AGH AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Python GUI 1

Celem ćwiczenia jest zapoznanie się z biblioteką PyQt, która umożliwia tworzenie graficznych interfejsów użytkownika (GUI – *graphic user interface*). Proszę zainstalować bibliotekę PyQt5 i zapoznać się z dokumentacją biblioteki, w szczególności z klasami: QPushButton, QLineEdit, QTextEdit oraz ich podstawowymi metodami. Dokumentacja dostępna jest na stronie: https://www.riverbankcomputing.com/static/Docs/PyQt4/qtgui.html

Przykład 1:

1) Pierwsza aplikacja okienkowa

Poniższy kod pozwala na uruchomienie pierwszej okienkowej aplikacji:

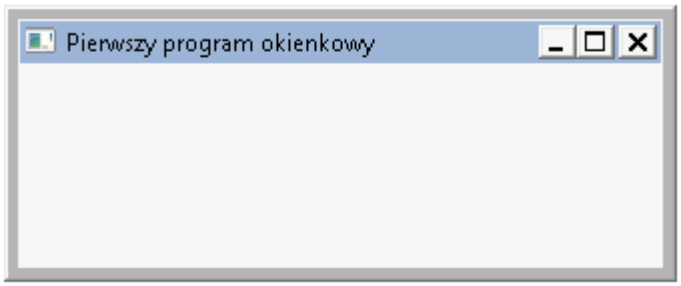
```
import sys
from PyQt5.QtWidgets import QApplication, QWidget

class App(QWidget):
    def __init__(self):
        super().__init__()
        self.title = 'Pierwszy program okienkowy'
        self.left = 100  # odległośc od lewej krawędzi ekranu w pixelach
        self.top = 100  # odległośc od górnej krawędzi ekranu w pixelach
        self.width = 300  # szerokośc okna
        self.height = 100  # wysokość okna

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

        ###############
        # Tu w kolejnych zadaniach będziemy dodwać przycisk typu QPushButton, Slot
oraz QLineEdit
        ###############
        self.show()


app = QApplication(sys.argv)
ex = App()
app.exec_()
```



Proszę zmienić nazwę wyświetlanego okna na swoje imię i nazwisko, zmienić rozmiar okna i jego położenie tak aby zajmował pełen ekran.

2) Przyciski (QPushButton)

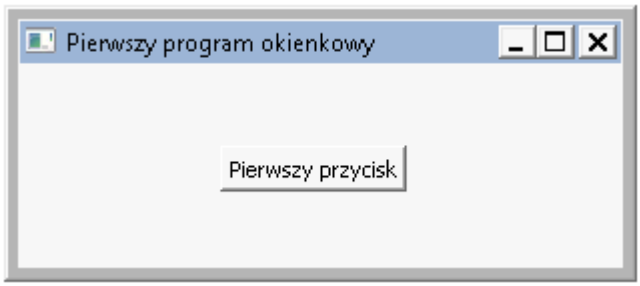
Z biblioteki pyQt5.QtWidgets proszę zaimportować QPushButton:

```
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton
```

Następnie linii z ustawieniem geometrii okna należy dodać:

```
self.button = QPushButton('Pierwszy przycisk', self)
self.button.setToolTip('Podpowiedź do pierwszego przycisku')
self.button.move(100, 40)   # położenie pierwszego przycisku
```

Powyższy fragment kodu spowoduje dodanie przycisku. Proszę uruchomić program i sprawdzić jego działanie. Wynikiem działania powinno być podobne okno jak na poniższym rysunku:



3) Obsługa naciśnięcia przycisku

Do przycisku należy dodać wywołanie sygnału od naciśnięcia przycisku:

```
self.button.clicked.connect(self.on_click)
```

A następnie do klasy App należy dodać metodę obsługującą naciśnięcie:

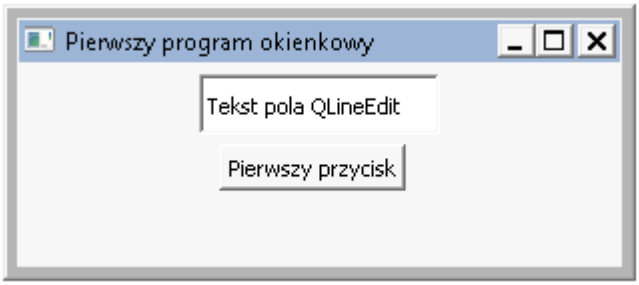
```
def on_click(self):
    print('Przycisk pierwszy zostal nacisniety')
```

4) Pole tekstowe QLineEdit

Aby dodać stworzyć pole tekstowe należy, podobnie jak poprzednio, zaimportować *QLineEdit* z *pyQt5.QtWidgets*, następnie należy dodać pod fragment kodu opisujący przycisk definicję pola tekstowego:

```
self.textbox = QLineEdit(self)
self.textbox.move(40, 5)
self.textbox.resize(220, 30)
self.textbox.setText("Tekst pola QLineEdit")
```

Proszę sprawdzić czy po uruchomieniu generowane jest pole tekstowe i znajduje się w odpowiednim miejscu.



