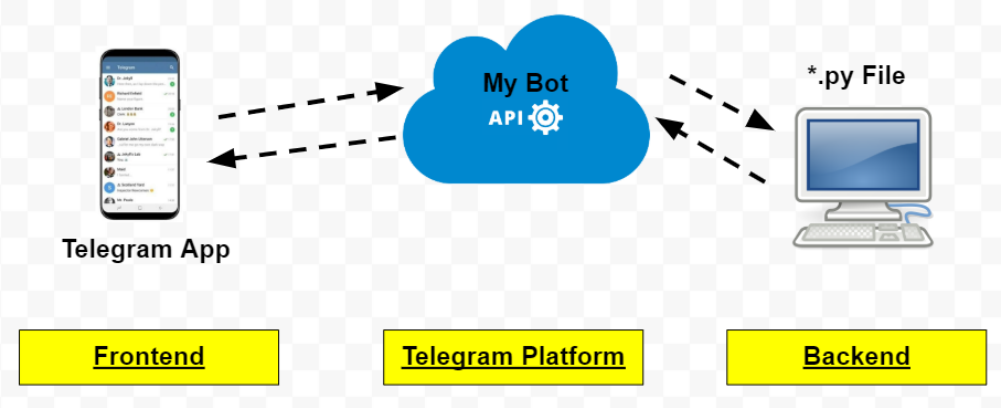
Introduction to Telegram Chatbot

1. Telegram Chatbot

How it works



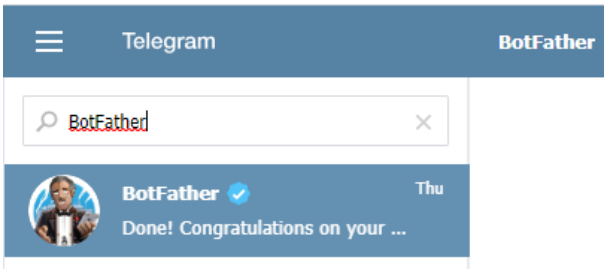
Create a Chatbot

Register Account

- Download telegram app on your mobile phone
- Register an account with your phone number

Telegram Web App

- Go to https://web.telegram.org/ (https://web.telegram.org/)
- Login with your phone number
- Search for `BotFather`



Talk to BotFather

BotFather is a chatbot to create and manage chatbots. Chat with BotFather to create your own bot.

- Send message `/start` to get started.

    - BotFather will reply with list of commands which he can understand.
- Send message `/newbot` to create a new bot.
    - Enter a `Name` for the bot, e.g. `<Your Name>'s Bot`
    - Enter a `username` for the bot, which must **end in `"bot"`** and must be **unique** among all bots.
- BotFather will reply with a API Token for your bot.
- Send message `/mybots` to list your bots.

Library `python-telegram-bot`

Telegram app provides an interface between your bot and users. To make your bot repsond to user, you need to implment a backend which will monitor user's input and respond accordingly

Install Python library `python-telegram-bot` .

In [18]:
```
1 !pip install python-telegram-bot --upgrade
```

```
Requirement already up-to-date: python-telegram-bot in c:\users\isszq\anaco
nda3\lib\site-packages (12.8)
Requirement already satisfied, skipping upgrade: decorator>=4.4.0 in c:\use
rs\isszq\anaconda3\lib\site-packages (from python-telegram-bot) (4.4.1)
Requirement already satisfied, skipping upgrade: certifi in c:\users\isszq
\anaconda3\lib\site-packages (from python-telegram-bot) (2019.11.28)
Requirement already satisfied, skipping upgrade: cryptography in c:\users\i
sszq\anaconda3\lib\site-packages (from python-telegram-bot) (2.8)
Requirement already satisfied, skipping upgrade: tornado>=5.1 in c:\users\i
sszq\anaconda3\lib\site-packages (from python-telegram-bot) (6.0.3)
Requirement already satisfied, skipping upgrade: six>=1.4.1 in c:\users\iss
zq\anaconda3\lib\site-packages (from cryptography->python-telegram-bot) (1.
14.0)
Requirement already satisfied, skipping upgrade: cffi!=1.11.3,>=1.8 in c:\u
sers\isszq\anaconda3\lib\site-packages (from cryptography->python-telegram-
bot) (1.14.0)
Requirement already satisfied, skipping upgrade: pycparser in c:\users\issz
q\anaconda3\lib\site-packages (from cffi!=1.11.3,>=1.8->cryptography->pytho
n-telegram-bot) (2.19)
```

