**Fundamentals of Programming project**

**Group:**

**Muhammad Amad Naeem 467469**

**Hunzala Asaad Qureshi 474644**

**Faisal Rasool Qazi 455689**

**Mechanical Section B**

**Batch : ME-15**

**Problem 1:**

****

Explanation:

This code creates a class called NewsStory. A class is like a blueprint for creating objects with similar properties and behaviors.

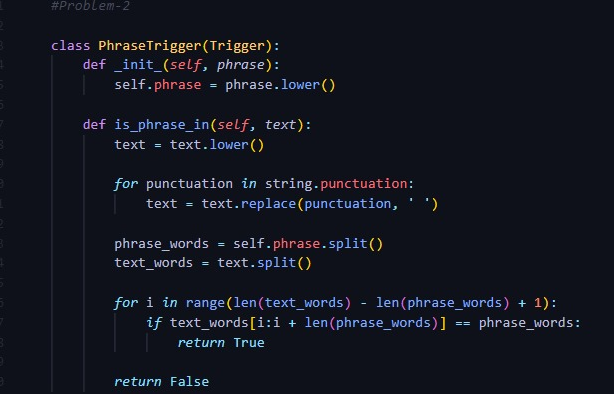
In this blueprint, there are some attributes like guid, title, description, link, and pubdate, which represent different pieces of information about a news story.

The \_\_init\_\_ method (the double underscores on each side) is a special method that initializes the object when it's created. It takes in arguments like guid, title, etc., and assigns them to the object's attributes.

Then there are several methods like get\_guid(), get\_title(), etc. These methods are like functions that allow you to access the values of the attributes from outside the class. For example, get\_title() returns the title of the news story.

So, you can use this NewsStory class to create objects that represent news stories, and you can get information about those news stories using the methods provided.

**Problem 2:**

****

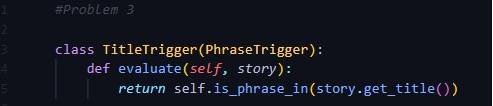
Explanation:

This code defines a class called PhraseTrigger that checks if a given phrase is present in a block of text.

The \_\_init\_\_ method initializes the object by storing the lowercase version of the input phrase.

The is\_phrase\_in method checks whether the stored phrase is present in a given text. It converts both the text and the phrase to lowercase for case-insensitive comparison. Then, it replaces punctuation marks with spaces in the text to ensure accurate word separation. After splitting both the phrase and the text into individual words, it iterates through the text to find if the phrase matches any continuous sequence of words. If a match is found, it returns True; otherwise, it returns False.

**Problem 3:**

**Top of Form**

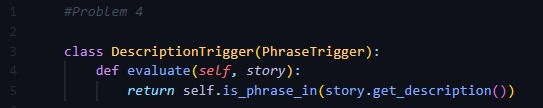
Explanation:

This code defines a class called TitleTrigger, which is a specific type of trigger that checks if a given phrase is present in the title of a news story.

This class inherits from another class called PhraseTrigger, which likely contains methods for checking if a phrase is present in a block of text.

The evaluate method in TitleTrigger takes a story as input. Inside the method, it retrieves the title of the story using the get\_title() method of the story object, and then it calls the is\_phrase\_in() method (likely inherited from PhraseTrigger) to check if the stored phrase is present in the title. Finally, it returns True if the phrase is found in the title, and False otherwise.

Problem 4:



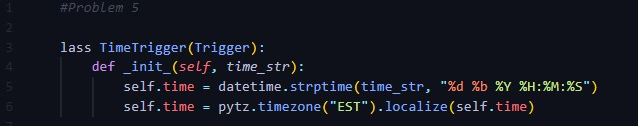
**Explanation:**

This code defines a class named DescriptionTrigger, which is a specific type of trigger. This trigger checks if a given phrase is present in the description of a news story.

Similar to the previous example, this class also inherits from another class named PhraseTrigger. That parent class likely contains methods for checking if a phrase is present in a block of text.

The evaluate method in DescriptionTrigger takes a story as input. Inside this method, it retrieves the description of the story using the get\_description() method of the story object. Then, it calls the is\_phrase\_in() method (likely inherited from PhraseTrigger) to check if the stored phrase is present in the description. Finally, it returns True if the phrase is found in the description, and False otherwise.

**Problem 5:**

****

Explanation:

TimeTrigger that represents a trigger based on a specific time.

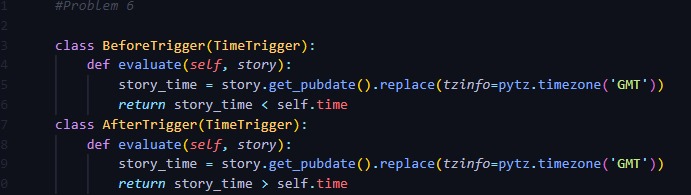
The class has an \_\_init\_\_ method, which is used for initializing objects. It takes a string time\_str as input, representing a date and time in a particular format ("%d %b %Y %H:%M:%S"). This format includes day, month (as a three-letter abbreviation), year, hour, minute, and second.

