**Create a module for the following exercises on dictionaries**

1. Create a function called create\_dict().   
   Create a dictionary to store information about software workshop I. The dictionary should store the following information:   
   type\_assess – t is tests and e is exams, term and weighting. For example, a dictionary could look as follows:  
   sww1 = {‘type\_assess’:’t’, ‘term’: 1, ‘weighting’: 100}  
   print a heading “Information regarding SWW1”  
   Using a for loop print each key-pair like this:  
   Information regarding software workshop I  
   ('type\_assess', 't')  
   ('term', 1)  
   ('weighting', 100)
2. Create another function called all\_modules().  
   Include sww1 – as above and 2 others (any so that you have 3 dictionaries each with a module name and the corresponding information).  
   Create a list that holds the dictionary names just created above.  
   Again, using a for loop print the contents in the following way:  
   Information regarding all modules:

How is the module assessed? t

Which term is the module taught? 1

How much weighting is given to the assessments? 100

How is the module assessed? e

Which term is the module taught? 2

How much weighting is given to the assessments? 80

How is the module assessed? e

Which term is the module taught? 1

How much weighting is given to the assessments? 100

1. Lastly, create a nested dictionary using the dictionaries above where the dictionary is called msc\_modules {} and each key is “sww1” (module name) followed by the rest of the dictionary as above. For example:  
   msc\_modules = {  
    “sww1” : {‘type\_assess’:’t’, ‘term’: 1, ‘weighting’: 100}  
   #include the others here  
    }  
   Create a for loop to output the data in the following way:

For sww1 the type of assessment is t The module is taught in term 1 The weighting is 100

For dsa the type of assessment is e The module is taught in term 2 The weighting is 80

For prof\_prac the type of assessment is e The module is taught in term 1 The weighting is 100

1. Create a contacts dictionary. Write a function that adds a new contact, updates an existing one and retrieves a contact by name. include a series of prompts and a menu to facilitate this. For example. Your contacts could include:  
   “Jack Smith”: {“phone”: “12345123”, “email”: “[jsm@gmail.com](mailto:jsm@gmail.com)”},  
   “Mary Hawkes”: {“phone”: “58678”, “email”: “[jha@gmail.com](mailto:jha@gmail.com)”},  
   **Add a contact:**  
   call the function and prompt for a name, phone number and email address; update the contacts {} – this is set as a global variable.  
   **Find a contact:** a prompt should ask which contact to find using the full name, the function should print out the corresponding phone number and email if the contact is found  
   **Update a contact:**  
   a prompt should ask for a contacts full name to update; if the contact is found another prompt should ask about what to update 1 – phone number or 2 – mail; create the code to perform the update (use a loop that iterates over each dictionary item, finds the contact and makes the appropriate update) -> Important to test this functionality very well.  
   list all contact:  
   List all of the contacts in the dictionary using an appropriate format / styling.  
   **Using ChatGPT for learning**

Write a function that reads a sentence and counts the number of times each word appears in the sentence. Make use of a dictionary so that each word has a word count, for example:  
“the”: 4, “many”: 2, “trouble”: 1  
Once you have a solution copy it into ChatGPT and ask for a different solution. Compare the 2 solutions. Answer the survey found here:

1. This exercise makes use of **dictionaries and tuples**. Create a schedule for yourself so that you know when you are setting aside time each day for studying. For example, on a Monday you can study from 3pm and 5pm, Tuesday 11am, and so on. The day becomes the key for the dictionary and the time is an element for the tuple part. So, the dictionary consists of key: tuple. Write function that returns the schedule for a particular day. The calling statement passes a day argument to the function. So, if you pass ‘Monday’ then the function returns ->   
   on Monday you have study time from 3pm to 5pm.
2. Create a dictionary that has a key for the invoice number and a tuple that shows the date as well as the amount. There are multiple invoice numbers that make up the dictionary. You can code this simply by calling a function to add\_invoice(), passing the inv nbr, date and amount to the function. For example:  
   add\_invoice(“A111”, “20102024”, 4500.45). the function must add the invoice to the dictionary/tuple. You must be able to call the function multiple times each time with a different invoice. You can put the calls in a function called call\_all() and call the call\_all() to start the code. Include get\_invoice() – pass the invoice number through - if you want to add another function to this or you can make it more sophisticated and include an entire solution as you did in exercise 4 above.   
   **Challenge:** Try and print date so that it reads like this:  
   DD: 20 MM: 10 YY: 2024