Import library and create a both with your token.

In [19]:
```
1 import telegram
2 telegram.__version__
```

Out[19]: `'12.8'`

In [21]:
```
1 TOKEN = '1142756283:AAFb5zMiJ-4n5rwlrgkmyrCjAKaP7J0LAlk'
2 bot = telegram.Bot(token=TOKEN)
```

If the token is valid, `get_me()` function will return information about the bot.

In [22]:
```
1 print(bot.get_me())
```

```
{'id': 1142756283, 'first_name': "Qinjie's Bot", 'is_bot': True, 'usernam
e': 'qinjie_bot', 'can_join_groups': True, 'can_read_all_group_messages': F
alse, 'supports_inline_queries': False}
```

Library Extensions

Above library includes a `telegram.ext` submodule. It provides an easy-to-use interface to ease development work.

It consists 2 most imporant classes `telegram.ext.Updater` and `telegram.ext.Dispatcher` .

2. Maths Bot

Lets get familiar with the library by creating a bot which can solve mathematical equations.

- Save following text in a python (`*.py`) file.

Updater

An **Updater** object receives the updates from Telegram and to delivers them to a dispatcher.

- Updater starts as a polling service to check on updates (new messages/commands) in Telegram server.
- The updates are kept in a queue.
- Updater works like a messenger between Telegram and developers.
- As `Updater.start_polling()` is a non-blocking function and eventually will stop, the `idele()` function is added to keep script running.

In [2]:
```
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters

# Part A
token = '<TOKEN>' # Replace <TOKEN> with your own token
token = '1142756283:AAFb5zMiJ-4n5rwlrgkmyrCjAKaP7J0LAlk' # Replace <TOKEN
updater = Updater(token, use_context=True)

# Part C (TODO, /start command)

# Part D (TODO, /help command)

# Part E (TODO, /rate command)

# Part F (TODO, reply USAGE for other messages)


# Part B
updater.start_polling()
updater.idle()
```

Above code creates a Updater and run it. But it does nothing because we haven't instruct him to do anything yet.

Dispatcher and Handler

Each `Updater` comes with a dispatcher. `Dispater` updates update from the queue and pass it to registered handlers.

A `Handler` is an instance of any subclass of the `telegram.ext.Handler` class. There are different subclasses of `telegram.ext.Handler` for different updates, e.g. `CommandHanlder`, `MessageHandler` etc.

In [30]:
```
type(updater.dispatcher)
```

Out[30]: `telegram.ext.dispatcher.Dispatcher`

There are many different types of Handler.

In [36]:
```
telegram.ext.*Handler?
```

Handle `/start` Command (Part C)

A `Update` can be a `Command` or a `Message`.

- User type a command by starting with a forward-slash `/`, for example `/start`

Following code adds a handler for `/start` command.

- It replies with a message `"Hi, I am a Mathematician!"` .

In [4]:
```
# Part C
def handle_start_cmd(update, context):
    update.message.reply_text('Hi, I am Mathematician!')

cmd = CommandHandler("start", handle_start_cmd)
updater.dispatcher.add_handler(cmd)
```

Exercise (Part D)

Add a CommandHandler for `"help"` command to the bot. Upon receiving the command, the bot will reply a message `"Type a maths equation and I will show you the answer."` .

