

O que aprendi ensinando Python?

Uma breve história da minha vida

Como vai ser?

- Uma breve introdução, sem promoções, de eu
- O que são comunidades
- Comunidades de software
 - Um pouco de história
 - Como isso mudou o mundo
 - O que a gente ganha com isso?
 - Posso contribuir com isso?
- Tá, mas e daí?





- Versão: 1.0
- Alias: Minduim
- Bebida preferida: Leite



- Versão: 2.0
- Alias: Du
- Bebida preferida: Coca-Cola



Comunidade!

Automação por comando de voz via Raspberry Pi

Desenvolvido por: Prof. MSc. Alexandre Pires-Filho
Desenvolvido por: Prof. MSc. Paulo de Oliveira
Assessor Técnico: Henrique Bernardino

atec
Tecnologia

Introdução

Este projeto tem como objetivo desenvolver um sistema de automação por comando de voz, utilizando um Raspberry Pi como controlador central. O sistema é capaz de interpretar comandos de voz e executar ações pré-definidas, como ligar e desligar dispositivos, controlar a temperatura, entre outros. A implementação utiliza a biblioteca PyAudio para capturar o áudio e o módulo SpeechRecognition para processar os comandos de voz. O sistema é desenvolvido em Python e utiliza o sistema de arquivos do Raspberry Pi para armazenar as configurações e os dados do sistema.

Objetivos

Os objetivos deste projeto são:

- Desenvolver um sistema de automação por comando de voz, utilizando um Raspberry Pi como controlador central.

- Implementar a biblioteca PyAudio para capturar o áudio e o módulo SpeechRecognition para processar os comandos de voz.

- Desenvolver o sistema em Python e utilizar o sistema de arquivos do Raspberry Pi para armazenar as configurações e os dados do sistema.

- Testar o sistema em um ambiente real, utilizando um Raspberry Pi e um sistema de áudio.

- Documentar o desenvolvimento do sistema, incluindo a descrição do projeto, a implementação e os resultados obtidos.

- Apresentar o sistema em uma feira de tecnologia, demonstrando suas funcionalidades e respondendo às perguntas dos visitantes.

- Concluir o projeto, avaliando os resultados obtidos e identificando as melhorias necessárias para o sistema.

- Publicar o projeto em um repositório de código aberto, permitindo que outros desenvolvedores possam aprender com o projeto e reutilizar o código.

- Participar de uma competição de robótica, demonstrando o sistema e competindo com outros projetos.

- Obter um certificado de conclusão de curso, comprovando a realização do projeto.

- Obter uma vaga de emprego, demonstrando a capacidade de desenvolver projetos de automação por comando de voz.

- Obter uma bolsa de estudos, demonstrando o potencial de pesquisa e desenvolvimento do projeto.

- Obter uma patente, demonstrando a originalidade e a inovação do projeto.

- Obter um prêmio, demonstrando a qualidade e a eficiência do projeto.

- Obter uma menção honrosa, demonstrando o reconhecimento do projeto.

- Obter uma medalha, demonstrando a conquista do projeto.

- Obter um diploma, demonstrando a conclusão do projeto.

- Obter um título, demonstrando a importância do projeto.

