LAPORAN PRAKTIKUM PBO

Aplikasi Perhitungan Dengan Metode PyQt Dan Qt Designer

Dosen Pengampu : Fredy Wicaksono, M.Kom

Disusun Oleh:

Muhammad Faiz : 220511139

Muhammad Hidayat : 220511088

Revan Fazry Huda : 220511179

Universitas Muhammadiyah Cirebon

Jl. Fatahillah

No. 40 Watubelah, Sumber, Cirebon

2023

KATA PENGANTAR

Praktikum ini bertujuan untuk memberikan pemahaman dan keterampilan dalam mengembangkan aplikasi berbasis GUI (Graphical User Interface) menggunakan perangkat lunak PyQt dan Qt Designer. Fokus utama dari praktikum ini adalah pembuatan kalkulator sederhana yang melibatkan desain

Kalkulator dipilih sebagai proyek praktikum karena menggabungkan beberapa konsep dasar dalam pengembangan perangkat lunak, termasuk desain UI, logika pengolahan input, dan manajemen sinyal dan slot. Melalui praktikum ini, diharapkan peserta dapat memahami prinsip-prinsip dasar pengembangan aplikasi berbasis GUI dan mampu mengimplementasikannya menggunakan PyQt dan Qt Designer.

Terima kasih kepada semua peserta dan pembimbing praktikum yang telah berkontribusi dalam kesuksesan praktikum ini. Semoga pengalaman ini menjadi landasan untuk memahami lebih dalam konsep-konsep pengembangan perangkat lunak berbasis GUI.

Cirebon, November 2023

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era perkembangan teknologi informasi, pemrograman aplikasi desktop menjadi salah satu keahlian yang sangat dibutuhkan. PyQt, sebagai salah satu kerangka kerja pengembangan antarmuka grafis (GUI) untuk aplikasi desktop menggunakan bahasa pemrograman Python, telah menjadi pilihan populer di kalangan pengembang perangkat lunak. PyQt memberikan kemudahan dalam membuat aplikasi desktop yang interaktif dan responsif.

Pemahaman mendalam tentang PyQt penting bagi mahasiswa atau pengembang perangkat lunak yang ingin menguasai pembuatan aplikasi desktop yang modern dan efisien. Melalui praktikum ini, diharapkan peserta dapat memahami dasar-dasar PyQt, termasuk konsep GUI, event handling, dan integrasi antara logika bisnis dengan antarmuka pengguna. Selain itu, praktikum ini juga memberikan kesempatan kepada peserta untuk mengimplementasikan pengetahuan mereka dalam membuat aplikasi sederhana menggunakan PyQt.

Penggunaan PyQt tidak hanya memberikan keunggulan dalam segi fungsionalitas, tetapi juga mempercepat proses pengembangan aplikasi desktop. Oleh karena itu, pemahaman dan penguasaan terhadap PyQt akan membuka peluang yang lebih luas dalam mengembangkan aplikasi desktop yang inovatif dan berkualitas tinggi.

1.2 Tujuan Laporan

a. Memahami Konsep GUI (Graphical User Interface):
   Menyajikan pemahaman yang kokoh terkait dengan konsep dasar antarmuka grafis (GUI) dan komponen-komponen utama yang digunakan dalam pengembangan aplikasi desktop dengan PyQt.
b. Integrasi Logika Bisnis dan Antarmuka Pengguna:
   Menyelaraskan logika bisnis aplikasi dengan antarmuka pengguna, sehingga peserta mampu membuat aplikasi yang tidak hanya menarik secara visual tetapi juga efektif dalam memproses data.

c. Memahami Penggunaan Widget dan Layout:

   Mengenal, menggunakan, dan mengelola widget serta layout dalam PyQt untuk menyusun antarmuka pengguna yang bersih dan terstruktur.

d. Membangun Aplikasi Sederhana:

   Menerapkan pengetahuan yang diperoleh untuk merancang dan mengembangkan aplikasi sederhana menggunakan PyQt, sehingga peserta dapat merasakan secara langsung penerapan konsep-konsep yang dipelajari.

e. Mengasah Keterampilan Debugging:

   Mengembangkan keterampilan dalam memecahkan masalah (debugging) melalui identifikasi dan perbaikan kesalahan yang mungkin muncul selama pengembangan aplikasi dengan PyQt.

f. Meningkatkan Kreativitas dalam Pengembangan Aplikasi:

   Mendorong peserta untuk mengembangkan kreativitas dalam merancang antarmuka pengguna dan meningkatkan fungsionalitas aplikasi menggunakan fitur-fitur yang disediakan oleh PyQt.

g. Menyajikan Hasil Praktikum dengan Jelas:

   Mampu menyajikan hasil praktikum secara sistematis dan jelas melalui laporan yang mencakup langkah-langkah yang diambil, kendala yang dihadapi, dan solusi yang diterapkan.

