Write bots for Telegram. What else?
Writing Chat Bots for Telegram (and Slack)
Who are we?

- Lorenzo Buonannno (Computer Science)
- Tommaso Sartor (Mechanical Engineering)
Who are you?

- OS
- Python experience
- Personal chat preferences
- What service would you like to develop a bot for?
- Chat Bot experience
What are we going to do?

- This is **NOT** a talk, it’s a training session!
  - If you have any question or any doubt, please **ask**!
- It will not be just us talking and showing some cool bots we (and other people) made
- Most of the time will be spent live coding bots!
- The goal of this session is to make you go home wanting to make an awesome bot at least slightly useful to at least a few people, and teaching you how to make it
Why should we start making bots?

- It’s easy
- It’s useful
- It’s fun
Why are bots great?

- **Easy**: much easier and faster to develop than a mobile app (or even a website) (@pyconbot took me a couple of hours)
  - You don’t have to build an UI!
- **Useful**: they can contact subscribed users at any time in a convenient way, and they can be very easily summoned when they’re needed
  - You can ask for lots of stuff (e.g. location)
  -Built-in authentication
What can be done?

- Success stories and opportunities:
  - LibreTaxi [http://libretaxi.org/](http://libretaxi.org/) more than 10 thousand registered users in less than 1 month from some random guy living in a remote “Siberian village”, looking for a way to connect drivers to passengers
  - Bot prize [https://telegram.org/blog/botprize](https://telegram.org/blog/botprize)
  - Transcriber bot got 3000 unique users
  - Thousands of bots with millions of users
What can a bot look like?

- Gitter gif
- Facebook Messenger Bot
- http://t.me/pyconbot
- Slack message builder
- http://t.me/xkcdchannel
- Twitter?
What do you want to do?

If you didn't already, start thinking now about any way in which the awesome bot that you’ll make (using what you’ll learn in this session) can do to make your (or even better somebody else’s) life easier, funnier or even just slightly better.
Some ideas

- Taylor made bots for small groups of people
  - Checking today’s menu from the canteen
  - Getting next talks information at a specific convention
  - Post updates from a website to a chat (or a group)
  - Scrape websites of local interest for useful events (tides in Venice, bikes in Milan, news, school)
  - Handle user interface for a task that’s done often (signing documents, google for a specific term and send back the first result to the chat)
  - Receive notifications for important events (production server doesn’t ping, emergency!)
A possible classification

- **User-Bot-User**
  - The bot only access to local information
  - I.e. do calculations, schedule reminders, manage lists

- **User-Bot-World**
  - User sends information or commands to something through the bot
  - Examples: home automation, open house door, print documents
  - Examples: Update/reboot servers, run some commands

- **World-Bot-User**
  - User get notified from events in the world or from the internet
  - Examples: Receive latest comics strip
  - Examples: Get notified from motion detection systems
What you need to run a bot

- Any internet connection (two possibilities):
  - (Default) Polling: No need for public IP
  - Web hooks: Bot host machine needs a public IP
- An API key
- A process
Bot deployment scenarios
Bot vs Users

Bots are different from users. Are they?

User accounts can be scripted but Bot API is different from "Users" API.
Before we begin

We will use:

- Wifi connection (password: pycon2017)
- Linux ;) or inferior other OSes
- Telegram! : [https://telegram.org/](https://telegram.org/) [http://t.me/pycon8](http://t.me/pycon8)
  - Telepot : [https://github.com/nickoala/telepot](https://github.com/nickoala/telepot)
- Slack? : [https://spyno.slack.com](https://spyno.slack.com)
  - slackclient : [http://github.com/slackapi/python-slackclient](http://github.com/slackapi/python-slackclient)
- Python (duh), pip, virtualenv
Hello World!

Almost time to code!
Bots interact with Telegram/Slack via an HTTP API, for which you’ll need a token

Telegram API:
https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe
Methods and info:
https://core.telegram.org/bots/api

Slack API:
https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
How to get your API token

Telegram:

- Send a message to @BotFather [https://telegram.me/botfather](https://telegram.me/botfather)
  - Follow instructions, maybe look around, we’ll ignore most commands for now

Slack:

- Go here [https://api.slack.com/bot-users](https://api.slack.com/bot-users) (bot users page) or directly here [https://my.slack.com/services/new/bot](https://my.slack.com/services/new/bot)
- Name your bot and click “Add bot Integration”
Attention!

- Absolutely make sure you keep your key SECRET
- Everyone with the key will have full control of your bot
- Make sure especially if you’re using github
  - Really
  - Seriously
  - I mean, why do people do this?
- Save it in a gitignored config.py file to import
- Save it in a gitignored config.json file to load
- Save it in an environment variable
Coding time!

Telegram users:

- We’ll follow https://telepot.readthedocs.io/en/latest/

```python
>>> import telepot

>>> bot = telepot.Bot('***** PUT YOUR TOKEN HERE *****

>>> bot.getMe()
```
Coding time!

