**Design Pattern:** Bridge (Structural)

**Module:**

[matplotlib](https://github.com/matplotlib/matplotlib)/[lib](https://github.com/matplotlib/matplotlib/tree/master/lib)/[matplotlib](https://github.com/matplotlib/matplotlib/tree/master/lib/matplotlib)/**ticker.py**

**matplotlib/lib/matplotlib/axis.py**

**Locator and Axis Classes:**

**The Locator class is used to design and determine the locations of the major ticks (scale numbers to explicitly show) and the minor ticks (scale numbers that are not shown and appears between the major ticks) of an axis. In other words, the overall purpose of the Locator class is to define the scale that is used and seen on an axis of a graph. The Axis class is used to define an axis of a graph.**

**The Locator class utilizes the Bridge design pattern. To recap, the Bridge design pattern is a structural design pattern that prefers composition over inheritance so that two abstract implementations can vary independently. An example is to have a single webpage with different selection of themes. One approach is to create two WebPage objects with one having a light theme, and the other a dark theme. However, by using the Bridge Design Pattern, a new Theme class is created. Theme objects are then used to change the theme of a WebPage object without creating multiple WebPage objects with different themes. In this case, each Axis object can use one Locator object and it can be interchanged with other Locator objects. Keep in mind that any Axis object can use any Locator object. The Axis object is analogous to the WebPage object and the Locator objects are analogous to the Theme objects.**

**Justification:**

**The Locator and Axis classes design pattern can be confused with the Composite or Decorator design pattern. The pair does not follow the Composite design pattern since the two classes cannot be treated as a single object and that the two classes must be independent from each other. It cannot be the Decorator design pattern since when a behaviour to a Locator object is added, it will change the behaviour of the underlying Locator objects. Moreover, an Axis object doesn’t create a Locator object, it is assigned to it. The two classes have two different responsibilities, Locator class for setting the tickers of the axis, the Axis class to define the axis of the graph. This does not conform to the Decorator design pattern.**

**Classes Used:**

1. Axis(artist.Artist) – *This class uses the Locator class*
   1. + set\_major\_locator()
   2. + set\_minor\_locator()
2. TickHelper(object) – *This class contains the axis for which the Locator class operates on.*
   1. - axis *(Attribute)*
   2. + set\_axis()
   3. + create\_dummy\_axis()
   4. + set\_view\_interval()
   5. + set\_data\_interval()
   6. + set\_bounds()
3. Locator(TickHelper)
   1. - MAXTICKS = 1000
   2. + tick\_values()
   3. + set\_params()
   4. - \_\_calls\_\_()
   5. + raise\_if\_exceeds()
   6. + view\_limits()
   7. + autoscale()
   8. + pan()
   9. + zoom()
   10. + refresh()
4. IndexLocator(Locator)
   1. - \_\_init\_\_()
   2. + set\_params() <<override>>
   3. - \_\_ call\_\_()
   4. + tick\_values() <<override>>
5. FixedLocator(Locator)
   1. - \_\_init\_\_()
   2. + set\_params() <<override>>
   3. - \_\_call\_\_()
   4. + tick\_values() <<override>>
6. NullLocator(Locator)
   1. - \_\_call\_\_()
   2. + tick\_values() <<override>>
7. LinearLocator(Locator)
   1. - \_\_init\_\_()
   2. + set\_params() <<override>>
   3. - \_\_call\_\_()
   4. + tick\_values() <<override>>
   5. - \_set\_numticks()
   6. + view\_limits() <<override>>