Name:_____ **Partner:** _____

Python Activity 26: Dictionaries

Lists are useful data structures, but what if the relationship between data isn't sequential?

Learning Objectives

Students will be able to:

Content:

- Define a **dictionary**.
- Identify the **key** and **value** pair of a dictionary.
- Explain why a dictionary is a good data structure for organizing data.

Process:

- Write code that accesses the keys, values, and length of a dictionary.
- Write code to create and modify dictionaries.
- Write code that iterates over a dictionary's keys.

Prior Knowledge

- Python concepts: lists of lists, indexing, mutability, len(..), for.. loops

Critical Thinking Questions:

1. Examine the sample code defining a list of lists, below:

Sample Code

```
dog2owner = [['pixel','iris'],['sally','jeannie'],['jerry','lida']]
print(dog2owner[0][0]) # prints: 'pixel'
```

a. What is stored at `dog2owner[1][1]`? _____

c. Write a line of code to print the name of Sally's owner using `dog2owner`:

_____

d. Write a line of code to access and print the name of Lida's dog via `dog2owner`:

_____

e. As `dog2owner` gets bigger and bigger (the CS department is growing!), will a list of a lists be an accessible way to continue storing this information?

_____

2. The following code occurs in interactive Python and introduces a new data structure:

```
0 >>> dt = {'pixel':'iris','sally':'jeannie','jerry':'lida'}
1 >>> dt['sally']
2 'jeannie'
```

a. What does `dt['sally']` do?

_____

b. How might python know that Sally (the dog) is mapped to Jeannie (the owner)? Where is that relationship defined?

c. In the line, `dt['sally']`, what does the value in the square brackets represent?

_____

_____

FYI: A ***dictionary*** is an unordered data structure that instead of storing *values* at numerical indices, ***values*** are mapped to ***keys.*** Keys must be an immutable data type.

d. Write a line of code to print the name of your CS134 instructor's name, accessed via the dictionary, `dt`: _____

e. Why might a dictionary be a better data structure for this data than a list of lists?

_____

f. How would you describe the ***keys*** and ***values*** for this dictionary, `dt`?

keys:_____ values:_____

g. What type of data is stored in the keys and the values for `dt`?

keys:_____ values:_____

3. The following code occurs in interactive Python and introduces a new data structure:

```
0 >>> dt = {'pixel':'iris','sally':'jeannie','jerry':'lida'}
1 >>> dt['jeannie']
2 KeyError: 'jeannie'
```

a. What is the programmer trying to do with the `dt['jeannie']` on line 1?

_____

b. Why might this code be throwing the error on line 2?

_____

c. What does this error tell you about what can go in the dictionary square brackets?

_____

4. Examine the following code from interactive Python:

```
0 >>> dt = {'pixel':'iris','sally':'jeannie','jerry':'lida'}
1 >>> dt['wally'] = 'steve'
2 >>> dt
3 {'pixel':'iris','sally':'jeannie','jerry':'lida','wally':'steve'}
```

a. What does the line `dt['wally'] = 'steve'` do?

_____

b. What might this imply about the ***mutability*** of dictionaries?

_____

c. What does the object in square brackets on the left hand side of the assignment operator in line 1 represent? (*Circle one*) key or value

d. What does the object on the right hand side of the assignment operator in line 1 represent? (*Circle one*) key or value

e. Write a line of code to add `Bill` and his dog, `Artie`, to our dictionary.

_____

5. Examine the following code from interactive Python:

```
0 >>> cs_pets = {'dogs':9, 'cats':4, 'bees':20000}
1 >>> len(cs_pets)
2 3
```

a. What type of data is stored in the keys and the values for `cs_pets`?

keys:_____ values:_____

b. How many keys does `cs_pets` have? _____

c. What is the length `cs_pets`? _____

d. How does python determine the length of a dictionary object?

_____

e. If we added a line 3 of code, `cs_pets['others'] = ['hamster', 'ferret']`, what might `len(cs_pets)` return? _____

7. Examine the following example code:

```
>>> coll = dict()          # can also do: coll = {}
>>> coll['colleges'] = 'williams'
>>> coll['colleges'] = 'amherst'
```

a. If we wrote a fourth line of code, `print(coll)`, what might be the output?

_____

b. At the end of this code execution, `coll` only has: `{'colleges': 'amherst'}` Why might this be?

_____

FYI: Dictionaries can only have <u>one</u> key of its value, any replicated key:value mappings added will simply overwrite the previous one!

8. Examine the following example code from interactive python:

```
0 >>> date = {'month':'dec', 'day':9, 'year':1906}
1 >>> for mykey in date:
2 ...     print("The", mykey, "is", date[mykey])
```

a. What data does the dictionary, `date`, appear to hold?

_____

b. If you had to guess, what might the programmer want to be output by line 2?

_____

c. For the first defined item of `date` what might `mykey` and `date[mykey]` refer to on lines 1 & 2?

`mykey:`_____ `date[mykey]:`_____

d. The first time through the loop defined on line 1, line 2 might print `'The month is dec'` What might be printed the second time through the loop?

_____

e. What does line 1, `for mykey in date:`, do?

_____

f. Write some code that will iterate over the items in `date` and print *only* the values:

_____

_____

_____

_____

8. Examine the following example code from interactive python:

```
0 >>> date = {'month': 'dec', 'day': 9, 'year': 1906}
1 >>> sorted(date)
```

a. What do you think will be the output of line 1?

_____

b. Line 1 actually outputs `['day', 'month', 'year']`. How do these values relate to the `data` dictionary?

_____

c. Write some lines of code that will construct a list of the *values* in date, sorted by the *keys* in date:

_____

_____

_____

_____

_____

_____

_____

Application Questions available in the digital file on Glow.

Application Questions: Use the Python Interpreter to check your work

1. Write a function that checks if a given dictionary, `d`, has a given `key`. If it doesn't, create a new list at `key` with the given `value` as its only element. If it does already have the `key`, append `value` to the existing list mapped to `key`.

```
def append_dict_lst(d, key, value):
```

2. Write a function, `data_entry` that collects data from the user to put into a dictionary. The user should be prompted for a key, and then value data to be added to a dictionary, and this process should be repeated until they enter the text `'done'`. For extra bonus points, use your previous function, `append_dict_lst`, to ensure that no data is overwritten, even if a key is duplicated! The `data_entry` function should return the dictionary when the process is done.

```
def data_entry():
```

Name:_____ **Partner:** _____

Python Activity 27: 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.