Slack users:

- We’ll follow
  https://www.fullstackpython.com/blog/build-first-slack-bot-python.html

```python
import os
from slackclient import SlackClient

BOT_NAME = 'YOUR BOT NAME'
slack_client = SlackClient('SLACK_BOT_TOKEN') #If you want you can use an environment variable
if __name__ == '__main__':
    api_call = slack_client.api_call("users.list") #Actual API call
    if api_call.get('ok'):
        print(api_call) #Print the result
        # retrieve all users so we can find our bot
        users = api_call.get('members')
        for user in users:
            if 'name' in user and user.get('name') == BOT_NAME:
                print("Bot ID for " + user['name'] + " is " + user.get('id'))
            else:
                print("could not find bot user with the name " + BOT_NAME)
```
CODING TIME!
How many of you used python standard logging library before?

```python
import logging

logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    filename='infolog.log', # If you want to save to a file, for now we should just print so you can see what you get
    level=logging.INFO)

logger = logging.getLogger(__name__)

logger.info("Test info, you'll see this")

logger.debug("If you want to see this, set level to logging.DEBUG")

logger.warning("warning! Logging is awesome")

logger.error("Error! You didn’t use logging before!")
```
Memory!

dataset: databases for lazy people

Google “python dataset” or go to https://dataset.readthedocs.io/

Example:

```python
import dataset
db = dataset.connect('sqlite:///mydb.db')
table = db['sometable']
table.insert(dict(chat_id='66666666', name='John Doe', age=37))
table.insert(dict(chat_id='99999999', name='Jane Doe', age=34))
john = table.find_one(name='John Doe')
print(john['chat_id'])
```
import telepot
def handle(msg):
    content_type, chat_type, chat_id = telepot.glance(msg)  # Useful telepot function!
    print(content_type, chat_type, chat_id)
    if content_type != 'text':
        return
    if msg['message']['text'].startswith('/help'):
        bot.sendMessage(chat_id, 'Hi! Let’s play /ping /pong')
    elif msg['message']['text'].startswith('/ping'):
        bot.sendMessage(chat_id, '/pong')
    elif msg['message']['text'].startswith('/pong'):
        bot.sendMessage(chat_id, '/ping')

bot = telepot.Bot("API TOKEN")  # or os.environ['BOT_TOKEN'] if you want to use an environment variable
bot.message_loop(handle)
# Keep the program running.
while 1:
    time.sleep(10)
There are better ways to do it!

def my_help_function(bot, msg):
    # Do stuff with msg
    bot....(...) #Call a bot method
    return

def my_start_function(bot, msg):
    # Do stuff with msg
    bot....(...) #Call a bot method
    return

command_functions={'/help': my_help_function,
                   '/start': my_start_function}

def handle(msg):
    if 'text' not in msg:
        return
    command = msg['text'].split()[0] if msg['text'] else ''
    command_functions[command](bot, msg)

Example:
https://git.peori.space/minala/MySubredditsBot/src/master
The magical way

Example [https://git.peori.space/porcospyno/omdb](https://git.peori.space/porcospyno/omdb)

@command(".omdb", ".imdb", flavors=["inline_query"])
@send_with(parse_mode="HTML", preview=True)
def omdb_inline_search(data):
    """Find list of movies matching title on omdb and returns it inline
    USAGE: call inline .omdb movie""
    res = []
    if not data:
        return "Please write something"
    movies = search_movies(data)
    if not movies:
        return "I couldn't find anything :(
    for movie in movies:
        res.append(
            {
                "title": movie["Title"] + ' (' + movie["Year"] + ')',
                "text": format_movie_info(movie),
                "description": movie["Type"],
                "thumb_url": movie["Poster"]
            })
    return res
A framework would be useful!

We are making one.
PorcoSpyno Framework Motivations

- Who are you?
  - Coders!

- What do you want?
  - Code fun and scalable stuff

- When do you want it?
  - In an hour or less
Design Concepts

● Solve problems **one time:**
  ○ **Modularity** by plugins
  ○ Your own bot, shared plugins

● **Make it easy** to write plugins:
  ○ Avoid as much **boiler plate** as possible
  ○ Framework takes care of **commons duties** as logs, notification and statistics
  ○ Provide ready to use **helpers** methods

● **Support** **multiple chat service** at fixed cost:
  ○ Abstraction layer over chat specific API
  ○ General, chat agnostic, API
Current implementation

- **Import machinery**
  - Plugin functions are registered to the specified spyno API Chat event (message, button, ecc) by function *decorators* at import time.

- **Inbound routing**
  - Incoming messages are wrapped in spyno API events.
  - Events are routed to the right registered plugin function if any.

- **Outbound possibilities**
  - Direct call to spyno API method
  - Returned or yielded values are consumed and converted in specific chat API method.

*Heavy WIP, do not trust this.*
Getting started with our framework:

https://git.peori.space/porcospyno/examplebot.git

Still no real documentation :(, examples plugin are a good place where to start!
Blue pill or red pill
CODING TIME!
HAVE FUN!