



PMR3201 Computação para Automação

Aula de Laboratório 6

PyQt: Geometry, Label, Box Layout, Menu, File Dialog

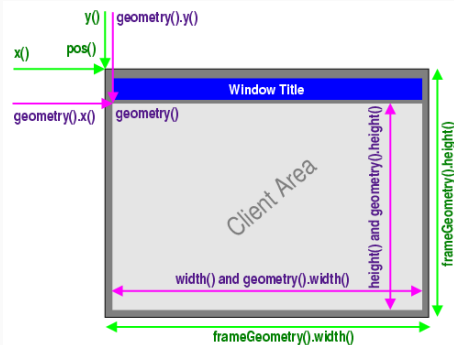
Newton Maruyama
Thiago de Castro Martins
Marcos S. G. Tsuzuki
Rafael Traldi Moura
André Kubagawa Sato
5 de maio de 2020

PMR-EPUSP

1. Geometria
2. Posicionamento absoluto
3. Classe BoxLayout
4. File Dialog
5. Para você fazer

Geometria

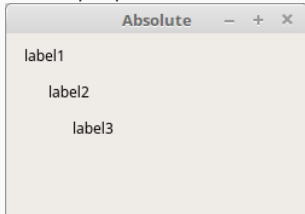
- ▶ Na aula anterior foi utilizado a função que define a posição e geometria da Janela, detalha-se aqui a utilização dessa função:
`QWidget.setGeometry(self,x,y,width,height)`
- ▶ Como visto na figura abaixo (x,y) define a posição da Janela na tela do computador e (width,height) define a largura e altura da janela.



Posicionamento absoluto

Exemplo: posicionamento absoluto de *Widgets*

- ▶ É possível posicionar cada um dos *widgets* que se deseja inserir utilizando o sistema de coordenadas da Janela.
- ▶ O canto superior esquerdo da Janela corresponde às coordenadas (0,0).
- ▶ O pixel do canto superior esquerdo do *widget* é utilizado como referência para ancoragem.
- ▶ O exemplo produz uma Janela contendo três *labels*:



- ▶ O exemplo está implementado no arquivo `AbsolutePos.py`.
- ▶ Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: AbsolutePos.py

- A seguir é apresentado o código fonte:

```
import sys
from PyQt5.QtWidgets import QWidget, QLabel, QApplication
class Janela(QWidget):
    def __init__(self):
        super(Janela, self).__init__()
        self.initUI()

    def initUI(self):

        lbl1 = QLabel('label1', self)
        lbl1.move(15, 10)

        lbl2 = QLabel('label2', self)
        lbl2.move(35, 40)

        lbl3 = QLabel('label3', self)
        lbl3.move(55, 70)

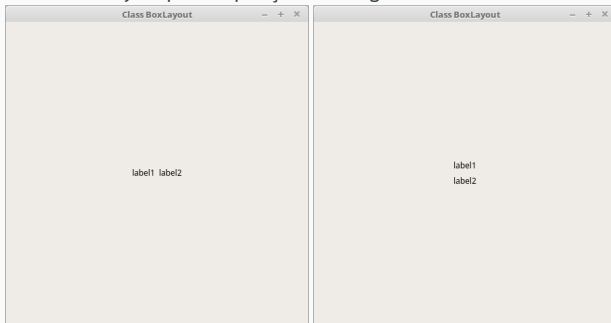
        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Absolute')
        self.show()

if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
    x = Janela()
    app.exec_()
```

Classe BorderLayout

Exemplo: utilização de *containers*

- ▶ Ao invés de posicionar os *labels* através da utilização de coordenadas podemos utilizar layout containers.
- ▶ Layout containers são estruturas que podem conter *widgets* filhos com disposição geométrica pré-definida.
- ▶ A classe **BoxLayout** é uma implementação de layout containers.
- ▶ Existem dois tipos de *BoxLayout*:
 - ▶ QHBoxLayout para disposição dos widgets horizontalmente.
 - ▶ QVBoxLayout para disposição dos widgets verticalmente.



