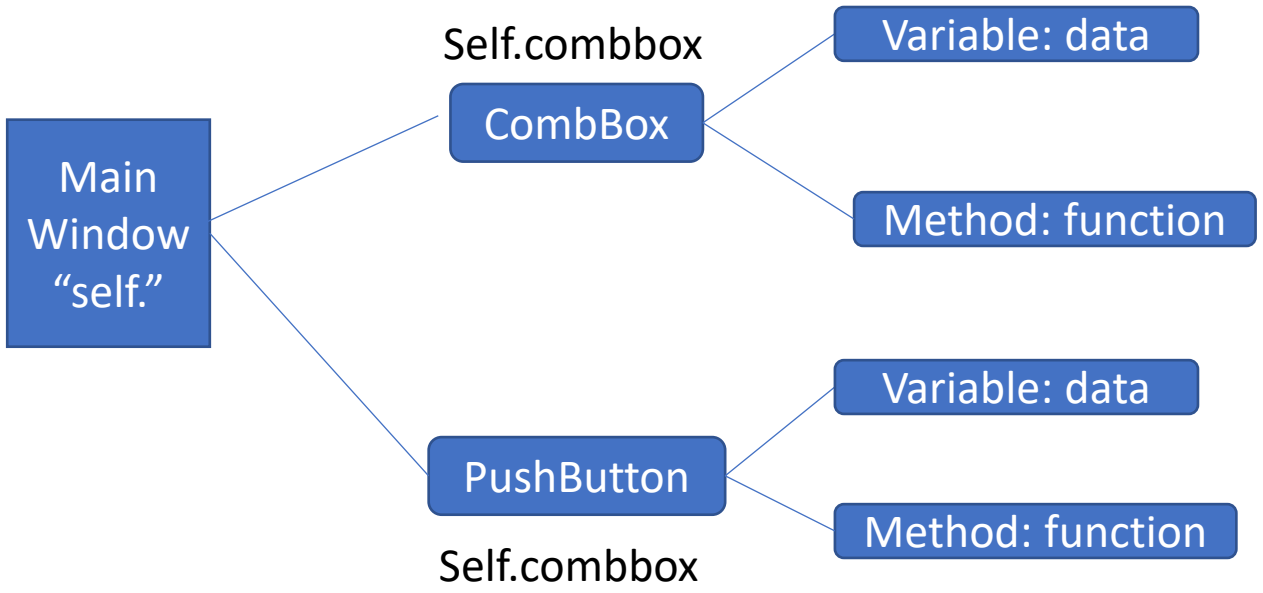
- Object oriented programming.

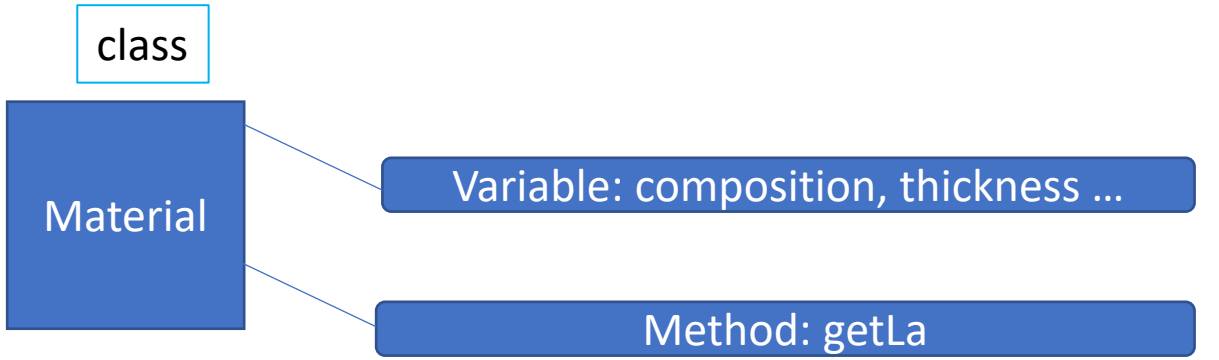



```python
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")
        layout = QVBoxLayout()

        #   add text line
        text1 = QLabel('Object-1')
        self.aCombo = QComboBox()
        self.aCombo.addItems(['Monday', 'Tuesday', 'Wednesday'])

        #   add combo-box
        text2 = QLabel('Object-2')
        self.aLineText = QLineEdit("test")

        # add checkbox
        text3 = QLabel('Object-3')
        self.aCheckBox = QCheckBox('Option?')

        #   add button
        text4 = QLabel('Object-4')
        self.aButton = QPushButton("Press Here!")
        self.aButton.setCheckable(True)
        # now connect object and function, when clicked
        self.aButton.clicked.connect(self.the_button_was_clicked)
```