1.3 Manfaat

a. Pemahaman Mendalam tentang PyQt:

   Menyusun laporan memerlukan pemahaman mendalam tentang konsep-konsep PyQt. Proses penulisan membantu peserta memproses dan memahami secara lebih baik bagaimana PyQt berfungsi dan bagaimana mengimplementasikannya dalam pembuatan aplikasi.

b. Kemampuan Memecahkan Masalah (Debugging):

   Ketika peserta menemui kesalahan atau bug dalam pengembangan aplikasi PyQt, menyusun laporan memerlukan kemampuan untuk menganalisis dan memecahkan masalah tersebut. Ini dapat meningkatkan keterampilan peserta dalam debugging.

c. Pengembangan Keterampilan Desain Antarmuka:

   Menyusun laporan mengharuskan peserta untuk merinci bagaimana mereka merancang antarmuka pengguna menggunakan PyQt. Ini membantu dalam mengembangkan keterampilan desain antarmuka yang efektif dan menarik.

d. Peningkatan Keterampilan Pemrograman Python:

   Penggunaan PyQt membutuhkan pemahaman yang kuat tentang bahasa pemrograman Python. Laporan praktikum membantu peserta memperdalam pemahaman mereka tentang Python, terutama dalam konteks pengembangan aplikasi desktop.

e. Memfasilitasi Kolaborasi Tim:

   Dalam situasi di mana proyek pengembangan melibatkan tim, laporan praktikum dapat berfungsi sebagai dokumen panduan dan rujukan bersama. Ini membantu dalam kolaborasi efektif antaranggota tim.

BAB II
LANDASAN TEORI

2.1 Definisi Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error.

2.2 Definisi PyQt

PyQt adalah jembatan yang secara mulus mengintegrasikan kerangka kerja lintas platform Qt C++ yang kuat dengan bahasa pemrograman Python yang fleksibel, yang terutama berfungsi sebagai modul GUI yang kuat .

Qt lebih dari sekadar perangkat GUI. Ini mencakup berbagai fungsi, menawarkan abstraksi untuk elemen seperti soket jaringan, thread, Unicode, SQL, database, SVG, OpenGL, dan XML . Khususnya, ia juga memiliki browser web internal dan sistem layanan canggih, yang didukung oleh banyak koleksi widget GUI.

Atribut penting dari Qt adalah mekanisme sinyal dan slotnya, yang merupakan inti dari fungsi kelasnya. Mekanisme ini menyederhanakan komunikasi antar objek, menyederhanakan pembuatan komponen perangkat lunak yang dapat digunakan kembali.

Selain itu, Qt dilengkapi dengan Qt Designer , alat intuitif untuk membuat antarmuka pengguna grafis. Hubungan sinergis terbentuk ketika PyQt dipasangkan dengan Qt Designer, karena PyQt dapat dengan mudah mengubah desain menjadi kode

Python. Persatuan ini semakin memperluas kapasitas untuk memperkenalkan kontrol GUI baru.

2.3 Qt Designer

Qt Designer adalah alat pengembangan grafis yang disediakan oleh Qt untuk merancang antarmuka pengguna secara visual. Qt Designer memungkinkan pengembang membuat desain antarmuka dengan cara yang intuitif tanpa menulis kode secara manual. Antarmuka yang dirancang dengan Qt Designer dapat diintegrasikan dengan proyek Qt dan diimplementasikan dengan menggunakan PyQt atau Qt untuk C++.

Berikut adalah beberapa fitur utama dari Qt Designer:

1. WYSIWYG (What You See Is What You Get): Qt Designer menyediakan tampilan langsung dari antarmuka pengguna yang sedang dirancang. Ini memungkinkan pengembang melihat dan mengedit antarmuka secara real-time sesuai dengan yang akan dilihat pengguna.
2. Widget Drag-and-Drop: Pengembang dapat menambahkan widget ke antarmuka dengan menyeret dan menjatuhkannya ke area desain. Ini menciptakan proses desain yang lebih cepat dan lebih intuitif.
3. Properti dan Gaya: Qt Designer menyediakan panel properti yang memungkinkan pengembang mengatur properti dan gaya widget dengan mudah. Ini termasuk warna, teks, ukuran, dan atribut lainnya.
4. Kustomisasi: Pengembang dapat membuat dan menyesuaikan widget khusus mereka sendiri untuk digunakan dalam antarmuka.
5. Integrasi dengan Kode: Setelah desain selesai, Qt Designer memungkinkan pengguna untuk menyimpan desain sebagai file XML dengan ekstensi .ui. File ini dapat diimpor dan diintegrasikan langsung ke dalam proyek PyQt, yang kemudian dapat dikonversi menjadi kode Python.
6. Dukungan untuk Internasionalisasi: Qt Designer menyediakan dukungan untuk pengembangan aplikasi multibahasa dan multikultural dengan mengintegrasikan fitur internasionalisasi langsung ke dalam antarmuka pengguna yang dirancang.

7. Konektivitas dengan Kode:* Qt Designer memungkinkan pengembang menentukan koneksi antara elemen antarmuka dan kode Python yang sesuai. Ini membuatnya mudah untuk menanggapi peristiwa seperti klik tombol atau perubahan nilai.

