Python for Tooling - 2

Prepared By

Ahmed Magdy

Agenda

- Regular Expressions (Regex)
- File Operations
- Python Graphical User Interface (GUI) using PyQT libray.

Lab 5 - Optional

Lab 5 - Optional

- Download the Python module file from the following link and solve it

https://drive.google.com/open?id=1Lb5539jIRbQ56oyIpuXhyaINWOH86NfI

Regular Expressions (Regex)

Regular Expressions (Regex)

- Regular expressions are a powerful language for matching text patterns.
- We use it if we want to search for a text that we know only its pattern and don’t know it exactly.

  i.e. Search for e-mail address or phone number inside a text block

- In Python a regular expression search is typically written as:

  match = re.search(pattern , string)

Regular Expressions (Regex)

- Example :-

```
import re

line = "Cats are smarter than dogs"

matchObj = re.match( r ‘ (.*) are (.*?) .*  ', line)

if matchObj:
   print "matchObj.group() : ", matchObj.group()
   print "matchObj.group(1) : ", matchObj.group(1)
   print "matchObj.group(2) : ", matchObj.group(2)
else:
   print "No match!!"
```

Regular Expressions (Regex)

- Basic Patterns :-

^	
Matches beginning of line.	
$	
Matches end of line.	
.	
Matches any single character except newline. Using m option allows it to match newline as well.	
[...]	
Matches any single character in brackets.	
[^...]	
Matches any single character not in brackets	

Regular Expressions (Regex)

- Basic Patterns :-

re*
Matches 0 or more occurrences of preceding expression.

re+
Matches 1 or more occurrence of preceding expression.

re?
Matches 0 or 1 occurrence of preceding expression.

re{ n}
Matches exactly n number of occurrences of preceding expression.

re*
Matches 0 or more occurrences of preceding expression.

Regular Expressions (Regex)

- Basic Patterns :-

\w
Matches word characters.

\W
Matches nonword characters.

\s
Matches whitespace. Equivalent to [\t\n\r\f].

\S
Matches nonwhitespace.

\d
Matches digits. Equivalent to [0-9].

\D
Matches nondigits.

Regular Expressions (Regex)

- Group Extraction:-

```
str = 'purple alice-b@google.com monkey dishwasher'
match = re.search('([\w.-]+)@([\w.-]+)', str)
if match:
  print match.group()   ## 'alice-b@google.com' (the whole match)
  print match.group(1)  ## 'alice-b' (the username, group 1)
  print match.group(2)  ## 'google.com' (the host, group 2)
```

Regular Expressions (Regex)

- **findall and Groups:-**

Before we used re.search() to find the first match for a pattern. findall() finds *all* the matches and returns them as a list of strings, with each string representing one match.

i.e.

```
str = 'purple alice@google.com, blah monkey bob@abc.com blah dishwasher'
tuples = re.findall(r'([\w\.-]+)@([\w\.-]+)', str)
print tuples  ## [('alice', 'google.com'), ('bob', 'abc.com')]
for tuple in tuples:
  print tuple[0]  ## username
  print tuple[1]  ## host
```

Regular Expressions (Regex)

- Note

Except for control characters, **(+ ? . * ^ $ ( ) [ ] { } | \)**, all characters match themselves. You can escape a control character by preceding it with a backslash.

File Operations

File Operations

The code f = open('name', 'r') opens the file into the variable f, ready for reading operations, and use f.close() when finished. Instead of 'r', use 'w' for writing, and 'a' for append.

```
# Echo the contents of a file
  f = open('foo.txt', 'r')
  for line in f:   ## iterates over the lines of the file
    print line,    ## trailing , so print does not add an end-of-line char
                 ## since 'line' already includes the end-of line.

  f.close()
```

