Python Queue Interaction

RISHI’s Python training

rishi.arun@yahoo.com

2021

**Types of Queue in Python**

There are mainly four types of queue in Python:

* First in First out Queue: For this, the element that goes first will be the first to come out.

To work with FIFO, you have to call **Queue()**class from queue module.

* Last in First out Queue: Over here, the element that is entered last will be the first to come out.

To work with LIFO, you have to call **LifoQueue()**class from the queue module.

import queue  
*# fifo = queue.Queue(size)*fifo = queue.Queue()  
lifo = queue.LifoQueue()  
simple = queue.SimpleQueue()  
priority = queue.PriorityQueue()  
  
*# Adding a message in queue - put method*message = **"This is a Message for queue"**fifo.put(message)  
lifo.put(message)  
simple.put(message)  
priority.put(message)

*# Checking Size*print(**"FIFO Size:: "**, fifo.qsize())  
print(**"LIFO Size:: "**, lifo.qsize())  
print(**"SIMPLE Size:: "**, simple.qsize())  
print(**"PRIORITY Size:: "**, priority.qsize())

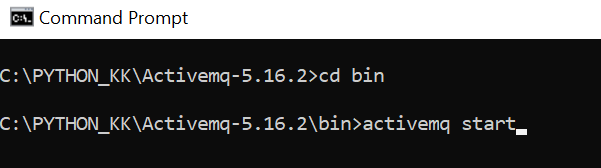
*# getting the message from queue*fifo\_get = fifo.get()  
print(**"FIFO :: "**, fifo\_get)  
lifo\_get = lifo.get()  
print(**"LIFO :: "**, lifo\_get)  
simple\_get = simple.get()  
print(**"SIMPLE :: "**, simple\_get)  
priority\_get = priority.get()  
print(**"PRIORITY :: "**, priority\_get)  
  
print(fifo.qsize())

*# Checking if queue is full*fifo\_size5 = queue.Queue(5)  
fifo\_size5.put(**"message1"**)  
fifo\_size5.put(**"message2"**)  
fifo\_size5.put(**"message3"**)  
fifo\_size5.put(**"message4"**)  
fifo\_size5.put(**"message5"**)  
if fifo\_size5.full():  
 print(**"queue is full"**)  
else:  
 print(**"you can add more messages"**)

[ActiveMQ](https://activemq.apache.org/) is an open source message-oriented middleware which is used to transfer messages between multiple applications. It consists of a queue which will hold in the messages and will only transfer when the receiver is available, hence we can be rest assured that the messages won’t be lost if the server is down. The applications could be entirely different and hosted on heterogenous platforms, ActiveMQ can be used to communicate between such systems as well.

Download ActiveMQ : <https://activemq.apache.org/components/classic/download/>

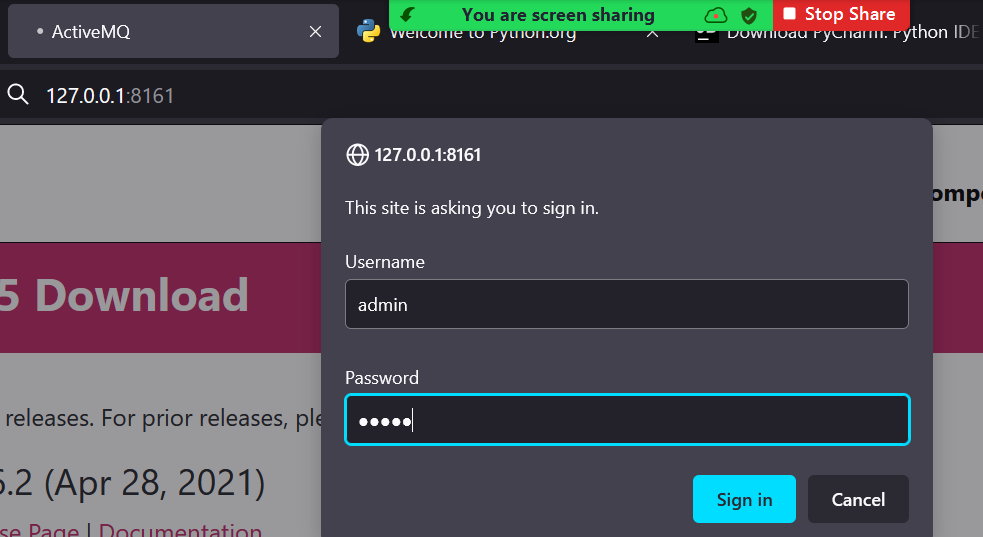
Start ActiveMQ:



Once started click on: <http://127.0.0.1:8161/index.html>

User Name: admin

Password : admin



Stomp will be used to receive and send messages to ActiveMQ.

**Stomp Message Sender:**

import stomp  
  
host = **"localhost"**port = 61613  
destination = **"/topic/event"**messages = 100  
data = **"Hello World from Python"**conn = stomp.Connection(host\_and\_ports = [(host, port)])  
  
conn.connect()  
  
for i in range(0, messages):  
 conn.send(destination=**"test.queue"**, body=data, persistent=**'true'**)  
  
  
conn.disconnect()

**Stomp Message Receiver:**

*'''  
https://github.com/apache/activemq/blob/main/assembly/src/release/examples/stomp/python/stomppy/listener.py  
'''*import stomp  
import time  
  
host = **"localhost"**port = 61613  
destination=**"test.queue"**messages = 100  
  
  
class MyListener:  
  
 def \_\_init\_\_(self, conn):  
 self.conn = conn  
 self.count = 0  
 self.start = time.time()  
  
 def on\_error(self, message):  
 print(**'received an error %s'** % message)  
  
 def on\_message(self, message):  
 if message == **"SHUTDOWN"**:  
 diff = time.time() - self.start  
 print(**"Received %s in %f seconds"** % (self.count, diff))  
 conn.disconnect()  
 else:  
 print(message)  
 print(message.headers)  
 print(message.body)  
  
  
  
conn = stomp.Connection(host\_and\_ports = [(host, port)])  
conn.set\_listener(**'RishizListener'**, MyListener(conn))  
conn.connect()  
conn.subscribe(destination=destination, id=1, ack=**'auto'**)  
print(**"Waiting for messages..."**)  
while 1:  
 time.sleep(10)