| **Session Objectives** | * Explain what a class is, how to create one, and its purpose * Introduce practical uses of a class (accessing attributes, etc) * Explore the concept of inheritance and practice using child classes * How to import classes into another piece of code |
| --- | --- |
| **Key Points** | * What is a class? * What are the components of a class? * Accessing attributes of a class * Introducing inheritance and the concept of child classes |
| **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-9 - Classes - Slides](https://docs.google.com/presentation/d/1JwH__9oEVOayDiepyNlbF6s1et8BG-hE9xB9R8qdWJw/edit?usp=sharing) * [TIFC1-PF-9 - Classes - Tasks](https://docs.google.com/document/d/1KMvrYAzbc4rQfgH_wHJgSTHUKSAAszkCXf1imxIYhZE/edit?usp=sharing) * Instructor Only: [Classes Task Solutions](https://docs.google.com/document/d/1OIofaVpe7guZ76w4d-krKt1l_jmY7GwZnkkr5OCMNpU/edit?usp=sharing) |

| **Time** | **Activity** |
| --- | --- |
| 5 minutes | **Slide 3: Class - What is it?**   * **Show Slide 1 - Explain the idea of object-oriented programming and that Python is an example of an object-oriented programming language. Then, explain the concept of a class. Explain that:** * **‘***Classes are like a blueprint for creating objects. The goal of a class is to explain or describe to Python what something is, what it does, or what it looks like. It is a general description of what something is, not a specific object. Now, let’s talk about how to create a class.’* |
| 10 minutes | **Slide 4-8: Creating and Using a Class**   * **Show slide 4:** Explain that you can model almost anything using a class and give some examples (example: robots, a house, animal). Then, say: * *‘To create a class, you first begin with the keyword, ‘class’ and then think of what you are trying to describe to Python. Remember, a class is a blueprint, so a general idea of what something is, not something specific.’* * **Show slide 5:** Introduce our topic that we will describe to Python through a class - a cat. * **Explain** to learners that Python cannot ‘see’ objects, so we need to describe objects to it. We need to think about what the object we are trying to describe is or what it can do. * **Emphasize** that at this point we are looking at specifically what a cat is and does, we are not looking at a specific cat, this will come later. * **Brainstorm** aspects of what makes a cat - cats have a name, fur, age, claws.. * **Brainstorm** with learners about what cats do - They like to sleep, climb, eat… * **Say**:   ‘ We will create a class that will store the name of a cat, the age of a cat, and give the cat the ability to climb and sleep.’   * **Slide 6:** **Show** learners the code on the slide. * **Explain** that what we described on the previous slide has now been put into a piece of code. Introduce the code to learners and walk through the concept of the piece of code. Look at the syntax and logic of the code, do not dwell too much on ‘self’ or the \_\_init\_\_() method for now, we will be breaking this down in the coming slides. |
| 10 minutes | **Slide 7-15: Let’s Break It Down**   * **Slide 7: Show and read from slide.** * **Slide 8 - Methods:** **Show slide 8 and say,** * *“A function that is part of a class is known as a method. This is the exact same thing as functions that we learned previously! Just named something different when within classes.“* * **Slide 9: Walk through with learners each method within the code provided. Say**, * *“We start a method by defining it, the exact same as when we start a function outside of a class. In this code, we have three methods.*   *The first, is a special method called the constructor. It is called when you first create a new object of a class. In this example, the code initializes the attributes ‘name’ and ‘age’ for the cat object that is being created, The ‘self’ parameter refers to the instance class itself.*  *The second method is the sleep method, which represents the action of a cat sleeping. When called, it prints a message to indicate that the cat is sleeping.*  *The third method represents the action of the cat climbing. When called, it prints a message that indicates the cat is climbing on a roof.*  *Each of these methods represents typical actions that a cat does; all cats sleep and climb, but Python does not know this so we need to tell it.”*   * **Slide 10 - \_\_init\_\_() Method:** Read the slide then move to slide 11 for use case in sample code. * **Slide 11: Show** where the \_\_init\_\_() method is located in the code. **Explain** that it is a special method that is always executed when a class is first being initiated. Say, * *“All classes have a function called \_\_init\_\_() that allows an object’s attributes to be initialized or to perform any setups needed for the object to function properly. “* * **Slide 12 - self:** Read from the slide then move to slide 13 for use case in sample code. * **Slide 13: Identify** where the self parameter is utilized in the code. **Say,** * ***“****The self parameter is a mandatory first parameter in method definitions of a class and that it references to the instance of a class so we can access attributes and methods within a class.”*   **Explain** the use case of self within the example code for each method, and why we need it.   * **Slide 14: Read from slide then show slide 15, which is a picture of Weasley, who we will be creating an instance of for our cat class.** |
| 10 minutes | **Slide 16-19: Making a Class of Weasley**   * **Slide 16: Show** code on slide 16. **Explain** to learners that we will be expanding on our class by creating an instance of the class. **Discuss** the syntax of creating an instance, then **demo** in an editor to show how it runs. * **Slide 17: Read from the slide**, which discusses what is happening in the background while the code is being run. * **Slide 18:** This slide has a visual representation of the relationship between our instance and class. **Say,** * *“Our class is just an idea of what something is. We are describing what things are to Python. We created a class that describes what actions all cats do - sleep and climb. Now, we are creating instances of the class cat to specify our objects, or what specific cats we want Python to interact with.”* * **Expand** to explain that we can create multiple instances of different cats, all coming from the same cat class that we created. * **Slide 19:** This is another example to explain the same concept. **Explain:** * Although houses are different with the interior, all houses share similarities. All houses have walls, doors, and windows. We can describe this to Python, then specify houses by creating an instance of the class. |