Penggunaan Qt Designer membantu mempercepat proses pengembangan antarmuka pengguna dengan menyediakan alat visual yang kuat dan intuitif untuk merancang aplikasi yang menggunakan framework Qt.

BAB III
PEMBAHASAN

3.1 Metode Praktikum

1. Persiapan Awal
    - Instalasi PyQt

Untuk menginstal PyQt di Windows,kita bisa menggunakan paket installer dari websiteqt-project.

Untuk Python versi 3.3 atau versi terbaru,kita bisa menginstall semua package pyqt dengan cata mencentang pada box features saat kita menginstall python.

Untuk python versi 3.6 keatas bisa menggunakan perintah dibawah ini:

"pip install pyqt5"

- Instal Qt Designer

- Unduh Qt Online Installer:

  Kunjungi situs unduhan Qt.

  Pilih opsi "Go open source" untuk versi open source atau pilih opsi sesuai kebutuhan.

  Unduh Qt Online Installer yang sesuai dengan sistem operasi dan arsitektur Anda (32-bit atau 64-bit).
- Jalankan Qt Online Installer:

  Buka file installer yang baru diunduh.

  Pilih opsi "MinGW" sebagai kompiler jika Anda tidak memiliki kompiler C++ lainnya yang sudah terinstal.
- Pilih Komponen yang Akan Diinstal:

  Pilih opsi "Desktop Qt" dan pastikan bahwa "Qt Designer" termasuk di dalamnya.

  Anda juga dapat memilih komponen lain yang mungkin Anda butuhkan.
- Pilih Lokasi Instalasi:

  Pilih lokasi di mana Anda ingin menginstal Qt. Pastikan untuk mengingat lokasi ini karena Anda akan membutuhkannya nanti.
- Pilih Lisensi:

  Setujui persyaratan lisensi untuk melanjutkan proses instalasi.

- Proses Instalasi:

  Klik tombol "Next" atau "Install" untuk memulai proses instalasi.

  Tunggu hingga proses instalasi selesai.
- Selesaikan Instalasi:

  Setelah instalasi selesai, klik tombol "Finish".
- Verifikasi Instalasi:

  Buka menu Start dan cari "Qt" atau "Qt Designer". Jalankan Qt Designer untuk memastikan bahwa instalasi berhasil.

2. Desain Antarmuka Kalkulator

   Penggunaan Qt Designer:

   Buka Qt Designer dan desain antarmuka kalkulator.

a. Tambahkan Widget PushButton:
   - Tambahkan tombol untuk angka 0 hingga 9.
   - Tambahkan tombol untuk operasi matematika: +, -, *, /.
   - Tambahkan tombol untuk %
   - Tambahkan tombol untuk hasil (=).
   - Tambahkan tombol untuk menghapus satu karakter (backspace) dan menghapus semua (clear).

b. Tambahkan Line Edit:

   Tambahkan Line Edit untuk menampilkan input dan hasil perhitungan.

c. Atur Layout:

   Atur layout secara logis, misalnya, susunan tombol seperti pada kalkulator biasa.

3. Ekspor Desain ke Kode Python:

   Setelah selesai merancang antarmuka di Qt Designer, ekspor desain tersebut ke dalam kode Python:
   - Pilih File -> Save As untuk menyimpan desain Anda sebagai file .ui.
   - Jalankan perintah berikut untuk mengonversi file .ui menjadi file .py:

     “pyuic5 your_design.ui -o your”

     Gantilah *your_design.ui* dengan nama file desain Anda.

4. Implementasi Logika Kalkulator

- Struktur Proyek:

  Buka file Python yang dihasilkan dari ekspor desain. Implementasikan logika kalkulator di dalamnya. Gunakan modul PyQt seperti QMainWindow, QLineEdit, dan QPushButton untuk mengaitkan fungsi-fungsi yang sesuai dengan tombol dan widget antarmuka.

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(339, 397)
        MainWindow.setStyleSheet("background-color: rgb(0, 0, 0);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setEnabled(True)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.gridLayoutWidget.setGeometry(QtCore.QRect(10, 20, 320, 331))
        self.gridLayoutWidget.setObjectName("gridLayoutWidget")
        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.gridLayout.setObjectName("gridLayout")
        self.p9 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("9") )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p9.sizePolicy().hasHeightForWidth())
        self.p9.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p9.setFont(font)
        self.p9.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p9.setObjectName("p9")
        self.gridLayout.addWidget(self.p9, 2, 2, 1, 1)
        self.mod = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("%") )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.mod.sizePolicy().hasHeightForWidth())
        self.mod.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
```

```
        self.mod.setFont(font)
        self.mod.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(94, 255, 250);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.mod.setObjectName("mod")
        self.gridLayout.addWidget(self.mod, 1, 0, 1, 1)
        self.p1 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("1") )
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p1.sizePolicy().hasHeightForWidth())
        self.p1.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p1.setFont(font)
        self.p1.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p1.setObjectName("p1")
        self.gridLayout.addWidget(self.p1, 4, 0, 1, 1)
        self.delete_2 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.remove_it())
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.delete_2.sizePolicy().hasHeightForWidth())
        self.delete_2.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.delete_2.setFont(font)
        self.delete_2.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(94, 255, 250);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.delete_2.setObjectName("delete_2")
        self.gridLayout.addWidget(self.delete_2, 1, 2, 1, 1)
        self.C = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("C"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.C.sizePolicy().hasHeightForWidth())
        self.C.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
```

