Lab 09 - Tuples and Dictionaries

Background

Python has a wide variety of built-in data types for storing numbers and text that we have already worked with (int, float, string). There are also common data structures that store values in a more complex way (list, tuple, dictionary). Today we will be focusing on tuples and dictionaries.

Content Learning Objectives

- Know how to create and use a tuple
- Explain what a tuple is in Python
- Know how to create and use a dictionary
- Explain what a dictionary is in Python

Part 1: Sequences

Lists, Strings, and Tuples are examples of ***sequence*** types in Python. Enter the Python code into a Python shell to complete the table below.

Python Code	Shell Output
seq1 = "one two"	n/a
type(seq1)	<'class str'>
len(seq1)	7
seq1[1]	'n'
seq1[1] = '1'	error
seq2 = "one", "two"	n/a
type(seq2)	<'class 'tuple''>
len(seq2)	2
seq2[1]	'two'
seq2[1] = "1"	error

seq3 = ("one", 2, "three")	n/a
type(seq3)	<'class tuple'>
len(seq3)	NameError
seq3[2] = 3	error
seq4 = ["one", 2, "three"]	n/a
type(seq4)	list
seq4[2]	'three'
seq4[2] = 3	n/a
print(seq4)	['one', 2, 3]
number = 12345	n/a
type(number)	<class int>
number[3]	error

Questions

1. What data types did we work with?

In [ ]:
tuples

2. There are two different syntax patterns you can create a tuple. What are they?

In [ ]:
- creating a variable equal to mulitple values
- multiple values in parenthesis

3. How does the syntax of creating a tuple compare to the syntax of creating a list?

In [ ]:
```
lists use brackets
```

4. Which sequence types allow for elements to be changed? Which do not?

In [ ]:
```
lists allow change and strings/tuples do not
```

5. Is it possible to store values of different types in a sequence? If so, give an example from the table. If not, explain how you know.

In [ ]:
```
yes because tuples store ints and strings: "one", 2
```

6. There is one BIG difference between the functionality of tuples and lists. What is it?

In [ ]:
Lists can undergo changes while tuples stay how they were created

Tuples are particularly useful in functions when you want to return multiple values in one variable. While useful, you will likely not work with them directly on a regular basis but you may see them in some function calls later in your education.

Part 2: Dictionaries

In Python, a ***dictionary*** stores "key:value" pairs. For each key, there is one value associated with it. This value could be anything: int, float, string, list, or even another dictionary. A key and its associated value are separated by colons while different key:value pairs are separated by commas all enclosed in curly braces. For example:

elements = {'C' : 'Carbon', "H": "hydrogen", "O" : "Oxygen", 'N' : "nitrogen"}

Key	Value
'C'	'Carbon'
"H"	"hydrogen"
"O"	"Oxygen"
'N'	"nitrogen"

Unlike a string, list or tuple that is a sequence type, a dictionary is a mapping type. This means that the values are referenced by a key rather than an index.

Type the following Python code into a shell to complete the table. Before you start, type the following line into your shell or run the line in your jupyter notebook.

In [28]:
```
activeWx = {'RA' : 'Rain', 'DZ': 'Drizzle',
            'IC' : 'Ice Crystals', 'SN': 'Snow',
            'GR': 'Hail'}
```

Python Code	Shell Output
type(activeWx)	<class dict>
activeWx.keys()	dict_values
activeWx.values()	dict_values
activeWx.items()	dict_items
activeWx['RA']	Rain
weather = 'SN'	n/a
activeWx[weather]	Snow
activeWx[SN]	NameError
activeWx['Snow']	KeyError
activeWx[1]	keyError
len(activeWx)	5
activeWx['PL'] = 'Ice Pellets'	n/a
activeWx.items()	dict_items
len(activeWx)	6

Questions

7. List all of the keys stored in the activeWx dictionary after completing the table.

In [ ]:
```
6
```

8. What is the data type of the keys in the elements dictionary?

In [ ]:
```
string
```

9. Explain why you got an error when you ran each of the following lines:

   a) activeWx[SN]

   b) activeWx['Snow']

   c) activeWx[1]

In [ ]:
```
SN is not a string, it is a value of weather
Snow is a value, not a key
1 is an index, not a key
```

10. Write a python expression that creates a dictionary named cloud_cover. The dictionary should have keys that are cloud cover abbreviations (SKC, FEW, SCT, BKN, OVC) and values that are the associated meaning (Skies Clear, Few, Scattered, Broken, Overcast)

In [ ]:
```
cloud_cover = { 'SKC' : 'Clear', 'FEW' : 'Few clouds', 'SCT' : 'Scattered
Clouds', 'BKN' : 'Mostly Cloudy', 'OVC' : 'Overcast'}
```

11. If you assign two different values to the same key, which of the values is stored in the dictionary? Justify your answer with an example.

