Richard Hayes Crowley

07/24/2021

CSC\_242\_Lab\_012\_QA

**(1)** In terms of collections, sequences, mutability and immutability, how are sets in Python similar to strings?

Refer to the following code snippet.

**print (set({1, 2, [3, 4]}))**

**The code snippet above returns an error: `list is unhashable type`**

**It errors because the hashing algorithm is unable to convert a list into a hash for the hash index for the set. Because sets are immutable, they use key / value hashes (similar to dictionaries) to store items in a set at machine addresses (the decimal hash converted to a binary machine address), which enables O(1) access and removal.**

**(2)** When performing operations on two defined subsets, it is necessary that the two subsets belong to some same universal set?

**Not at all, the two sets can be subsets of two completely different universal sets.**

**(3)** If sets are considered as immutable, why does Python have an intrinsic **frozenset()** function?

Refer to the following code snippet for an example of this intrinsic method.

|  |
| --- |
| **# from a set**  **print (set({"a", "e", "i", "o", "u"}))**  **# from a dictionary**  **print (set({"a" : 1, "e": 2, "i": 3, "o": 4, "u": 5}))**  **# from a frozen set**  **frozen\_set = frozenset(("a", "e", "i", "o", "u"))**  **print (set(frozen\_set))** |

**The “frozenset” is a read-only version of a set that only allows set operations (union, difference, etc.) and is truly immutable. The “frozen set” is to a tuple as a set is to a list.**

**(4)** Explain what the following code segment will perform when the statements are executed.

.

**setA = set({1, 3, 10, 0})**

**setB = set({2, 7, -4})**

**blnCheck1 = setA.isdisjoint(setB)**

**blnCheck2 = setA.issubset(setB)**

**print (blnCheck1 and blnCheck2)**

**print (blnCheck1 or blnCheck2)**

**The code segment instantiates two sets, runs a statemet to see if setA does not have an intersection with setB (no common elements), and another statement to see if setA is a subset of setB.   
  
It then prints a couple Boolean statements that look for whether both or either of the prior statements are truthy. The first statement returns false because while there is no intersection (common elements) between setA and setB, setA is not a subset of setB. The second statement (“or” statement) is true because blnCheck1 is True.**

**(5)** For our Brainstorming Sessions application, what if teams that are formed via a set operation, such as an intersection, are limited in the size of the members? Explain how you would code such a variation in the project.

**I would include some selection for users of team members when forming subsets and limit the number of team members that can be selected for each subset.**