**Solutions:**

**Problem #1 - Data Stream Ingestion**:

class DataStream:  
 def \_\_init\_\_(self):  
 # Dictionary to store the last printed timestamp for each string  
 self.last\_printed = {}  
  
 def should\_output\_data\_str(self, timestamp: int, data\_string: str) -> bool:  
 # Check if the data\_string can be printed  
 if data\_string not in self.last\_printed or timestamp >= self.last\_printed[data\_string] + 5:  
 self.last\_printed[data\_string] = timestamp # Update the last printed time  
 return True  
 return False  
data\_stream = DataStream()  
output = []  
output.append(data\_stream.should\_output\_data\_str(timestamp=0, data\_string="hello"))   
output.append(data\_stream.should\_output\_data\_str(timestamp=1, data\_string="world"))   
output.append(data\_stream.should\_output\_data\_str(timestamp=6, data\_string="hello"))   
output.append(data\_stream.should\_output\_data\_str(timestamp=7, data\_string="hello"))   
output.append(data\_stream.should\_output\_data\_str(timestamp=8, data\_string="world"))  
print("[" + " ".join(map(lambda x: str(x).lower(), output)) + "]")

**output:**

**[true, true, true, false, true]**

**Explanation:**

The code defines a DataStream class that ensures each string is printed at most once for every 5 seconds. It provides a method, **should\_output\_data\_str**, which decides whether a given string should be printed based on the timestamp of when it was last printed.

**Timestamp**: The current time

data\_stream = DataStream() **#which creates an object to the DataStream class**

**print ("[" + " ".join (map(lambda x: str(x).lower(), output)) + "]")**

Converts the output list (which contains True/False) to lowercase strings.

Joins the strings with a space and formats them in square brackets.