**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23 may 2020** | **Name:** | **Veronica gudagur** |
| **Course:** | **python** | **USN:** | **4al16ec091** |
| **Topic:** | **List, Tuples and Dictionary** | **Semester & Section:** | **8-B** |
| **Github Repository:** | **veronica-g** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **C:\Users\Nelson\Desktop\p1.PNG**  **C:\Users\Nelson\Desktop\p2.PNG** |
| **How to store list of elements….**  List of an elements   1. words = stuff.split(' ') 2. a\_list = ['foo', 2, [4, 5]] 3. a\_list[2] = (3, 4) 4. a\_list   What is twin and copy?  When your twin is in pain do you feel it?   1. a = [1, 2, 3] 2. b=a.copy() 4. a.append(4) 5. b 6. def isiterable(obj): 7. try: 8. iter(obj) 9. return True 10. except TypeError: # not iterable 11. return False 13. isiterable('a string') 15. isiterable([1, 2, 3]) 17. iter(5)   Difference between is and ==?   1. a = [1, 2, 3] 2. b = a 3. c = list(a) 5. a is b 7. a is c 9. a==c   What is range?  # This creates a list using a range   1. list(range(10))   Adding two list, and extending a list as you can see.  lst1 =[4, None, 'foo'] + [7, 8, [2, 3]]   1. lst1[5][1]  4. x = [4, None, 'foo'] 5. x.extend([7, 8, (2, 3)]) 6. x   **Key Value Pairing :**  When you wife asks you buy something does she gives you a list, tuple or dict?  Two dictionaries and play with the two dictionaries.  Looping through the key value pairs.  Read about JSON the equilvanet in JS and MongoDB.   1. # create a mapping of state to abbreviation 2. states = { 3. 'Oregon': 'OR', 4. 'Florida': 'FL', 5. 'California': 'CA', 6. 'New York': 'NY', 7. 'Michigan': 'MI' 8. } 10. # create a basic set of states and some cities in them 11. cities = {'CA': 'San Francisco', 12. 'MI': 'Detroit', 13. 'FL': 'Jacksonville'} 15. # add some more cities 16. cities['NY'] = 'New York' 17. cities['OR'] = 'Portland' 19. # print out some cities 20. print('-' \* 10) 21. print("NY State has: ", cities['NY']) 22. print("OR State has: ", cities['OR']) 24. # print some states 25. print('-' \* 10) 26. print("Michigan's abbreviation is: ", states["Michigan"]) 27. print("Florida's abbreviation is: ", states['Florida']) 29. # do it by using the state then cities dict 30. print('-' \* 10) 31. print("Michigan has: ", cities[states['Michigan']]) 32. print("Florida has: ", cities[states['Florida']]) 34. # print every state abbreviation 35. print('-' \* 10) 36. for state, abbrev in states.items(): 37. print(f"{state} is abbreviated {abbrev}") 39. # print every city in state 40. print('-' \* 10) 41. for abbrev, city in cities.items(): 42. print(f"{abbrev} has the city {city}") 44. # now do both at the same time 45. print('-' \* 10) 46. for state, abbrev in states.items(): 47. print(f"{state} state is abbreviated {abbrev}") 48. print(f"and has city {cities[abbrev]}") 50. print('-' \* 10) 51. # safely get an abbreviation by state that might not be there 52. state = states.get('Texas') 54. if not state: 55. print("Sorry, no Texas.") 57. # get a city with a default values 58. city = cities.get('FL', 'Does not Exist') 59. print(f"The city for the state 'TX' is: {city}")   Long code.. to reverse things...  Value from key is simple just write a square bracket, or use get. But if you want to do the opposite which is find key for a value then you have write this code.   1. # run the below 2. print(list(cities.keys())[list(cities.values()).index('New York')]) 3. #list from dict 4. list( cities.values()).index('New York') 5. #list of keys 6. list(cities.keys()) 7. #looping through all 8. for key, value in states.items(): 9. print(f"{key} key is abbreviated {value} value") 10. #get items 11. states.items() 12. city = cities.get('XXX', 'Does not Exist') 13. print(f"The city for the state is: {city}") |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |