Notifier - Home assignment

The purpose of this module is to provide a way to notify an external API when something of interest has happened in our system.

When something changes (any change) on any supported entity - this module gets used. (You decide on the API. Is it a class? A single function?) [[1]](#footnote-0)

The currently supported entity types are:

* Company
* Event
* Webinar
* Content Item
* CompanyForEvent
* CompanyCompetitor
* CompanyForWebinar

Simple class implementations are provided so that you can explore them.

However you decide on implementing it, the input should be:

* entity\_obj: an object representing an entity in its NEW/UPDATED state (might be None, if the object is now physically deleted)
* original\_entity\_obj: an object representing the same entity BEFORE the updated state (might be None, if this is an added entity)
* type: a string representing the entity type.

(you can assume that AT LEAST entity\_obj or original\_entity\_obj are not None)

The module should decide whether to notify the external system if at least one of the “notify if either” conditions are true, and notify on the correct object (meaning that the external service thinks that the “notify on” object is the one that actually changed):

For Company:

* Notify if either:
* New object
* Object is now physically deleted
* “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
* “crawling\_status” attribute has changed and is now one of TEXT\_ANALYZED, TEXT\_UPLOADED
* Notify on:
  + Itself (the company)

For Event:

* Notify if either:
* New object
* Object is now physically deleted
* “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
* “is\_blacklisted” attribute has changed (boolean, from true to false or vice-versa)
* “crawling\_status” attribute has changed and is now one of TEXT\_ANALYZED, TEXT\_UPLOADED
* Notify on:
  + Itself (the event)

For Webinar:

* Notify if either:
* New object
* Object is now physically deleted
* “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
* “is\_blacklisted” attribute has changed (boolean, from true to false or vice-versa)
* “crawling\_status” attribute has changed and is now one of TEXT\_ANALYZED, TEXT\_UPLOADED
* Notify on:
  + Itself (the webinar)

For Content Item:

* Notify if either:
* New object
* Object is now physically deleted
* “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
* “is\_blacklisted” attribute has changed (boolean, from true to false or vice-versa)
* “crawling\_status” attribute has changed and is now one of TEXT\_ANALYZED, TEXT\_UPLOADED
* Notify on:
  + Its “company” attribute

For CompanyForEvent:

* Notify if either:
  + New object
  + Object is now physically deleted
  + “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
  + “is\_blacklisted” attribute has changed (boolean, from true to false or vice-versa)
* Notify on:
  + Its “event” attribute.

For CompanyCompetitor:

* Notify if either:
  + New object
  + Object is now physically deleted
  + “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
* Notify on:
  + Its “company” attribute.

For CompanyForWebinar:

* Notify if either:
  + New object
  + Object is now physically deleted
  + “is\_deleted” attribute has changed (boolean, from true to false or vice-versa)
  + “is\_blacklisted” attribute has changed (boolean, from true to false or vice-versa)
* Notify on:
  + Its “webinar” attribute.

# Deliverables:

* You can use any language you’d like -   
  If you use our starter code and Python - please use Python 3.8+ (note we are using keyword-only arguments which is not supported in earlier versions of Python)
* For the purpose of this assignment, printing the “notify on” object to the console will suffice.
* Simple tests to make sure your logic works (no need for unittest or pytest, just a simple script that implements different scenarios would do)
* Though using external packages is not needed for our version of the solution - you may use as many as needed if you feel your approach requires one.
* Hint: will the conditions for notifying ever change? How about the “notify on”? Maybe more entities will be added?
* Please allow about 2 hours for this assignment, and hand-in whatever you got at the end. It doesn’t have to be perfect for us to assess your code.
* If you require any assistance or have any questions - please contact us. Someone should have been assigned to help.

1. \*\* How? In real life we use Django’s ORM and the base classes for these “notifiable” entities wrap the “save” and “update” (and other similar methods) to call this module. The decision on actually notifying the service remains a logic in this module. [↑](#footnote-ref-0)