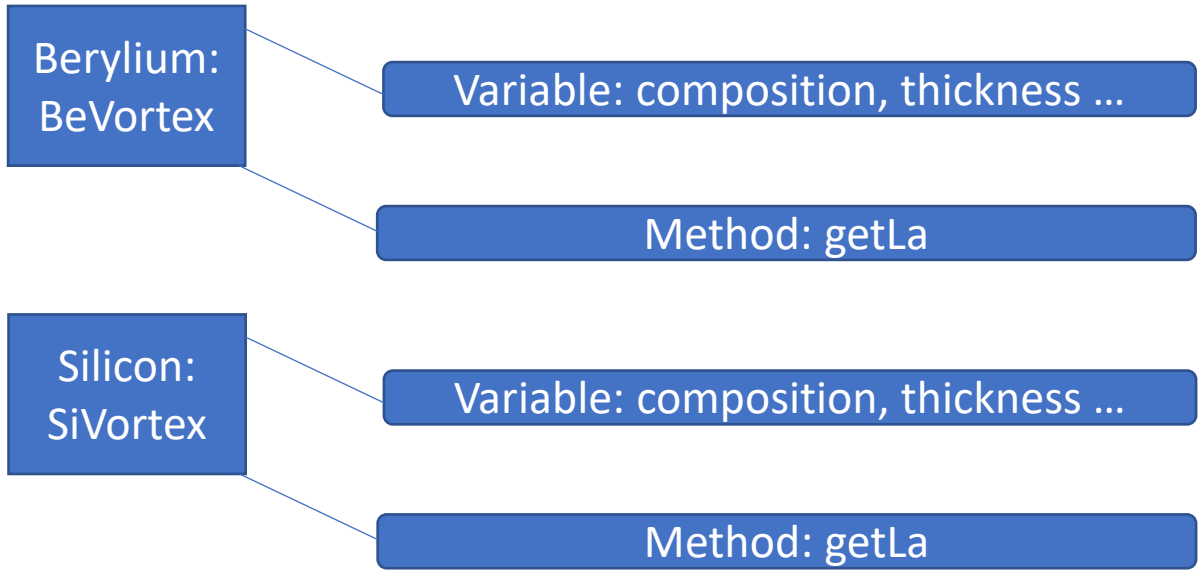
Test_qt2.py

7/11: python-Object-2

- Class, instance



Instances: members belonging to class




```python
class Material:
    def __init__(self, composition, density, thickness=1):
        self.composition=composition        # ex) SiO2
        self.density=density               # in g/cm^3
        self.thickness=thickness            # in cm
        self.la=1.0                         #absoprtion length in cm
        self.trans=0.5                      # transmission.
        self.absrp=0.5                  # absorption
        self.delta=0.1                   # index of refraction, real part
        self.beta=0.1                    # index of refraction, imaginary part
        temp=f1f2.get_ChemName(composition)
        AtomList=temp[0]
        AtomIndex=temp[1]
        AtomWeight=0
        for (ii, atom) in enumerate(AtomList):
            AtWt=f1f2.AtSym2AtWt(atom)
            index=AtomIndex[ii]
            AtomWeight = AtomWeight + index*AtWt
        NumberDensity=density*Nav/AtomWeight
        self.NumDen=NumberDensity       # number of molecules per cm^3
        self.AtWt=AtomWeight            # weight per mole
    def getLa(self, energy, NumLayer=1.0):    # get absorption legnth for incid
        temp=f1f2.get_delta(self.composition, self.density, energy)
        # returns delta, beta, la_photoE, Nat, la_total
        self.delta=temp[0];     self.beta=temp[1]
        self.la=temp[4]  # temp[4] (total) instead of temp[2] (photoelectric)
```

```python
def Assemble_QuadVortex(eV1):
    # quad vortex detector efficiency. eV1: fluo energy
    net=1.
    BeVortex=Material('Be', 1.85, 0.00125)
    SiO2Vortex=Material('SiO2', 2.2, 0.00001)
    SiVortex=Material('Si', 2.33, 0.035)
    BeVortex.getLa(eV1, 1)  # one Be layer in Vortex
    SiO2Vortex.getLa(eV1, 1)    # oxide layer on Si detec
    SiVortex.getLa(eV1, 1)  # Si detection layer, what's
    net=net*BeVortex.trans*SiO2Vortex.trans*SiVortex.absr
    if (print2screen):
        print ('%.3f eV : BeVortex.trans=%.3e , SiO2Vorte
    return net
```

fluo_det.py

Next steps,

- Make a layout with options (buttons, combobox…)
- Add tab functionality (tab1 for options, tab2 for graph, tab3 for table)

- Make a function to read input parameters from multiple objects in the main GUI
- Connect this function to a button (See test_qt.py in github).