Como debugar código Python?

Live de Python # 197

Roteiro

1. Afinal, o que é debugar código?

Conceitos básicos de depuração

2. Ferramentas que podem ajudar

Sim, vamos parar de usar print()

3. Um olhar um pouco mais avançado

Como isso funciona internamente?

4. Codar para exemplificar

Se der tempo ou até o tempo acabar

picpay.me/dunossauro

apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3

Acássio Anjos, Ademar Peixoto, A Earth, Alexandre Harano, Alexandre Souza, Alexandre Takahashi, Alexandre Villares, Alex Lima, Alynne Ferreira, Alysson Oliveira, Ana Carneiro, Ana Padovan, Andre Azevedo, André Rocha, Aquiles Coutinho, Arnaldo Turque, Artur Zalewska, Ayrton Freeman, Bloquearsites Farewall, Bruno Barcellos, Bruno Freitas, Bruno Guizi, Bruno Oliveira, Bruno Ramos, Caio Nascimento, César Almeida, Christiano Morais, Clara Battesini, Cleber Santos, Daniel Haas, Danilo Segura, Dartz Dartz, David Kwast, Delton Porfiro, Dhyeives Rodovalho, Diego Guimarães, Dilenon Delfino, Donivaldo Sarzi, Douglas Bastos, Douglas Braga, Douglas Martins, Douglas Zickuhr, Emerson Rafael, Eric Niens, Érico Andrei, Eugenio Mazzini, Euripedes Borges, Evandro Avellar, Fabiano Gomes, Fabio Barros, Fábio Barros, Fabio Castro, Fábio Thomaz, Felipe Rodrigues, Fernanda Prado, Flávio Meira, Flavkaze Flavkaze, Franklin Silva, Gabriel Barbosa, Gabriel Simonetto, Geandreson Costa, Guilherme Felitti, Guilherme Gall, Guilherme Ostrock, Gustavo Dettenborn, Heitor Fernandes, Henrique Junqueira, Igor Taconi, Ismael Ventura, Israel Gomes, Italo Silva, Jair Andrade, Janael Pinheiro, João Lugão, Johnny Tardin, Jonatas Leon, Jonatas Oliveira, Jônatas Silva, Jorge Plautz, Jose Mazolini, Juan Gutierrez, Juliana Machado, Julio Silva, Kaio Peixoto, Kaneson Alves, Leandro Miranda, Leonardo Cruz, Leonardo Mello, Leonardo Nazareth, Lucas Adorno, Lucas Mello, Lucas Mendes, Lucas Oliveira, Lucas Polo, Lucas Praciano, Lucas Teixeira, Lucas Valino, Luciano Silva, Luciano Teixeira, Luiz Junior, Luiz Lima, Maiquel Leonel, Marcelino Pinheiro, Marcelo Matte, Márcio Martignoni, Marco Mello, Marco Yamada, Maria Clara, Marina Passos, Mario Deus, Matheus Silva, Matheus Vian, Murilo Andrade, Murilo Cunha, Murilo Viana, Natan Cervinski, Nicolas Teodosio, Osvaldo Neto, Patricia Minamizawa, Patrick Brito, Paulo Tadei, Pedro Henrique, Pedro Pereira, Peterson Santos, Priscila Santos, Rafael Barbosa, Rafael Lopes, Rafael Romão, Ramayana Menezes, Reinaldo Silva, Renan Moura, Renato Veirich, Richard Nixon, Riverfount Riverfount, Rodrigo Ferreira, Rodrigo Freire, Rodrigo Junior, Rodrigo Vaccari, Rogério Sousa, Ronaldo Silva, Rui Jr, Samanta Cicilia, Sara Selis, Thiago Araujo, Thiago Borges, Thiago Bueno, Thiago Curvelo, Thiago Moraes, Tony Dias, Victor Wildner, Vinícius Bastos, Vinicius Figueiredo, Vítor Gomes, Vitor Luz, Vlademir Souza, Vladimir Lemos, Wellington Abreu, Wesley Mendes, William Alves, Willian Lopes, Wilson Neto, Yury Barros

Obrigado você

Que raios é isso?

Debug ar

A arte de descobrir QQ tácontecenu

Debug

O que você faz quando o código não funciona como você esperava que ele funcionasse?

Pergunta!

