1. **Data Classes**  
   Video link [https://vimeo.com/851027228](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTEwMjcyMjg_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.vWJpcIXwuS0x_KO2lxHZvOcVs0zZqY5NWlsTju4e6no)  
   Length is 9 minutes
2. **Class Attributes**  
   Video link [https://vimeo.com/851410842](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTE0MTA4NDI_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.cwLK58kMeeObdt0mOeLBE9b3vwBHgoTn9t_ovvbZcBg)  
   Length is 7 minutes
3. **Inheritance**  
   Video link [https://vimeo.com/853191178](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTMxOTExNzg_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.EoPsuUljadLvnv2kRiu8U8mxwWHJZewkudFMnUdLSc0)  
   Length is 7 minutes
4. **Inheritance and Super**  
   Video link [https://vimeo.com/853424172](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTM0MjQxNzI_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.wV-sAPyBJnWkhPV6bmSzdLOe30usjV-kDAhtCWyfjmw)  
   Length is 6 minutes
5. **Inheritance Conclusion**  
   Video link [https://vimeo.com/853581589](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTM1ODE1ODk_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.AY2Wq0fgEqwB2teVM4zXW6EwapPbFHgeq9NR8xziZtQ)  
   Length is 4 minutes
6. **Composition**  
   Video link [https://vimeo.com/854820755](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84NTQ4MjA3NTU_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.eRsQ9JSuaZ-0YTVvsZG0-mK3kena9TVlkvs8e6B5QQ4)  
   Length is 12 minutes

**Collateral Material (programs used in the videos):**  
  
*In collateral, we try to provide the Python scripts and related files shown in the videos. In other words, we try to make it easy for you to reproduce the examples from the videos.*  
  
[Lesson8 Collateral Programs](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..eaqzoxlLdGmVYeQ4Oi-Z4AII84xHXT39FEWcThjiMZI)  
  
  
  
  
**Additional Content:**  
  
**Data Classes**  
  
[Data Classes in Python3.7+](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3JlYWxweXRob24uY29tL3B5dGhvbi1kYXRhLWNsYXNzZXMvP19fcz1tenBtNm5xM2NtdHp4MDN0azVkZyJ9.4ynBtLKYEIAXSD77uhsUdBOKpHkrlcItwxgwlT1FRlQ)  
Real Python article discussing data classes. This article covers data class syntax, compares data classes to (somewhat) similar things like tuples, dictionaries, and named tuples. The article also shows you how to set default values for attributes, and how to add methods to data classes. I recommend reading through the section named, "Adding Methods".  
  
  
**Class Attributes**  
  
[Python Class Attributes: An Overly Thorough Guide](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3d3dy50b3B0YWwuY29tL3B5dGhvbi9weXRob24tY2xhc3MtYXR0cmlidXRlcy1hbi1vdmVybHktdGhvcm91Z2gtZ3VpZGU_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.N5O8y9Qc_hRf1jTBANCj8VkKWLRk_I17LEK6zC3YZyM)  
Article by Charles Marsh that discusses class attributes. The article compares class attributes to instance attributes including discussing namespaces for class attributes and instance attributes. The article concludes by covering some good use cases for class attributes.  
  
  
**Inheritance and Super**  
  
[Object-Oriented Programming (OOP) in Python 3](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..iK8TLdBx4BVhDAAgZcfJw3e0CV2J9-EecxYdIVi08tM)  
Real Python article on object-oriented programming, read the section titled, "How Do You Inherit From Another Class in Python?".  This section covers inheritance basics including creating classes from a parent class. It also has a brief example of overriding a parent method and then using super() to call the parent class.  
  
[Supercharge Your Classes With Python super()](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3JlYWxweXRob24uY29tL3B5dGhvbi1zdXBlci8_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.9ABSzmJba1c4jQABUDVQQaM9TC_FnCdABOvwVnIjQZ4)  
While this article is very focussed on super(), it also covers a lot of Python inheritance basics. It provides an example of inheriting from a parent class and of calling super() in the dunder-init() method. It then expands on this and uses super() in other methods. I would read through the section titled, "What can super() do for you".  
  
  
**Composition**  
  
[The Composition Over Inheritance Principle](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3B5dGhvbi1wYXR0ZXJucy5ndWlkZS9nYW5nLW9mLWZvdXIvY29tcG9zaXRpb24tb3Zlci1pbmhlcml0YW5jZS8_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.kpXyXRoK5ltaF7D5XSNazvJEnmQJwmUn3zMsDQY0jcQ)  
This is a technical article, but nicely illustrates the problems that you can run into with class hierarchies. It also provides significant details regarding how composition can be used to solve some of these problems.  
  