Następnie do stworzonej wcześniej metody on_click proszę dodać poniższy fragment kodu, który ustawia tekst QLineEdit : „Przycisk został naciśnięty” po naciśnięciu przycisku.

```
self.textbox.setText("Przycisk został naciśnięty")
```

Uwaga!
Jeżeli chcemy wpisać w *textbox* zmienną liczbową musimy przekonwertować ją na łańcuch znaków używając funkcji str():

```
zmienna_liczbowa = 5.6
self.textbox.setText(str(zmienna_liczbowa))
```

Zadanie1.

Proszę napisać GUI zawierający odpowiednio rozmieszczone 3 pola tekstowe, oraz dwa przyciski. Pierwsze pole tekstowe powinno domyślnie zawierać wartość „Ala ma” natomiast drugie powinno zawierać domyślnie wartość „ kota”. Po naciśnięciu pierwszego przycisku zawartość pól tekstowych jeden i dwa powinna zostać połączona i przepisana do pola trzeciego.

Po naciśnięciu drugiego przycisku ma nastąpić skasowanie zawartości wszystkich pól tekstowych (metoda clear() klasy QLineEdit).

Podpowiedź:
Pobranie tekstu z QLineEdit i przepisanie go do zmiennej tekstowej można wykonać w następujący sposób:

```
zmienna = self.textbox1.text()
```

Zadanie 2.
Proszę zmodyfikować kod z poprzedniego zadania tak aby po wpisaniu liczb zmiennoprzecinkowych w pierwsze i drugie pole tekstowe i naciśnięciu pierwszego przycisku obliczyć sumę obu liczb i wyświetlić ją w polu tekstowym numer trzy.

Podpowiedź:
Pobranie z pola tekstowego wartość i zrzutowanie jej na zmienną typu float można wykonać w następujący sposób:

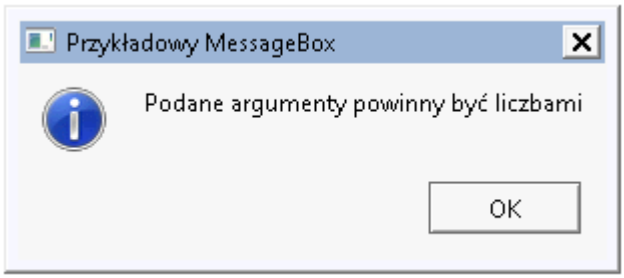
```
zmienna = float(self.textbox1.text())
```

Uwaga, jeżeli pole tekstowe będzie zawierało inne znaki niż liczbowe, program przy próbie rzutowania do *float'a* zakończy działanie zakończy działanie wyświetlając w konsoli komunikat o błędzie. Aby uniknąć takiej sytuacji należy wykorzystać mechanizm przechwytywania wyjątków *try and except* (Przykład tego mechanizmu był przedstawiony w instrukcji „Python funkcje").

Można poinformować użytkownika o wystąpieniu wyjątku poprzez wyświetlenie odpowiedniego okna dialogowego QMessageBox. W tym celu należy podobnie jak we wcześniejszych przykładach zaimportować klasę QMessageBox a następnie do naszej klasy App dodać metodę *showDialogBox* która będzie wywoływana podczas wystąpienie wyjątku (w except):

```
def showDialogBox(self):
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Information)
    msg.setText("Podane argumenty powinny być liczbami")
    msg.setWindowTitle("Przykładowy MessageBox")
    msg.setStandardButtons(QMessageBox.Ok)
    msg.exec_()
```

Po wprowadzeniu łańcucha znaków zamiast liczby powinien pojawić się komunikat w formie okna dialogowego:



Zadanie domowe:

Proszę napisać kalkulator z graficznym interfejsem użytkownika. Kalkulator powinien zawierać:
10 odpowiednio rozmieszczonych przycisków odpowiadający cyfrom od 0-9,
5 przycisków służących do wyboru operacji matematycznej (+, -, *, /, =, przecinek, znak minus, pierwiastek).

Napisanie samego kalkulatora z GUI oraz udowodnienie jego poprawnego działania dla zmiennoprzecinkowych liczb dodatnich i ujemnych oraz zabezpieczenie przed dzieleniem przez zero stanowi 70% oceny.

Dodatkowo:
Zastosowanie layoutów do rozmieszczenia przycisków 20%,
Wyświetlanie MessageBoxa w sytuacji dzielenia przez zero 10%