Soluções

1. **print**

```
1   match (variavel):
2       case "batata":
3           print("Entrou aqui")
4           chama_alguém()
5           print("Deu bom")
6       case "xpto":
7           print("??????????")
```

Soluções

1. print
2. **Chama alguém**

Soluções

1. print
2. Chama alguém
3. **logs**

```python
from logging import getLogger

logger = getLogger()

match (variavel):
    case "batata":
        logger.debug("Entrei na batata")
        chama_alguém()
        logger.debug("Deu bom!")
    case "xpto":
        logger.debug("???????")
```

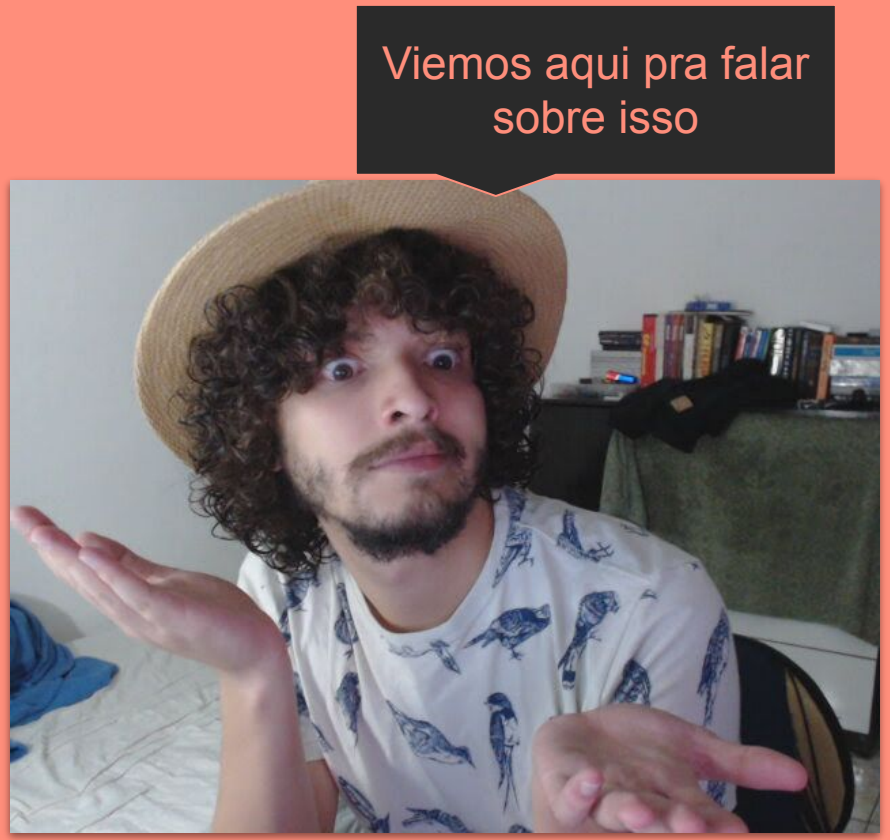
Soluções

1. print
2. Chama alguém
3. logs
4. **trace**

```
$ python -mtrace --trace seu_arquivo.py
```

Soluções

1. print
2. Chama alguém
3. logs
4. trace
5. **debug**



Hora de abandonar os prints

Ferramentas

Quais ferramentas podemos usar?

- Python DeBuger (**PDB**)
  - Embutido
- IPython DeBuger (**IPDB**)
  - pip install ipdb
- Remote Python DeBuger (**RPDB**)
  - pip install rpdb
  - Debuger remoto, via netcat
- Web Python DeBuger (**Web PDB**)
  - pip install web_pdb.server
  - Debuger remoto utilizando o browser
- **PySnooper**
  - pip install pysnooper
- Integrações de **IDEs**
- ...

Vamos do começo [exemplo_01.py]

```python
def formatação(quem, prog, n):
  return '{quem} está apresentando a {prog} #{n}'


nome = 'Eduardo'
programa = 'Live de Python'
numero = '197'

formatado = formatação(nome, programa, numero)

# TESTE!
assert formatado == 'Eduardo está apresentando a Live de Python #197'
```