```
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.C.setFont(font)
        self.C.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(94, 255, 250);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.C.setObjectName("C")
        self.gridLayout.addWidget(self.C, 1, 1, 1, 1)
        self.kurang = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("-"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.kurang.sizePolicy().hasHeightForWidth())
        self.kurang.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.kurang.setFont(font)
        self.kurang.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(255, 130, 232);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.kurang.setObjectName("kurang")
        self.gridLayout.addWidget(self.kurang, 2, 3, 1, 1)
        self.tambah = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("+"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.tambah.sizePolicy().hasHeightForWidth())
        self.tambah.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.tambah.setFont(font)
        self.tambah.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(94, 255, 250);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.tambah.setObjectName("tambah")
        self.gridLayout.addWidget(self.tambah, 1, 3, 1, 1)
        self.koma = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.dot_it())
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
```

```
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.koma.sizePolicy().hasHeightForWidth())
        self.koma.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.koma.setFont(font)
        self.koma.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.koma.setObjectName("koma")
        self.gridLayout.addWidget(self.koma, 5, 2, 1, 1)
        self.bagi = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("/"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.bagi.sizePolicy().hasHeightForWidth())
        self.bagi.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.bagi.setFont(font)
        self.bagi.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(255, 130, 232);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.bagi.setObjectName("bagi")
        self.gridLayout.addWidget(self.bagi, 4, 3, 1, 1)
        self.p7 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("7"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p7.sizePolicy().hasHeightForWidth())
        self.p7.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p7.setFont(font)
        self.p7.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p7.setObjectName("p7")
        self.gridLayout.addWidget(self.p7, 2, 0, 1, 1)
```

```
        self.p0 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("0"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p0.sizePolicy().hasHeightForWidth())
        self.p0.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p0.setFont(font)
        self.p0.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.p0.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p0.setObjectName("p0")
        self.gridLayout.addWidget(self.p0, 5, 0, 1, 2)
        self.p4 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("4"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p4.sizePolicy().hasHeightForWidth())
        self.p4.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p4.setFont(font)
        self.p4.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p4.setObjectName("p4")
        self.gridLayout.addWidget(self.p4, 3, 0, 1, 1)
        self.p3 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("3"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p3.sizePolicy().hasHeightForWidth())
        self.p3.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p3.setFont(font)
        self.p3.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
```

```
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p3.setObjectName("p3")
        self.gridLayout.addWidget(self.p3, 4, 2, 1, 1)
        self.p6 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("6"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p6.sizePolicy().hasHeightForWidth())
        self.p6.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p6.setFont(font)
        self.p6.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p6.setObjectName("p6")
        self.gridLayout.addWidget(self.p6, 3, 2, 1, 1)
        self.p2 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("2"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p2.sizePolicy().hasHeightForWidth())
        self.p2.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p2.setFont(font)
        self.p2.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p2.setObjectName("p2")
        self.gridLayout.addWidget(self.p2, 4, 1, 1, 1)
        self.p5 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("5"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p5.sizePolicy().hasHeightForWidth())
        self.p5.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p5.setFont(font)
        self.p5.setStyleSheet("QPushButton{\n"
```

```
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p5.setObjectName("p5")
        self.gridLayout.addWidget(self.p5, 3, 1, 1, 1)
        self.p8 = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("8"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.p8.sizePolicy().hasHeightForWidth())
        self.p8.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.p8.setFont(font)
        self.p8.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    background-color : rgb(166, 166, 166);\n"
"    border-radius : 30px;\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.p8.setObjectName("p8")
        self.gridLayout.addWidget(self.p8, 2, 1, 1, 1)
        self.kali = QtWidgets.QPushButton(self.gridLayoutWidget, clicked = lambda:
self.press_it("*"))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.kali.sizePolicy().hasHeightForWidth())
        self.kali.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.kali.setFont(font)
        self.kali.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(255, 130, 232);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.kali.setObjectName("kali")
        self.gridLayout.addWidget(self.kali, 3, 3, 1, 1)
        self.sama_dengan = QtWidgets.QPushButton(self.gridLayoutWidget, clicked =
lambda: self.equal_it())
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(14)
        sizePolicy.setHeightForWidth(self.sama_dengan.sizePolicy().hasHeightForWidth()
)
        self.sama_dengan.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
```

