How to install Python IBM MQ module PyMQI in Linux and Windows

https://www.ibm.com/support/pages/node/6856737

Date last updated: 23-Jan-2023

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Acknowledgements

- Thanks to Morag Hughson for her assistance regarding the installation of PyMQI to work with the server (local bindings via shared memory) connection. (Note: the default installation and usage is for the MQ Client connections via TCP).

- Thanks to my coworker Bob Gibson for his assistance during some troubleshooting tasks!

+++ Disclaimer of technical support for PyMQI

- The IBM MQ Support team does not provide technical assistance for questions or defects for PyMQI.
- For how-to questions and defects for PyMQI please visit:
https://pypi.org/project/pymqi/

+++ Objective +++

Some IBM MQ customers use Python to write administration scripts and in some instances, customers have used the PyMQI module in Python to interact with an IBM MQ queue manager.
The PyMQI package is a production-ready, open-source Python extension for IBM MQ.
This document provides the steps for ensure that the requisites for PyMQI are properly installed and then how to use the "pip" Python Installer Package to install PyMQI.

For more information on PyMQI see:
https://pypi.org/project/pymqi/
PyMQI
pymqi 1.12.10 (Released: Jan 7, 2023)
PyMQI is a production-ready, open-source Python extension for IBM MQ

The chapters for this document are:
- Chapter 1: Installing PyMQI in Linux
- Chapter 2: Installing PyMQI in Windows
- Chapter 3: Troubleshooting

++ Requirements

+ The MQ Client shared libraries need to be installed in the server.
See articles:
https://www.ibm.com/support/pages/node/97549
Determining if the MQ client fileset package is installed

https://www.ibm.com/support/pages/node/6565783
Using Windows wmic and reg to find the IBM MQ Installations and their IBM MQ Components

+ In Linux, optionally you can install PyMQI to also use the MQ Server shared libraries.
pip3 install pymqi --install-option server
Then you can use specify the following environment variable:
export MQ_CONNECT_TYPE=LOCAL
… and without any changes to the source code, at runtime your program can use the local server (or local bindings or shared memory) connection.

+ In Linux, at runtime, PyMQI requires to access the MQ client shared library from /usr/lib. By default, the installation of MQ does not automatically creates the symbolic links because there is no automatic assignation of a Primary installation, even if you have only a single installation.
Thus, you MUST designate explicitly one of the installations (even if it is the only one) as the Primary installation (for more details see Chapter 1).

+ In Linux, the installation of PyMQI requires the Python Development package (which is not usually installed by default).

+ In Windows, the installation of PyMQI requires the Microsoft Visual Studio BuildTools.

+ In Windows, it is not necessary to designate a Primary installation of MQ, but "setmqenv" needs to be used to setup the MQ environment variables and to set the PATH, which will let PyMQI find the MQ Client DLLs.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++ Chapter 1: Installing PyMQI in Linux
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

For this tutorial the following version of RHEL was used:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# cat /etc/redhat-release
Red Hat Enterprise Linux release 8.6 (Ootpa)
```

++ Requisites:

- MQ Client
- Ensure that there is a Primary installation of MQ
- Python3
- Python Development Package: python3-devel

+ MQ Client:

The following Linux command confirms that the MQ Client package is installed.

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# rpm -q MQSeriesClient
MQSeriesClient-9.2.0-5.x86_64
```

If it is not installed, then you MUST install the MQ Client before proceeding.

+ Ensure to have a Primary installation of MQ

Please keep in mind that even if you have only 1 installation of MQ, by default, it is NOT marked as Primary.

Issue the MQ command:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# /opt/mqm/bin/dspmqinst
InstName:      Installation1
InstDesc:
Identifier:    1
InstPath:      /opt/mqm
Version:       9.3.0.0
Primary:       No
State:         Available
```

If the attribute "Primary" is "No" in the installation stanzas, then you will encounter runtime errors with PyMQI because PyMQI expects to access the MQ Client shared libraries directly from /usr/lib and by default, there are no symbolic links in /usr/lib that point to the actual installed files from MQ.

You need to designate one of the installations as Primary, and this will create symbolic links under /usr/lib for the MQ client shared library.

To mark an installation as Primary:

Login as root and issue:

```
#  /opt/mqm/bin/setmqinst -i -n Installation1
'Installation1' (/opt/mqm) set as the primary installation.
```

This will provide the necessary symbolic link that is needed by PyMQI.