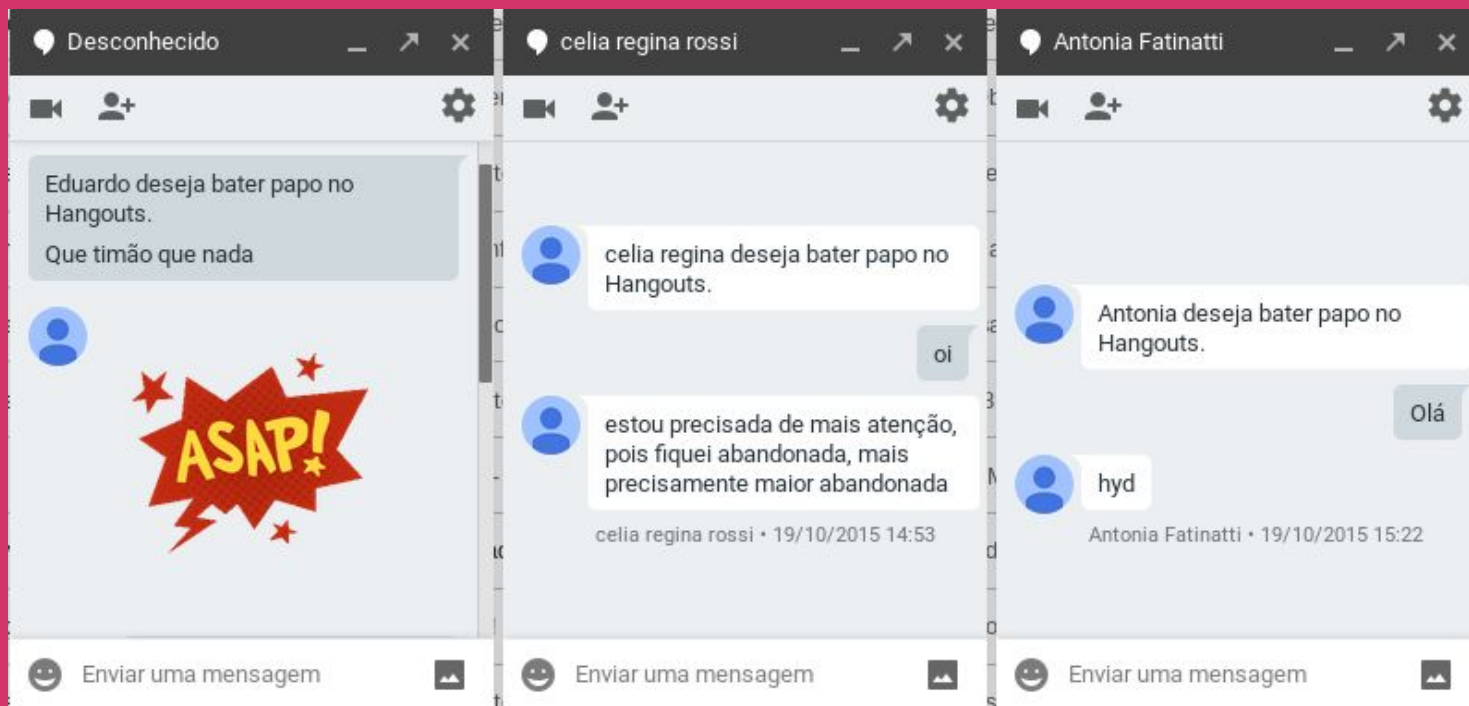
- Obter um reconhecimento, demonstrando a valorização do projeto.

- Obter uma homenagem, demonstrando a gratidão pelo projeto.

- Obter uma lembrança, demonstrando a memória do projeto.

Voluntariado

Voluntariado




```

10
11 • #Importacao
12 import sys
13
14 FOUND = 0
15 NOT_FOUND = 1
16 ERROR = 2
17
18 • def find(search, filename):
19 •
20 •     file = open(filename)
21 •
22 •     for line in __file:
23 •         if search in line:
24 •             yield line.strip()
25
26 •     file.close()
27
28 • def main(args):
29 •
30 •     try:
31 •
32 •         search = args[1]
33 •
34 •     except IndexError:
35 •
36 •         print >>sys.stderr,("Usage: grep [OPTION]... PATTERN [FILE]...")
37 •         return(ERROR)
38
39 •     filenames = args[2:]
40
41 •     ret = NOT_FOUND
42 •
43 •     #iteracao
44 •     for filename in filenames:
45 •

```

```

26 •         file.close()
27
28 • def main(args):
29 •
30 •     try:
31 •
32 •         search = args[1]
33 •
34 •     except IndexError:
35 •
36 •         print >>sys.stderr,("Usage: grep [OPTION]... PATTERN [FILE]...")
37 •         return(ERROR)
38
39 •     filenames = args[2:]
40
41 •     ret = NOT_FOUND
42 •
43 •     #iteracao
44 •     for filename in filenames:
45 •
46 •         try:
47 •
48 •             for line in find(search, filename):
49 •                 print("%s:%s" % (filename, line))
50 •                 ret = FOUND
51 •
52 •         except IOError, ex:
53 •
54 •             print >>sys.stderr,("grep.py: %s: %s" % (ex.filename, ex.strerror))
55 •             return(ERROR)
56
57 •     return(ret)
58
59 • args = sys.argv
60 ret = main(args)
61 sys.exit(ret)

```

Luciano Ramalho





- Versão: 3.0
- Alias: Zara
- Bebida preferida: Cerveja



G.E.C.A.