```
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.sama_dengan.setFont(font)
        self.sama_dengan.setStyleSheet("QPushButton{\n"
"    color : rgb(0, 0, 0);\n"
"    border-radius : 30px;\n"
"    background-color: rgb(255, 130, 232);\n"
"}\n"
"QPushButton:Hover{\n"
"    background-color: rgb(255, 255, 255);\n"
"}\n"
"")
        self.sama_dengan.setObjectName("sama_dengan")
        self.gridLayout.addWidget(self.sama_dengan, 5, 3, 1, 1)
        self.label = QtWidgets.QLabel(self.gridLayoutWidget)
        self.label.setEnabled(True)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(20)
        sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
        self.label.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setStyleSheet("color: rgb(255, 255, 255);")
        self.label.setFrameShape(QtWidgets.QFrame.Box)
        self.label.setFrameShadow(QtWidgets.QFrame.Raised)
        self.label.setLineWidth(2)
        self.label.setAlignment(QtCore.Qt.AlignLeft|QtCore.Qt.AlignLeading|QtCore.Qt.A
lignVCenter)
        self.label.setObjectName("label")
        self.gridLayout.addWidget(self.label, 0, 0, 1, 4)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 339, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

5. Manajemen Event dan Responsifitas Antarmuka

- Penanganan Peristiwa (Event Handling):

Implementasikan penanganan peristiwa untuk tombol-tombol kalkulator, seperti tombol angka dan operasi matematika.

```
#fungsi sama dengan
    def equal_it(self):
        screen = self.label.text()
        answer = eval(screen)
        self.label.setText(str(answer))


#Menghapus karakter
```

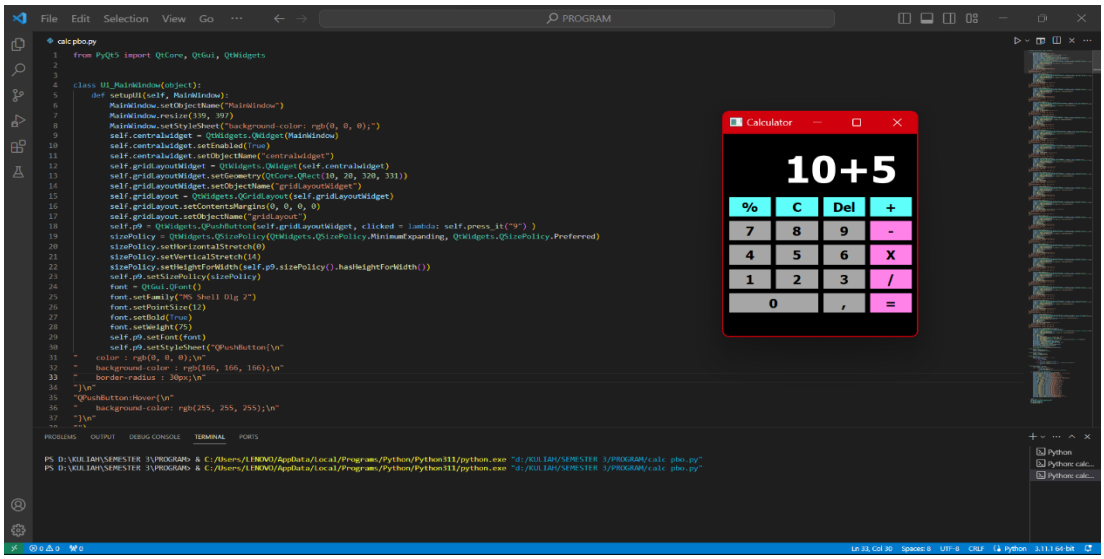
```
    def remove_it(self):
        screen = self.label.text()
        screen = screen [:-1]
        self.label.setText(screen)


#Membuat Decimal
    def dot_it(self):
        screen = self.label.text()
        signs_tuple = ("/", "x", "+", "-","%")
        for i in signs_tuple:
            if i not in screen:
                if "." in screen:
                    pass
                else:
                    self.label.setText(f"{screen}.")
            else:
                if screen[-1].isnumeric() and "." not in screen[screen.rfind(i):]:
                    self.label.setText(f"{screen}.")
                else:
                    pass

#Biar bisa ditekan
    def press_it(self, pressed):
        if pressed == "C":
                self.label.setText('0')
        else:
                if self.label.text() == "0":
                   self.label.setText('')
                self.label.setText(f'{self.label.text()}{pressed}')


    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Calculator"))
        self.p9.setText(_translate("MainWindow", "9"))
        self.p9.setShortcut(_translate("MainWindow", "9"))
        self.mod.setText(_translate("MainWindow", "%"))
        self.mod.setShortcut(_translate("MainWindow", "%"))
        self.p1.setText(_translate("MainWindow", "1"))
        self.p1.setShortcut(_translate("MainWindow", "1"))
        self.delete_2.setText(_translate("MainWindow", "Del"))
        self.delete_2.setShortcut(_translate("MainWindow", "Backspace"))
        self.C.setText(_translate("MainWindow", "C"))
        self.C.setShortcut(_translate("MainWindow", "C"))
        self.kurang.setText(_translate("MainWindow", "-"))
        self.kurang.setShortcut(_translate("MainWindow", "-"))
        self.tambah.setText(_translate("MainWindow", "+"))
        self.tambah.setShortcut(_translate("MainWindow", "+"))
        self.koma.setText(_translate("MainWindow", ","))
        self.koma.setShortcut(_translate("MainWindow", ","))
        self.bagi.setText(_translate("MainWindow", "/"))
        self.bagi.setShortcut(_translate("MainWindow", "/"))
        self.p7.setText(_translate("MainWindow", "7"))
        self.p7.setShortcut(_translate("MainWindow", "7"))
        self.p0.setText(_translate("MainWindow", "0"))
        self.p0.setShortcut(_translate("MainWindow", "0"))
        self.p4.setText(_translate("MainWindow", "4"))
        self.p4.setShortcut(_translate("MainWindow", "4"))
        self.p3.setText(_translate("MainWindow", "3"))
        self.p3.setShortcut(_translate("MainWindow", "3"))
        self.p6.setText(_translate("MainWindow", "6"))
        self.p6.setShortcut(_translate("MainWindow", "6"))
        self.p2.setText(_translate("MainWindow", "2"))
        self.p2.setShortcut(_translate("MainWindow", "2"))
        self.p5.setText(_translate("MainWindow", "5"))
        self.p5.setShortcut(_translate("MainWindow", "5"))
        self.p8.setText(_translate("MainWindow", "8"))
        self.p8.setShortcut(_translate("MainWindow", "8"))
        self.kali.setText(_translate("MainWindow", "X"))
        self.kali.setShortcut(_translate("MainWindow", "*"))
```