```
# ls -l /usr/lib/libmqic_r.so
lrwxrwxrwx 1 root root 25 Jan 20 05:26 /usr/lib/libmqic_r.so ->
/opt/mqm/lib/libmqic_r.so
```

Notice that dspmqinst will show that the installation is Primary (value "Yes")

```
# /opt/mqm/bin/dspmqinst
InstName:      Installation1
InstDesc:
Identifier:    1
InstPath:      /opt/mqm
Version:       9.3.0.0
Primary:       Yes
```

+ Python 3

Some systems have installed "Python 2" and the commands "python" and "pip" are version 2.
In this host, Python 2 was not installed:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# python
-bash: python: command not found
```

PyMQI needs "Python 3" instead of "Python 2".
The commands for Python 3 are:
python3
pip3

You can issue the following command to find out the version.
(Thanks to Bobbee Broderick for mentioning this option)

```
mqm@suvereto1.fyre.ibm.com: /home/mqm
$ python3 -V
Python 3.6.8
```

Or you can issue the following command without any additional arguments, to get into interaction mode. This shows the version.
Notice that to exit you must type: `quit()` or `exit()`

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# python3
Python 3.6.8 (default, Jan 14 2022, 11:04:20)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-7)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

+ Python Development Package: python3-devel

Need to install the Python Development package "python3-devel", because the installation of PyMQI requires a recompilation.

Note: If the development package is not installed, then the installation of PyMQI will fail with error:

```
    code/pymqi/pymqe.c:110:10: fatal error: Python.h: No such file or directory
     #include "Python.h"
              ^~~~~~~~~~
    compilation terminated.
    error: command 'gcc' failed with exit status 1
```

To install the Python Development package, login as root and issue:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# yum install python3-devel
Updating Subscription Management repositories.
Red Hat Enterprise Linux 8 for x86_64 - AppStre  34 kB/s | 2.8 kB     00:00
Red Hat Enterprise Linux 8 for x86_64 - BaseOS   28 kB/s | 2.4 kB     00:00
Dependencies resolved.
================================================================================
 Package                Arch   Version   Repository                        Size
================================================================================
Installing:
 python36-devel         x86_64 3.6.8-38.module+el8.5.0+12207+5c5719bc
                                          rhel-8-for-x86_64-appstream-rpms  17 k
Installing dependencies:
 platform-python-devel  x86_64 3.6.8-45.el8
                                          rhel-8-for-x86_64-appstream-rpms 250 k
 python3-rpm-generators noarch 5-7.el8   rhel-8-for-x86_64-appstream-rpms  25 k
…
Installed:
  platform-python-devel-3.6.8-45.el8.x86_64
  python3-rpm-generators-5-7.el8.noarch
  python36-devel-3.6.8-38.module+el8.5.0+12207+5c5719bc.x86_64
Complete!
```

+ Use pip3 to install PyMQI

a) Using ONLY the MQ Client shared libraries

Login as root and issue:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# pip3 install pymqi
WARNING: Running pip install with root privileges is generally not a good idea. Try
`pip3 install --user` instead.
Collecting pymqi
  Downloading
https://files.pythonhosted.org/packages/20/dc/a1975fc640d1dabdabe7877f28eebec2bd4d5
13227322eb35c3753ee2d0b/pymqi-1.12.10.tar.gz (91kB)
    100% |âââââââââââââââââââââââââââââââââââ| 92kB 2.5MB/s
Installing collected packages: pymqi
  Running setup.py install for pymqi ... done
Successfully installed pymqi-1.12.10
```