- ▶ Um exemplo da utilização de *BoxLayout* está implementado no arquivo `BoxLayout.py`.
- ▶ Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: arquivo BoxLayout.py

- A seguir é apresentado o código fonte:

```
import sys
from PyQt5.QtWidgets import QWidget, QLabel, QHBoxLayout, QVBoxLayout, QApplication
from PyQt5 import QtCore
class Janela(QWidget):
    def __init__(self):
        super(Janela, self).__init__()
        self.initUI()
    def initUI(self):
        # box = QHBoxLayout()
        box = QVBoxLayout()
        lbl1 = QLabel('label1')
        lbl2 = QLabel('label2')
        box.addWidget(lbl1)
        box.addWidget(lbl2)
        self.setLayout(box)

        #box.setAlignment(QtCore.Qt.AlignTop)
        #box.setAlignment(QtCore.Qt.AlignBottom)
        box.setAlignment(QtCore.Qt.AlignCenter)
        #box.setAlignment(QtCore.Qt.AlignHCenter)
        #box.setAlignment(QtCore.Qt.AlignVCenter)
        #box.setAlignment(QtCore.Qt.AlignLeft)
        #box.setAlignment(QtCore.Qt.AlignRight)

        self.setGeometry(300, 300, 450, 450)
        self.setWindowTitle('Class BoxLayout')
        self.show()

if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
```

Observações:

- ▶ Nesse exemplo podemos escolher entre disposição vertical ou horizontal escolhendo entre um dos comandos abaixo:

```
#box = QHBoxLayout()  
box = QVBoxLayout()
```

- ▶ Os labels são inseridos no *Container* através dos seguintes comandos:

```
box.addWidget(lbl1)  
box.addWidget(lbl2)
```

- ▶ Em seguida o container deve ser associado à janela principal como a seguir:

```
self.setLayout(box)
```

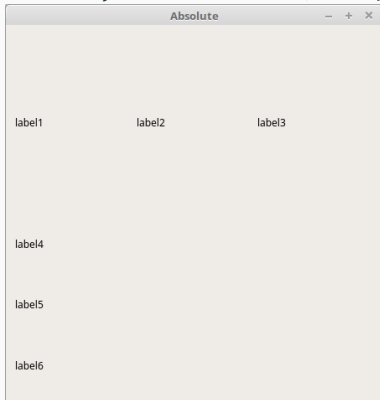
- ▶ Além da disposição horizontal ou vertical dos *labels* é necessário definir a posição do container na janela principal.
- ▶ No exemplo, foi escolhido que o container ficara no centro da janela principal, entretanto existe a possibilidade de escolha entre os vários tipos de posicionamento.

```
#box.setAlignment(QtCore.Qt.AlignTop)
#box.setAlignment(QtCore.Qt.AlignBottom)
box.setAlignment(QtCore.Qt.AlignCenter)
#box.setAlignment(QtCore.Qt.AlignHCenter)
#box.setAlignment(QtCore.Qt.AlignVCenter)
#box.setAlignment(QtCore.Qt.AlignLeft)
#box.setAlignment(QtCore.Qt.AlignRight)
```

- ▶ **Realize testes dos vários tipos de posicionamento.**

Exemplo: *containers* dentro de *containers*

- ▶ Muitas vezes, a utilização de *BoxLayouts* dentro de outros *BoxLayouts* é a melhor maneira de criar um *Layout* com muitos elementos.
- ▶ A Janela abaixo por exemplo foi criado com três *labels* dentro de um *QHBoxLayout* e três *labels* dentro de um *QVBoxLayout*. Posteriormente coloca-se tais *BoxLayouts* dentro de um *QVBoxLayout*.



