com/python/list/list-of-dictionaries

**Dictionaries**

A dictionary is different from a list in at least two ways. The indexes of a list are not visible and are always a sequence of numbers starting with the digit zero, whereas dictionary indexes are visible. They are the keys.

Like a list that cannot have duplicate indexes, a dictionary cannot have duplicate keys.

With a list, to find a value, you must search the list until you find the value that you are interested in. However, with a dictionary, if you know the key, you go straight to the value. There are lots of examples below.

**Defining/creating a Dictionary**

Dictionaries are defined by something like this: ***my\_dictionary = {},*** or we could define it with predefined elements: ***my\_dictionary= {'a' : 'something', 'd' : 'something else' }***

A dictionary has a **key and value**. The **key** is on the **left** side of a **colon**, and the **value** is on the **right**. The individual items are separated by commas.

**Remember, no keys can be duplicated**. If you try to add a duplicate key, the previous entry using that same key **will be thrown away**.

We can add an item to the dictionary by using the following syntax:

***my\_dictionary***[" I\_am\_a\_key"] = "here is yet another value"

Once again, our values can be any data type,

**Getting a Value from a Dictionary**

With a dictionary, the way you get a value is by referencing the key:

To get the **value** of "***something else***" we would use ***my\_dictionary ["d"]***

Notice that apostrophes and double quotation marks are interchangeable if you're consistent within the statement.

Also, notice from the project that the value of a dictionary can be a list. In fact, the value can be any data structure, including compound lists, which contain a mixture of data types (int, float, string, etc.)

**Setting a Value in a Dictionary**

in addition to initializing a dictionary with multiple items when creating a dictionary, as mentioned above, we can also add items individually. For example, when we’re looping through a list.

Direct command:

***my\_dictionary*****= {}**

***my\_dictionary ["compound\_name"] = my\_list***

Or initializing a new dictionary

***my\_dictionary = {"compound\_name": my\_list}***

Of course, a dictionary will usually have a lot more items than what I'm showing here.

**More on Getting a Value from a Dictionary – getting an item from a “list that is a dictionary value”**

Using the examples here, we could get one of the values from the list that's found in the dictionary by doing something like:

# Initialize a list

***my\_list = [52, 19, 32, “Jeep”, “Toyota”]***

# Initialize a dictionary with the compound list as above

***my\_dictionary = {"compound\_name": my\_list}***

***# So now I have a dictionary with only one item. The dictionary has a “key” of “compound\_name”,***

***# however, that single value of “my\_list”, has 5 elements, or 5 values.***

***# I can now use this dictionary to pull an item from the list with the following:***

***A\_list = my\_dictionary ["compound\_name"]***

***print (A\_list [1]) #*** would produce an integer of **19**

***print (A\_list [3]) #*** would produce the string “**Jeep**”

Here are a few more examples, this one using a **dictionary with values that are compound list items**.

\_dictionary = {

    "a": [2, "x", 20],

    "b": [72, "y", 21],

    "e": [29, "z", 20]

    }

# Notice I can't use the "b" (Using " character) as a key in the following statement

# because that would interfere with the F string format.

# However, because the key is a string of "b", An apostrophe works just the same.

print(f"I'm getting my value directly {\_dictionary['b'][2]}")

#  the following is the same as above

My\_variable = \_dictionary["b"][2]

print(f"I'm getting my value a little less directly {My\_variable}")

#  the following is the same as above

a\_1 = \_dictionary["b"]

a\_2 = a\_1[2]

print(f"I'm getting my value a lot less directly {a\_2}")

# Let's iterate through the dictionary. Rarely do this.

# Might as well just use a list instead of a dictionary.

# The Next line of code is necessary. Otherwise, the first time in the loop,

# the program will fail because My\_sum is uninitialized.

# When the statement My\_sum=My\_sum+a\_value, the My\_sum on the right

# of the = is uninitialized/Unknown.

My\_sum = 0

for k, v in \_dictionary.items():

    My\_sum += v[2]

    print( f"The key is {k}, the running sum is {My\_sum}, Getting the string from dictionary: {\_dictionary[k][1]}"  )

# Value associated with the key "e"

# This is the most common way a dictionary is used.

# Typically, you would have a dictionary loaded. Then the user would enter

# a key. For example, the student's ID number, which is the key,

# and that would give you the information found in the dictionary

# (or whatever is in the value area for that key.)

print(\_dictionary["e"])

Running this program produces the following results:

I'm getting my value directly 21

I'm getting my value a little less directly 21

I'm getting my value a lot less directly 21

The key is a, the running sum is 20, Getting the string from dictionary: x

The key is b, the running sum is 41, Getting the string from dictionary: y

The key is e, the running sum is 61, Getting the string from dictionary: z

[29, 'z', 20]

***You can even have a dictionary of dictionaries.*** See the following website for some examples:

https://linuxhint.com/python\_dictionary\_of\_dictionaries/