GRUPO DE ESTUDOS CRIPTOGRAFICOS DE AMERICANA



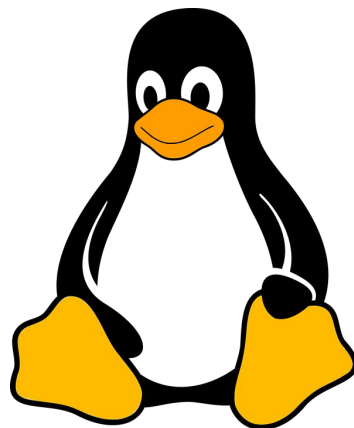
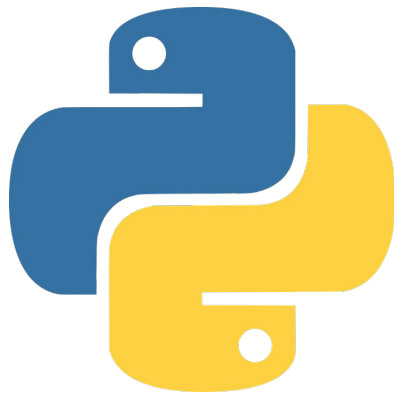


As quatro liberdades

0. Executar o programa como você desejar, para qualquer propósito
1. Estudar como o programa funciona, e adaptá-lo às suas necessidades
2. Redistribuir cópias de modo que você possa ajudar ao próximo
3. Redistribuir cópias de suas versões modificadas a outros



O que a gente ganha com isso?



Posso contribuir com isso?

- Não sou programador
- Ainda não estou preparado
- Eles vão falar mal do meu código
- Tenho dificuldades com inglês
- Não tenho tempo
- Não sei por onde começar



Posso contribuir com isso?

- Não sou programador
- Ainda não estou preparado
- Eles vão falar mal do meu código
- Tenho dificuldades com inglês
- Não tenho tempo
- Não sei por onde começar

