| **Session Objectives** | * Understand what a function is, how to define it, and when to use it * Understand the difference between an argument and a parameter * Understand when to use keyword and positional arguments |
| --- | --- |
| **Key Points** | * Functions, what they are and how to define it * Arguments and parameters * Keyword and positional arguments * Default and return values * Using functions with lists and dictionaries * Modules and how to import them |
| **Assessment** | * Via practical challenges - see task sheet below. * Via in class practice modifying code with peer feedback |
| **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-8 - Functions - Slides](https://docs.google.com/presentation/d/1TLXku0ntjtczhUsIhnNl2tnVzf70T9saBKnZh0z-7z4/edit?usp=sharing) * [TIFC1-PF-8 - Functions - Tasks](https://docs.google.com/document/d/1EtTYxuhdvGXFxKdoQ7ExFmnQi90D4B88bZR9pUhWPJI/edit?usp=sharing) * Instructor Only: [Functions Task Solutions](https://docs.google.com/document/d/1mwssDMy_JRgnfzwVeAByb_pSuQiQrOv2yGS2bdUzHmY/edit?usp=sharing) |

| **Time** | **Activity** |
| --- | --- |
| 5 minutes | **Slide 2: Learning Objectives**   * **Read from slide** |
| 10 minutes | **Slide 3: What is a Function?**   * Read from the slide to give a brief introduction to what a function is. |
| 10 minutes | **Slide 4: A Very Simple Function**   * **Ask learners what they think will happen if the code were to run. Ask them to explain the code to the class.** * **When done with discussion, run the code to show its functionality.** |
| 10 minutes | **Slide 5-6: Expand By Passing Information**   * **Slide 5:** Again, ask learners what they think would happen if this code were to run, ask one learner to walk through the code and how it works. When done, demo the code in an editor. * **Slide 6:** Indicate where the parameter and the argument is on the slide, and what their purposes are. Emphasize that often programmers use these terms interchangeably, so be cautious of this. |
| 20 minutes | **Slide 7-12: Positional and Keyword Arguments**   * **Slide 7: Read from the slide, then say,** * *“In the next slides, we will explore both positional and keyword arguments and their use case.”* * **Slide 8: Discuss what a positional argument is, say,** * *“ A positional argument is a type of argument in a function where the value is determined by its position in the function’s argument.*   *In the example on the slide, ‘animal\_type’ and ‘pet\_name’ are the positional arguments. They are passed to the ‘describe\_pet’ function in the order at which they are defined on line 1 of the code.”*   * **Demo the code in an editor to show the functionality.** After showing the code run, swap the arguments on line 5 so it looks like this: describe\_pet(‘weasley’, ‘cat’). Then, run the code again. The output should be: *“I have a weasley. My weasley’s name is Cat.”* This will show that the position of the arguments matter in positional arguments, and explain this concept to the learners. * **Say,**   - *“Use positional arguments when the order of arguments is consistent and predictable. Positional arguments are simple to use, but can get complicated when there are a large number of arguments.”*   * **When finished, read the slide.** * **Slide 9: Discuss what a keyword argument is, say,** * *“Order does not matter with a keyword argument. With a keyword argument, each argument is preceded by a keyword then an equal sign, to specify which parameter the argument corresponds to in a function.*   *In the example code, you can see that ‘animal\_type’ and ‘pet\_name’ are now keyword arguments. They have an equal sign after to explicitly associate the arguments with parameters.”*   * **Demo the code in an editor to show the functionality.** After showing the code, change line 5 so it says, ‘describe\_pet(pet\_name='weasley', animal\_type='cat')’. Run the code again and emphasize that the order does not matter with keyword arguments. * **Say,**   - *“Use keyword arguments when you need clarity and/or have a large number of arguments. Keyword arguments are also great since they let you specify only the arguments that you want, which helps function calls be more clear and readable.”*   * **Slide 10: Read from the slide, then discuss the code on the slide**. Mention that on line 1 animal\_type now has a default value set to it, and why this can be useful. **Say,** * *“ With default values, we can assign initial values to our parameters in a function, so that if no argument is provided for that parameter, the default value will be used instead.”