Inside the \_\_init\_\_ method, the string time\_str is converted into a datetime object using Python's strptime function from the datetime module. This function converts a string representation of a date and time into a datetime object according to the specified format.

Then, the code localizes the datetime object to the "EST" timezone using the localize function from the pytz module. This ensures that the datetime object is interpreted correctly within the Eastern Standard Time (EST) timezone.

So, essentially, this class allows you to create a time trigger object by specifying a specific date and time, which is then converted into a datetime object and localized to the Eastern Standard Time timezone.

**Problem-06**



**Explanation:**

defines two classes, BeforeTrigger and AfterTrigger, both of which inherit from a TimeTrigger class. These classes are used to evaluate whether a given news story's publication date is before or after a specified time.

BeforeTrigger Class

Purpose: To check if a news story was published before a certain time.

Method:

evaluate(self, story): Takes a story object as input.

Retrieves the publication date of the story using story.get\_pubdate().

Converts the publication date to GMT timezone.

Compares this date with the trigger's specified time (self.time).

Returns True if the story's publication date is before self.time, otherwise False.

AfterTrigger Class

Purpose: To check if a news story was published after a certain time.

Method:

evaluate(self, story): Takes a story object as input.

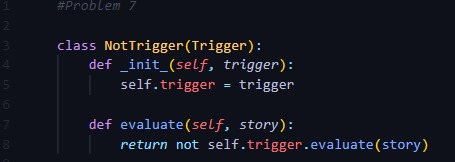
Retrieves the publication date of the story using story.get\_pubdate().

Converts the publication date to GMT timezone.

Compares this date with the trigger's specified time (self.time).

Returns True if the story's publication date is after self.time, otherwise False.

Problem 07



Explanation:

defines a NotTrigger class, which is used to invert the result of another trigger. Here's a detailed breakdown:

NotTrigger Class

Inheritance: Inherits from the Trigger class.

Purpose: To create a trigger that negates the result of another trigger.

Methods

\_\_init\_\_(self, trigger):

Purpose: Constructor to initialize the NotTrigger object.

Parameters: Takes a trigger object as an argument.

Implementation: Assigns the provided trigger object to an instance variable self.trigger.

Issue: There is a typo in the method name \_\_init\_\_. It is written as \_init\_ which should be \_\_init\_\_. Also, setf should be self.

evaluate(self, story):

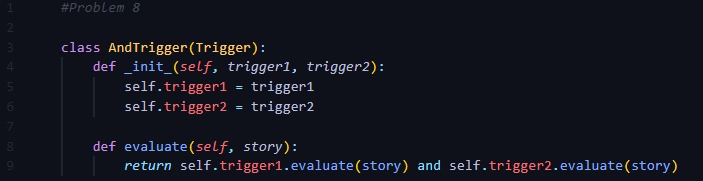
Purpose: Evaluates whether the given story object does not satisfy the wrapped trigger.

Parameters: Takes a story object as an argument.

Implementation: Calls the evaluate method of the wrapped trigger and negates its result using not.

Issue: There is a typo in the parameter list. It should use a comma , instead of a semicolon ; between self and story.

**Problem 08**

****

**Explanation:**

defines an AndTrigger class, which is used to combine two triggers such that the combined trigger evaluates to True only if both of the individual triggers evaluate to True for a given story. Here is a breakdown:

AndTrigger Class

Inheritance: Inherits from the Trigger class.

Purpose: To create a composite trigger that combines two triggers using a logical AND operation.

Methods

Constructor (\_\_init\_\_(self, trigger1, trigger2)):

Purpose: Initializes the AndTrigger object with two triggers.

Parameters: Takes two trigger objects, trigger1 and trigger2.

Implementation: Assigns the provided triggers to instance variables self.trigger1 and self.trigger2.

Issues: The method name \_\_init\_\_ is missing the double underscores (\_\_), and triggerl and trigger2 should be trigger1 and trigger2.

evaluate(self, story):

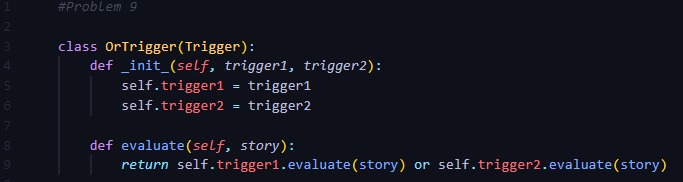
Purpose: Evaluates whether both triggers evaluate to True for the given story.

Parameters: Takes a story object as an argument.

Implementation: Calls the evaluate method of both stored triggers with the story object and combines their results using the logical AND operator (and).

Issues: There is a typo in the parameter list for the evaluate method. setf should be self. Also, the second evaluation should be self.trigger2.evaluate(story) instead of repeating self.trigger1.evaluate(story).

**Problem 09**



Explanation:

defines an OrTrigger class, which is used to combine two triggers such that the combined trigger evaluates to True if at least one of the individual triggers evaluates to True for a given story. Here's a breakdown:

OrTrigger Class

Inheritance: Inherits from the Trigger class.

Purpose: To create a composite trigger that combines two triggers using a logical OR operation.