Não tenho vontade de contribuir



Eventos de Python

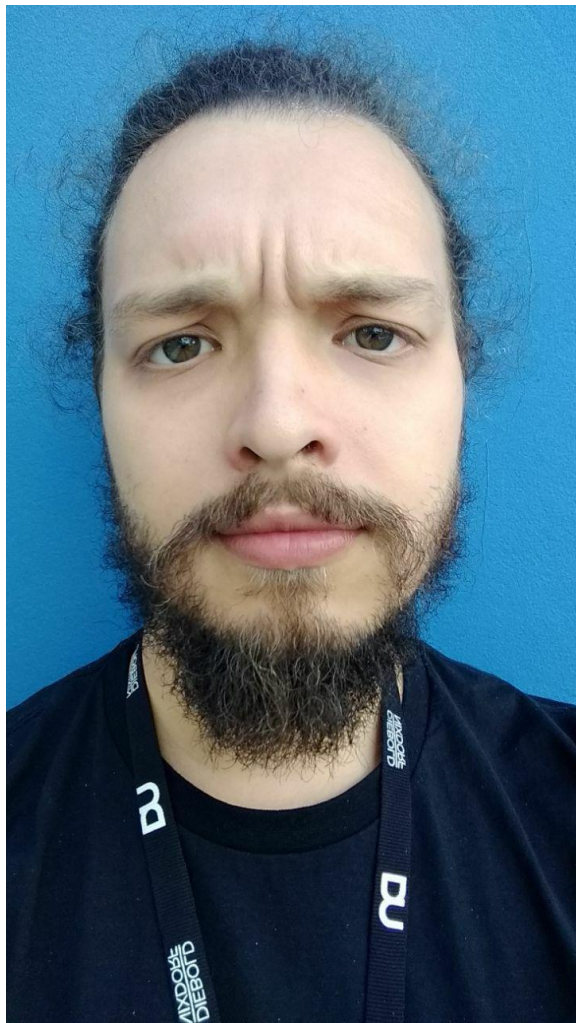


Eventos de Python



Eventos de Python



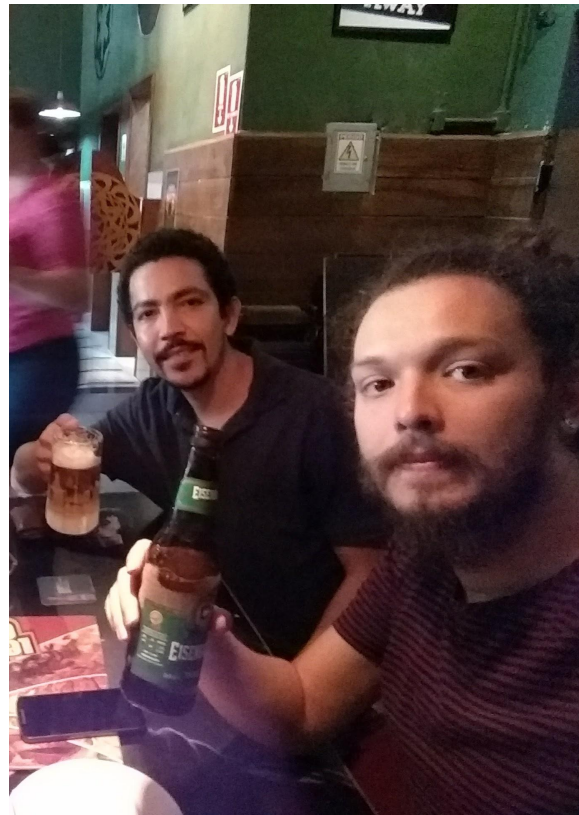


- Versão: unstable
- Alias: Du
- Bebida preferida: Cerveja





QA



A decisão

- Como vou contribuir se não sou um bom programador?
- Que raios eu posso fazer se não documentações?



Alguns vídeos/roteiros/códigos e qualquer coisa que o valha

[Edit](#)[Add topics](#)

138 commits

2 branches

0 releases

6 contributors

Branch: master

[New pull request](#)[Create new file](#)[Upload files](#)[Find file](#)[Clone or download](#)

dunossauro Update README.md

Latest commit df16aaf a minute ago



.github

template para issues

a year ago



funclib

small revision

6 months ago



roteiros

Substituição de vídeo por tópico 🔥

10 days ago



tools

WIP; Roteiro 10 decoradores

10 days ago



README.md

Update README.md

a minute ago



_config.yml

Set theme jekyll-theme-leap-day

a year ago



bibliotecas_externas.md

#20 - Finalizando o texto da construção da nossa lib;

a year ago



referencias.md

atualização das referências

a year ago



sumario.md

FIX sumário #25

2 months ago

README.md

[Say Thanks](#)[waffle](#)

Repositório sobre os vídeos de python funcional



Eduardo Mendes

1.852 inscritos

PERSONALIZAR O CANAL

ESTÚDIO DE CRIAÇÃO

INÍCIO

VÍDEOS

PLAYLISTS

CANAIS

DISCUSSÃO

SOBRE



Uploads ▾

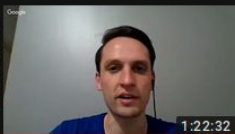
REPRODUZIR TODOS

CLASSIFICAR POR



Live de Python #45 - Rows
(Com Turicás)

438 visualizações • 2 dias atrás



Live de Python #44 - Criando
um bot para telegram (Com

466 visualizações •
1 semana atrás



Live de Python #43 -
Gerenciadores de contexto

313 visualizações •
3 semanas atrás



Live de Python #42 - Testes
de mutação

238 visualizações • 1 mês atrás



Live de Python #41 -
Cobertura e testrunners

288 visualizações • 1 mês atrás



Live de Python #40 - Análise
estática de código (linters)

305 visualizações • 1 mês atrás



Live de Python #39 - Guias
de estilo e padronização de

407 visualizações • 1 mês atrás



Live de Python #38 -
Introdução a qualidade de

472 visualizações •
2 meses atrás



Rapidinha Pythonica #6 -
Slice

292 visualizações •
2 meses atrás



Live de Python #37 -
Arquivos de configuração

464 visualizações •
2 meses atrás



Live de Python #36 -
Argumentos em linha de

315 visualizações •
2 meses atrás



Live de Python #35 - Redes
neurais usando a biblioteca

1,2 mil visualizações •
2 meses atrás



Live de Python #34 -
Trabalhando com arquivos

501 visualizações •
3 meses atrás



Rapidinha Pythonica #5 -
Testando stdout

348 visualizações •
3 meses atrás



Live de Python #33 - Python
para arquitetos, designers e

512 visualizações •
3 meses atrás



Qual o melhor sistema
operacional para programar?

662 visualizações •
3 meses atrás



Editor de texto ou IDE?