In [ ]:
```
the last assgined value

activeWx = {'RA' : 'Rain', 'DZ': 'Drizzle',
'IC' : 'Ice Crystals', 'SN': 'Snow',
'GR': 'Hail', 'RA' : 'myomu'}
dict_values(['myomu', 'Drizzle', 'Ice Crystals', 'Snow', 'Hail'])
```

12. Another way you could store data is with two lists:

    keys = ['RA' , 'DZ', 'IC' , 'SN', 'GR']

    values = ['Rain', 'Drizzle', 'Ice Crystals', 'Snow', 'Hail']

    What is a disadvantage of this approach? Explain your reasoning.

In [ ]:
Takes more time to type and takes up more space and is also harder to store the correct values with the correct keys

Part 3: Dictionary Practice

For this part use the dictionary below to complete the functions and answer the questions.

In [29]:
```
sept_temps = {'09-01': 57.0,'09-02': 53.51,'09-03': 51.66,
              '09-04': 43.11,'09-05': 41.51,'09-06': 39.03,
              '09-07': 35.44,'09-08': 43.74,'09-09': 39.41,
              '09-10': 45.87,'09-11': 45.18,'09-12': 44.87,
              '09-13': 34.36,'09-14': 27.53,'09-15': 29.53,
              '09-16': 38.77,'09-17': 39.15,'09-18': 40.29,
              '09-19': 40.9,'09-20': 28.04,'09-21': 24.62,
              '09-22': 32.32,'09-23': 27.94,'09-24': 38.26,
              '09-25': 35.4,'09-26': 41.47,'09-27': 38.63,
              '09-28': 32.29,'09-29': 34.45,'09-30': 33.32}


oct_temps = {"10-01": 31.1,"10-02": 31.1,"10-03": 32.2,"10-04": 25.0,
             "10-05": 15.0,"10-06": 14.4,"10-07": 12.2,"10-08": 19.4,
             "10-09": 22.2,"10-10": 19.4,"10-11": 21.7,"10-12": 20.0,
             "10-13": 11.7,"10-14": 7.2,"10-15": 11.1,"10-16": 18.3,
             "10-17": 14.4,"10-18": 13.9,"10-19": 15.0,"10-20": 10.0,
             "10-21": 8.3,"10-22": 7.2,"10-23": 10.0,"10-24": 13.9,
             "10-25": 16.1, "10-26": 20.0,"10-27": 15.0,"10-28": 11.1,
             "10-29": 10.0,"10-30": 8.9, "10-31": 6.7}
```

In [30]:
```
def warmest_day(dictionary):
        #loop through each day in the dictionary
        #test to see if it is the warmest
        #save the warmest day

    return date

#call the function here
```

13. Which day in September was the warmest? How about October?

In [ ]:
```
09-01

10-03
```

In [31]:
```
def coolest_day (dictionary):
        #very similar to warmest_day but opposite


    return date

#call function here
```

14. Which day in September was the coolest? How about October?

In [ ]:
```
09-21

10-31
```

In [32]:
```
def average_temp(dictionary):
    #loop through each day in the dictionary
    #add the temperatures together
    #find the length of the dictionary
    #divide total by length

    return temp
```

```
#call function here
```

15. What are the average temperatures of each month? Which month had a warmer average?

In [ ]:
09 - 38.5867
10 - 15.887

september was warmer

In [33]:
```
def days_above_temp(dictionary, temp_limit):
        #loop through each day in dictionary
        #test to see if day is above, below, or equal to the temp_limi
        #add day if it meets criteria

    return num_of_days

#call function here
```

16. How many days are above freezing in September? How about October?

In [ ]:
09 - 25
10 - 1