Lab 6

Lab 6

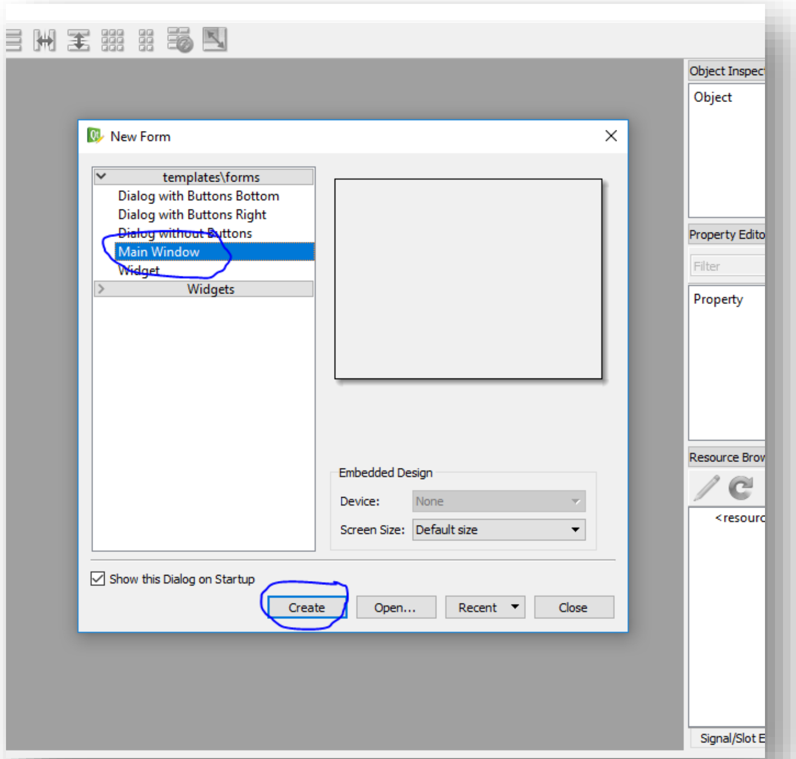
- Download the following file and use Python to open it and extract function names and input parameters and the return types.

  https://drive.google.com/open?id=1HMSV2hxdkR0pO7ZFIOXp4_ptw7x57WVK

Python Graphical User Interface (GUI) using PyQT libray.

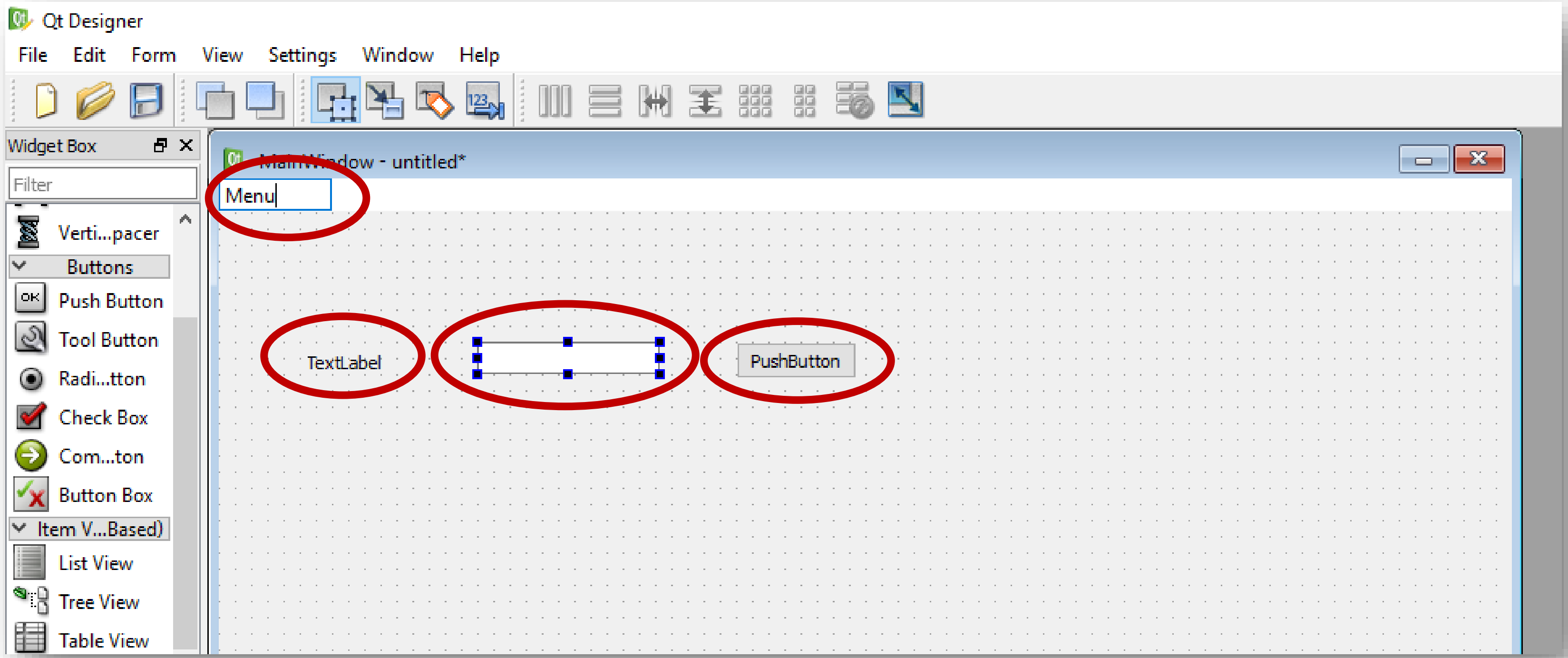
Python Graphical User Interface (GUI) using PyQT libray.