In [ ]:
```
# Part D
def handle_help_cmd(update, context):
    update.message.reply_text('Type a maths equation and I will show you
cmd = CommandHandler('help', handle_help_cmd)
updater.dispatcher.add_handler(cmd)
```

Exercise (Part E)

The `update.message.text` will always return the text sent by user. For example, user can type `"/help about"` . For such message, it will be handled by `"/help"` command handler since it starts with `/help` .

Handling `/maths` Command

The main function of this chatbot is to evaluate maths expression. For example, `"/maths 1 + 2"` can be evaluated as a command `/maths` followed by a maths expresssion `"1 + 2"` . The bot will return the result of this maths expression.

- If the string fails to be evaluated as a maths equation, it will return `"Invalid maths equation"` .
- Hint: Use `eval()` function.

In [ ]:
```python
# Part E
def handle_maths_cmd(update, context):
    s = update.message.text
    print(s)
    exp = s.replace('/maths', '')
    try:
        result = eval(exp)
    except:
        result = 'Invalid maths expression'
    update.message.reply_text(result)

cmd = CommandHandler('maths', handle_maths_cmd)
updater.dispatcher.add_handler(cmd)
```

Handle All Other Messages (Part F)

Dispatcher will look for a suitable handler for an update one by one. It will stop once it finds a suitable handler. Thus the **order of adding** handlers to dispatcher is important.

For those unhandled updates, we want to remind user on how to use our chatbot.

Implement another handler for all unhandled text updates.

- Use `Filters.text` to match all updates of text type.

In [ ]:
```python
# Part F
def handle_any_msg(update, context):
    update.message.reply_text('USAGE: /maths <maths-expression>')

cmd = MessageHandler(Filters.text, handle_any_msg)
updater.dispatcher.add_handler(cmd)
```

3. Assignment

The Foreign Exchange Rates API website http://exchangeratesapi.io/ (http://exchangeratesapi.io/) is a free service for current and historical foreign exchange rates published by the European Central Bank.

Some useful APIs:

- GET https://api.exchangeratesapi.io/latest?base=SGD (https://api.exchangeratesapi.io/latest?base=SGD)
- GET https://api.exchangeratesapi.io/latest?base=SGD&symbols=USD (https://api.exchangeratesapi.io/latest?base=SGD&symbols=USD)
- GET https://api.exchangeratesapi.io/latest?base=SGD&symbols=USD,GBP (https://api.exchangeratesapi.io/latest?base=SGD&symbols=USD,GBP)

In [6]:
```
import requests

SERVER = 'https://api.exchangeratesapi.io'
RESOURCE = '/latest'

url = SERVER + RESOURCE
params = {'base':'SGD', 'symbols':'USD,GBP'}
resp = requests.get(url, params)
print(resp.status_code)
print(resp.text)
print(resp.json())
```

```
200
{"rates":{"USD":0.7180793479,"GBP":0.5704005604},"base":"SGD","date":"2020-
07-10"}
{'rates': {'USD': 0.7180793479, 'GBP': 0.5704005604}, 'base': 'SGD', 'dat
e': '2020-07-10'}
```

In [7]:
```
result = resp.json()
print(result['base'])
for k,v in result['rates'].items():
    print(k,v)
```

```
SGD
USD 0.7180793479
GBP 0.5704005604
```

Create a chatbot with following commands and replies, where rates are retreived from API http://exchangeratesapi.io/ (http://exchangeratesapi.io/).

- /start

  ```
  Hi, I am Money Changer!
  ```

- /help

  ```
  Ask me about exchange rate!
  USAGE:
      /rate <SGD> <USD>
      /rate <SGD> <USD,GBP>
  ```

- /rate SGD USD

  ```
  1 SGD = 0.7180793479 USD
  ```

- /rate SGD USD,GBP

  ```
  1 SGD = 0.7180793479 USD, 0.5704005604 GBP
  ```

In [ ]:
```

```