```
# 2024 Â© Idan Hazay
# Import libraries

from PyQt6.QtWidgets import QApplication, QDialog, QVBoxLayout, QLabel, QTextEdit, QPushButton, QHBoxLayout
from PyQt6.QtGui import QPixmap, QIcon
from PyQt6.QtCore import Qt

import os
from docx import Document


class FileViewer:
    """Displays files (text, images, and documents) in a PyQt dialog."""
    def __init__(self, file_path, title):
        self.file_path = file_path
        self.title = title
        self.file_viewer_dialog()

    def open_in_native_app(self):
        """Opens the file with the system's default application."""
        try:
            os.startfile(self.file_path)
        except Exception as e:
            print(f"Error opening file in native app: {e}")

    def file_viewer_dialog(self):
        """Creates and displays a file viewer dialog based on file type."""
        app = QApplication.instance()  # Get existing QApplication instance
        if app is None:
            app = QApplication([])  # Create a new instance if needed

        file_extension = os.path.splitext(self.file_path)[1].lower()
        dialog = QDialog()
        dialog.setStyleSheet("font-size:15px;")
        layout = QVBoxLayout()
        dialog.resize(600, 400)

        # Set window icon if available
        icon_path = f"{os.path.dirname(os.path.dirname(os.path.abspath(__file__)))}/assets/icon.ico"
        if os.path.isfile(icon_path):
            dialog.setWindowIcon(QIcon(icon_path))

        dialog.setWindowTitle(self.title)

        close_button = QPushButton('Close', dialog)
        close_button.clicked.connect(dialog.close)
        layout.addWidget(close_button)

        content_widget = None

        # Display images
        if file_extension in ['.jpg', '.jpeg', '.png', '.bmp', '.gif']:
            content_widget = QLabel(dialog)
            pixmap = QPixmap(self.file_path)
            content_widget.setPixmap(pixmap.scaled(600, 800, Qt.AspectRatioMode.KeepAspectRatio))  # Maintain aspect ratio
            layout.insertWidget(0, content_widget)

        # Display .docx files
        elif file_extension == '.docx':
            content_widget = QTextEdit(dialog)
            content_widget.setReadOnly(True)
            try:
                doc = Document(self.file_path)
                full_text = '\n'.join([paragraph.text for paragraph in doc.paragraphs])  # Extract text from all
paragraphs
                content_widget.setPlainText(full_text)
            except Exception as e:
                content_widget.setPlainText(f"Error reading document: {str(e)}")
            layout.insertWidget(0, content_widget)

        else:
            # Attempt to open unsupported file types as plain text
            try:
                with open(self.file_path, 'r', encoding='utf-8') as f:
                    content = f.read()
                content_widget = QTextEdit(dialog)
                content_widget.setReadOnly(True)
                content_widget.setPlainText(content)
                layout.insertWidget(0, content_widget)

            except Exception as e:
                # If opening as text fails, display a fallback message
                content_widget = QLabel(dialog)
                content_widget.setText(f"Cannot open {os.path.basename(self.file_path)[5:]} in file viewer.\nTry opening
it in its default app.")
                content_widget.setStyleSheet("font-size: 20px")
                content_widget.setAlignment(Qt.AlignmentFlag.AlignCenter)
        layout.insertWidget(0, content_widget)

        # Add a button to open in the default system application
        open_native_button = QPushButton(f"Open {os.path.splitext(self.file_path)[1]} in default app", dialog)
        open_native_button.clicked.connect(self.open_in_native_app)
        layout.addWidget(open_native_button)

dialog.setLayout(layout)
dialog.exec()
```