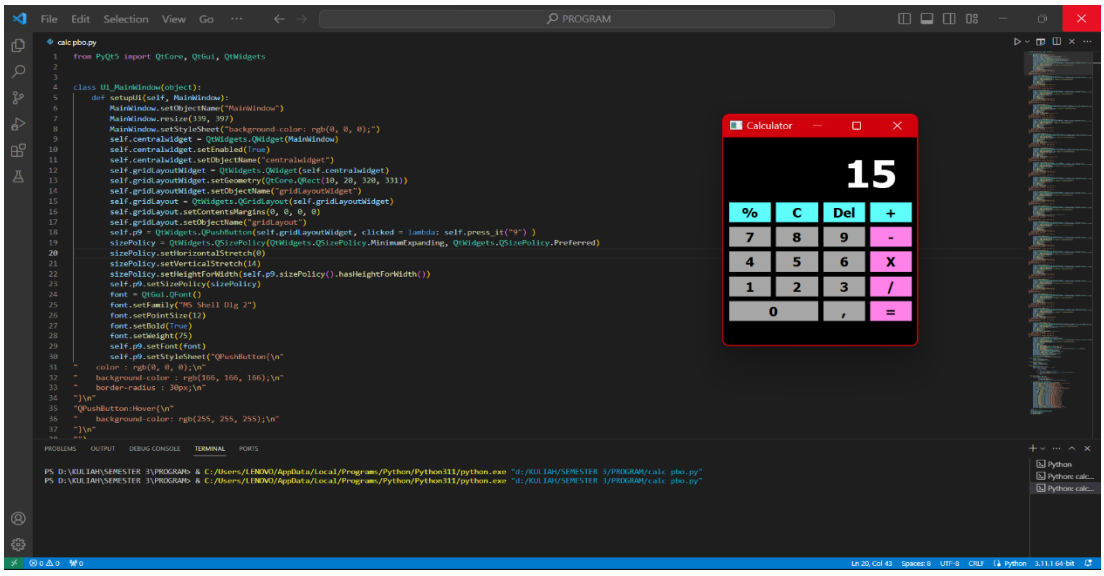
```
        self.sama_dengan.setText(_translate("MainWindow", "="))
        self.sama_dengan.setShortcut(_translate("MainWindow", "="))
        self.label.setText(_translate("MainWindow", "0"))


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

3.2 Uji Coba Aplikasi Kalkulator

- Uji Fungsionalitas:

Uji fungsi kalkulator untuk memastikan semua operasi matematika berfungsi dengan benar.





3.3 Kendala yang dihadapi

Membuat program kalkulator menggunakan PyQt dan Qt Designer dapat memberikan beberapa tantangan. Berikut adalah beberapa kendala umum yang mungkin dihadapi beserta solusi atau tips untuk mengatasi mereka:

1. Ketidakfahaman Penggunaan Qt Designer:

   Solusi: Qt Designer adalah alat desain grafis untuk membuat antarmuka pengguna dengan mudah. Bila Anda kesulitan menggunakan Qt Designer, Anda bisa memeriksa tutorial atau dokumentasi resmi PyQt. Terdapat banyak sumber daya di luar sana yang dapat membantu Anda memahami cara menggunakan Qt Designer.

2. Integrasi Kode Python dengan Qt Designer:

   Solusi: Setelah Anda mendesain antarmuka pengguna di Qt Designer, Anda perlu mengintegrasikannya dengan kode Python. Pastikan Anda memahami cara memuat UI yang telah Anda desain ke dalam program Python Anda menggunakan QUiLoader atau metode serupa. Perhatikan juga untuk menetapkan fungsi yang akan dipanggil ketika tombol di antarmuka diklik.