- Make sure that you have installed both the "Python Environment" and the "PyQT Environment".
- Open "QtDesigner"



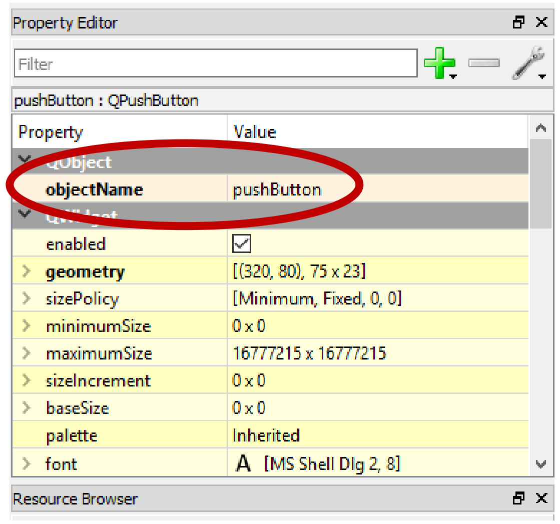
Python Graphical User Interface (GUI) using PyQT libray.

- You can add Pushbutton, LineEdit , Label or Menu --→ then save .ui file



Python Graphical User Interface (GUI) using PyQT libray.

- Note while making your GUI:
  - objectName ---> Name of the variable inside the code
  - Text ---> The string that appears to the User



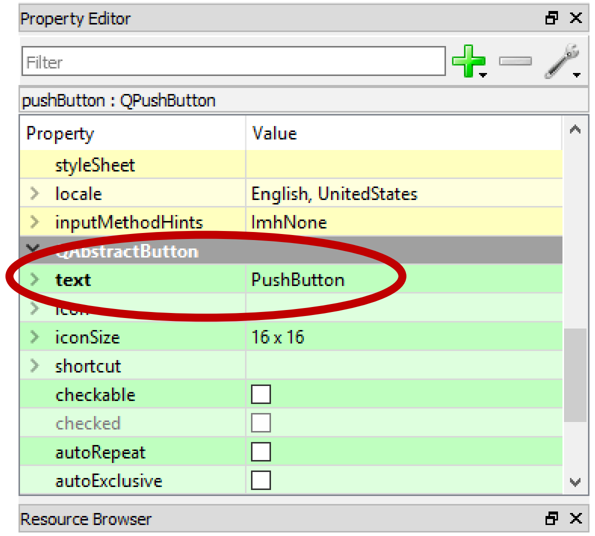
Property Editor

Filter

pushButton : QPushButton

Property	Value
QObject	
objectName	pushButton
QWidget	
enabled	☑
geometry	[(320, 80), 75 x 23]
sizePolicy	[Minimum, Fixed, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [MS Shell Dlg 2, 8]

