# PyQt5 [Python 2.7] & Binary Ninja

## Quick Overview

Okay, so this is going to be a pretty straightforward guide. The first sections are going to describe how to setup your system so you can actually use PyQt5 without python losing its mind. It probably doesn't need to be noted, but I'll mention it anyways – This is for Python 2.7. Binary Ninja has Python 2.7 bindings, but most of the PyQt5 packages that are available are for Python 3. Yay.

To test to see if you have access to PyQt5 from within Binary Ninja, run the following in the embedded Python prompt. If it errors out, you don't have access to PyQt5 ☹.

```
> import PyQt5
```

If you want to see some examples of how to interact with Binary Ninja from PyQt5, check out my git repos:

- http://github.com/nbsdx/binja-ui-api
- http://github.com/nbsdx/binja-ui-tweaks

UI hacking is not supported by the Vector 35 team, so it's unlikely that they'll offer too much help – they're working on their own API for extending the UI, I started this work just for fun until the official API is out ☺. If you run into issues, I can try to help out as much as I can, ping me in the binaryninja slack: @neil.

## Platforms

Binary Ninja supports Windows, OSX, and Ubuntu, so I'll talk to each platform here. On Ubuntu and OSX, Binary Ninja uses the Python 2.7 install that is already installed on your computer. On Windows, it comes bundled with its own version of Python 2.7, so some of the directions will differ for Windows.

On Linux, Binary Ninja reports to be using Qt 5.5.1, but docs.binary.ninja reports Qt 5.6 as the version being used. I went with that version for this guide, but compiling for older (or new) versions should have the same steps.

When building PyQt5, you can configure which modules you want to build, and which to skip. These instructions build all modules.

## UBUNTU

This one is easy if you have a version of Ubuntu newer than 14.04. Use the following command to install PyQt5 directly from the command line:

```
$ sudo apt-get install python-pyqt5
```

And you're done! If you DON'T have newer than 14.04, well, sorry. Try following the build instructions below. I haven't tried it, so have fun! It should be pretty much identical, just a different download path for Qt, and different file system paths.

## Windows

### PreReqs

You'll need a couple things for this to work on Windows.

1. Visual Studio 2013
2. Python 2.7
3. Qt5.6.0
    a. https://download.qt.io/official_releases/qt/5.6/5.6.0/qt-opensource-windows-x86-msvc2013_64-5.6.0.exe
4. PyQt5 Sources
    a. https://sourceforge.net/projects/pyqt/files/PyQt5/PyQt-5.6/PyQt5_gpl-5.6.zip/download
5. Sip
    a. https://sourceforge.net/projects/pyqt/files/sip/sip-4.18.1/sip-4.18.1.zip/download

### Building

Did you know that Windows has a command line interface? Yeah, me neither. Break out cmd.exe and navigate to wherever you unpacked the sip and pyqt5 sources (tip: shift+right click in the directory, "Open Command Window Here"). The steps here are pretty straightforward (I'm assuming the sip and pyqt5 folders are in the same directory).

```
> "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64\vcvars64.bat"
> cd sip-4.18.1
> python configure.py
> nmake
> nmake install
```

This installs SIP, which is needed to generate the bindings for Qt in Python. SIP is installed next to wherever your python.exe file lives (generally C:\Python27\sip.exe).

Now we have to build PyQt5. This requires some setup. Once you've installed Qt 5.x, you need to add the Qt5 lib and bin directories to your path so the configure script can find them. On my computer, Qt puts them in the following locations:

- Bin: C:\Qt\Qt5.6.1\5.6\msvc2013_64\bin
- Lib: C:\Qt\Qt5.6.1\5.6\msvc2013_64\lib

Note, that I have Qt 5.6.1 installed. I downloaded the wrong one and didn't want to fix it (but I think 5.6.0 is required for OSX), and it works, so I'm keeping it. Add those directories to your Windows PATH environment variable, and you should be good to go.

```
> "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64\vcvars64.bat"
> cd PyQt5_gpl-5.6
> python configure.py -q "C:\Qt\Qt5.6.1\5.6\msvc2013_64\bin\qmake" ^
                      -d "C:\Python27\Libs\site-packages" ^
                      --sip "C:\Python27\sip.exe"
> set CL=/MP # Remove this line if you DON'T want multicore build support
> nmake
> nmake install
```

You can probably change the -d parameter to the location of Binary Ninja's site-packages folder, or even to the Binary Ninja plugins directory to ensure that it survives updates (currently, updates to the Windows version of Binary Ninja destroys any modifications you make to site-packages.

***NOTE*** If building PyQt5 fails to complete (I had issues with certain modules failing to generate bindings), then you can disable them through the configure script (I think), or by commenting out the lines referencing that module in the configure.py script.

Now, navigate to wherever your -d parameter points to, "C:\Python27\Libs\site-packages" in my case. Copy the following directories/files into "C:\Program Files\Vector35\BinaryNinja\plugins\Lib\site-packages":

1. [dir] PyQt5
2. [file] sip.pyd
3. [file] sip.pyi (maybe optional)

You should now be able to access PyQt5 from Binary Ninja's python console, and plugins.

## OS X

I was hoping that OS X would be as easy as “brew install pyqt5 --withoutpython3 --withpython", but it wasn’t. Damn. Building for OS X is more or less the same as Windows.

### PREREQS

1. Clang/XCode
2. Python 2.7
3. Qt5.6.0
    a. https://download.qt.io/official_releases/qt/5.6/5.6.0/qt-opensource-mac-x64-clang-5.6.0.dmg
4. PyQt5 Sources
    a. https://sourceforge.net/projects/pyqt/files/PyQt5/PyQt-5.6/PyQt5_gpl-5.6.zip/download
5. Sip
    a. https://sourceforge.net/projects/pyqt/files/sip/sip-4.18.1/sip-4.18.1.tar.gz/download

### BUILDING

Building follows the same process as in Windows:

```
$ cd sip-4.18.1
$ python configure.py
$ make
$ make install
```

```
$ cd PyQt5_gpl-5.6
$ python configure.py \
         -q /Users/<name>/Qt5.6.0/5.6/clang_64/bin/qmake \
         -d /Library/Python/2.7/site-packages \
         --sip /System/Library/Frameworks/Python.framework/Versions/2.7/bin/sip
$ make # -j<num_cores> if you want to parallelize
$ make install
```

### USING PYQT5 IN OS X

This is where OS X finally decides to be a pain in the ass. It’s not smart enough to know that we want to use the frameworks that have already been loaded by the application for Qt, and since its sees the frameworks elsewhere, OS X loses its mind and just fails to do anything. The solution to this is pretty awful, but there are far worse things we could be doing ☺

We’re going to symlink the frameworks that PyQt5 expects to see so there are only one instance of the Framework in the applications library path.

```
$ cd /Applications/Binary\ Ninja.app/Contents/Frameworks
$ for x in *; do
     sudo mv $x $x.orig;
     sudo ln -s /Users/<name>/Qt5.6.0/5.6/clang_64/lib/$x $x;
  done
```

Basically, this shell script will rename the original Qt libraries that ship with Binary Ninja, and replace them with links to the ones we downloaded so we could build PyQt5.