PyQt5 Cheatsheet

Getting Started

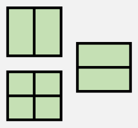
*from PyQt5.QtWidgets import **
*from PyQt5.QtCore import **
*from PyQt5.QtGui import **

QtCore - core classes, signal and slot mechanism, animations, applications settings
QtGui - Graphical user interface components
QtWidgets - Classes for creating classic desktop-style Uis

```
#A simple structure example
class Window(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Example')
if __name__ == "__main__":
    import sys
    app = QApplication(sys.argv)
    ui = Window()
    ui.show()
    sys.exit(app.exec_())
```

Layouts

```
layout = QVBoxLayout(parent)
#add a widget to current layout?
layout.addWidget(widget, alignment)
#add a child layout to current layout?
layout.addLayout(QLayout layout, alignment)
#setting the layout for the parent
parent.setLayout(layout)
```

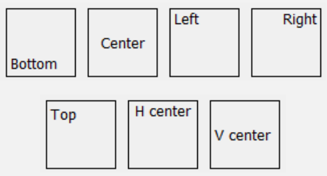


QVBoxLayout(parent)
QHBoxLayout(parent)
QGridLayout(parent)
obs: addWidget(*widget*, int row, int column, int height, int row)

Alignments

my_widget = setAlignment(Qt.AlignTop)

Qt.AlignRight
Qt.AlignCenter
Qt.AlignLeft
Qt.AlignBottom
Qt.AlignTop



Size Policy

*SizePolicy = **QSizePolicy**(QSizePolicy.Policy horizontal, QSizePolicy.Policy vertical)*
*mywidget.**setSizePolicy**(sizePolicy)*

Type of policies
QSizePolicy.Fixed Never grow or shrink
QSizePolicy.Minimum Cannot be smaller than the size provided by sizeHint().
QSizePolicy.Maximum Cannot be larger than the size provided by sizeHint().
QSizePolicy.Expanding Uses the whole available size
QSizePolicy.MinimumExpanding Expands from a minimum size

Stylesheets

widget.setStyleSheet(stylesheet)

```
stylesheet= """Qwidget{
        background-color: rgb(50,65,75);
        font: 9pt \"Segoe UI\";
        color: rgb(232, 232, 232);
        border: 2px black;
        border-radius: 10px;}"""
```

Threads

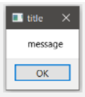
```
class Window(...):
        ...
        def function(self):
            #create a thread
            self.thread = QThread()
            #create a worker
            self.worker = Worker()
            #move worker to thread
            self.worker.moveToThread(self.thread)
            #run method
            self.thread.started.connect(self.worker.method1)
            self.worker.finished.connect(self.thread.quit)
            self.worker.finished.connect(self.worker.deleteLater)
            self.thread.finished.connect(self.thread.deleteLater)
            #link a signal to a new function
            self.worker.data.connect(lambda x: print(x))
            #Start the thread
            self.thread.start()

class Worker(QObject):
   finished = pyqtSignal() #signals to communicate with main
   data = pyqtSignal(list) #should be class attributes

   def method1(self):
      QThread.sleep(3)
      data=[1,0]
      self.data.emit(data)
      self.finished.emit()
```

Info Dialogs

QMessageBox.about(parent, "title", "message")



Spacers

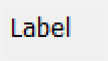
QSpacerItem(int w, int h, QSizePolicy.Policy hPolicy = QSizePolicy.Minimum, QSizePolicy.Policy vPolicy = QSizePolicy.Minimum)

Separation Lines (Horizontal)

```
line =QFrame(parent)
line.setStyleSheet("background-color: rgb(R,G,B);")
Line.setFrameShape(QtWidgets.QFrame.HLine)
```

Labels

```
label = QLabel(parent)
label.setText("Label")
#Picture?
pixmap = QPixmap('logo.png')
pixmap = pixmap.scaledToWidth(300) #scale image
label.setPixmap(pixmap)
```



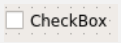
Buttons

```
Button = QPushButton("my button", parent)
#optional – set a name for object, size policy
Button.setObjectName("name")
Button.setSizePolicy(my_policy)
#optiona – icon?
button.setIcon(QIcon('icon.jpg'))
button.setIconSize(QSize(300, 300))
#optional - checkable,enabled?
Button.setCheckable(bool)
Button.setEnabled(bool)
#link click to function
Button.clicked.connect(function/method)
```

