**Introduction to Programming with Python**

**Homework 5**

***Due on August 5, 2020***

1. **Conversions**
2. In IDLE or the Python development environment of your choice, edit the provided **hw5.1.py** file. This script contains initializations of a **str**, a **list**, a **tuple**, a **set**, and a **dict**:

**str1 = 'This is a test of the emergency broadcast system.'**

**m1 = [4, 1, 5, 5, 1, 3, 8, 9, 7, 8, 2, 2, 6]**

**tup1 = ('curiouser', 'and', 'curiouser', 'cried', 'Alice')**

**set1 = {4, 7, 'a', True, 12.5, None}**

**d1 = {'hello': 5, 'goodbye': 3, 'you': 2, 'say': 8, 'I': 5, 'why': 4}**

Run the script to confirm that it works, and confirm for yourself that the output makes sense.

1. Using the appropriate conversion function, define the variable **m2** to refer to a **list** of each of the 1-character strings from **str1**. Display **m2** with **print("m2: ", m2)**.
2. Using the appropriate conversion function, define the variable **tup2** to refer to a **tuple** of each of the 1-character strings from **str1**. Display **tup2**.
3. Using the appropriate conversion function, define the variable **set2** to refer to a **set** of each of the 1-character strings from **str1**. Display **set2**. Does **set2** appear to be ordered? Make a comment in your code file.
4. Using the appropriate conversion function, define the variable **m3** to refer to a **list** of each of the items from **tup1**. Display **m3**.
5. Using the appropriate conversion function, define the variable **m4** to refer to a **list** of each of the items from **set1**. Display **m4**.
6. Using the appropriate conversion function, define the variable **tup3** to refer to a **tuple** of each of the items from **m1**. Display **tup3**.
7. Using the appropriate conversion function, define the variable **tup4** to refer to a **tuple** of each of the items from **set1**. Display **tup4**.
8. Using the appropriate conversion function, define the variable **set3** to refer to a **set** of each of the items from **tup1**. Display **set3**. How does **set3** compare with **tup1**? Make a comment in your code file.
9. Using the appropriate conversion function, define the variable **listk1** to refer to a **list** of each of the *keys* from **d1**. Display **listk1**.
10. Using the appropriate conversion function, define the variable **setk1** to refer to a **set** of each of the *keys* from **d1**. Display **setk1**. How do the items in **listk1** compare with the items in **setk1**? Are there the same number of items? Is this just “luck”?
11. Display the lengths of (number of items in) **d1**, **listk1**, and **setk1**.
12. Using the appropriate conversion function, define the variable **listv1** to refer to a **list** of each of the *values* from **d1**. Display **listv1**.
13. Using the appropriate conversion function, define the variable **setv1** to refer to a **set** of each of the *values* from **d1**. Display **setv1**. How do the items in **listv1** compare with the items in **setv1**? Are there the same number of items? Is this just “luck”?
14. Display the lengths of (number of items in) **d1.values()**, **listv1**, and **setv1**. What can you say about the number of unique *keys* in a **dict** vs. the number of unique *values* in a **dict**?
15. **m2** and **tup2** contain the same item values in the same order. Is **m2 == tup2**? Is **m2[7] == tup2[7]**? Test, and make a comment in your code file.
16. Which of these assignments will work to create a **list**?
    1. **mq1 = list(())**
    2. **mq2 = list((5))**
    3. **mq3 = list((9,))**
    4. **mq4 = list(1, 2, 3)**
    5. **mq5 = list({1, 1, 1})**
    6. **mq6 = list([5, 4, 3])**
    7. **mq7 = list(range(4))**
17. **Comprehensions**
18. Using a **list** *comprehension*, define the variable **mc1** to refer to a **list** of 10 items with value 0. Display **mc1**.
19. Using a **list** *comprehension*, define the variable **mc2** to refer to a **list** of 10 items with values 0, 1, 2, 3, … 9. Display **mc2**.
20. Using a **list** *comprehension*, define the variable **mc3** to refer to a **list** of 10 items with values square root of 0, square root of 1, square root of 2, …, through square root of 9. Display **mc3**.
21. Using a **list** *comprehension*, define the variable **mc4** to refer to a **list** of 15 items with values 0, 1, 2, 0, 1, 2, 0, 1, 2, …, 2. Display **mc4**. (***Hint:*** Remember the modulus operator, **%**)
22. Using a **list** *comprehension*, define the variable **mc5** to refer to a **list** of 10 3-**tuple**s, with values (0, 1, 4), (1, 2, 9), (2, 3, 16), …, (9, 10, 121). Display **mc5**. Do you agree that this is *Pythonic*? Make a comment in your code.
23. **String Handling**
24. Define the variable **str3a** as a copy of **str1** but with all characters converted to lower case. Display **str3a**.
25. Define the variable **str3b** as a copy of **str3a** but with the first character of each word converted to upper case. Display **str3b**.
26. Define the variable **m3c** as a **list** of the words in **str1**, in the same order as the words exist in **str1**. Display **m3c**.
27. Define the variable **m3d** to refer to a concatenation of *slices* of **m3c**:

**m3d = m3c[:2] + m3c[5:7] + m3c[-1:]**

Predict what **m3d** contains, then display **m3d** to see whether you were correct.

1. Define the variable **str3e** to refer to a string (**str**) consisting of the items in **m3d** separated with individual space characters. Display **str3e**.
2. Define the variable **str3f** to refer to a string (**str**) consisting of the items in **m3d** separated with **'+'** characters, and with **'+'** characters at the beginning and end. Display **str3f**.
3. Define the variable **str3g** to refer to a copy of **str3f** converted to ALL UPPERCASE LETTERS! Display **str3g**.
4. Perform some experiments of your own with **center()**, **ljust()**, **rjust()**, **strip()**, **lstrip()**, **rstrip()**, **islower()**, **isupper()**, **isalpha()**, **isnumeric()**, and **isalnum()**.