b) Using both the MQ Client and MQ Server shared libraries

Login as root and issue:

```
# pip3 install pymqi --install-option server
/usr/lib/python3.6/site-packages/pip/commands/install.py:204: UserWarning:
Disabling all use of wheels due to the use of --build-options / --global-options /
--install-options.
  cmdoptions.check_install_build_global(options)
WARNING: Running pip install with root privileges is generally not a good idea. Try
`pip3 install --user` instead.
Collecting pymqi
  Using cached
https://files.pythonhosted.org/packages/20/dc/a1975fc640d1dabdabe7877f28eebec2bd4d5
13227322eb35c3753ee2d0b/pymqi-1.12.10.tar.gz
Installing collected packages: pymqi
  Running setup.py install for pymqi ... done
Successfully installed pymqi-1.12.10
```

+ How to confirm that PyMQI was successfully installed (MQ Client shared libraries)

We can use an extremely simple sample to do a very basic test.
Ensure to modify the "pymqi.connect" line in the sample to reflect your queue manager.

1: Create file: putmsg.py

```
import pymqi
print("Starting putmsg")
queue_manager = pymqi.connect('QM93', 'SYSTEM.DEF.SVRCONN', 'localhost(1414)')
q = pymqi.Queue(queue_manager, 'Q1')
q.put('Hello from Python!')
print("Putting 1 message into queue")
print("Ending putmsg")
```

2: Run in Linux:

```
$ python3 putmsg.py
Starting putmsg
Putting 1 message into queue
Ending putmsg
```

Yeah!!!

+ (Optional) How to confirm that PyMQI was successfully installed (MQ Server shared libraries)

If you have installed PyMQI to also allow the use of the MQ Server shared libraries, then you can use the small sample "putmsg.py" shown above.

The only difference is that you need to set an MQ environment variable (MQ_CONNECT_TYPE) that will allow for the use of the MQ Server connection shared libraries (local bindings):

```
$ export MQ_CONNECT_TYPE=LOCAL

$ python3 putmsg.py
Starting putmsg
Putting 1 message into queue
Ending putmsg
```

+ Additional notes about using the MQ Server libraries

In case that you doubt that the MQ Server libraries are being used, then you could stop the MQ listener, confirm that the runmqlsr process is not running (which will prevent remote network connections) and run again the sample program.

Use runmqsc to stop the MQ listener:
stop LISTENER(SYSTEM.LISTENER.TCP.1)
2 : stop LISTENER(SYSTEM.LISTENER.TCP.1)
AMQ8706I: Request to stop IBM MQ Listener accepted.

Confirm that runmqlsr is no longer running:
$ ps -ef | grep runmqlsr
(none)

```
$ export MQ_CONNECT_TYPE=LOCAL

$ python3 putmsg.py
Starting putmsg
Putting 1 message into queue
Ending putmsg
```

The trace of the MQ client application will indicate that server library will be used:

```
...
 10:45:32.968342    8208.1         :          xcsGetEnvironmentString[MQ_CONNECT_TYPE] = 'LOCAL'
 10:45:32.968346    8208.1         :      ---} xcsGetEnvironmentString rc=OK FunctionTime=7
 10:45:32.968349    8208.1         :          MQ_CONNECT_TYPE says only attempt server connection
 10:45:32.968354    8208.1         :          ConnectType: 3, ServerOnly: TRUE, ClientOnly: FALSE, Fastpath:
FALSE, Shared: FALSE
 10:45:32.968357    8208.1         :      --} zswChooseLibraries rc=OK FunctionTime=23
 10:45:32.968361    8208.1         :          Attempting to connect using the server library
 10:45:32.968365    8208.1         :          Trying optimized connect using internal interface 0x7f52c9c928a0
```

++
+++ Chapter 2: Installing PyMQI in Windows
++

++ Requisites:

MQ Client
Python 3
Microsoft Visual Studio BuildTools

++ Check if the MQ Client is already installed

You can use the following Windows command.
Notice that it is a very long, compound command. Needs to be done in a single line.