  
  
  
**Exercises**  
  
Reference code for these exercises is posted on GitHub at:   
  
[https://github.com/twin-bridges/learning\_python/tree/main/lesson8/exercises](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzYwNDQxMDAyLCJuYmYiOjE3NjA0NDEwMDIsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJkMG9wNnp6ZGFvZjlrbGs3a29udyIsInRva2VuIjoiZDBvcDZ6emRhb2Y5a2xrN2tvbnciLCJzZW5kX2F0IjoxNzYwNDU0MDAwLCJlbWFpbF9pZCI6NzUwNDk2OCwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL2dpdGh1Yi5jb20vdHdpbi1icmlkZ2VzL2xlYXJuaW5nX3B5dGhvbi90cmVlL21haW4vbGVzc29uOC9leGVyY2lzZXM_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.IbMO0a6NS59xr-stC9Kkk4yL_xH2Hsk-W4oI00xoEkk)  
  
  
1. Create a data class named RouterFacts.  
  
This class should have the following attributes: hostname, vendor, network\_os, model, os\_version, interfaces, uptime\_sec, and serial\_number. All the attributes should be strings (type str), except for interfaces (which should be a list, type List) and uptime\_sec (which should be an integer, type int).  
  
Create an instance of this class using the following data.

{

'hostname': 'la-rtr1',

'vendor': 'cisco',

'network\_os': 'iosxr',

'model': '8201-SYS',

'os\_version': '7.0.12',

'interfaces': [

'HundredGigE0/0/0/24',

'HundredGigE0/0/0/25',

'HundredGigE0/0/0/26',

'HundredGigE0/0/0/27',

'HundredGigE0/0/0/28',

'HundredGigE0/0/0/29',

'HundredGigE0/0/0/30',

'HundredGigE0/0/0/31',

'HundredGigE0/0/0/32',

'HundredGigE0/0/0/33',

'HundredGigE0/0/0/34',

'HundredGigE0/0/0/35',

'MgmtEth0/RP0/CPU0/0'

],

'uptime\_sec': 93073,

'serial\_number': 'FOC2291AVYB'

}

Print out this object to standard output.  
  
Note, for type List, you will need to import the following:

from typing import List

2. Create a Router class that has two class attributes: count and all\_hosts.  
  
"count" should be initialized to 0 and all\_hosts should be initialized to a blank list.  
  
The class should have one instance attribute named host (instance attribute i.e. bound to self in dunder-init()).  
  
As part of dunder-init(), each Router object that you create should set the aforementioned "host" attribute and should increment the class "count" attribute. Additionally, each Router object that you create should add the "host" name to the "all\_hosts" list.  
  
In other words, "count" should count the total number of Router objects that exist and "all\_hosts" is a list of all the hostnames (of Router objects). Note, we are not taking any action for deleted objects (in other words, you do not need to handle Router object deletion).  
  
Create four Router objects.  
  
Print out the two class attributes: count, and all\_hosts.  
  
Verify that the count is 4 and that all\_hosts has all four of the "host" names.  
  
  
3. Create a class hierarchy where the parent class is NetworkDevice and takes a "host" attribute in dunder-init().  
  
Create three child classes named Router, Switch, and AccessPoint. Each of these three classes should inherit from NetworkDevice and should use the dunder-init() method.  
  
The three child classes should have one additional method dunder-repr (i.e. \_\_repr\_\_ ). This method is a representation of an object. It generally shows you what it would look like to recreate the object.  
  
Consequently, an example dunder-repr output for a Router object would be:

Router('[rtr1.domain.com](http://rtr1.domain.com/)')

Create one object of each child class (Router, Switch, and AccessPoint).  
  
For each of these objects, print out the "host" attribute and also print out the object itself. Note, if you have not defined dunder-str, Python will fallback and use dunder-repr. Consequently, printing out the object here will print out the dunder-repr that you created.  
  
  
4. Expand on exercise3, such that the Router class can accept an additional argument named "model" in dunder-init().  
  
This "model" parameter should default to None. The dunder-init() method should assign the "model" to the object. Additionally, you should use super() to call the dunder-init() method of the parent class (so that the "host" attribute gets set properly).  
  
Additionally create a print\_model() method in the Router class that prints out the model assigned to the object (if the model is not None).  
  
Create a Router class object that specifies both a host and a model. Print out the host attribute, the model attribute, and the object itself (you do not need to update dunder-repr to account for the model attribute). Finally, execute the print\_model() method on this Router object.  
  
  
5. [Optional] Create two classes: a Router class and a Channel class.  
  
The Channel class should have four methods: dunder-init(), connect(), read(), and write().  
  
Create two child classes of the Channel class: SSHChannel and TelnetChannel.  
  
Both the SSHChannel class and the TelnetChannel class should have a dunder-init() method that calls super() on the parent class and that sets the transport attribute. Additionally, these two classes should also have a connect() method.  
  
The connect() method of each of these child classes should just print a pretend message (i.e. we are pretending we are actually connecting). For example, "Fictional SSH connection to [host1.domain.com](http://host1.domain.com/)".  
  
