```python
#utils.py

# coding: utf-8

import os
import config
from PyQt5 import QtWidgets, QtCore, QtGui
import json


class AbstractFunction(object):
    def move_to_center(self):
        qtRectangle = self.frameGeometry()
        centerPoint = QtWidgets.QDesktopWidget().availableGeometry().center()
        qtRectangle.moveCenter(centerPoint)
        self.move(qtRectangle.topLeft())

    @classmethod
    def show_warning_message(
        cls,
        message: str,
        title: str = "Warning",
        detail: str = None,
        extra: str = None,
        parent=None,
        only_yes: bool = False,
    ):
        """show warning msg"""
        msg_box = QtWidgets.QMessageBox(parent=parent)
        msg_box.setIcon(QtWidgets.QMessageBox.Warning)
        msg_box.setText(message)
        msg_box.setWindowTitle(title)
        if isinstance(extra, str) and detail:
            msg_box.setInformativeText(extra)
        if isinstance(detail, str) and detail:
            msg_box.setDetailedText(detail)
        if only_yes is True:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
        else:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes | 
QtWidgets.QMessageBox.No)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
            btn_no = msg_box.button(QtWidgets.QMessageBox.No)
            btn_no.setText("Ignore")

        msg_box.setDefaultButton(QtWidgets.QMessageBox.Yes)
        msg_box.setEscapeButton(QtWidgets.QMessageBox.No)
        msg_box.setTextInteractionFlags(QtCore.Qt.TextSelectableByMouse)  # 
QtCore.Qt.NoTextInteraction
        r = msg_box.exec_()
        if r == QtWidgets.QMessageBox.Yes:
            return True
        else:
            return False

    @classmethod
    def show_info_message(
        cls,
        message: str,
        title="Notification",
        detail: str = None,
        extra: str = None,
        parent=None,
        only_yes: bool = False,
    ):
        """show notification msg"""
        msg_box = QtWidgets.QMessageBox(parent=parent)
        msg_box.setIcon(QtWidgets.QMessageBox.Information)
        msg_box.setText(message)
        msg_box.setWindowTitle(title)
        if isinstance(extra, str) and detail:
            msg_box.setInformativeText(extra)
        if isinstance(detail, str) and detail:
            msg_box.setDetailedText(detail)
        if only_yes is True:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
        else:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
            btn_no = msg_box.button(QtWidgets.QMessageBox.No)
            btn_no.setText("No")

        msg_box.setDefaultButton(QtWidgets.QMessageBox.Yes)
        msg_box.setEscapeButton(QtWidgets.QMessageBox.No)
        msg_box.setTextInteractionFlags(QtCore.Qt.TextSelectableByMouse)  # QtCore.Qt.NoTextInteraction
        r = msg_box.exec_()
        if r == QtWidgets.QMessageBox.Yes:
            return True
        else:
            return False

    @classmethod
    def get_last_directory(cls):
        data = cls.__load_default_config()
        return data.get("last_dir", config.base_dir)

    @classmethod
    def save_last_directory(cls, dir_path):
        if not isinstance(dir_path, str):
            raise TypeError
        if not os.path.exists(dir_path):
            raise FileNotFoundError
        if not os.path.isdir(dir_path):
            raise ValueError
        data = cls.__load_default_config()
        data["last_dir"] = dir_path
        cls.__save_default_config(data=data)

    @classmethod
    def __save_default_config(cls, data: dict):
        if not isinstance(data, dict):
            raise TypeError
        with open(config.app_config_fp, "w", encoding="utf-8") as f:
            json.dump(data, f)

    @classmethod
    def __load_default_config(cls):
        data = {}
        if not os.path.exists(config.app_config_fp):
            return data
        if not os.path.isfile(config.app_config_fp):
            return data
        with open(config.app_config_fp, "r", encoding="utf-8") as f:
            data = json.load(f)
        return data

```