C:\> **reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\WebSphere MQ\Installation" /s | findstr "Client"**
    Local Clients\Windows NT Client   REG_SZ   Installed

For more details see:
https://www.ibm.com/support/pages/node/6565783
Using Windows wmic and reg to find the IBM MQ Installations and their IBM MQ Components

++ Install MQ Client

If the MQ Client is not installed, then you can download the MQ 9.3 Client for Windows from

The MQ Client code that is downloaded from IBM Fix Central:
- It does not require a license.
- It is a "manufacturing refresh". For example, MQ 9.2.0.6 includes base MQ 9.2.0.0 + 9.2.0.1 + ... + 9.2.0.6

https://www.ibm.com/docs/en/ibm-mq/9.3?topic=roadmap-mq-downloads
IBM MQ / 9.3 / IBM MQ downloads
Any MQ version, all downloads
.
Scroll to section:
  Resource adapter, clients and other resources

See the subsection:
  Clients:

Click on:
    IBM MQ C and .NET clients
        https://ibm.biz/mq93clients

You will be taken to a selected list of items from IBM Fix Central.
Some example items are:
.
Item 3: release level: 9.3.0.0-IBM-MQC-Win64
Long Term Support: 9.3.0 Client install image for IBM MQ on Windows x64

Item 10: release level: 9.3.0.0-IBM-MQC-LinuxX64
Long Term Support: 9.3.0 Client rpm install packages for IBM MQ on Linux x86-64

Note that in the name of the installation file to be downloaded, the "C" in "MQC" means "Client" and not "C-language".
That is, the MQ Client includes the interfaces for C-language and Java/JMS.

And follow the installation instructions:

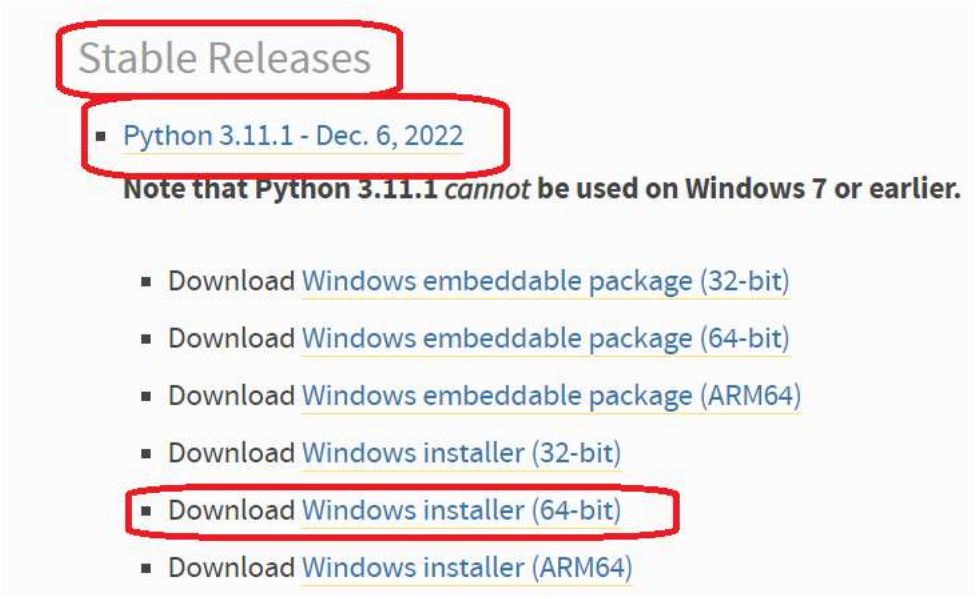
https://www.ibm.com/support/pages/node/598547
Installing MQ 9.0 in Windows, using only the defaults

++ Install Python 3

See the following tutorial:
https://www.digitalocean.com/community/tutorials/install-python-windows-10
// Tutorial //
How to Install Python on Windows 10
Published on August 3, 2022
.
The installation procedure involves these steps:

