Name:________________________________ **Partner:** __________________________

Python Activity 28: Dictionaries of Dictionaries

We can use dictionaries to represent all sorts of structures of data.

Learning Objectives
Students will be able to:
Content:
- Define a **nested dictionary** or **dictionary of dictionaries**

Process:
- Write code to construct and add elements to dictionaries of dictionaries
- Write code to access elements of dictionaries of dictionaries
- Write code to iterate over dictionaries of dictionaries

Prior Knowledge
- Python concepts: dictionaries, data types, \n

Critical Thinking Questions:

1. Examine the sample code below, declaring several dictionaries, which maps ice cream flavors as keys to the number of cones sold. Each dictionary represents a different year of sales.

lickety.py
```
0  yr2022 = {'Purple Cow':1027,'Sweet Cream':1509,'Mudpie':2231}
1  yr2021 = {'Purple Cow':992, 'Sweet Cream':1623,'Mudpie':2064}
2  yr2020 = {'Purple Cow':891, 'Sweet Cream':955, 'Mudpie':520}
   #yr2019 = ...
   # Imagine we had 20 (or more!) years' worth of data
25 year = input("Year of ice cream sales? ")
```

a. Given the code in its current state, write a single line of code to display the expected output if the user entered `2020` on line 25: ____

b. Would your proposed approach work if we wanted to allow the user to input *any* year? *Summarize* what we would need to do to support user input of any year:

____

____

____

c. We could imagine a solution like the one outlined in the code below:

```
26 year_table = [{}] * 2023 # Adds 2023 empty dictionaries to this list
27 year_table[2022] = yr2022
28 year_table[2021] = yr2021
29 year_table[2020] = yr2020
   # Imagine this continued for 20 more years' of data
50 year = input("Year of ice cream sales? ")
```

What is the *type* of the keys in `year_table`? ______________

What is the *type* of the values in `year_table`?______________

Write a line of code to display the expected output if the user entered `2020` on line 50 (*Hint: remember what the keys' type is!)*:

____

d. How many lines of code (approximately) does this solution require? ~ __________lines of code

e. Is this a *good/efficient/convenient* solution? Why or why not?

____

f. Instead of a *list of dictionaries*, what might be a different data structure that allows us to access the data by *year* more efficiently?

a int | str | bool | function | tuple | set | dictionary *(circle one)* of dictionaries.

2. Examine the sample *incomplete* code below, which *should* be a better solution than the one proposed in Question 1c.

lickety.py

```
0  yr2022 = {'Purple Cow':1027,'Sweet Cream':1509,'Mudpie':2231}
1  yr2021 = {'Purple Cow':992, 'Sweet Cream':1623,'Mudpie':2064}
2  yr2020 = {'Purple Cow':891, 'Sweet Cream':955, 'Mudpie':520}
   #yr2019 = ...
   # Imagine we had 20 (or more!) years' worth of data

20 year_table = # (i) What type of data structure?
21 # (ii) How to add our dictionaries to year_table?



30 year = input("Year of ice cream sales? ")
31 print(year_table[int(year)])

32 flavor = input("Flavor of interest? ")
33 for icecream_year in year_table:
34    print(      # (iii) Year: Number Sold                  )
```

a. Given the call to `year_table[year]` on line 31 and how we intend to iterate over ***all*** the data in `year_table` on line 33, what *type* of data structure might `year_table` be? _____________

b. Complete the line of code on line 20, creating a new, empty object for `year_table`:

`20 year_table =` ____

c. Write a few lines of code, representing how you would add the first three dictionaries to `year_table` on lines 21-30:

____

____

____

d. Examine the code on lines 32-34. When a user inputs "Sweet Cream" the output should be something similar to: `'2022: 1509 \n 2021: 1623 \n 2020: 955'`. Write a line of code, for line 34, to do this:

____

e. It is possible that a particular ice cream flavor might have only received sales in *some* years. In that case, the number 0 should be stored in the dictionary and then displayed when printing on line 34. Rewrite the code around line 34 to handle this situation:

`for icecream_year in year_table:`

____

Application Questions: Use the Python Interpreter to check your work

1. We don't *typically* begin with 20+ dictionaries hard-coded in a Python script! It's much more realistic to read-in the data from a file, and accumulate the data into a nested data structure (much like we've previously done with *lists of lists*). This allows us to write fewer lines of code.

   Given the sample data file below, read-in the data into a data structure that allows us to access the data as specified by the sample code in Question 2.

lickety.csv *(could have 60+ lines!)*

```
2022,Purple Cow,1027
2022,Sweet Cream,1509
2022,Mudpie,2231
2021,Purple Cow,992
2021,Sweet Cream,1623
2021,Mudpie,2064
2020,Purple Cow,891
2020,Sweet Cream,955
2020,Mudpie,520
```

2. a. We want to hire an effective offensive player (i.e., someone who scores a lot) for our new football (soccer) team. We're pursuing this goal with a data-driven approach, and have a comma-separated values files containing data on the top goal-scorers for the past several years. The first several lines of the file are shown below, and each row has the season (year), player's name, number of goals, number of passes, and number of fouls. Write a function, `read_goal_data(filename)`, that takes a string filename and returns a dictionary of dictionaries, mapping the year to each season of data (just the names and their number of goals).

all_seasons.csv *(first 9 lines)*

```
2018,Pierre-Emerick Aubameyang,22,692,13
2018,Sadio Mané,22,1,34
2018,Mohamed Salah,22,1,25
2018,Sergio Agüero,21,771,21
2018,Jamie Vardy,18,416,19
2018,Eden Hazard,16,1,12
2018,Callum Wilson,14,440,41
2018,Raúl Jiménez,13,1,42
2018,Alexandre Lacazette,13,771,51
```

   b. Write a function, `get_top_scorers(season_table)`, that takes a dictionary of dictionaries as an argument and returns a list of player names that appear for all seasons of our data.