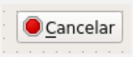
QPushButton(QIcon icon, QString text, QWidget parent)

QToolButton(QIcon icon, QWidget parent)

QRadioButton(QWidget *parent)



QCheckBox(QWidget *parent)

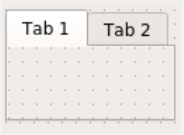


QDialogButtonBox(QDialogButtonBox.StandardButtons buttons, QWidget *parent)
Eg. QDialogButtonBox.Cancel

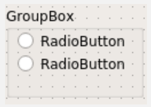
Containers

```
frame = Qframe(parent)
#add a button
Button = QPushButton("my button", frame)
#optional – shadow
shadow = QGraphicsDropShadowEffect(xOffset=3,
yOffset=3,blurRadius=5,color=Qt.gray)
frame.setGraphicsEffect(shadow)
```

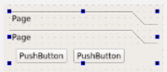
```
Container = QTabWidget(parent)
#create and add a widget
widget=QtWidget(Container)
...define widget, set layout, etc...
Container.addTab(widget, "Tab 1"/Icon)
```



```
Container = QGroupBox(parent)
layout = QGridLayout(Container)
layout.addWidget(widget, 0,0)
Container.setLayout(layout)
```



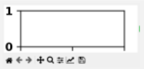
```
Container = QToolBox(parent)
Container.addItem(widget, title)
```



Matplotlib

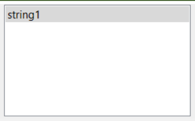
```
import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib.backends.backend_qt5agg import
          FigureCanvasQTAgg, NavigationToolbar2QT
from matplotlib.figure import Figure
class MplCanvas(FigureCanvasQTAgg):
          def __init__(self, parent=None):
                    self.fig = Figure(figsize=(5, 5), dpi=300)
                    self.axes = self.fig.add_subplot(111)
                    super(MplCanvas, self).__init__(self.fig)
                    self.toolbar = NavigationToolbar2QT(self, parent)
                    self.fig.tight_layout()

widget=MplCanvas(self)
toolbar=widget.toolbar
```

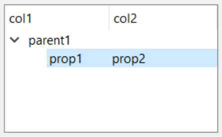


Item List

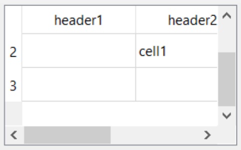
```
ls = QListWidget(parent)
ls.addItem("string 1")
#connect with a function (e.g. print)
ls.itemClicked.connect(lambda x: print(x.text()))
```



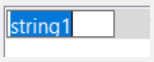
```
view = QTreeWidget(parent)
view.headerItem().setText(0, "col1")
view.headerItem().setText(1, "col2")
parent1 = QTreeWidgetItem(view)
parent1.setText(0, 'parent1')
child1 = QTreeWidgetItem(parent1)
child1.setText(0,"prop1")
child1.setText(1,"prop2")
#eg access in 1st column
view.currentItem().text(0)
```



```
view = QTableWidget(parent)
view.setRowCount(3)
view.setColumnCount(2)
view.setHorizontalHeaderLabels(
          ['header1','header2'])
cell1 = QTableWidgetItem('cell1')
cell2 = QTableWidgetItem('cell2')
view.setItem(1,1,cell1)
```

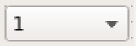


```
#making items editable
item1.setFlags(Qt.ItemIsEnabled | Qt.Item
Qt.ItemIsEditable)
```



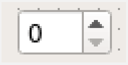
Input Widgets

```
CB = QComboBox(parent)
CB.addItem("1")
```



```
qline = QLineEdit(parent)
qline.setText(string)
```

```
SB = QSpinBox(parent)
SB.setMinimum(min)
SB.setMaximum(max)
```



```
dial = QDial(parent)
dial.setNotchesVisible(True)
dial.setMinimum(min)
dial.setMaximum(max)
```

```
S = QSlider(Qt.Horizontal, self)
S.setMinimum(0)
S.setMaximum(100)
```

Progress Bar

```
Bar = QProgressBar()
Bar.setMaximum(max)
Bar.setMinimum(min)
Bar.setValue(value)
```