- ▶ Repare que não se controla o posicionamento dos *labels* dentro do *container*.
- ▶ Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: arquivo Boxlayout1.py

- A seguir é apresentado o código fonte

```
import sys
from PyQt5.QtWidgets import QWidget, QLabel, QHBoxLayout, QVBoxLayout, QApplication
class Example(QWidget):
    def __init__(self):
        super(Example, self).__init__()
        self.initUI()

    def initUI(self):

        hbox1 = QHBoxLayout()
        vbox1 = QVBoxLayout()

        lbl1 = QLabel('label1')
        lbl2 = QLabel('label2')
        lbl3 = QLabel('label3')
        lbl4 = QLabel('label4')
        lbl5 = QLabel('label5')
        lbl6 = QLabel('label6')
```

► Continuação ...

```
hbox1.addWidget(lbl1)
hbox1.addWidget(lbl2)
hbox1.addWidget(lbl3)

vbox1.addWidget(lbl4)
vbox1.addWidget(lbl5)
vbox1.addWidget(lbl6)

vbox2 = QVBoxLayout(self)
vbox2.addLayout(hbox1)
vbox2.addLayout(vbox1)

self.setLayout(vbox2)

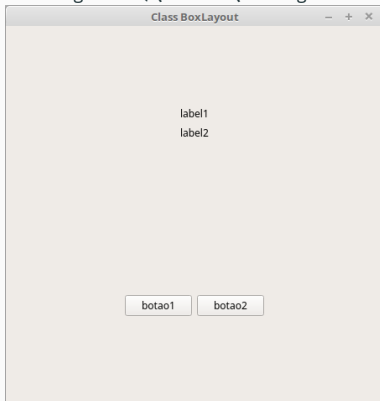
self.setGeometry(300, 300, 450, 450)
self.setWindowTitle('Absolute')
self.show()

if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()

    x = Example()
    app.exec_()
```

Exemplo: controlando o posicionamento dos elementos

- Utiliza-se aqui dois labels dentro de um objeto da classe `QVBoxLayout` de dois botões dentro de um objeto da classe `QHBoxLayout` e ambos são colocados dentro de um outro objeto da classe `QVBoxLayout`.
- Realiza-se aqui a posição dos elementos através do comando `setAlignment(QtCore.Qt.AlignCenter)`.



- Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: arquivo BoxLayout2.py

- A seguir apresenta-se o código fonte:

```
import sys
from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QHBoxLayout, QVBoxLayout, QApplication
from PyQt5 import QtCore
class Janela(QWidget):
    def __init__(self):
        super(Janela, self).__init__()
        self.initUI()

    def initUI(self):
        box1 = QVBoxLayout()

        lbl1 = QLabel('label1')
        lbl2 = QLabel('label2')

        box1.addWidget(lbl1)
        box1.addWidget(lbl2)
        box1.setAlignment(QtCore.Qt.AlignCenter)

        box2 = QHBoxLayout()

        button1 = QPushButton('botao1')
        button2 = QPushButton('botao2')

        box2.addWidget(button1)
        box2.addWidget(button2)
        box2.setAlignment(QtCore.Qt.AlignCenter)
```

► Continuação ...

```
        mainbox = QVBoxLayout()
        mainbox.addLayout(box1)
        mainbox.addLayout(box2)

        self.setLayout(mainbox)

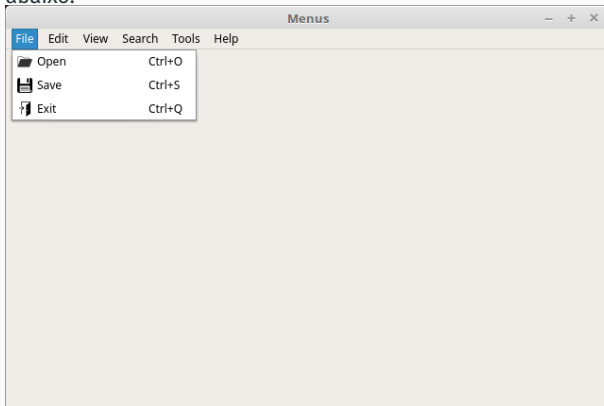
        self.setGeometry(300, 300, 450, 450)
        self.setWindowTitle('Class BoxLayout')
        self.show()

if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
    x = Janela()
    app.exec_()
```