3. Manajemen Sinyal dan Slot:

   Solusi: PyQt menggunakan konsep sinyal dan slot untuk menghubungkan tindakan pengguna (seperti mengklik tombol) dengan fungsi yang harus dijalankan. Pastikan Anda memahami konsep ini dan mengonfigurasi koneksi yang diperlukan di dalam kode Python Anda.

4. Kesulitan dalam Logika Kalkulator:

   Solusi: Implementasi logika kalkulator dapat menjadi bagian yang rumit. Pastikan Anda memisahkan tata letak antarmuka pengguna dari logika kalkulator. Gunakan metode yang didefinisikan dengan baik untuk setiap operasi kalkulator.

5. Debugging Kode:

   Solusi: Gunakan alat debugging Python seperti pdb atau tambahkan pernyataan cetak untuk membantu melacak bug dalam kode Anda. Juga, pastikan Anda memahami pesan kesalahan yang dihasilkan oleh PyQt.

6. Pemeliharaan Kode dan Perubahan Desain:

   Solusi: Jangan ragu untuk merombak atau memperbarui kode Anda jika diperlukan. Sesuaikan kode Python Anda jika ada perubahan dalam desain antarmuka pengguna atau jika Anda menemui masalah yang perlu diperbaiki.

BAB IV
PENUTUP

3.1 Kesimpulan

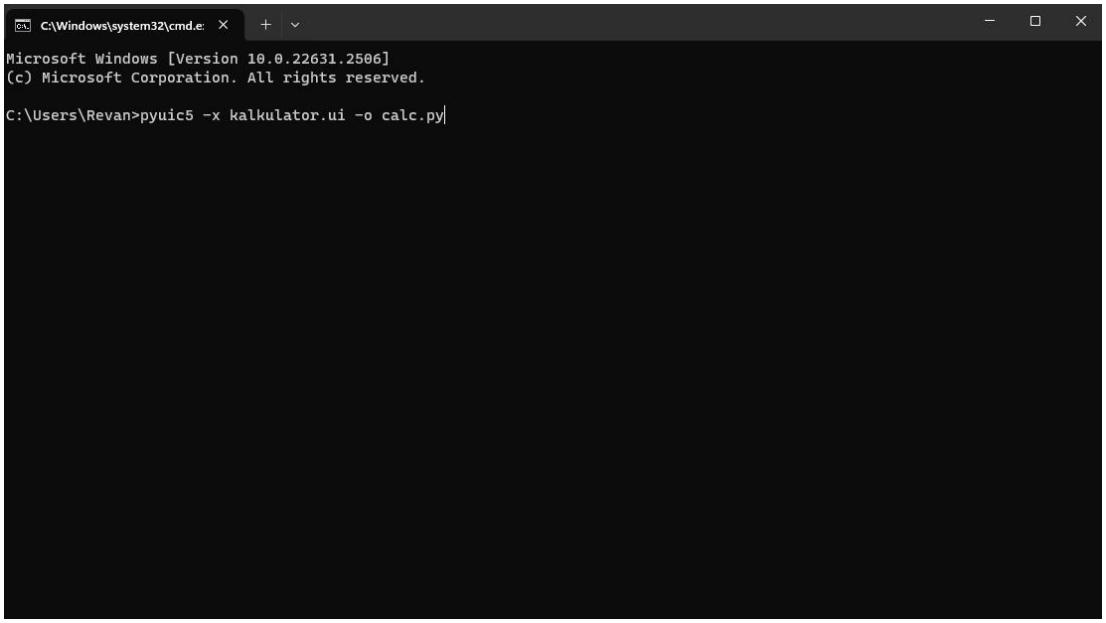
