| **Session Objectives** | * Identify dictionaries in Python * Identify Key Value Pairs * Explain how KVPs can be used to model objects * Explain how dictionaries are declared and used |
| --- | --- |
| **Key Points** | * Dictionaries are collections of KVPs which can be used to model objects by representing the object’s attributes. |
| **Assessment** | * Via practical challenges - see task sheet below |
| **Instructor prep** | * **Note**: These session plans include sections where long explanations are provided simply because it is important to ensure that these tricky concepts are communicated as clearly as possible. However, it is not expected that the instructor will recite these verbatim, feel free to use your own refined and well-honed approach as long as the learning point is covered. The provided narrative is always available if needed. * **Learning and delivery** can be more effective if resources and tasks are personalised. If time permits, feel free to update slides with your own examples such as replacing cat examples with your own pets, or anything else. |
| **Materials** | * [TIFC1-PF-6 - Dictionaries - Slides](https://docs.google.com/presentation/d/1z-YGC49oBZV7AOCoNQbaSXak0PFJC7h2gt2Fa-eaX_U/edit?usp=sharing) * [TIFC1-PF-6 - Dictionaries - Tasks](https://docs.google.com/document/d/13TO9cV__AeARXsrtAmKBl1OCFJ0tmviE2fvMpMRj6W4/edit?usp=sharing) |

| **Time** | **Activity** |
| --- | --- |
| 5 minutes | **Slide 2 - Objectives**   * **Show Slide 2 - Read** session objectives |
| 15 minutes | **Slide 3 - 5 - Introduction to IF statements**   * **Show Slide 3 -** Introduce your ‘dictionaries’ * **Show Slide 4 - Ask** for a volunteer to **explain** what the code in the example does. If some positive responses are forthcoming, thank the contributors then **explain** if not already covered. * Note: Flick back and forth between slides 4 & 5 as necessary as you explain and point out the following:   *“This code demonstrates the next data structure in Python we need to review, dictionaries.*  *Dictionaries are simply collections of key value pairs, which are very common and will come up in a variety of contexts and environments.*  *A Key Value Pair (KVP) is often how we list the attributes of an item or resource, and is very useful for modelling objects in code. For example, think of a cake; What are the attributes of that cake? Some of them could be type, flavour, weight, portions, etc. each of these keys could have an appropriate value. What other examples can you think of?*  **Pause for responses** and discuss any forthcoming  *Most things can be thought of as a list of attributes in this way. When we move onto the cloud you will deploy resources and provide their attributes in the same way, so some KVPs that you need to consider might be the network to deploy your VM, the number of CPU cores and amount of RAM it needs, what type of storage is attached, and so on.*  *Back to the code, here we’ve modelled three cats using dictionaries. Three variables are declared on lines 1, 6, and 11, each one has been given the value of a dictionary, and each dictionary has three attributes or KVPs. In this case each dictionary has the same KVPs.*  *Notice the syntax, the dictionary is declared using curly brackets (braces), each key and value is linked by a colon, and different KVP are separated by a comma. You can write your dictionaries all on one line, but here each KVP starts on a new line which makes it more readable - observe the indentation though.*  *Our three variables, each of which contains an entire dictionary, are then stored in a list called ‘cats’ at line 17. This is an interesting illustration of how a lot of information can be packed into variables, then managed relatively simply in a collection.*  *At the bottom a For loop is used to simply print each item in the list, or print out each dictionary in this example. There are many other ways in which dictionaries are just like lists.”* |
| 5 minutes | **Slides 6 - 7 - Accessing Values in a Dictionary**   * **Show Slide 6 - Ask** for a volunteer to **explain** what the code in the example does. If some positive responses are forthcoming, thank the contributors then **explain** if not already covered.   *“In this code you can see just one dictionary has been declared, and it’s all on one line rather than starting each key on a new one.*  *Lines 3 - 5 demonstrate the syntax to recall specific values from the dictionary, notice to recall the value you provide the key”*   * **Show Slide 7 - Explain** that here the same syntax is used to concatenate the value matching the key of ‘fur’ into an f-string. |
