The solution to this bug follows precisely the original plan for it from deliverable 3. A handler for the legend Annotation was added and it was mapped to Annotation objects like the other legend items. It additionally makes use of handlers for Texts and Arrows (that we also added), because annotations can be composed of texts and/or arrows which are themselves separate entities. Text objects (super class of annotations) were added to the list of objects to be added to the legend and import statements were added where necessary.

As for composing the actual legend item, all types of annotations have been broken down into 3 cases: blank, text, arrow (with or without text). Blank annotations show up as blank on the legend. Making the legend omit adding this item by default would require going outside of this self-contained solution and this presents a niche use case, anyways. A user making a blank annotation constitutes a very specific action and if they wanted to hide it, they could just add the appropriate argument label onto the annotation. Otherwise, it would likely be intended behaviour on their part. Texts, again, are only omittable by user action for the same reasons. The legend item will be a replication of the text in the original font, colour, and style (scaled to an appropriate size). Texts that exceed the width of the legend item field will simply be cut off. This was chosen because sizing down texts was deemed to be more useless to the end user if they cannot read it in the first place. Annotations with arrows are perfectly replicated into the legend area. Colour, style, and thickness of different components are respected.

There is a competing pull request with regards to this bug. As explained, an unrelated contributor worked on this a year ago and left it inactive after failing to pass error tests. Recently, it was picked up by who we suspect are fellow D01 students. As such, we spent time running and profiling their solution to see what we were up against in terms of winning the pull request.

Our solution differs in that texts are represented based on the actual text string even when cut off while the other one replaces the text with a base string set (“Aa” or similar) in the same styling as the original when exceeding a certain limit character limit (the limit is often premature and depends on a character limit rather than the actual space available). We do not like this because there is no way to differentiate between long texts that have the same style in every other respect other than the actual string. Additionally, we have a strict improvement when handling arrows. The “linestyle” property of FancyArrows is not respected in the other solution and thus does not reflect certain aspects of arrows (like whether it is dashed). The other solution chooses to make “text + arrow” annotations a special case and displays both the text (usually in base text form “Aa”) and arrow within the small confines of the legend. We do not believe this is useful for anyone for the same reasons we do not like how texts are handled and because it takes away emphasis from the arrow (reducing its size and identifiability). Another difference is that we also decided not to legend Text and Arrow items on the plot outside of actual Annotation objects (by not mapping the handlers for them to legend creation). This was not the purpose of the bug fix nor do we see any useful functionality for it that wouldn’t interfere with normal operation. The default nature of entries being added to the legend unless explicitly argued against can prove to be annoying for users who simply want to add simple texts and arrows without having them showing up on the legend. We think that making it so that Texts/FancyArrows in Annotations being added to the legend by default and raw Texts/FancyArrows not being added to the legend by default offers the most flexibility without having to resort to omission arguments on the part of the user. If a user wants to make an arrow/text to show up on the legend, they just need make it an Annotation. However, this functionality, should anyone choose to do so, can be trivially added onto our solution with the addition of 2 lines of code in the “\_default\_handler\_map” object in legend\_handler.py with appropriate import statements. The handlers we implemented in for Texts and FancyArrows as a side effect of the Annotation handler will already suffice for this simple addition. Overall, we think we have a more complete and well-thought out solution.