7940 Lab3-4

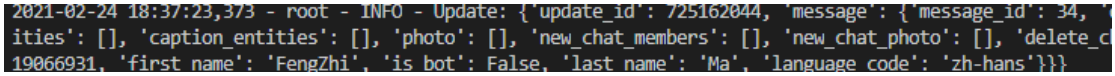
STN: MaFengzhi STO: 20401043

Please describe the architecture of the current chatbot system. Identify the components and check where are they running now.

For its back-end: First, we need to install the python packages that we will use by using pip. Then there's a config file, we need to make sections we need, and push the data into it. This file can ensure the data security of the project. Last for the python file, we should import the package we installed before, write down the functions that we will use in def main. Then we will use the functions we wrote before, the functions of the packages, the data we put in the config to make the main function. At last, we write a code to run this python file. When we run this code, it will using API of Telegram to build a back-end of this robot on Telegram's Server.

For its front-end: just the Telegram software on your mobile.

It was running on the sever of Telegram, and here's the data coming back from the sever.



Explain how do your chatbot handle the special command. You need to trace the code and explain that.

```
# register a dispatcher to handle message: here we register an echo dispatcher
echo_handler = MessageHandler(Filters.text & (~Filters.command), echo)
dispatcher.add_handler(echo_handler)
```

At first, we should register a dispatcher to handle the message that the robot will receive. Then we could using dispatcher.add_handler() function to add the handler(contains echo_handler or command handler). So how we create a handler? The next picture shows it.

```
def echo(update, context):
    reply_message = update.message.text.upper()
    logging.info("Update: " + str(update))
    logging.info("context: " + str(context))
    context.bot.send_message(
        chat_id=update.effective_chat.id, text=reply_message)


# Define a few command handlers. These usually take the two arguments update and
# context. Error handlers also receive the raised TelegramError object in error.
def help_command(update: Update, context: CallbackContext) -> None:
    """Send a message when the command /help is issued."""
    update.message.reply_text('Helping you helping you.')
```

The first function is echo_handler's echo function. The second function is help_command function. You can write everything you want to reply in the reply text when the robot receive the /help command.

At last, we should push the command handler function into the dispatcher.

```
# on different commands - answer in Telegram
dispatcher.add_handler(CommandHandler("add", add))
dispatcher.add_handler(CommandHandler("help", help_command))
dispatcher.add_handler(CommandHandler("hello", hello_command))
```

Update your code so that when user type /hello Kevin, it will reply Good day, Kevin!. Write down the change you have made.

```
    dispatcher.add_handler(CommandHandler("hello", hello_command))


def hello_command(update: Update, context: CallbackContext) -> None:
    """Send a message when the command /hello is issued."""
    try:
        logging.info(context.args[0])
        msg = context.args[0]
        update.message.reply_text(
            'Good day, ' + msg + "!")
    except (IndexError, ValueError):
        update.message.reply_text('Usage: /hello <keyword>')
```

Make a few screen caps to prove that you have applied your own Redis account, used it in your chatbot, and push the code on GitHub (at least 2 commits - lab3/lab4).