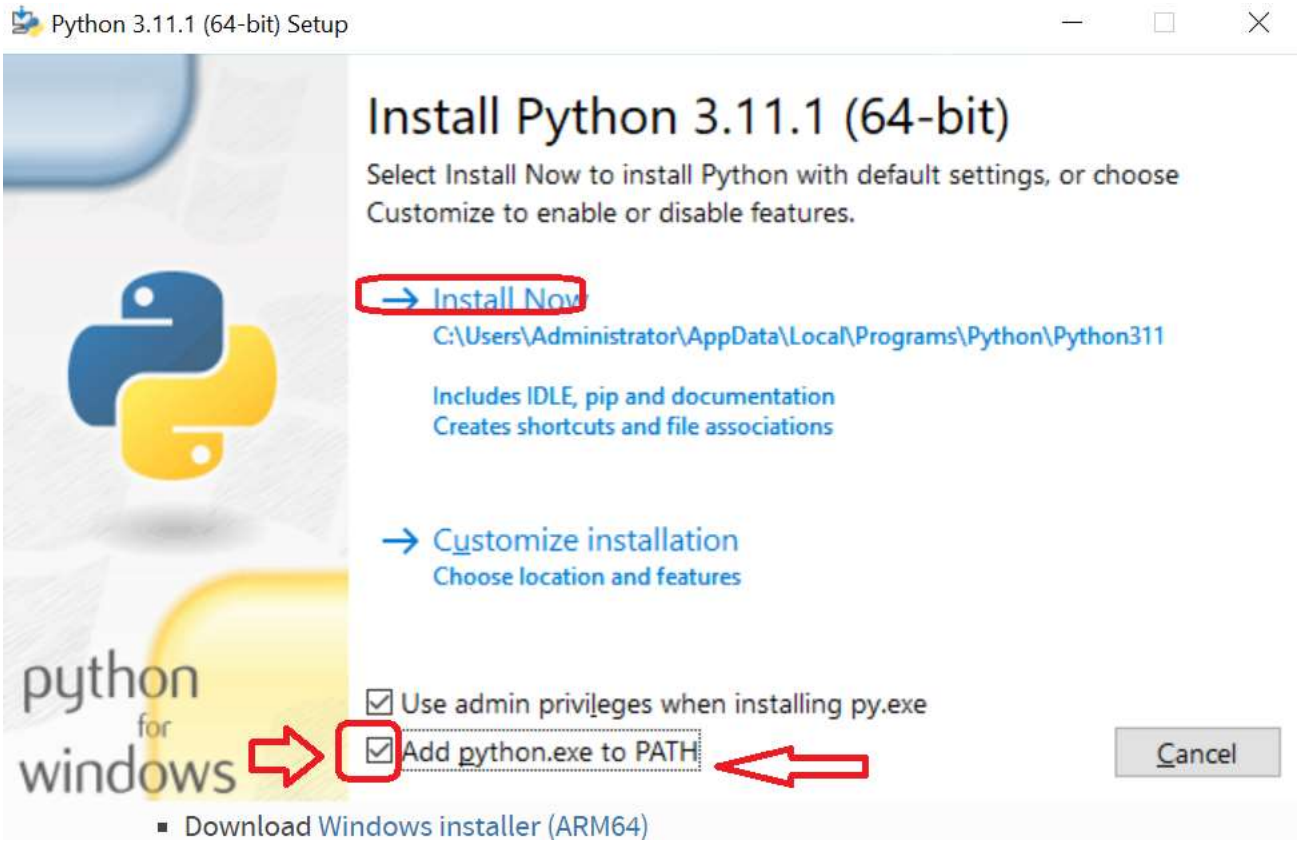
1: Download the installation file for Python
https://www.python.org/downloads/windows/

Stable Releases
Python 3.11.1 - Dec. 6, 2022
Download Windows installer (64-bit)



For this tutorial, the file:
  python-3.11.1-amd64.exe
… was downloaded into the directory:
  C:\downloads\python
2: Run the Executable installer.
You will see the following dialog.

Note: It is HIGHLY recommended that you check mark:
(*) Add python.exe to PATH



3: Add Python to PATH environmental variables
(Done during the installation by doing a Check Mark on the option to add to PATH)

4: Verify the Python Installation

You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation.

