

Relatório do Segundo Sprint

Manipulação de tempo, Enviar e-mails e mensagens, Web Scraping e Manipulação de imagens

Daniel Gomes, Érica Costa, Marcela Malaquias, Paulo Vicktor Felix

23 de Maio de 2017

Resumo

Neste momento, focamos em aprender funções uteis para o desenvolvimento do nosso projeto final, portanto, os capítulos em questão serão de grande ajuda para que seja possível concluir o que desejamos.

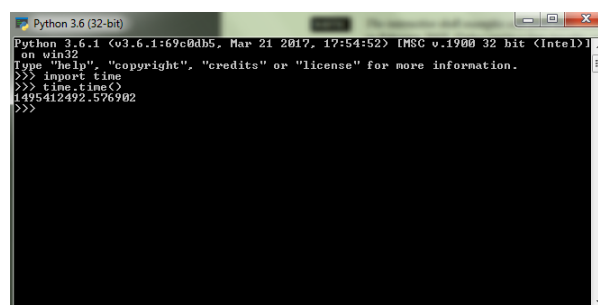
Para o nosso objetivo, será necessário estabelecer uma comunicação entre o usuário e o Bot do telegram, logo, é útil que o bot tenha algumas funcionalidades diferenciadas, para se destacar entre os outros que já existem. Desse modo, iremos incluir em uma única plataforma funções como: verificar o clima, enviar um email, receber imagens de câmeras de segurança, acionar dispositivos remotos (Raspberry pi) e agendar tarefas.

1 Manipulando o Tempo

1.1 O módulo time

O relógio do sistema do computador está definido para uma data, hora e fuso horário específico. O módulo de time embutido permite que seus programas Python leiam o relógio do sistema para a hora atual. As `time.time()` e `time.sleep()` são as mais úteis no módulo de time.

- Função `time.time()`: retorna o número de segundos desde aquele momento como um valor float.]



```
Python 3.6.1 (32-bit)
>>> on win32
>>> type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> time.time()
1495412492.576902
>>>
```

Figura 1: Função `time.time`

Obs: o valor de retorno é quantos segundos se passaram entre a época do Unix e o momento em que `time.time()` foi chamado.

- Função `time.sleep()`: para pausar o programa por um tempo, chame a função `time.sleep()` e passe o número de segundos que você deseja que o programa fique pausado. A função `time.sleep()` liberará seu programa para executar outro código - até que o número de segundos que você passou para `time.sleep()` tenha acabado. Por exemplo, se você inserir `time.sleep(5)`, verá que o próximo prompt não aparece até que cinco segundos tenham passado.
- Rounding numbers: Para tornar os valores float mais fáceis de trabalhar, pode-se encurtá-los com a função `round()`, que arredonda um

float para a precisão específica. Basta passar o número que se deseja arredondar, além de um segundo argumento opcional representando quantos dígitos após o ponto decimal que se deseja arredondá-lo. O segundo, `round()` arredonda seu número para o inteiro mais próximo.

```
Python 3.6.1 <v3.6.1:69c0db5, Mar 21 2017, 17:54:52> [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> now = time.time()
>>> now
1495414282.0030622
>>> round(now, 2)
1495414282.0
>>> round(now, 4)
1495414282.0031
>>> round(now)
1495414282
>>> -
```

Figura 2: Uso de Rounding Numbers

1.2 O módulo datetime

O módulo de `time` é útil para obter um timestamp de época Unix para trabalhar. Mas caso deseja exibir uma data em um formato mais conveniente, ou fazer aritmética com datas (por exemplo, descobrir qual data foi de 205 dias atrás ou data que é de 123 dias a partir de agora), pode-se deve usar o módulo `datetime`.

```
Python 3.6.1 <v3.6.1:69c0db5, Mar 21 2017, 17:54:52> [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import datetime
>>> datetime.datetime.now()
datetime.datetime(2017, 5, 21, 22, 27, 34, 70297)
>>> dt = datetime.datetime(2015, 5, 20, 21, 27, 5)
>>> dt.date + datetime.timedelta(days=205)
datetime.datetime(2015, 5, 20)
>>> dt.hour, dt.minute, dt.second
(21, 27, 5)
>>> -
```

