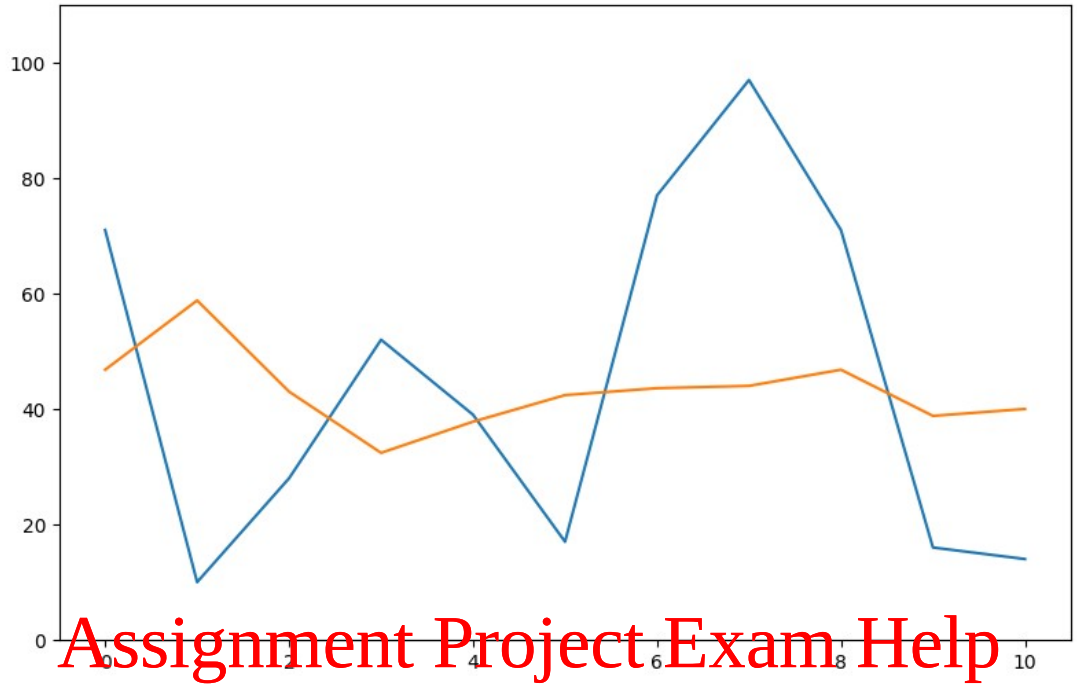
In [2]:
```
# import statements
from time import sleep
from kafka import KafkaConsumer
import datetime as dt
import matplotlib.pyplot as plt
import statistics

# this line is needed for the inline display of graphs in Jupyter Notebook
%matplotlib notebook

topic = 'Week9-Topic'

def annotate_max(x, y, ax = None):
    ymax = max(y)
    xpos = y.index(ymax)
    xmax = x[xpos]
    text = 'Max: Time={}, Value={}'.format(xmax, ymax)
    if not ax:
        ax=plt.gca()
    ax.annotate(text, xy=(xmax, ymax), xytext=(xmax, ymax+5),
arrowprops=dict(facecolor='red', shrink=0.05),)

def annotate_min(x, y, ax = None):
    ymin = min(y)
    xpos = y.index(ymin)
    xmin = x[xpos]
    text = 'Min: Time={}, Value={}'.format(xmin, ymin)
    if not ax:
        ax=plt.gca()
    ax.annotate(text, xy=(xmin, ymin), xytext=(xmin, ymin-5),
arrowprops=dict(facecolor='orange', shrink=0.05),)

def connect_kafka_consumer():
    _consumer = None
    try:
         _consumer = KafkaConsumer(topic,
                                   consumer_timeout_ms=10000, # stop iteration if no
message after 10 sec
                                   auto_offset_reset='earliest', # comment this if
you don't want to consume earliest available message
                                   bootstrap_servers=['localhost:9092'],
                                   api_version=(0, 10))
    except Exception as ex:
        print('Exception while connecting Kafka')
        print(str(ex))
    finally:
        return _consumer

def init_plots():
    try:
        width = 9.5
        height = 6
        fig = plt.figure(figsize=(width,height)) # create new figure
        fig.subplots_adjust(hspace=0.8)
```

```
        ax1 = fig.add_subplot(111)
        ax1.set_xlabel('Time')
        ax1.set_ylabel('Value')
        fig.suptitle('Real-time uniform stream data visualization with interesting
points') # giving figure a title
        fig.show() # displaying the figure
        fig.canvas.draw() # drawing on the canvas
        return fig, ax1
    except Exception as ex:
        print(str(ex))

def consume_messages(consumer, fig, ax1):
    try:
        # container for x and y values
        x1, y1, y2 = [], [], []
        check = 0
        # print('Waiting for messages')
        for message in consumer:
            data = str(message.value.decode('utf-8')).split(', ')
            x1.append(data[0])
            y1.append(int(data[1]))
            if len(y1) > 5:
                # print (y1[:5])
                y2.append(statistics.mean(y1[:5]))
            else:
                y2.append(0)
                # print(y)
            # we start plotting only when we have 10 data points
            if len(y1) > 10:

                ax1.clear()
                ax1.plot(x1, y1)
                ax1.plot(x1, y2)
                ax1.set_xlabel('Creation Time')
                ax1.set_ylabel('Value')
                ax1.set_title('Creation Time Vs Value')
                ax1.set_ylim(0,110)
                ax1.set_yticks([0,20,40,60,80,100])
                annotate_max(x1, y1, ax1)
                annotate_min(x1, y1, ax1)

                fig.canvas.draw()
                x1.pop(0) # removing the item in the first position
                y1.pop(0)
                y2.pop(0)

        plt.close('all')
    except Exception as ex:
        print(str(ex))

if __name__ == '__main__':

    consumer = connect_kafka_consumer()
    fig, ax1 = init_plots()
    consume_messages(consumer, fig, ax1)
```




```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-2-d572de577c13> in <module>
    102     consumer = connect_kafka_consumer()
    103     fig, ax1 = init_plots()
--> 104     consume_messages(consumer, fig, ax1)
    105
    106

<ipython-input-2-d572de577c13> in consume_messages(consumer, fig, ax1)
     78             if len(y1) > 10:
     79
---> 80                 ax1.clear()
     81                 ax1.plot(x1, y1)
     82                 ax1.plot(x1, y2)

~/.local/lib/python3.8/site-packages/matplotlib/axes/_base.py in clear(self)
   1175     def clear(self):
   1176         """Clear the axes."""