* * **Run the code for learners to see the functionality.** * **Change line 5 to**: describe\_pet(pet\_name='weasley', animal\_type='dog') * **Run the code again to show that default values can also be overridden if needed.** * **Slide 11: Read from the slide, then look at the code.** Read each line of code to learners, then explain what the function is of each line. Explain how the return value works in this example. **Run the code to show its functionality.** * **Slide 12: Show the code, then say:** * *“ In this code, ‘pet\_last\_name’ is an optional argument. There is a default value of an empty string, which means that if the argument is not provided when calling the function it will default to an empty string.*   *The first call on line 8 does not provide a ‘pet\_last\_name’, so it uses a default value. The second function call on line 11, however, does provide an argument for this parameter. Therefore, the value is included when the code is run.”*   * **Run the code for learners to see the functionality. Say,**   - *“This code shows that optional arguments can give flexibility to our functions by allowing users to provide information only if needed, but also providing a default value if information is omitted.*” |
| 10 minutes | **Slide 13-16: Use of Functions with Lists, Dictionaries, and Loops**   * **Slide 13: Read from the slide.** * **Slide 14 - Dictionaries and Functions: Explain t**hat functions can be used with concepts that were learned previously in the course, dictionaries being one of them Discuss the code with the learners and show that there is a dictionary on line 2. Explain the functionality of the code with the learners, then run the code to show its functionality. * **Slide 15 - Lists and Functions: Read from the slide, then discuss the code with learners. Run the code to show functionality.** * **Slide 16 - Modifying Lists from a Function: Explain that one can also modify lists from a function. Say,** * *“This code simulates Weasley and Noche eating each food item stored in, ‘dinner\_items’, moving the food items into an empty list called, ‘empty\_items’, then displaying the eaten items. This example code shows that we can modify lists from within a function.”* * **Walk through each line of code** with the learners, explaining the concepts in the code (pop(), while loops, etc…). **Then run the code.** |
| 5 minutes | **Slide 17: Passing an Arbitrary Number of Arguments**   * **Say,** * *“At times, it may be uncertain how many arguments a function should accept in advance. Fortunately, Python permits a function to gather an unspecified number of arguments from the calling statement. You can utilize an asterisk (\*) in front of a parameter to transmit an arbitrary number of arguments to a function.”* * **Run the code to show its functionality.** Emphasize that line 5 only has one argument, but line 6 has three. |
| 5 minutes | **Slide 18: Mixing Positional and Arbitrary Arguments**   * **Say,** * *“Mixing positional and an arbitrary number of arguments in a function allows you to define a function with a fixed number of parameters, or a positional argument while also allowing for flexibility to accept an unknown number of additional arguments. This allows our function calls to be more versatile and accommodating to varying needs without needing to define multiple functions.*   *In the example code, by mixing positional arguments and an arbitrary number of arguments, it allows us to be flexible when specifying dinner ingredients for Weasley and Noche. The code accommodates different combinations of ingredients for their dinner bowls without needing to define functions with different sets of parameters.* **”**   * **Read the code line by line, explaining the concepts along the way. Run the code to show its functionality.** |
| 5 minutes | **Slide 19: Using Arbitrary Keyword Arguments**   * **Read from slide, then walk through code from slide with learners line by line before running the code.** |
| 10 minutes | **Slide 20-23: Modules**   * **Read from the slide.** |
| 60 mins | **Slide 24: Hands-On Challenges**   * **Share link** to task sheet, instruct learners that they now have 60 minutes to work through the task sheet. Once the time is up they should take a screenshot/snip of their last completed challenge and submit it on Canvas. * **Open breakout rooms** - instructor to select number/mix |