Methods

Constructor (\_\_init\_\_(self, trigger1, trigger2)):

Purpose: Initializes the OrTrigger object with two triggers.

Parameters: Takes two trigger objects, trigger1 and trigger2.

Implementation: Assigns the provided triggers to instance variables self.trigger1 and self.trigger2.

Issues: The method name \_init\_ is missing the double underscores (\_\_), and seLjc should be self. Also, triggerl and trigger2 should be trigger1 and trigger2.

evaluate(self, story):

Purpose: Evaluates whether at least one of the triggers evaluates to True for the given story.

Parameters: Takes a story object as an argument.

Implementation: Calls the evaluate method of both stored triggers with the story object and combines their results using the logical OR operator (or).

Issues: There is a typo in the parameter list for the evaluate method. seljc should be self.

**Problem 10**



Explanation:

defines a function filter\_stories that filters a list of stories based on a list of triggers. Here's a very short explanation:

filter\_stories Function

Purpose: To return a list of stories that satisfy at least one trigger in the trigger list.

Parameters:

stories: A list of story objects.

triggerlist: A list of trigger objects.

Implementation:

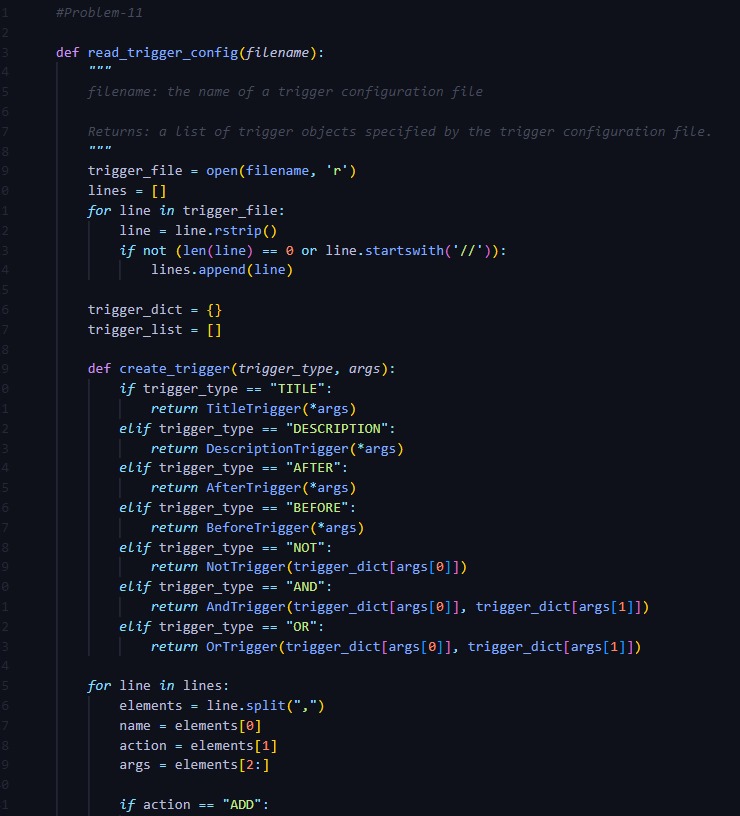
Initializes an empty list filtered\_stories.

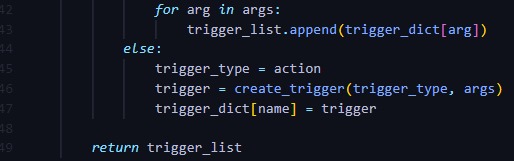
Iterates over each story in stories.

For each story, iterates over each trigger in triggerlist.

If a trigger evaluates to True for the story, the story is added to filtered\_stories, and the inner loop breaks to avoid checking further triggers.

Returns the filtered\_stories list.

**Problem 11**



Explanation:

Function Definition: The function read\_trigger\_config takes a filename as an argument.

Open File: The file is opened for reading.

Read Lines: The file is read line-by-line, and each line is stripped of trailing whitespace.

Ignore Empty/Comment Lines: Lines that are empty or start with // (comments) are ignored.

Store Valid Lines: Non-empty, non-comment lines are stored in a list called lines.

Initialize Dictionaries and Lists:

trigger\_dict: A dictionary to map trigger names to trigger objects.

trigger\_list: A list to store the final set of triggers.

Helper Function: A nested function create\_trigger is defined to create different types of trigger objects based on the trigger type and arguments.

This function uses conditional statements to create specific trigger objects like TitleTrigger, DescriptionTrigger, AfterTrigger, BeforeTrigger, NotTrigger, AndTrigger, and OrTrigger.

Process Each Line: Each line from the lines list is processed:

Split Line: The line is split by commas into elements.

Extract Components: The first element is the trigger name, the second is the action (trigger type or "ADD"), and the rest are arguments.

Add Triggers: If the action is "ADD", the triggers specified in the arguments are added to trigger\_list.

Create Triggers: Otherwise, the action is treated as a trigger type, and a trigger object is created using create\_trigger and stored in trigger\_dict with the given name.

Return Result: The function returns trigger\_list, which contains all the triggers specified in the configuration file.