599 visualizações •
3 meses atrás



Rapidinha Pythonica #4 -
Especial de natal

267 visualizações •
3 meses atrás


```

1  from re import sub #NOQA
2  from functools import partial
3  from collections import Counter
4  from multiprocessing.dummy import Pool
5
6  map_loko = Pool(4)
7
8
9  def pipe(*funcs):
10     def wrapper(arg):
11         result = arg
12         for func in funcs:
13             result = func(result)
14         return result
15     return wrapper
16
17
18  remove_blank_lines = partial(filter, lambda
19  remove_new_lines = partial(map_loko.map, lar
20  lower = partial(map, str.lower)
21  remove_punctuation = partial(map, lambda x:
22  join = partial(str.join, ' ')
23  split = partial(str.split, sep=' ')
24
25  parse = pipe(open, remove_blank_lines,
26              remove new lines. lower. remove

```

```

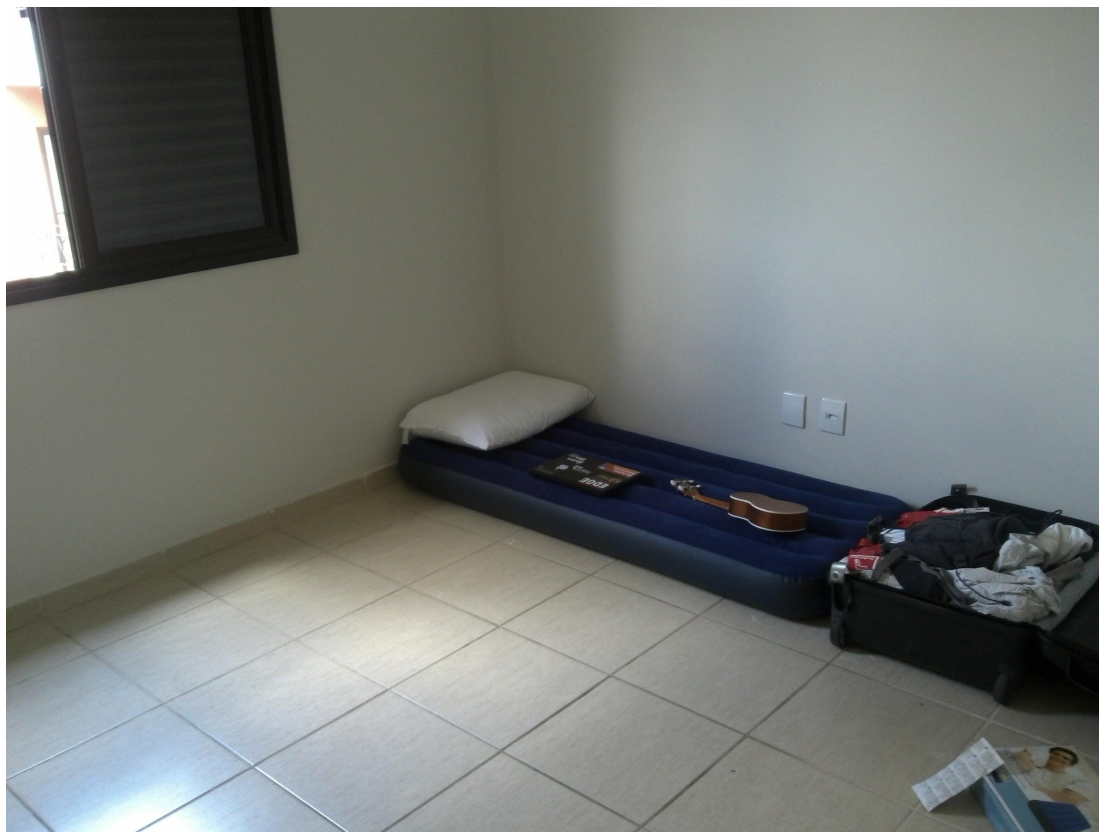
7
8
9  def pipe(*funcs):
10     def wrapper(arg):
11         result = arg
12         for func in funcs:
13             result = func(result)
14         return result
15     return wrapper
16
17
18  remove_blank_lines = partial(filter, lambda x: x != '\n')
19  remove_new_lines = partial(map_loko.map, lambda x: str.strip(x, '\n'))
20  lower = partial(map, str.lower)
21  remove_punctuation = partial(map, lambda x: sub(r'[\.,?!\\-\(\);]', '', x))
22  join = partial(str.join, ' ')
23  split = partial(str.split, sep=' ')
24
25  parse = pipe(open, remove_blank_lines,
26              remove_new_lines, lower, remove_punctuation,
27              join, split)
28
29  count_parse = pipe(parse, Counter)
30
31  xpto = count_parse('tabacaria.txt')
32

```

Um papo sobre educação

- Educação como forma de ativismo
- Educação para transformar





XOXO

Dúvidas/sugestões

agora é a hora de dizer algo