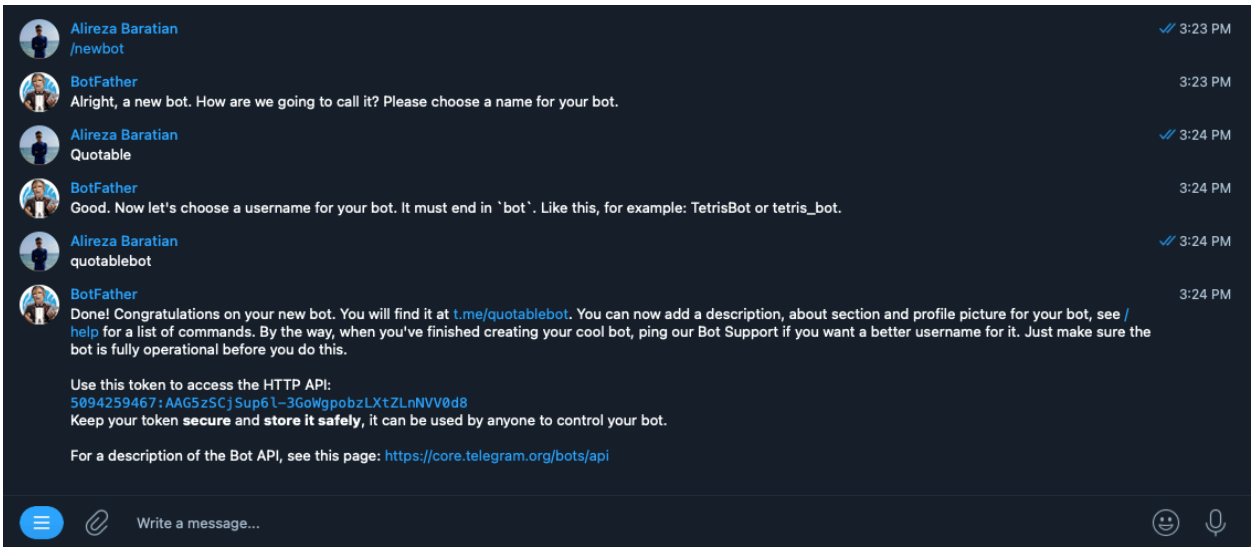
راهنمایی اولیه

ثبت ربات

می‌خواهیم ربات @quotablebot را بسازیم. وظیفه آن نمایش یک نقل قول است.
ابتدا با استفاده از دستور /newbot در @botfather، ربات خودتون رو ایجاد کنید.
اول یک نام نمایشی و بعد یک یوزرنیم برای ربات انتخاب کنید، توجه کنید که یوزرنیم باید با عبارت "bot" تمام شود. مثل تصویر زیر:



در نهایت یک API Token نمایش داده می‌شود که در حقیقت ابزاری برای احراز هویت ربات شماست و در ادامه به آن اشاره می‌کنیم. البته می‌توانید API Token را هر زمان که خواستید با دستور /mybots و انتخاب ربات و منوی Edit bot آن را تغییر دهید.
همچنین برای اضافه کردن تصویر، توضیحات و غیره می‌توانید از دستور /mybots استفاده کنید.

راه اندازی اولیه

همان طور که گفته شد می‌توانیم از کتابخانه python-telegram-bot استفاده کنیم که مستندات استفاده آن کامل و قابل فهم بیان شده است.
ابتدا با استفاده از PIP، پکیج را نصب می‌کنیم:

```
pip install python-telegram-bot
```

توصیه می‌شود حتما از یک محیط مجازی ترجیحا virtualenv برای جلوگیری از مشکل نسخه کتابخانه ها با یکدیگر استفاده کنیم. (در جلسه آخر درس مبانی توضیح داده شد)

نکته: برای انجام این پروژه نیازی به درک مبحث شیء گرایی نیست و تنها این قابلیت‌ها را فراخوانی می کنیم و تنها باید منطق کلی را متوجه شوید، نیازی به دقیق شدن در جزئیات ندارد و اصطلاحات شیءگرایی برای جلوگیری از پیچیدگی اضافی گفته نمی‌شود و سعی می‌شود ساده بیان شود.