-> 1177         self.cla()
   1178
   1179     def get_facecolor(self):

~/.local/lib/python3.8/site-packages/matplotlib/axes/_base.py in cla(self)
   1053
   1054         for name, spine in self.spines.items():
-> 1055             spine.cla()
```

```
   1056
   1057         self.ignore_existing_data_limits = True

~/.local/lib/python3.8/site-packages/matplotlib/spines.py in cla(self)
    236         self._position = None  # clear position
    237         if self.axis is not None:
--> 238             self.axis.cla()
    239
    240     def _adjust_location(self):

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in cla(self)
    786                 mpl.rcParams['axes.grid.which'] in ('both', 'minor'))
    787
--> 788         self.reset_ticks()
    789
    790         self.converter = None

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in reset_ticks(self)
    809             pass
    810         try:
--> 811             self.set_clip_path(self.axes.patch)
    812         except AttributeError:
    813             pass

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in set_clip_path(self,
clippath, transform)
    899     def set_clip_path(self, clippath, transform=None):
    900         martist.Artist.set_clip_path(self, clippath, transform)
--> 901         for child in self.majorTicks + self.minorTicks:
    902             child.set_clip_path(clippath, transform)
    903         self.stale = True

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in __get__(self, instance,
cls)
    616             if self._major:
    617                 instance.majorTicks = []
--> 618                 tick = instance._get_tick(major=True)
    619                 instance.majorTicks.append(tick)
    620                 return instance.majorTicks

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in _get_tick(self, major)
   2011         else:
   2012             tick_kw = self._minor_tick_kw
-> 2013         return XTick(self.axes, 0, major=major, **tick_kw)
   2014
   2015     def set_label_position(self, position):

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in __init__(self, *args,
**kwargs)
    424         self.tick2line.set(
    425             xdata=[0], ydata=[1],
--> 426             transform=self.axes.get_xaxis_transform(which="tick2"),
    427             marker=self._tickmarkers[1],
    428         )
```

```
~/.local/lib/python3.8/site-packages/matplotlib/axes/_base.py in
get_xaxis_transform(self, which)
    718         elif which == 'tick2':
    719             # for cartesian projection, this is top spine
--> 720             return self.spines['top'].get_spine_transform()
    721         else:
    722             raise ValueError('unknown value for which')

~/.local/lib/python3.8/site-packages/matplotlib/spines.py in
get_spine_transform(self)
    391     def get_spine_transform(self):
    392         """Return the spine transform."""
--> 393         self._ensure_position_is_set()
    394
    395         position = self._position

~/.local/lib/python3.8/site-packages/matplotlib/spines.py in
_ensure_position_is_set(self)
    217             # default position
    218             self._position = ('outward', 0.0)  # in points
--> 219             self.set_position(self._position)
    220
    221     def register_axis(self, axis):

~/.local/lib/python3.8/site-packages/matplotlib/spines.py in set_position(self,
position)
    381         self.set_transform(self.get_spine_transform())
    382         if self.axis is not None:
--> 383             self.axis.reset_ticks()
    384         self.stale = True
    385

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in reset_ticks(self)
    809             pass
    810         try:
--> 811             self.set_clip_path(self.axes.patch)
    812         except AttributeError:
    813             pass

~/.local/lib/python3.8/site-packages/matplotlib/axis.py in set_clip_path(self,
clippath, transform)
    898
    899     def set_clip_path(self, clippath, transform=None):
--> 900         martist.Artist.set_clip_path(self, clippath, transform)
    901         for child in self.majorTicks + self.minorTicks:
    902             child.set_clip_path(clippath, transform)

~/.local/lib/python3.8/site-packages/matplotlib/artist.py in set_clip_path(self,
path, transform)
    751         if transform is None:
    752             if isinstance(path, Rectangle):
--> 753                 self.clipbox = TransformedBbox(Bbox.unit(),
    754                                                path.get_transform())
    755                 self._clippath = None
```

```
~/.local/lib/python3.8/site-packages/matplotlib/transforms.py in unit()
    797     def unit():
    798         """Create a new unit `Bbox` from (0, 0) to (1, 1)."""
--> 799         return Bbox([[0, 0], [1, 1]])
    800
    801     @staticmethod

~/.local/lib/python3.8/site-packages/matplotlib/transforms.py in __init__(self,
points, **kwargs)
    772         """
    773         BboxBase.__init__(self, **kwargs)
--> 774         points = np.asarray(points, float)
    775         if points.shape != (2, 2):
    776             raise ValueError('Bbox points must be of the form '

~/.local/lib/python3.8/site-packages/numpy/core/_asarray.py in asarray(a, dtype,
order)
     81
     82     """
---> 83     return array(a, dtype, copy=False, order=order)
     84
     85

KeyboardInterrupt:
```

In [ ]:

In [ ]: