

Quem sou eu?

Programação em Python, v0.1

Prof. Eduardo Vasconcelos @ SENAI Araraquara
Araraquara, 05/05/2025

Oi, eu sou o Eduardo!

- 🎓 Microlins – Montagem e Manutenção de Computadores (2007);
- 🎓 USP – Engenharia de Computação (2012–2017);
- 🎓 Unicamp – Especialização em Engenharia de Software (2019);
- 🎓 USP – Mestrado em Ciências de Computação (2022–atualmente);
- 💼 SiDi (Samsung) – Arquiteto de Segurança de Aplicações (atualmente);
- 👣 Outras experiências: iFood, Hacker Rangers (Perallis Security), Embraer, SigMEDIA (Trinity College).

MedEsc 10.4.0 - EvaSoft

	SIMULADO 1	PROVA	SIMULADO 2	
POR	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
LIT	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
MAT	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
GEO	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
FIS	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
BIO	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
LIN	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
QUI	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
HIS	<input type="text"/>	<input type="text"/>	<input type="text"/>	MÉDIA
	PROVA 1	PROVA 2		
RED	<input type="text"/>	<input type="text"/>		MÉDIA
	MÉDIA GERAL			

MedEsc 10.4.0 - EvaSoft®. Software desenvolvido por EvaSoft: Eduardo Vasconcelos Software Enterprises®. Cópia proibida.

Meu primeiro programa de computador 🖱️

Algumas coisas legais que eu já fiz usando Python (I)

Um programa para automatizar a geração da escala da liturgia da minha igreja:

```
gerar-escala.py X
001-quem-sou-eu > gerar-escala.py > ...
1  #!/usr/bin/python3
2
3  from random import randint
4
5  domingos = 4
6  data_inicial = 1
7  mes = 5
8
9
10
11
12
13
14  urna_comentaristas = comentaristas.copy()
15  urna_leitores = leitores.copy()
16  urna_cestinhas = cestinhas.copy()
17  urna_cestinhas_dizimistas = cestinhas_dizimistas.copy()
18
19  for i in range(0, domingos + 1):
20      print('\n~~~~~ {}/{} ~~~~'.format(data_inicial + (i * 7), mes))
21
22      ja_escalados = []
23
24      # Comentarista
25
26      if len(urna_comentaristas) < 1:
27          urna_comentaristas = comentaristas.copy()
28
29      comentarista = urna_comentaristas[randint(0, len(urna_comentaristas) - 1)]
30      ja_escalados.append(comentarista)
31      urna_comentaristas.remove(comentarista)
```

```
eduardo@thinkpad: ~/teaching/
eduardo@thinkpad: ~/teaching/python/001-quem-sou-eu$ python3 gerar-escala.py

~~~~~ 1/5 ~~~~
- Comentarista
- Leitor 1
- Leitor 2
- Precizes
- Cestinha
- Cestinha

~~~~~ 8/5 ~~~~
- Comentarista
- Leitor 1
- Leitor 2
- Precizes
- Cestinha
- Cestinha

~~~~~ 15/5 ~~~~
- Comentarista
- Leitor 1
- Leitor 2
- Precizes
- Cestinha
- Cestinha

~~~~~ 22/5 ~~~~
- Comentarista
- Leitor 1
- Leitor 2
- Precizes
```

Algumas coisas legais que eu já fiz usando Python (II)

Um programa que foi a base de um artigo científico premiado:

