Introduction to chatbots

A chatter robot (chatbot) is a type of conversational agent, a computer program designed to simulate an intelligent conversation with one or more human users in natural language via auditory or textual methods.

In short, chatbots are virtual assistant programmed to automatically answer users requests

applications ¶

- Customer support - User account information, Order tracking and delivery, Product technical support, Incident report
- Sale and advice - Make an order, reservation, Ask for personalized advices
- Internal support - Helpdesk (office applications, etc), HR (leave balances, etc), Practical services (room booking…)
- Gaming platforms - interactive, even multiuser text-image based games

advangages

- Immediate answers
- Available 24/7
- 90% routine answers
- Personalized answers
- Customer autonomy
- Innovative customer service

telegram bot creation

- search for botfather
- start conversation
- type /newbot command for creating new bot - provide name (how everybody will call your bot) and identificator, that must end with 'bot'
- use generated token to authorize your bot

In computer systems, an ***access token*** *contains the security credentials for a login session and identifies the user, the user's groups, the user's privileges, and, in some cases, a particular application*

In [ ]:
```
# for security reasons my token is hidden :)
from my_token import MY_TOKEN
```

installing module to work with telegram module

$ pip install pytelegrambotapi

pip is a recursive acronym for "Pip Installs Packages". pip is a standard package-management system used to install and manage software packages written in Python.

In [ ]:
```
# lets import library and initiate our new bot
import telebot
bot = telebot.TeleBot(MY_TOKEN)
```

In [ ]:
```
# our bot can do nothing yet.
# lets program it to responce on /start command
@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, "let's start")

# here @bot is a decorator
### Decorators provide a simple syntax for calling higher-order functions.
### By definition, a decorator is a function that takes another function
### and extends the behavior of the latter function without explicitly modifying it.
# message_handler catches messages
# commands=['start'] determines what type and keyword should it catch
### i.e. @bot.message_handler(commands=['start']) is a decorator that catches 'start' command
# def start_message(message): is a name and argument of response function
# bot.send_message(message.chat.id, "let's start") is a response command
###. i.e. bot sends message "let's start" to the chat that was asked to /start
```

In [ ]:
```
# yet this is not enough
# our new not is not listening to us
# let's turn it on
bot.polling()
# now bot can respond on /start command
```

In [ ]:

```
# now let's extend functionality of our bot
# let's teach itreply on 'hi', 'bye' messages
@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text.lower() == 'hi':
        bot.send_message(message.chat.id, 'Good day')
    elif message.text == 'bye':
        bot.send_message(message.chat.id, 'cya')
    else:
        bot.send_message(message.chat.id, 'try again')
# make sure that us stopped bot.polling()
# and start it after this function
```

In [ ]:

```
# you will get something like this
import telebot
bot = telebot.TeleBot(MY_TOKEN)

@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, "let's start")

@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text.lower() == 'hi':
        bot.send_message(message.chat.id, 'Good day')
    elif message.text == 'bye':
        bot.send_message(message.chat.id, 'cya')
    else:
        bot.send_message(message.chat.id, 'try again')

bot.polling()
```