طبق این مقاله، ابتدا کلاس Updater از پکیج telegram.ext را وارد می‌کنیم گه وضیفه آن دریافت اطلاعات و دستورات جدید از سمت تلگرام است و به آن برای ورودی token که همان API Token است که در بخش پیش گفتیم.
همچنین با استفاده از قابلیت dispatcher دستورات و پیام‌ها رو شناسایی و مدیریت می‌کنیم:

quotable.py > ...

```python
from telegram.ext import Updater

api_token = "Paste your API Token here"

updater = Updater(token=api_token, use_context=True)
dispatcher = updater.dispatcher
```

به طور کلی ربات‌ها هم پیام را خوانده و هم دستوراتی که با / اغاز می‌شوند که برای هر کدام روش متفاوتی برای پیاده سازی سازی وجود دارد، ابتدا دستور start که اولین دستوری است که هر کاربر به ربات می‌دهد و بعد از آن نحوه پردازش پیام‌ها را پیاده سازی می‌کنیم.

دستور شروع

دستور شروع در واقع هر زمان که کاربری شروع به استفاده از ربات کند، اولین دستوری است که ربات شما دریافت می‌کند. به طور کلی برای دریافت دستورها و انجام کارهای متناظر آن باید برای هر دستور یک تابع تعریف کرده و آن را به dispatcher که در بخش قبلی گفته شد معرفی کنیم، پس ابتدا ماژول Update از پکیج telegram و ماژول CallBackContext را از پکیج telegram.ext وارد برنامه می‌کنیم و هر عملی که می‌خواهیم کاربر بعد از شروع استفاده از ربات برای وی انجام شود را در تابعی با اسم همان دستور که اینجا

start نامیده می‌شود را قرار می‌دهیم. ووردی‌های تابع طبق کد بوده و به عنوان مثال ما تنها در چتی که همان کاربر با ربات ایجاد کرده پیام خوش‌آمد ارسال می‌کنیم:

```python
quotable.py > ...
from telegram import Update
from telegram.ext import CallbackContext

def start(update: Update, context: CallbackContext):
    context.bot.send_message(chat_id=update.effective_chat.id, text="😃 Welcome!")
```

سپس همانطور که گفته شد به کمک ماژول CommandHandler از پکیج telegram.ext تابعی که تعریف کردیم را به دستور /start متصل می‌کنیم. ورودی تابع سازنده CommandHandler() ابتدا عنوان دستور و سپس نام تابعی متناظر آن است. در آخر حاصل را به dispatcher که از قبل ساخته بودیم معرفی می‌کنیم:

```python
quotable.py > ...
from telegram.ext import CommandHandler
start_handler = CommandHandler('start', start)
dispatcher.add_handler(start_handler)
```

تست ربات

حالا کافیه خط زیر رو به کد اضافه کنیم تا ربات ما شروع به ارتباط گرفتن با تلگرام کنه:

```
updater.start_polling()
```

برای خطایابی بهتر می‌توانید با کتابخانه logging که به شکل پیشفرض توسط پایتون نصب شده، یک گزارش از وضعیت و اتفاقات ربات را در کنسول پایتون در زمان اجرا مشاهده کنید:

حالا می‌توانید ربات خود را تست کنید. (البته برای ارتباط کد با تلگرام نیاز به فیلترشکن دارید)

پردازش پیام‌ها

همانطور که گفته شد، به طور کلی ورودی کاربر می‌تواند دستور یا پیام باشد. محتوای پیام هم می‌تواند متنوع باشد مثل متن، ایموجی(نوعی متن)، استیکر، عکس، فیلم، صوت و ...