Pada praktikum ini, kami berhasil merancang dan mengimplementasikan kalkulator sederhana menggunakan PyQt dan Qt Designer. Dalam proses pengembangan, kami mengalami beberapa tantangan, namun berhasil mengatasi mereka dengan mencari solusi yang sesuai. Praktikum ini juga memberikan wawasan yang berharga dalam memahami kerja sama antara Qt Designer dan PyQt untuk pengembangan aplikasi. Kami merasa lebih siap untuk menjelajahi proyek-proyek yang lebih besar dan kompleks di masa depan. Kesempatan ini memperkaya pemahaman kami tentang pembuatan antarmuka pengguna dan logika aplikasi, mempersiapkan kami untuk tantangan di dunia pengembangan perangkat lunak.Keseluruhan, praktikum ini berhasil mengintegrasikan konsep desain dan implementasi logika program dengan baik, dan kami optimis untuk menerapkan keterampilan yang diperoleh ke dalam proyek-proyek berikutnya.

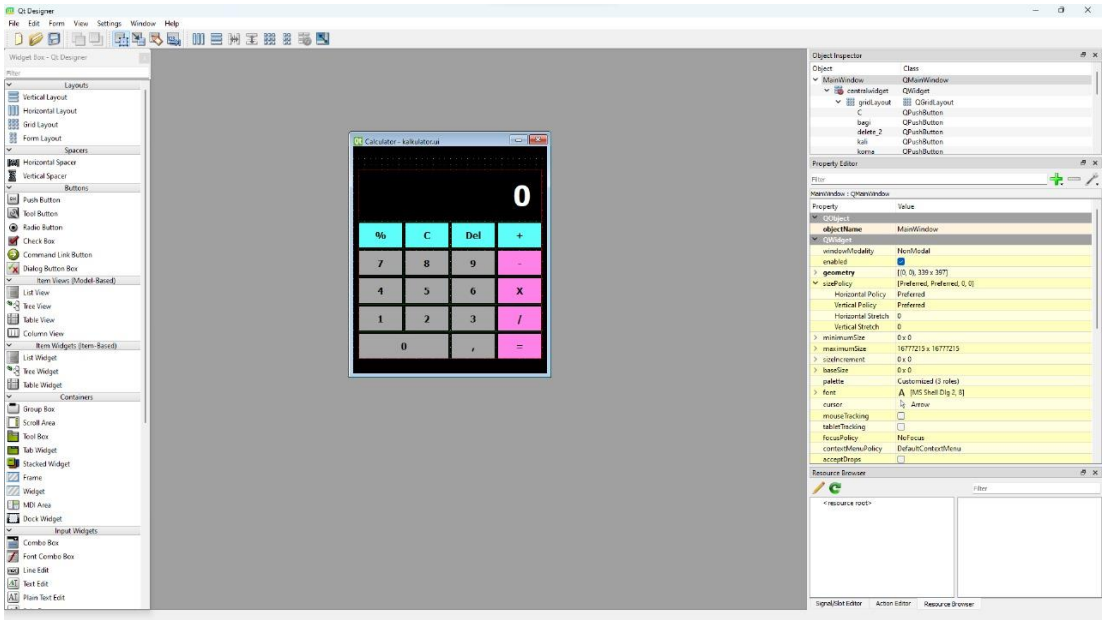
DAFTAR PUSTAKA

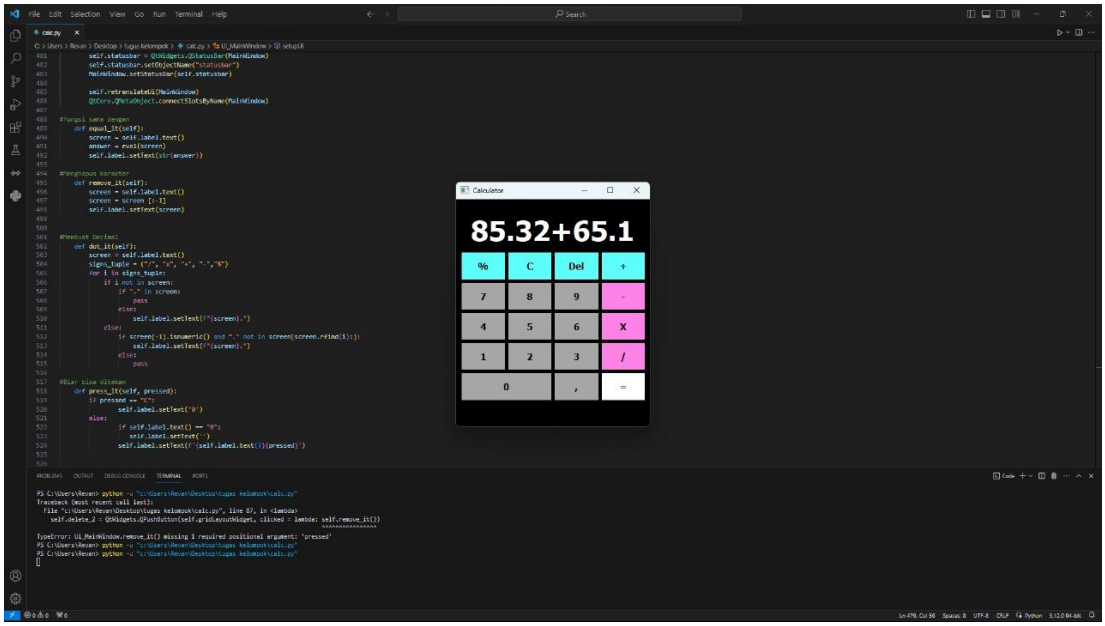
Definisi Python Pendahuluan Python - Belajarpython – Situs Open Source Tutorial Pemrograman Python Bahasa Indonesia

Konsep Dasar Qt Designer https://doc.qt.io/qt-6/q tdesigner-manual.html

LAMPIRAN


C:\Windows\system32\cmd.e:
Microsoft Windows [Version 10.0.22631.2506]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Revan>pyuic5 -x kalkulator.ui -o calc.py


Qt Designer
Calculator - kalkulator.ui
0
%
C
Del
+
7
8
9
-
4
5
6
X
1
2
3
/
0
,
=


Calculator
85.32+65.1
%
C
Del
+
7
8
9
-
4
5
6
X
1
2
3
/
0
,
=