File Dialog

Exemplo: Menu Bar

- ▶ Inicialmente ilustra-se como se constroe um *Menu Bar* como ilustrado na figura abaixo:



- ▶ O exemplo encontra-se no arquivo `TesteMenuo.py`.
- ▶ Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: arquivo TesteMenuo.py

- A seguir é apresentado o código fonte:

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QAction
from PyQt5.QtGui import QIcon

class App(QMainWindow):

    def __init__(self):
        super().__init__()
        self.title = 'Menuo'
        self.left = 10
        self.top = 10
        self.width = 640
        self.height = 400
        self.initUI()

    def initUI(self):
        self.setWindowTitle( self.title )
        self.setGeometry( self.left, self.top, self.width, self.height )

        mainMenu = self.menuBar( )
        fileMenu = mainMenu.addMenu( 'File' )
        editMenu = mainMenu.addMenu( 'Edit' )
        viewMenu = mainMenu.addMenu( 'View' )
        searchMenu = mainMenu.addMenu( 'Search' )
        toolsMenu = mainMenu.addMenu( 'Tools' )
        helpMenu = mainMenu.addMenu( 'Help' )
```

Exemplo: arquivo TesteMenuo.py

► Continuação ...

```
openButton = QAction( QIcon('openfile.png'), 'Open', self )
openButton.setShortcut( 'Ctrl+O' )
openButton.setStatusTip( 'Open File' )
# openButton.triggered.connect(self.close)
fileMenu.addAction(openButton)

saveButton = QAction( QIcon('savefile.png'), 'Save', self )
saveButton.setShortcut( 'Ctrl+S' )
saveButton.setStatusTip( 'Save File' )
# saveButton.triggered.connect( ?? )
fileMenu.addAction( saveButton )

exitButton = QAction( QIcon('exit.png'), 'Exit', self )
exitButton.setShortcut( 'Ctrl+Q' )
exitButton.setStatusTip( 'Exit application' )
exitButton.triggered.connect( self.close )
fileMenu.addAction( exitButton )

self.show()

if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()

    x = App()
    app.exec_()
```

- ▶ Inicialmente uma barra de menu é criada através da classe **menuBar**.
- ▶ Em seguida adiciona-se à barra as diversas opções do menu como ilustrado a seguir:

```
mainMenu = self.menuBar( )  
fileMenu = mainMenu.addMenu( 'File' )  
editMenu = mainMenu.addMenu( 'Edit' )  
viewMenu = mainMenu.addMenu( 'View' )  
searchMenu = mainMenu.addMenu( 'Search' )  
toolsMenu = mainMenu.addMenu( 'Tools' )  
helpMenu = mainMenu.addMenu( 'Help' )
```

- ▶ Somente para a opção do menu identificada como 'File' são criadas opções internas: 'Open', 'Save' e 'Exit' utilizando a classe **QAction**.
- ▶ Associa-se *Icons* para cada uma dessas opções no formato *.png.
- ▶ Note que é possível associar uma tecla de *Short Cut* através de `setShortcut()`, mensagem de status através de `setStatusTip()`, e uma ação através de `triggered.connect()`.
- ▶ Note que somente para 'Exit' existe uma ação definida que encerra a Janela.