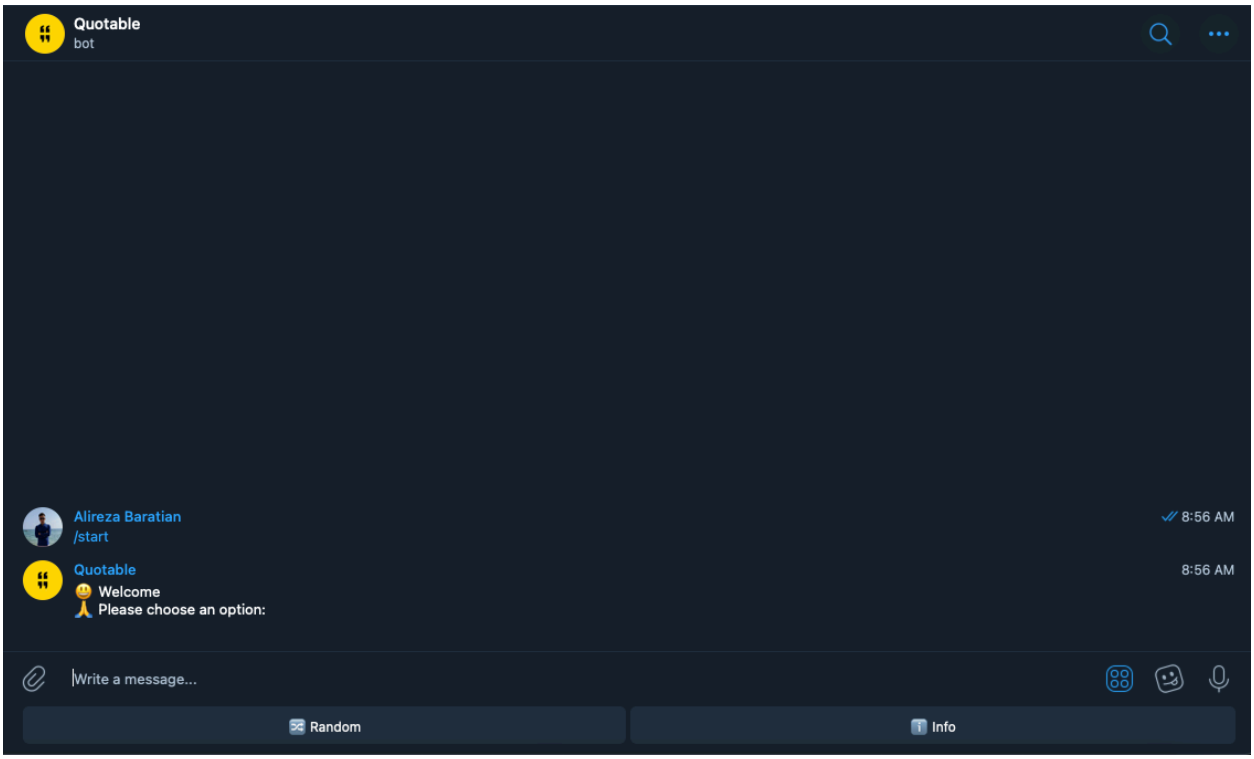
برای پردازش این نوع ورودی از دو ماژول Messagehandler و Filters استفاده می‌شود و برای انجام کار متناظر با پیام ورودی که به عنوان مثال در اینجا شناسایی دستور متنی و فراخوانی تابع مورد نظر است پرداخته و مثل بخش قبلی یک Messagehandler با ورودی (Filters.command) & Filters.text به معنای پیام متنی و شناسایی آن به عنوان دستور استفاده می‌کنیم: (برای Filterهای بیشتر اینجا را مطالعه کنید)

```python
from telegram.ext import MessageHandler, Filters

def parse_input(update: Update, context: CallbackContext):
    context.bot.send_message(chat_id=update.effective_chat.id, text="🙏 Please choose an option")

input_handler = MessageHandler(Filters.text & (~Filters.command), parse_input)
dispatcher.add_handler(input_handler)
```

حالا باید یک منوی کیبوردی برای رابط کاربری بهتر تعریف کنیم که هر زمان شخصی شروع به استفاده از ربات کرد به او لیست قابلیت‌ها و منوی ربات نشان داده شود.
برای این کار کافی است ماژول ReplyKeyboardMarkup از telegram را اضافه کرده و در پیام خروجی به کاربر از آن استفاده کیم. به عنوان مثال در این ربات، با متد update.message.reply_text() ورودی replymarkup را آن کیبورد مورد نظر تعریف می‌کنیم.
طبق مستندات تلگرام هر کیبورد از یک آرایه تو در تو از گزینه‌های کیبورد تشکیل شده است و البته یکسری تنظیمات نمایشی دیگر هم دارد.
در این ربات قصد داریم با گزینه random یک نقل قول تصادفی نمایش داده شده و به عناون مثال گزینه دیگری هم برای اطلاعات درباره ربات با نام info در نظر بگیریم، در واقع تابع start ما به شکل زیر تبدیل می‌شود:

```python
from telegram import ReplyKeyboardMarkup
def start(update: Update, context: CallbackContext):
    keyboard_options = [["🔀 Random", "ℹ️ Info"]]
    update.message.reply_text("😃 Welcome\n🙏 Please choose an option:",
     reply_markup=ReplyKeyboardMarkup(keyboard_options))
```

خروجی:



توجه کنید که حالا باید تابع مربوط به پردازش پیام ارسالی کاربر را متناسب با گزینه انتخاب شده با کیبورد و شرط های مختلف تغییر دهیم چون هر گزینه کیبورد کلیک شود پیام متنی آن به ربات ارسال می شود و باید در بدنه تابع بررسی پیام ورودی بررسی شود و مناسب با آن تابع مروبط به آن ورودی فراخوانی شود.