Resource Browser



Property Editor

Filter

pushButton : QPushButton

Property	Value
styleSheet	
locale	English, UnitedStates
inputMethodHints	ImhNone
QAbstractButton	
text	PushButton
icon	
iconSize	16 x 16
shortcut	
checkable	☐
checked	☐
autoRepeat	☐
autoExclusive	☐

Resource Browser

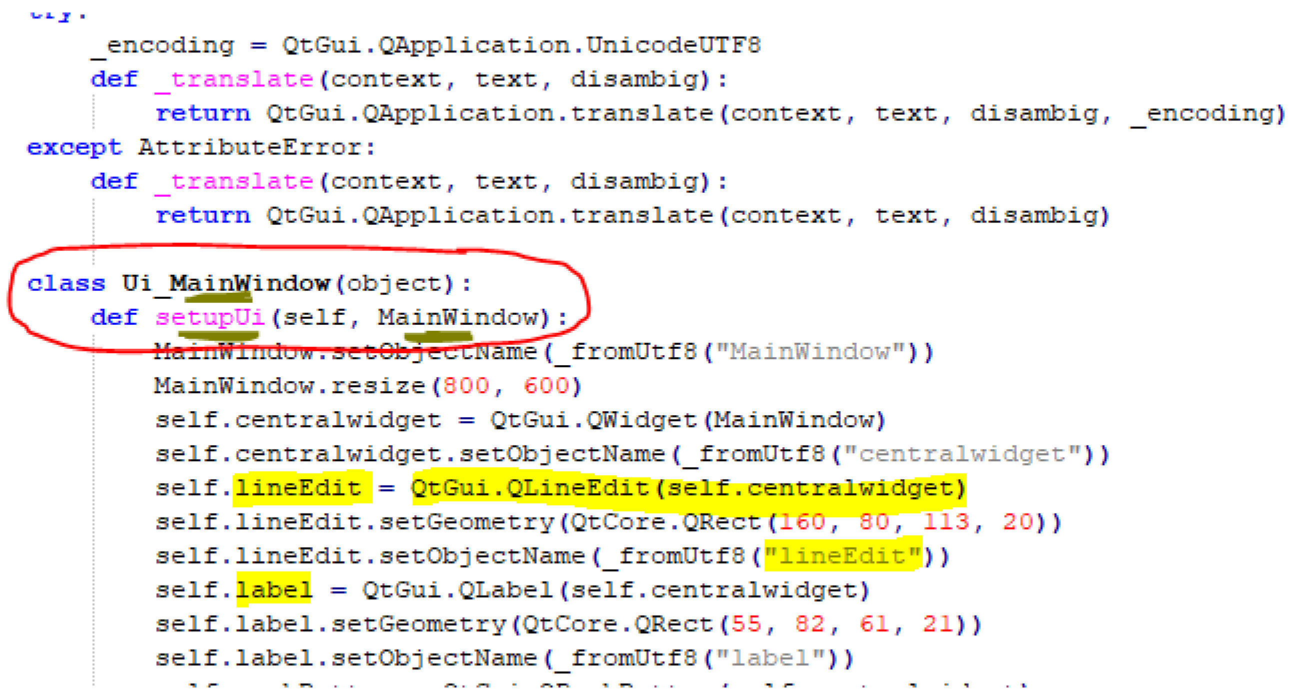
Python Graphical User Interface (GUI) using PyQT libray.

- After saving the .ui file convert it into Python module using the following pyuic4.bat tool by typing into the cmd in the same directory of the .ui file :

```
pyuic4.bat -x filename.ui -o filename.py
```

Python Graphical User Interface (GUI) using PyQT libray.

- After saving converting the ui file to py. Open it.