```
openButton = QAction( QIcon( 'openfile.png' ), 'Open', self )
openButton.setShortcut( 'Ctrl+O' )
openButton.setStatusTip( 'Open File' )
# openButton.triggered.connect(self.close)
fileMenu.addAction(openButton)

saveButton = QAction( QIcon( 'savefile.png' ), 'Save', self )
saveButton.setShortcut( 'Ctrl+S' )
saveButton.setStatusTip( 'Save File' )
# saveButton.triggered.connect( ?? )
fileMenu.addAction( saveButton )

exitButton = QAction( QIcon( 'exit.png' ), 'Exit', self )
exitButton.setShortcut( 'Ctrl+Q' )
exitButton.setStatusTip( 'Exit application' )
exitButton.triggered.connect( self.close )
fileMenu.addAction( exitButton )
```


- ▶ Partindo do exemplo anterior vamos acrescentar ações associadas às opções 'Open' e 'Save'.
- ▶ A opção 'Open' abre um *File Dialog* que permite que o usuário selecione um arquivo *.txt.
- ▶ O conteúdo do arquivo é colocado na área de texto provida pela classe **QTextEdit**.
- ▶ Se o usuário selecionar a opção 'Save' um outro *File Dialog* é aberto para salvar o texto num arquivo *.txt.
- ▶ O exemplo encontra-se no arquivo TesteMenu1.py.
- ▶ Carregue o arquivo na IDE Spyder e verifique o seu funcionamento.

Exemplo: arquivo TesteMenu1.py

- A seguir é apresentado o código fonte:

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QAction, QFileDialog, QTextEdit
from PyQt5.QtGui import QIcon
class App(QMainWindow):
    def __init__(self):
        super().__init__()
        self.title = 'Menus'
        self.left = 10
        self.top = 10
        self.width = 640
        self.height = 400
        self.initUI()
    # Create an Editor Widget
    def editor(self):
        self.textEdit = QTextEdit()
        self.setCentralWidget(self.textEdit)
    # launches File Dialog Widget
    def file_open(self):
        options = QFileDialog.DontUseNativeDialog
        filenames = QFileDialog.getOpenFileName( self, 'Open File', '', 'text files (*.txt)', None, options )
        name = filenames[0]
        file = open(name, 'r+')
        with file:
            text = file.read()
            self.textEdit.setText( text )
        file.close()
    def file_save(self):
        options = QFileDialog.DontUseNativeDialog
        filenames = QFileDialog.getSaveFileName( self, 'Open File', '', 'text files (*.txt)', None, options )
        name = filenames[0]
        file = open(name, 'w')
        file.write( str(self.textEdit.toPlainText()) )
        file.close()
```

► Continuação ...

```
def initUI( self ):
    self.setWindowTitle( self.title )
    self.setGeometry( self.left, self.top, self.width, self.height )

    mainMenu = self.menuBar( )
    fileMenu = mainMenu.addMenu( 'File' )
    editMenu = mainMenu.addMenu( 'Edit' )
    viewMenu = mainMenu.addMenu( 'View' )
    searchMenu = mainMenu.addMenu( 'Search' )
    toolsMenu = mainMenu.addMenu( 'Tools' )
    helpMenu = mainMenu.addMenu( 'Help' )

    openButton = QAction( QIcon('openfile.png'), 'Open', self )
    openButton.setShortcut( 'Ctrl+O' )
    openButton.setStatusTip( 'Open File' )
    openButton.triggered.connect( self.file_open )
    fileMenu.addAction( openButton )
```

► Continuação ...

```
saveButton = QAction( QIcon('savefile.png'), 'Save', self )
saveButton.setShortcut( 'Ctrl+S' )
saveButton.setStatusTip( 'Save File' )
saveButton.triggered.connect( self.file_save )
fileMenu.addAction( saveButton )

exitButton = QAction( QIcon('exit.png'), 'Exit', self )
exitButton.setShortcut( 'Ctrl+Q' )
exitButton.setStatusTip( 'Exit application' )
exitButton.triggered.connect( self.close )
fileMenu.addAction( exitButton )

self.editor()
self.show()
if __name__ == '__main__':
    if not QApplication.instance():
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
    x = App()
    app.exec_()
```

Observações

- Note que a área de texto é criada através do seguinte código:

```
# Create an Editor Widget
def editor(self):
    self.textEdit = QTextEdit()
    self.setCentralWidget(self.textEdit)
```