Open a new Windows command prompt and type:
C:\> **python**
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> **exit()**

++ (Optional) Download and install PyCharm IDE Community Edition (from JetBrains)

The following is a very good Integrated Development Environment

https://www.jetbrains.com/pycharm/download/#section=windows

Installed in:
"C:\Program Files\JetBrains\PyCharm Community Edition 2022.3.1\"

Main executable:
"C:\Program Files\JetBrains\PyCharm Community Edition 2022.3.1\bin\pycharm64.exe"

Location of projects:
C:\Users\USERID\PycharmProjects\

++ Install Microsoft Visual Studio C++ Build Tools

+ Why is this component needed?

Because if you do not have it, then when you try to install PyMQI, a compilation will be required and you will get the following error:

<begin excerpt>

Error:
...
  building 'pymqi.pymqe' extension
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
  [end of output]

  note: This error originates from a subprocess, and is likely not a problem with pip.
  ERROR: Failed building wheel for pymqi
ERROR: Could not build wheels for pymqi, which is required to install pyproject.toml-based projects

<end excerpt>

+ For installation instructions, see the tutorial:

https://www.ibm.com/support/pages/node/358087
How to compile IBM MQ samples using Microsoft Visual Studio Build Tools

After the installation of the Build Tools you need to create a batch file inside a directory that is in the PATH, to run "vcvars64.bat" which is the batch file that is supplied by VC++ to setup the proper variables for compiling and linking.

Example:
You can add a batch file in the directory that is in your PATH. In this case, it is:
  c:\wintools
File:
  c:\wintools\set-visual-studio-env.bat

The contents of this file is:

```
REM Set environment variables for Microsoft Visual Studio 2022, BuildTools
"C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\Auxiliary\Build\vcvars64.bat"
```

++ Install PyMQI

As an Administrator, open a Windows Command prompt and issue the batch command file to setup the Microsoft Visual Studio Build Tools

C:\> **set-visual-studio-env.bat**

C:\>REM Set environment variables for Microsoft Visual Studio 2022, BuildTools

C:\>"C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\Auxiliary\Build\vcvars64.bat"
****
** Visual Studio 2022 Developer Command Prompt v17.4.4
** Copyright (c) 2022 Microsoft Corporation
****
[vcvarsall.bat] Environment initialized for: 'x64'

Now proceed to use "pip":

C:\> **pip install pymqi**
Collecting pymqi
  Downloading pymqi-1.12.10.tar.gz (91 kB)
     -- 91.7/91.7 kB 2.6 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: pymqi
  Building wheel for pymqi (pyproject.toml) ... done
  Created wheel for pymqi: filename=pymqi-1.12.10-cp311-cp311-win_amd64.whl size=91920 sha256=81badf2c9da458ff0aaea46edb27598d117d16466abebf43c7ba3a3f9593ff5c
  Stored in directory: c:\users\administrator\appdata\local\pip\cache\wheels\fd\30\6d\c24d4392310238ce18b5fb7156f1650e13d5ef490d2155d987
Successfully built pymqi
Installing collected packages: pymqi
Successfully installed pymqi-1.12.10

+ How to confirm that PyMQI was successfully installed

We can use an extremely simple sample to do a very basic test.
Ensure to modify the "pymqi.connect" line in the sample to reflect your queue manager.

1: Create file: putmsg.py

```
import pymqi
print("Starting putmsg")
queue_manager = pymqi.connect('QM93', 'SYSTEM.DEF.SVRCONN',
'localhost(1414)')
q = pymqi.Queue(queue_manager, 'Q1')
q.put('Hello from Python!')
print("Putting 1 message into queue")
print("Ending putmsg")
```

2: Run in Windows:

```
C:\> python putmsg.py
Starting putmsg
Putting 1 message into queue
Ending putmsg
```

Yeah!!

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++ Chapter 3: Troubleshooting
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

++ Linux:

Scenario:
You issue the following command:
pip3 install pymqi
… and if fails:

```
    code/pymqi/pymqe.c:110:10: fatal error: Python.h: No such file or directory
     #include "Python.h"
              ^~~~~~~~~~
    compilation terminated.
    error: command 'gcc' failed with exit status 1
```

Answer:
The problem is that you must install the Python Development package.