Figura 3: Resultado da aplicação de datetime

2 E-mails e mensagens

No capítulo 16 aprendemos a enviar e encontrar um email. Para enviar usamos

funções como `smtplib.SMTP` para determinar o tipo de uma conta de email e `.login` para logar em uma conta e também `.sendmail` para enviar texto. Já para encontrar o email temos funções como `.select_folder` para determinar onde iremos procurar o email (INBOX, lixo, enviados) temos também `.search` para buscar especificamente como a partir de uma data. Segue exemplos de um programa enviando e outro procurando.

```
1 a = input('digite seu email')
2 b = input('digite sua senha')
3 c = input('digite o seu destinatario')
4 d = input('digite sua mensagem')
5 import smtplib
6 smtpObj = smtplib.SMTP('smtp.gmail.com', 587)
7 type(smtpObj)
8 smtpObj.ehlo()
9 smtpObj.starttls()
10 smtpObj.login(a, b)
11 smtpObj.sendmail(a, c, d)
12 smtpObj.quit()
```

Figura 4: Resultado da aplicação de datetime

```
1 a = input('digite seu email')
2 b = input('digite sua senha')
3 import imapclient
4 imapObj = imapclient.IMAPClient('imap.gmail.com', ssl=True)
5 imapObj.login(a, b)
6 imapObj.select_folder('INBOX', readonly=True)
7 UIDS = imapObj.search(['SINCE 10-May-2017'])
8 UIDS
9 rawMessages = imapObj.fetch([UIDS[0]], ['BODY[]', 'FLAGS'])
10 import pyzmail
11 message = pyzmail.PyzMessage.factory(rawMessages[50040]['BODY[]'])
12 message.get_subject()
13 message.get_addresses('from')
14 message.get_addresses('to')
15 message.get_addresses('cc')
16 message.get_addresses('bcc')
17 message.text_part != None
18 message.text_part.get_payload().decode(message.text_part.charset)
19 imapObj.logout()
```

Figura 5: Resultado da aplicação de datetime

3 Web Scraping

Utilizamos a Web Scraping para consultar o clima, por meio de uma API chamada OpenWeatherMap, que disponibiliza planos gratuitos e pagos para soluções em softwares. Assim, foi feito um cadastro no site da API e conseguimos uma licença gratuita para desenvolver a nossa aplicação com até 60 requisições por minuto.

```
daniel@daniel-ubuntu:~/Documentos/ESOF/erica$ python3 prog.py
Escreva sua cidade: uberlandia
Condição do tempo: Clouds
Temperatura: 23.0
daniel@daniel-ubuntu:~/Documentos/ESOF/erica$ python3 prog.py
Escreva sua cidade: caldas novas
Condição do tempo: Rain
Temperatura: 20.994000000000028
daniel@daniel-ubuntu:~/Documentos/ESOF/erica$ python3 prog.py
Escreva sua cidade: sao paulo
Condição do tempo: Drizzle
Temperatura: 18.100000000000023
```

Figura 6: Resultado da aplicação de WebScraping para Clima e Temperatura.

4 Manipulando Imagens

Para usar imagens em python, utilizaremos o módulo PIL. Este, conta com vasta aplicação, como cortar, rotacionar e salvar imagens. Abaixo, seguem exemplos de aplicação.

```
from PIL import Image
im = Image.new('RGBA', (100, 200), 'purple')
im.save('purpleImage.png')
```

Figura 7: Criando uma imagem roxa.



Figura 8: Imagem criada nas dimensões escolhidas.

O próximo programa, abre uma imagem existente e muda o formato dela, no caso, estava em .jpg e mudou para .png.

```
import sys
import HYP_Utils
from PIL import Image

scriptDir = HYP_Utils.GetDemoDir()

PIL_Version = Image.VERSION

img_filename = "%s/teste.jpg" % scriptDir
im = Image.open(img_filename)
im.save("%s/teste.png" % scriptDir)
```

Figura 9: Imagem criada nas dimensões escolhidas.

5 Referências

[1] SWEIGART, Al. "Automate the boring stuff with python", 2015.

[2] Documentation API OpenWeather-Map: <https://openweathermap.org/current>. Acesso em 20 de Maio de 2017, as 20h.

[3] Primeiros passos com PIL (IMB); <https://www.ibm.com/developerworks/community/blogs/fd26864d-cb41-49cf-b719-d89c6b072893/entry/primeiros-passos-com-pil-a-biblioteca-de-imagens-do-python2?lang=en>. Acesso em 15 de Maio de 2017, as 15h30.