- A criação do *File Dialog* para leitura do arquivo e inserção do conteúdo do arquivo na área de texto é realizada pela função `file_open()`:

```
def file_open(self):
    options = QFileDialog.DontUseNativeDialog
    filenames = QFileDialog.getOpenFileName( self, 'Open File', '',
        'text files (*.txt)', None, options )
    name = filenames[0]
    file = open(name, 'r+')
    with file:
        text = file.read()
        self.textEdit.setText( text )
    file.close()
```

- A função `getOpenFile()` devolve uma lista de Strings na variável `filenames` e o nome do arquivo selecionado corresponde a primeira posição da lista, i.e., `filenames[0]`.

- A função que cria um *File Dialog* para salvar o conteúdo da área de texto em um arquivo *.txt é apresentado abaixo:

```
def file_save(self):  
    options = QFileDialog.DontUseNativeDialog  
    filenames = QFileDialog.getSaveFileName( self, 'Open File', '',  
        'text files (*.txt)', None, options )  
    name = filenames[0]  
    file = open(name, 'w')  
    file.write( str(self.textEdit.toPlainText()) )  
    file.close()
```

Group box

- ▶ Um *group box* é um container que organiza botões (*radio buttons* e *push buttons*) e *check boxes* em grupos.
- ▶ No dia a dia nos deparamos sempre com a utilização desses elementos em formulários eletrônicos por exemplo.
- ▶ A figura abaixo ilustra a utilização numa interface gráfica:

Exclusive Radio Buttons

Non-Exclusive Checkboxes

☒ Radio button 1

☐ Radio button 2

☐ Radio button 3

☐ Checkbox 1

☒ Checkbox 2

☐ Tri-state button

☐ Exclusive Radio Buttons

☒ Push Buttons

☒ Radio button 1

☐ Radio button 2

☐ Radio button 3

☒ Independent checkbox

Normal Button

Toggle Button

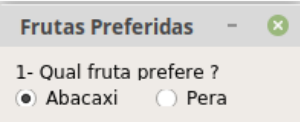
Flat Button

Popup Button

- ▶ Os *push-buttons* já foram apresentados.
- ▶ Os *check-boxes* tem comportamento não exclusivo, ou seja, é possível selecionar mais de uma opção, enquanto que os *radio-buttons* tem comportamento exclusivo.

Um exemplo utilizando Radio Buttons

- A interface a ser construída é ilustrada na figura abaixo:



Frutas Preferidas - ✕

1- Qual fruta prefere ?

☒ Abacaxi ☐ Pera

Exemplo: arquivo testegroupbox1

```
import sys
from PyQt5.QtWidgets import QWidget, QMainWindow, QRadioButton,
                             QLabel, QGroupBox, QHBoxLayout, QVBoxLayout, QApplication
from PyQt5 import QtCore

def adeus(x):
    for i in range(0,2):
        if x[i].isChecked() == True:
            print("A minha fruta escolhida e: ",x[i].text())

class Radiodemo(QWidget):    # Radiodemo sub-classe de QWidget

    def __init__(self):
        QWidget.__init__(self)    # Inicializacao da super-classe

        self.mainlayout = QVBoxLayout() # layout principal do tipo vertical

        # Ha apenas uma pergunta mas se existissem N perguntas -> Perguntas[0..N-1]
        self.Perguntas=[None]
        self.Perguntas[0] = QLabel('1- Qual fruta prefere ?')

        # As alternativas para cada questao formam um groupbox horizontal
        # Se N perguntas -> [0..N-1]
        self.LayoutDasAlternativas = [None]
        self.LayoutDasAlternativas[0] = QHBoxLayout() # layout horizontal

        # Matrix de radio buttons
        # Pre-alocacao com elementos vazios
        # Se N perguntas -> [0..N-1,0..N-1]
        # Por exemplo se 2 perguntas -> MatrixRadioButtons[[None,None],[None,None]]
        self.MatrixRadioButtons=[None,None]
```