| 10 minutes | **Slides 8 - 9 - Add new key-value pairs**   * **Show slide 8 - Explain** that this same syntax, dictionary\_name[‘key’], can also add new key value pairs to the dictionary as seen here. * **Show slide 9 - Explain** that it can be sometimes useful to start with an empty dictionary. * Optional: Open browser, navigate to a shopping site which allows you to search by attribute, such as Amazon or Autotrader. Advise learners that:   “*We don’t know if this is exactly how this page has been coded, but this could be a use for an empty dictionary”*  *Then navigate by category to for example computers, then filter by CPU, RAM, HDD, etc. by using checkboxes. On Autotrader you may filter by fuel type, drive train, etc.*”   * Explain to learners that you can imagine how this search was carried out based on attributes not a string I typed. This may have been an empty dictionary, which was then populated with KVPs based on my selections. |
| 5 minutes | **Slides 10 - 11 - Modifying key-value pairs**   * **Show slide 10 - Explain** that this syntax is also used to modify values by overwriting the value for an existing key * **Show slide 11 - Explain** that this slide summarises the last few points in one block of code. **Ask** for a volunteer to **explain** what the code in the example does. |
| 15 minutes | **Slides 12 - 13 - Removing key value pairs**   * **Show slide 12 - Explain** that removing KVPs is also very similar to how we work with lists by using the ‘del’ keyword”, **move to slide 13**, “as seen near the bottom in this example” |
|  | **Slides 14 - 18 - Using loops with Dictionaries**   * **Show slide 14 - Explain** that using a for loop with a dictionary is a little trickier than a list because you need to decide whether you want to loop through the keys and return the values, loop through the values and return the keys (which might be useful if you need to know what attributes are actually stored), or loop through both keys and values returning both. * **Show slide 15 - Explain** that this example shows the syntax to loop through everything, there are two important points to notice. First is that two temporary variables are required, one represents the key and the other the value for each iteration of the loop, but they don’t need to be named ‘key’ and ‘value’. The next is the items() method which gives the behaviour that we want i.e. return both key and value on each loop. * **Show slide 16 - Explain** that if you only want to loop through the dictionary returning just the keys or just the values for each KVP then you only require one temporary variable, and use either the keys() or values() methods respectively. * **Show slide 17 -** Summarise syntax points for looping through keys/values/items * **Show slide 18 - Ask** for a volunteer to **explain** what is happening in this code example. If some positive responses are forthcoming, thank the contributors then **explain** if not already covered.   *“In this example we’ve used the sorted() method, just like we can do with lists, to work with a sorted copy of the dictionary which could be useful when you need to print out large dictionaries with a high number of KVPs.”* |
| 15 mins | **Slides 19 - 22 - Nesting**   * **Show slide 19 - Explain** that nesting is when we’re using one code structure entirely inside another, we’ve done it already when we’ve nested an If statement within a For loop for example. There are many other common examples… * **Show slide 20 - Explain** that we’ve already seen this example, and it is technically nesting, a collection of dictionaries have been nested within a list. * **Show slide 21 - Explain** that this example is the opposite, instead of a list containing dictionaries, here each dictionary contains a list. Specifically, each dictionary has, amongst others, a KVP with the key of ‘favourite foods’ and the corresponding value of a list containing several strings. Notice, the syntax for declaring the list is just the same as we’ve used previously, i.e. square brackets and commas. * **Show slide 22 - Ask** for a volunteer to **explain** the syntax in this code example. If some positive responses are forthcoming, thank the contributors then **explain** if not already covered.   “*There are two things to note in this example, the first is that it demonstrates nesting by storing several dictionaries inside a ‘parent’ dictionary. The first key in the ‘cats’ dictionary is ‘weasley’ and its value is a new dictionary containing the KVPs (attributes) for Weasley. Then the next key of ‘noche’ has the value of another dictionary with Noche’s KVPs, and the same for Bigglesworth.*  *Next notice the for loop at the bottom, at the end of line 19 you can see that our For loop needs to iterate through all items in the cats dictionary (rather than just the keys or values), so two temporary variables are needed, they’re declared as cat\_name and cat\_info.*  *As we run the code the For loop iterates through the cats dictionary, on the first pass cat\_name will contain the first key ‘weasley’ and cat\_info will contain the whole dictionary of Weasley’s KVPs. On the next pass cat\_name will be noche, and cat\_info will be the next dictionary, and so on.*” |
| 50 mins | **Slides 23 - Practical Tasks**   * Share the following task sheet with learners: [TIFC1-PF-6 - Dictionaries - Tasks](https://docs.google.com/document/d/13TO9cV__AeARXsrtAmKBl1OCFJ0tmviE2fvMpMRj6W4/edit?usp=sharing) * Advise learners that they have XX minutes to complete as many tasks as they can. |