```
 1: #
 2: # File: context.py
 3: # Description: Middleman between python and QML. Passes data from python to
 4: #               QML by converting python objects to QVariant types.
 5: # Project: Carberry Pi
 6: # Author: Ryan McHugh
 7: # Year: 2020
 8: #
 9: from PyQt5.QtCore import QObject, QThread
10: from PyQt5 import QtCore
11: import time
12: import sys
13:
14: class Main_Context(QObject):
15:     # Refactor with Dictionary (State) - Done*
16:     # rpmValueChanged = QtCore.pyqtSignal(int)
17:     # speedValueChanged = QtCore.pyqtSignal(int)
18:     # tempValueChanged = QtCore.pyqtSignal(int)
19:     handlerChanged = QtCore.pyqtSignal(QtCore.QVariant)
20:     diagnosticsChanged = QtCore.pyqtSignal(QtCore.QVariant)
21:     timeChanged = QtCore.pyqtSignal(str)
22:     configChanged = QtCore.pyqtSignal(QtCore.QVariant)
23:
24:     counter = 0
25:
26:
27:     def __init__(self, parent=None):
28:         super(Main_Context, self).__init__(parent)
29:         # self.m_rpmValue = 1
30:         # self.m_speedValue = 0
31:         # self.m_tempValue = 0
32:         self.m_handler = {}
33:         self.m_diagnostics = {}
34:         self.m_time = QtCore.QDateTime.currentDateTime().toString("h:mm ap")
35:         self.m_config = {}
36:
37:
38:
39:
40:
41:     # Handler for main dashboard
42:     @QtCore.pyqtProperty(QtCore.QVariant, notify=handlerChanged)
43:     def handler(self):
44:         return QtCore.QVariant(self.m_handler);
45:
46:     @handler.setter
47:     def handler(self, val):
48:         self.m_handler.update(val)
49:         self.handlerChanged.emit(self.m_handler);
50:
51:     def getHandler(self):
52:         return self.m_handler
53:
54:     # Current time in seconds
55:
56:     @QtCore.pyqtProperty(str, notify=timeChanged)
57:     def time(self):
58:         return self.m_time
59:
60:     @time.setter
61:     def time(self, val):
62:         if self.m_time == val:
63:             return
64:         self.m_time = val;
65:         self.timeChanged.emit(self.m_time);
66:
67:     def updateTime(self):
68:
```

```
 69:         # self.time =
 70:         # QtCore.QLocale.setDefault(QtCore.QLocale("en_DE"))
 71:         self.time = QtCore.QDateTime.currentDateTime().toString("h:mm ap")
 72:
 73:     # Configuration values
 74:
 75:     @QtCore.pyqtProperty(QtCore.QVariant, notify=configChanged)
 76:     def config(self):
 77:         return QtCore.QVariant(self.m_config)
 78:
 79:     def getConfig(self):
 80:         return self.m_config
 81:
 82:     @config.setter
 83:     def config(self, val):
 84:         self.m_config.update(val)
 85:         self.configChanged.emit(self.m_config)
 86:
 87:     @QtCore.pyqtSlot(QtCore.QVariant, QtCore.QVariant)
 88:     def updateConfigFromQML(self, key, value):
 89:
 90:
 91:         if value.lower() == "true":
 92:             value = True
 93:         elif value.lower() == "false":
 94:             value = False
 95:
 96:         updatedDict= self.getConfig()[key]
 97:         updatedDict['current'] = value
 98:         # print({key : updatedDict})
 99:         self.config = {key : updatedDict}
100:         # print(self.getConfig())
101:
102:
103:     def close(val):
104:         sys.exit(val)
105:
106:
107:     @QtCore.pyqtProperty(QtCore.QVariant, notify=diagnosticsChanged)
108:     def diagnostics(self):
109:         return QtCore.QVariant(self.m_diagnostics)
110:
111:     @diagnostics.setter
112:     def diagnostics(self, val):
113:         self.m_diagnostics.update(val)
114:         self.diagnosticsChanged.emit(self.m_diagnostics)
115:
116:     def getDiagnostics(self):
117:         return self.m_diagnostics
```