Exemplo: arquivo testegroupbox1

```
self.MatrixRadioButtons[0]= QRadioButton("Abacaxi") # Cria opcao abacaxi
self.MatrixRadioButtons[0].setChecked(True) # Resposta default
self.MatrixRadioButtons[0].toggled.connect(lambda:self.btnstate(self.MatrixRadioButtons[0]))
# Acao associada a essa opcao - btnstate()

self.MatrixRadioButtons[1]= QRadioButton("Pera") # Cria opcao abacaxi
self.MatrixRadioButtons[1].toggled.connect(lambda:self.btnstate(self.MatrixRadioButtons[1]))
# Acao associada a essa opcao - btnstate()

# RadioButtons sao inseridos no Groupbox
self.LayoutDasAlternativas[0].addWidget(self.MatrixRadioButtons[0])
self.LayoutDasAlternativas[0].addWidget(self.MatrixRadioButtons[1])

# Construcao do layout principal
self.mainlayout.addWidget(self.Perguntas[0]) # Label da pergunta
self.mainlayout.addLayout(self.LayoutDasAlternativas[0]) # group box das alternativas

# coloca o main layout na janela
self.setLayout(self.mainlayout)
self.setWindowTitle("Frutas Preferidas") # Titulo da janela
self.setFixedSize(200,50) # Dimensoes da janela
```

Exemplo: arquivo testegroupbox1

```
def btnstate(self,b): # Identifica o estado dos RadioButtons True/False
    if b.text() == "Abacaxi":
        if b.isChecked() == True:
            print(b.text()+" foi selecionada")
        else:
            print(b.text()+" foi desselecionada")
    if b.text() == "Pera":
        if b.isChecked() == True:
            print(b.text()+" foi selecionada")
        else:
            print(b.text()+" foi desselecionada")

def closeEvent(self, event): # Definicao da funcao que executa se fechar a janela
    for i in range(0,2):
        if self.MatrixRadioButtons[i].isChecked() == True:
            print("A minha fruta escolhida",self.MatrixRadioButtons[i].text())

if __name__ == '__main__':
    if not QApplication.instance(): # Verifica se ja existe uma instancia de QApplication
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
    x = Radiodemo()
    x.show()
    app.exec_()
```

- ▶ A classe Radiodemo é sub-classe de QWidget
- ▶ O layout principal é vertical

```
self.mainlayout = QVBoxLayout()  
# layout principal do tipo vertical
```

- ▶ A pergunta é colocada através de um label:

```
# Ha apenas uma pergunta mas se existissem N perguntas -> Perguntas[0..N-1]  
self.Perguntas=[None]  
self.Perguntas[0] = QLabel('1- Qual fruta prefere ?')
```

- ▶ Cria-se um box horizontal que vai funcionar como um group box para os radio buttons:

```
# As alternativas para cada questao formam um groupbox horizontal  
# Se N perguntas -> [0..N-1]  
self.LayoutDasAlternativas = [None]  
self.LayoutDasAlternativas[0] = QHBoxLayout() # layout horizontal
```

LayoutDasAlternativas é declarado como array porque eventualmente pode haver a necessidade de N perguntas.

Observações

- ▶ Os widgets do tipo radio buttons serão alocados através de um array unidimensional de 2 posições. Eventualmente se houverem mais perguntas deve ser alocado um array de 2 dimensões:

```
# Matrix de radio buttons
# Pre-alocacao com elementos vazios
# Se N perguntas -> [0..N-1,0..N-1]
# Por exemplo se 2 perguntas -> MatrixRadioButtons = [[None,None],[None,None]]
# No caso so precisamos de um array de 2 posicoes
self.MatrixRadioButtons=[None,None]
```