Login as root and issue:

```
+++ROOT+++ suvereto1.fyre.ibm.com: /root
# yum install python3-devel
```

++ Linux:

Scenario:
At runtime you encounter an error about not finding the MQ client shared library:

```
mqm@suvereto1.fyre.ibm.com: /home/mqm
$ python3 putmsg.py
Traceback (most recent call last):
  File "/usr/local/lib64/python3.6/site-packages/pymqi/__init__.py", line
132, in <module>
    from . import pymqe # type: ignore
ImportError: libmqic_r.so: cannot open shared object file: No such file or
directory

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "putmsg.py", line 1, in <module>
    import pymqi
  File "/usr/local/lib64/python3.6/site-packages/pymqi/__init__.py", line
134, in <module>
```

```
    import pymqe # type: ignore # Backward compatibility
ModuleNotFoundError: No module named 'pymqe'
```

Answer:
PyMQI is expecting the MQ Client shared library to be in /usr/lib.
But the installation of MQ, by default, does NOT add any symbolic links under /usr/lib.
As user root you need to designate a Primary MQ installation, which will create the symbolic links in /usr/lib to point to the real shared libraries for the MQ Client.

```
#  /opt/mqm/bin/setmqinst -i -n Installation1
'Installation1' (/opt/mqm) set as the primary installation.
```

This will provide the necessary symbolic link that is needed by PyMQI.

```
# ls -l /usr/lib/libmqic_r.so
lrwxrwxrwx 1 root root 25 Jan 20 05:26 /usr/lib/libmqic_r.so ->
/opt/mqm/lib/libmqic_r.so
```

+ Windows

Scenario:
PyMQI can work fine with PRIMARY installation defined,
or without one being defined.
.
If there is no Primary installation, then ensure to setup the MQ environment:
  setmqenv
.
Windows:
If no Primary is define and if no "setmqenv" is defined, then

C:\> cd \angel\coding\mq\python

C:\angel\coding\MQ\python> set MQ

C:\angel\coding\MQ\python>dspmqver
'dspmqver' is not recognized as an internal or external command,
operable program or batch file.

C:\angel\coding\MQ\python>python putmsg.py
Traceback (most recent call last):
  File "C:\Users\594079897\AppData\Local\Programs\Python\Python311\Lib\site-packages\pymqi__init__.py", line 132, in <module>
    from . import pymqe # type: ignore
    ^^^^^^^^^^^^^^^^^^^
ImportError: DLL load failed while importing pymqe: The specified module could not be found.

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\angel\coding\MQ\python\putmsg.py", line 1, in <module>
    import pymqi
  File "C:\Users\594079897\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pymqi\__init__.py", line 134, in <module>
    import pymqe # type: ignore # Backward compatibility
    ^^^^^^^^^^^^
ModuleNotFoundError: No module named 'pymqe'

"C:\Program Files\IBM\MQ\bin\setmqenv"  -n Installation1
```

+ Windows

Scenario:
What happens when queue manager is not running

```
C:\angel\coding\MQ\python>python putmsg.py
Starting putmsg
Traceback (most recent call last):
  File "C:\angel\coding\MQ\python\putmsg.py", line 3, in <module>
    queue_manager = pymqi.connect('QM931', 'SYSTEM.DEF.SVRCONN', 'localhost(1414)')
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\594079897\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pymqi\__init__.py", line 3214, in connect
    qmgr.connect_tcp_client(queue_manager or '', CD(), channel, conn_info, user, password)
  File "C:\Users\594079897\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pymqi\__init__.py", line 1772, in connect_tcp_client
    self.connect_with_options(name, **kwargs)
  File "C:\Users\594079897\AppData\Local\Programs\Python\Python311\Lib\site-
packages\pymqi\__init__.py", line 1747, in connect_with_options
    raise MQMIError(rv[1], rv[2])
pymqi.MQMIError: MQI Error. Comp: 2, Reason 2538: FAILED: MQRC_HOST_NOT_AVAILABLE
```

+++ end +++