| 10 minutes | **Slide 20-21: Calling Methods**   * **Slide 20:** Read from the slide, then move on to slide 21. * **Slide 21:** Walk through the code with the learners, emphasizing the syntax for calling the sleep and climbing methods in the example code. Once completed, demo the code in an editor for learners to see. * Walk through with the learners what is happening in the background while the code is running, so they understand how Python is interpreting the code. |
| 5 minutes | **Slide 22-23: Accessing Attributes**   * **Slide 22: Read from the slide, then move to the next slide.** * **Slide 23: Explain** that one can also access specific attributes from our instance. Discuss the syntax and usage of this from the example code. When finished, demo the code so learners can see how it runs. |
| 5 minutes | **Slide 24-25: Adding multiple instances - Noche**   * **Slide 24: Read from slide then move on**. * **Slide 25: Read** from slide. Once done, walk through each line of code, explaining what each line does as well as the syntax for the code. Then, run the code to show the functionality. |
| 5 minutes | **Slide 26: Default Values**   * **Explain** what a default value is, and when it can be useful. * **Read from the slide**, then walk through the code, showing the changes specified on the slide. * **Run the code** for the learners to see the functionality. |
| 10 minutes | **Slide 27-29: Modifying Attribute Values**   * **Slide 27: Read from the slide**. * **Slide 28: Read** from the slide, which says there has been a change to line 25. **Explain** the syntax of line 25 in the code, then run it to show the functionality and changes. **Emphasize** where the changes are after the code has been run. * **Slide 29: Read** the slide. Then show the change in line 27 of the code to the learners. **Explain** the syntax, and what will happen if the code were to run. **Demonstrate** the code running in an editor to show how it functions. |
| 10 minutes | **Slide 30-32: Inheritance**   * **Slide 30: Read from the slide.** * **Slide 31: Introduce the child class**, Tiger(Cat). Explain the syntax from the code on the slide, from line 24-30. Explain that the child class inherits all characteristics/methods from its parent class, but it can now have new methods. **Say,** * *“ There are many different types of cats, some share common characteristics like sleeping and climbing, but others may have their own characteristics that they do not share with others. For example, although Weasley and Noche sleep and climb like a tiger does, they do not need to hunt for their own food or like swimming in pools of water. These characteristics are unique for big cats, such as a tiger.*   *Much like how children inherit qualities from their parents. The overall genetic makeup of a child will be from their parents, but they may have different likes and dislikes compared to their parents. We can create child classes to differentiate between different people or cats to specify their characteristics.”*   * **Read from the slide**, which explains the use of the super() function and the function of the \_\_init\_\_ method in a child class. * **Run the code** to show its functionality, explaining to the learners what is happening while the code runs. * **Slide 32: Read the slide,** then walk through the code with the learners. Focus on describing the syntax and what each line of code does. Once done, run the code to show the functionality. |
| 5 minutes | **Slide 33: Overriding Methods from the Parent Class**   * **Read from the slide.** * Show the learners that on line 14 the method climbing() was defined for the cat class. Then show the learners that in the new child class, on line 33 a new method was defined under the tiger class. Discuss the syntax for calling a child class method, Explain that we can override the parent class methods if needed. You can make comparisons again to children and their parents here. For example, saying that parents can enjoy eating spicy food, but children may not enjoy it as much so we have to emulate this to Python. * **Run the code to show that the original method has been overridden.** |
| 10 minutes | **Slide 34-36: Instances as Attributes**   * **Slide 34: Say,** * *“When modeling something from the real world in code, you may find that you’re adding more and more detail to a single class. You’ll find that you have a growing list of attributes and methods and that your files are becoming lengthy and too complex. In these situations, you might recognize that part of one class can be written as a separate class. You can break your large class into smaller classes that work together. Breaking down a large class into smaller, more concise classes is known as refactoring.*   *For example, if we find we are talking about a tiger’s attributes more than a cat’s, we should make a big cat instance as an attribute in the cat class. Then, we can include an instance in a BigCat() classes an attribute to represent the relationship between cats and big cats. “*   * **Slide 35: Walk through the code with learners. Explain the following points:** * We renamed the Cat class to DomesticCat to better represent domestic cats. * We created instances of DomesticCat named noche and weasley. * We accessed methods meow() and describe() for each domestic cat instance. * Both noche and weasley have access to the BigCat class methods through the big\_cat\_instance attribute, allowing them to roar like big cats if needed. This instance also shows the relationship between cats and big cats. * **Walk through the code line by line with the learners, then run the code to see the functionality** * **Slide 36: Read from slide.** |
| 5 minutes | **Slide 37: Importing Classes**   * **Read from the slide.** * **Demo the code provided in the notes section of the slide, to show that classes can be imported.** |
| 5 minutes | **Slide 38: Python Standard Library**   * **Read from the slide.** * **Provide learners with the following links:**   <https://pymotw.com/3/>  <https://docs.python.org/3/library/index.html> |
| 45 minutes | **Slide 39: Hands-On Challenges**   * **Share link** to task sheet, instruct learners that they now have 45 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 |