- ▶ Dois objetos da classe QRadioButtons são criados, um associado à string Abacaxi e outro associado à string Pera.

```
self.MatrixRadioButtons[0]= QRadioButton("Abacaxi") # Cria opcao abacaxi
self.MatrixRadioButtons[0].setChecked(True) # Resposta default
self.MatrixRadioButtons[0].toggled.connect(lambda:self.btnstate(self.MatrixRadioButtons[0]))
# Acao associada a essa opcao - btnstate()

self.MatrixRadioButtons[1]= QRadioButton("Pera") # Cria ocao pera
self.MatrixRadioButtons[1].toggled.connect(lambda:self.btnstate(self.MatrixRadioButtons[1]))
# Acao associada a essa opcao - btnstate()
```

- ▶ A opção Abacaxi se torna a opção default através de setChecked(True)
- ▶ Uma ação está associada quando se clica o radio button através de toggled.connect.
- ▶ No caso há a chamada da função btnstate()

- Os dois radio buttons são inseridos no groupbox:

```
# RadioButtons sao inseridos no Groupbox
self.LayoutDasAlternativas[0].addWidget(self.MatrixRadioButtons[0])
self.LayoutDasAlternativas[0].addWidget(self.MatrixRadioButtons[1])
```

- Inicialmente coloca-se o label no layout vertical e depois o layout horizontal que contem os radio buttons:

```
# Construcao do layout principal
self.mainlayout.addWidget(self.Perguntas[0])
# Label da pergunta
self.mainlayout.addLayout(self.LayoutDasAlternativas[0])
# group box das alternativas
```

Note que o label é inserido através de `addWidget()` e o layout horizontal através de `addLayout()`

- Ao final associa-se o layout principal à janela, coloca-se o título da janela e setamos suas dimensões:

```
# coloca o main layout na janela
self.setLayout(self.mainlayout)
self.setWindowTitle("Frutas Preferidas") # Titulo da janela
self.setFixedSize(200,50) # Dimensoes da janela
```

- ▶ A função `btnstate()` consegue identificar a ação realizada no radio button:

```
def btnstate(self,b): # Identifica o estado dos RadioButtons True/False
    if b.text() == "Abacaxi":
        if b.isChecked() == True:
            print(b.text()+" foi selecionada")
        else:
            print(b.text()+" foi desselecionada")
    if b.text() == "Pera":
        if b.isChecked() == True:
            print(b.text()+" foi selecionada")
        else:
            print(b.text()+" foi desselecionada")
```

- ▶ A função `closeEvent()` é uma função especial que é executada quando a janela for fechada:

```
def closeEvent(self, event): # Definição da função que executa se fechar a janela
    for i in range(0,2):
        if self.MatrixRadioButtons[i].isChecked() == True:
            print("A minha fruta escolhida é: ",self.MatrixRadioButtons[i].text())
```

- ▶ No programa principal inicialmente é feito um teste para saber se já existe uma instância de um objeto da classe `QApplication` criando um novo objeto somente se necessário:

```
if __name__ == '__main__':
    if not QApplication.instance(): # Verifica se já existe uma instância de QApplication
        app = QApplication(sys.argv)
    else:
        app = QApplication.instance()
    x = Radiodemo()
    x.show()
    app.exec_()
```

Para você fazer

- ▶ Deseja-se construir uma interface gráfica para a realização de um teste de aritmética básica.
- ▶ Especificações:
 - ▶ Título da Janela: Teste de Aritmetica
 - ▶ O teste consiste de apenas 3 perguntas com duas alternativas possíveis (a) e (b):
 - 1-) $1 + 1 = ?$
(a) 2 (b) 0
 - 2-) $12 - 3 = ?$
(a) 15 (b) 9
 - 3-) $10 - 10 = ?$
(a) 0 (b) -1
- ▶ Ao final a função `closeEvent()` calcula quantas questões o usuário acertou.
- ▶ Não é necessário implementar a função `btnstate()`.
- ▶ Utilize como base o exemplo do arquivo `testegroupbox1.py`.