برای کار با APIها نیز اغلب مواقع به کتابخانه requests نیاز خواهید داشت که در جلسه آخر درس، درباره آن توضیحاتی داده شد. اینجا میخواهیم با دستور random یک نقل قول تصادفی از مسیر
https://api.quotable.io/random

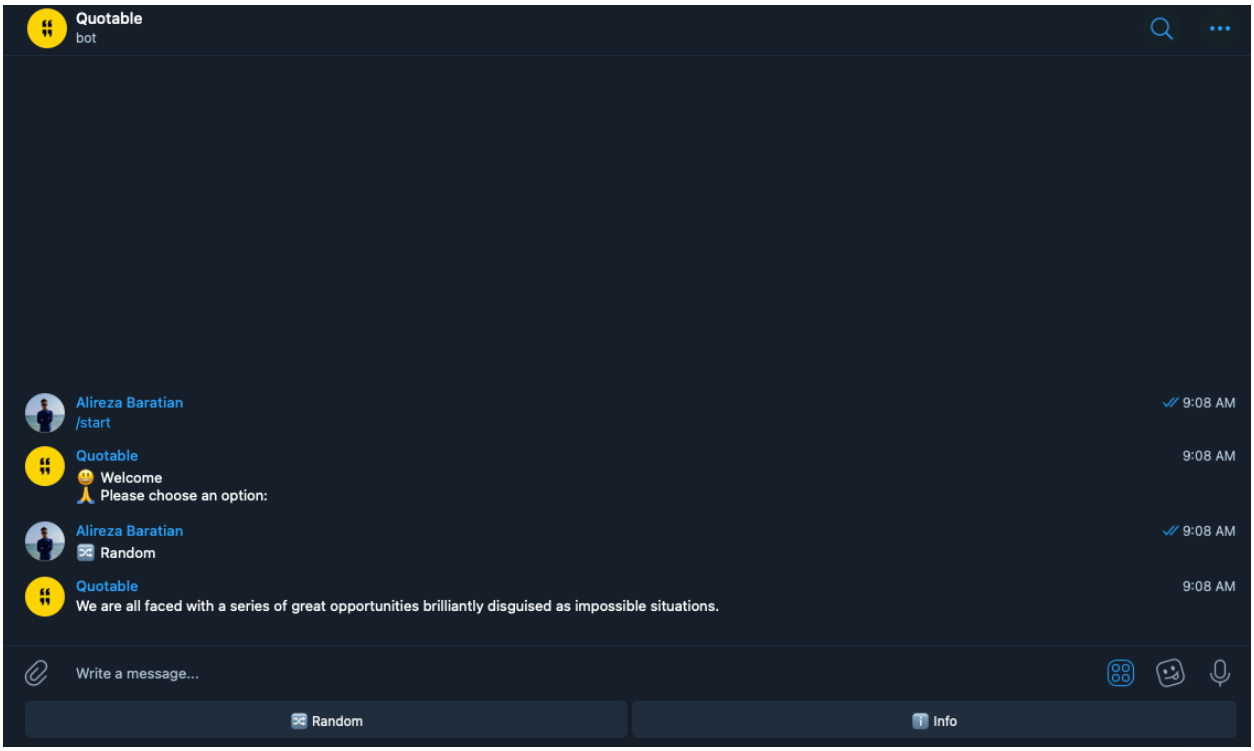
دریافت کنیم و آن را به کاربر نشان دهیم:

کد آن به شکل زیر است: (البته همانطور که گفته شد باید در بدنه تابع parse_input در این مثال اگر پیام ورودی random بود تابع زیر را صدا زده و خروجی آن را به کاربر پیام دهیم)

quotable.py > ...

```
def get_random_quote():
    result = requests.get("https://api.quotable.io/random").json()
    quote = result["content"]
    return quote
```

در نهایت یک تست از خروجی ربات



سخن پایانی

هدف کلی از این پروژه ایجاد علاقه و آشنایی با مجموعه ای از کتابخانه ها و تلاش برای درک آن‌ها و خواندن مستندات مربوطه است. پس سعی کنید مستندات کتابخانه اصلی در این پروژه برای کار با ربات را استفاده کنید و اگر به سوالی برخوردید بدانید و آگاه باشید که سرچ انگلیسی در گوگل یار مهربان شماست. باز هم اگر مشکل یا سوالی داشتید می‌توانید با شرح دقیق با من مطرح کنید.

موفق باشید.