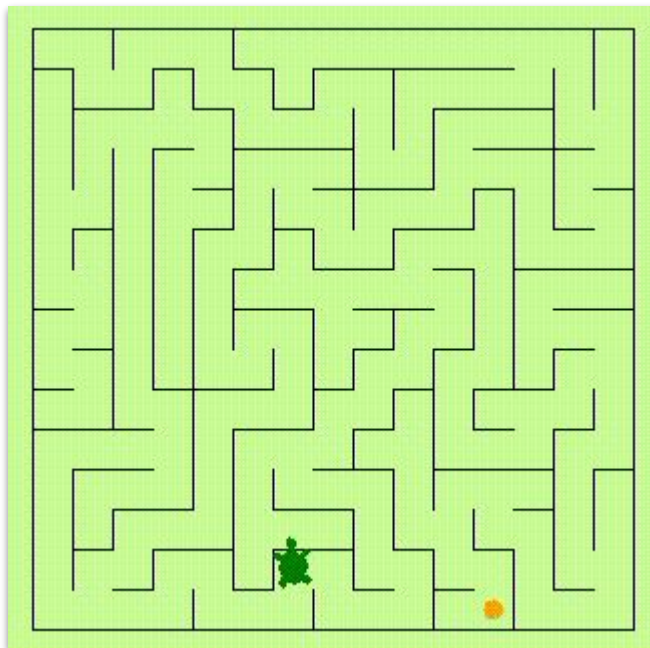
```
seed-vulns.py X
seed-vulns.py > ...
11 from operators.java.tapjacking_set_full_overlay_windows import TapjackingSetFullOverlayWindows
12 from operators.operators import OperatorNames, OperatorTypes
13 from operators.xml.improper_export import ImproperExport
14 from operators.xml.debuggable_application import DebuggableApplication
15 from operators.xml.plaintext_http import PlaintextHttp
16 from operators.xml.tapjacking_full_occlusion import TapjackingFullOcclusion
17 from resources.resources_handler import ResourcesHandler
18 from source.source_handler import SourceHandler
19
20 def main():
21     # Setup logging and print banner to console
22     log = setupLogging()
23     log.info("===== seed-vulns =====")
24
25     # Parse arguments and log them. Need sourcePath,
26     # destinationPath, and operators
27     log.info("Parsing arguments...")
28     args = parseArguments()
29     sourcePath = args.sourcePath
30     destinationPath = args.destinationPath
31     operators = args.operators.split(',')
32     single = args.single
33     commentMutations = args.comment_mutations
34     allMutants = args.all_mutants
35     logArguments(log, sourcePath, destinationPath, operators, single, commentMutations)
36     if allMutants and single:
37         log.error("Conflicting arguments: cannot output all mutants and a single mutant at the
38             same time. Exiting...")
39         exit(1)
40
41     # Copy the source path to the destination path. Mutations
42     # shall overwrite the files in the destination path
43     if single:
```



Algumas coisas legais que eu já fiz usando Python (III)

Um programa que resolve um labirinto sozinho:

```
tartarugaPq_resposta.py X
001-quem-sou-eu > tartarugaPq_resposta.py >...
285 # e paredes = [1,1,1,1] (todas as paredes estao completas inicialmente)
286 def criaCelula(p1, p2, p3, p4):
287     celula = {}
288     celula['visitado'] = 0
289     celula['paredes'] = [p1,p2,p3,p4]
290     return celula
291
292 # Manda a tartaruga virar a direita (muda heading do turtle)
293 def viraDireita() :
294     turtle.setheading((turtle.heading() - 90) % 360)
295
296 # Manda a tartaruga virar a esquerda (muda heading do turtle)
297 def viraEsquerda() :
298     turtle.setheading((turtle.heading() % 360) + 90)
299
300 # Verifica se ha uma parede a frente da tartaruga, com relacao ao
301 # seu valor de heading (do turtle)
302 def temParedeFrente() :
303     global x
304     global y
305     if (turtle.heading() == 0) :
306         return labirinto[y][x]['paredes'][0]
307     elif (turtle.heading() == 90) :
308         return labirinto[y][x]['paredes'][3]
309     elif (turtle.heading() == 180) :
310         return labirinto[y][x]['paredes'][2]
311     else :
312         return labirinto[y][x]['paredes'][1]
313
314 # Verifica se ha uma parede atras da tartaruga, com relacao ao
315 # seu valor de heading (do turtle)
316 def temParedeAtras() :
```



E você?

- Como você se chama?
- Por que você está fazendo esse curso?
- Esse é o seu primeiro contato com programação?