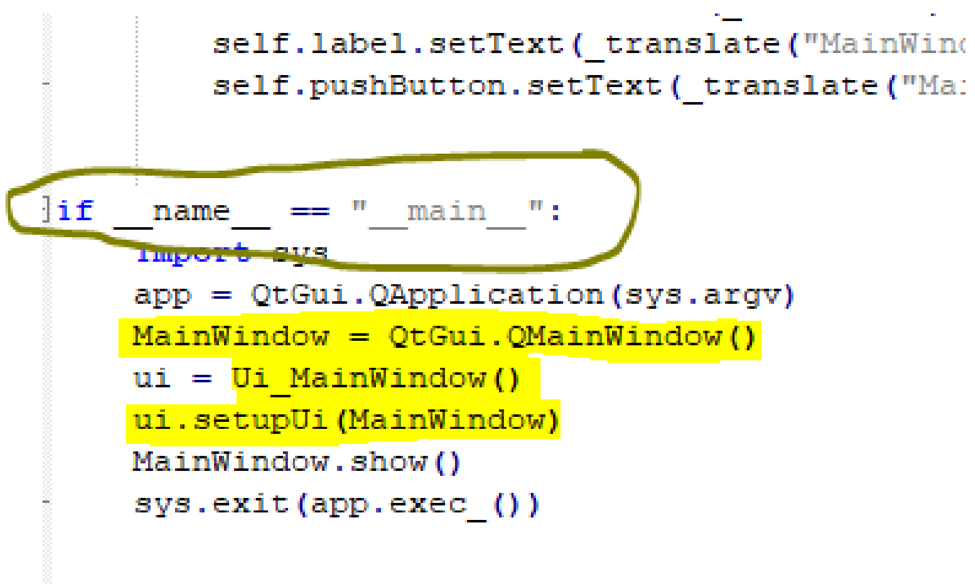


```
try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName(_fromUtf8("MainWindow"))
        MainWindow.resize(800, 600)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName(_fromUtf8("centralwidget"))
        self.lineEdit = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(160, 80, 113, 20))
        self.lineEdit.setObjectName(_fromUtf8("lineEdit"))
        self.label = QtGui.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(55, 82, 61, 21))
        self.label.setObjectName(_fromUtf8("label"))
```

Python Graphical User Interface (GUI) using PyQT libray.

- After saving converting the ui file to py. Open it. (BoilerPlate)



```
        self.label.setText(_translate("MainWin
        self.pushButton.setText(_translate("Ma

if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    MainWindow = QtGui.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

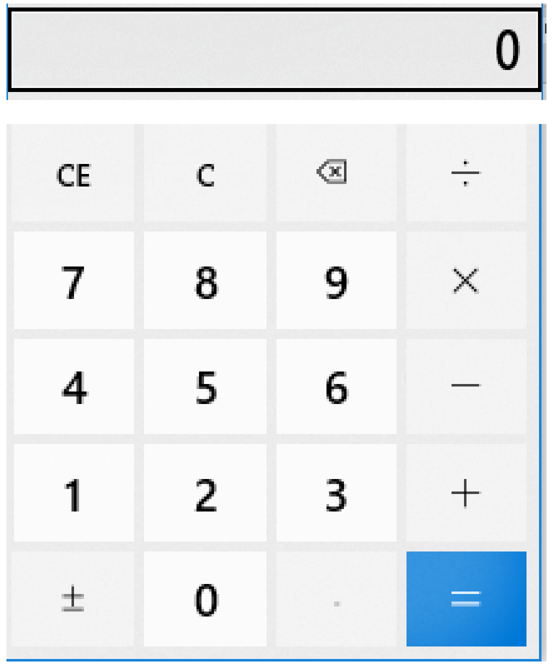
Python Graphical User Interface (GUI) using PyQT libray.

- After saving converting the ui file to py. Open it. You can *import* this module and call it from your Python code.

Lab 7

Lab 7

- Make a Calculator using Python and PyQT



Any Questions or Comments ?

References

References

- Google's Python Class
- Tutorialspoint