The parent Channel class should accept three arguments in dunder-init(): host, username, and password. It should bind all of these arguments to the object.  
  
The read() and write() method in the parent class should just print a message. Once again we are doing a fictional read and write. The write() method should accept a second argument named "data" (i.e. the data we would write out to the channel).  
  
For the other main class, the Router class, you should accept five arguments: host, device\_type, username, password, and transport. The transport parameter should default to "ssh". In dunder-init(), you should assign both the "host" and the "device\_type" arguments to the object.  
  
You should also check the transport: if the transport is "ssh", then you should create an instance of the SSHChannel class and assign this to self.channel of the Router object. In other words, the Router object will refer to this other object using "self.channel". You could call the connect() method on this SSHChannel object.  
  
Similarly if the transport is "telnet", then you should create an instance of the TelnetChannel class and assign this object to self.channel of the Router object. Once again you should call the connect() method on this TelnetChannel object.  
  
Finally, the Router class should have both a read() and write() method. The "read" method should call self.channel.read() and the "write" method should call self.channel.write(data). The "write" method should require a second argument named "data" (i.e. the data to write out the channel).  
  
Create an instance of your Router class using a transport of "ssh". Do a test write() call and read() call on this Router object. Verify that you see the fictional messages that you expect.  
  
  
  
  
  
**CLASS OUTLINE**

1. **Data Classes**
   1. Syntax [0:19]
      1. Import “dataclass” decorator (PY3.7 or later) [0:20]
      2. Decorate your class [0:30]
      3. Define your class [0:55]
         1. Type hints [1:34]
   2. Data Class Equivalency (as far as argument passing and binding arguments) [2:44]
   3. Using this Class [3:34]
   4. Default Values [4:20]
      1. Default values must go last [4:25]
   5. Data classes generate some additional boilerplate automatically [4:45]
      1. \_\_repr\_\_ is automatically created [4:49]
      2. \_\_repr\_\_ is similar to \_\_str\_\_, \_\_str\_\_ will fallback and use \_\_repr\_\_ [5:20]
      3. \_\_eq\_\_ is also created automatically [5:59]
   6. Data classes - you can have methods, you can have more complex default values and more complex initialization [6:22]
2. **Class Attributes**
   1. Class Attributes - Attributes that are shared by all instances of the class [0:25]
      1. Referring to the name of the class and incrementing attribute [1:20]
      2. Referring to the attribute using self, read-only [1:36]
         1. Must be very careful about assignment at the instance level [2:05]
   2. Conflicts between instance attributes and class attributes - instance attribute will win [3:24]
   3. If there is not instance attribute, then Python will look for the attribute as a class attribute [4:05]
   4. We can also directly access the class attribute [4:31]
   5. Class Attribute Example [5:23]
3. **Inheritance**
   1. Class Hierarchy and Inheritance [0:14]
      1. Automatically inherits from “object” [0:34]
   2. We can expand on the default hierarchy by creating our own parent and child classes [1:30]
   3. The Python base class (“object”) has a set of methods built into it [2:03]
   4. Inheritance [2:50]
      1. When we implement \_\_init\_\_(), it overrides (replaces) the default \_\_init\_\_() in the base class (“object”) [3:06]
   5. Method Resolution Order (MRO) - Single Inheritance [4:36]
   6. Why are we doing this? [6:04]
      1. DRY - Don’t Repeat Yourself [6:20]
4. **Inheritance and Super**
   1. What if we want to augment an existing method [0:10]
   2. super() - allows us to explicitly call the method in the parent class [1:06]
      1. Problem - What if the parent class changes? [1:50]
         1. A good case for \*args and \*\*kwargs [2:20]
   3. super() - A second example [4:05]
5. **Inheritance Conclusion**
   1. Class Hierarchy and Inheritance [0:05]
      1. Avoid code duplication “DRY” [0:12]
      2. We can use super() if we need to augment parent class behavior [0:19]
      3. Single Inheritance, Method Resolution Order (MRO) [0:45]
      4. Multiple inheritance [0:55]
   2. Inheritance - The Good - reduction of code [1:10]
   3. Inheritance - The Bad [1:16]
      1. Implied code [1:20]
      2. Parent code can change [2:06]
      3. Maintaining consistency between method definitions [2:45]
6. **Composition**
   1. We can have other relationships between objects [0:13]
      1. Object attributes can refer to other object(s) [1:20]
   2. We can pass in objects via dunder-init. [1:41]
   3. Composition - Why?  [2:06]
      1. We do not want to multiply our classes [3:17]
      2. How could these be integrated together? [3:43]
   4. Composition vs Inheritance [7:27]
      1. You can use both inheritance and composition (you are not limited to either-or) [9:07]
   5. Example from Nornir [10:08]
   6. Key Takeaway - There are other ways to establish relationships between objects besides inheritance [12:11]