O debuger em sua forma mais simples

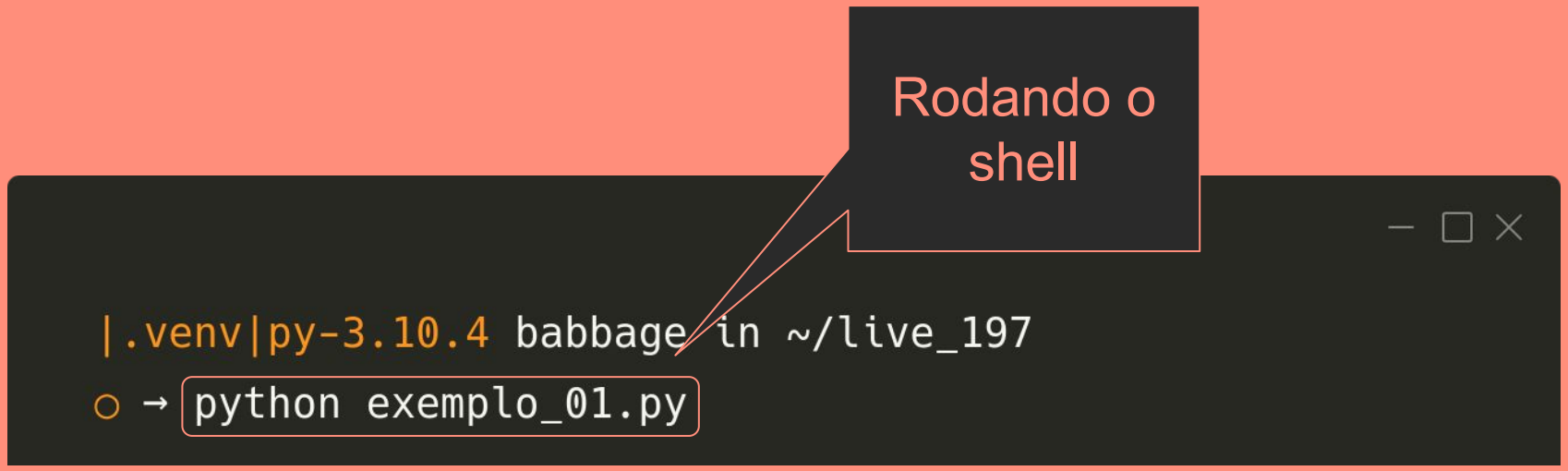
```
breakpoint()
```

Colocando no código

```
1   def formatação(quem, prog, n):
2     return '{quem} está apresentando a {prog} #{n}'
3
4
5   nome = 'Eduardo'
6   programa = 'Live de Python'
7   numero = '197'
8
9   breakpoint()
10
11  formatado = formatação(nome, programa, numero)
12
13  # TESTE!
14  assert formatado == 'Eduardo está apresentando a Live de Python #197'
```

Pare na linha 9

O resultado da arte



O resultado da arte

```
|.venv|py-3.10.4 babbage in ~/live_197
○ → python exemplo_01.py

> /home/dunossauro/live_197/exemplo_01.py(11)<module>()
-> formatado = formatação(nome, programa, numero)
(Pdb)
```

Arquivo

O resultado da arte

```
|.venv|py-3.10.4 babbag                                    7
○ → python exemplo_01.

> /home/dunossauro/live_197/exemplo_01.py(11)<module>()
-> formatado = formatação(nome, programa, numero)
(Pdb)
```

Conteúdo da linha

Número da linha atual

O resultado da arte

```
|.venv|py-3.10.4 babbage in ~/live_197
○ → python exemplo_01.py

> /home/dunossauro/live_197/exemplo_01.py(11)<module>()
-> formatado = formatação(nome, programa, numero)
(Pdb)
```

Shell do debugger

Shell

```
(Pdb) 1 + 1
2

(Pdb) '.'.join('abcd')
'a.b.c.d'

(Pdb) [x * 2 for x in [1, 2, 3, 4]]
[2, 4, 6, 8]

(Pdb) nome
'Eduardo'

(Pdb)
```

Variável do nosso arquivo

Os comandos do debugger

Existe uma série de comandos que podemos usar no debugger pra facilitar nossa vida. Os comandos tem a sua forma completa e a forma abreviada. Um exemplo:

h(elp)

h

Abreviada

help

Completo

Vamos rodar o Help, pra ver o que rola

```
(Pdb) h

Documented commands (type help <topic>):
========================================
EOF     c          d        h         list      q        rv       undisplay
a       cl         debug    help      ll        quit     s        unt
alias   clear      disable  ignore    longlist  r        source   until
args    commands   display  interact  n         restart  step     up
b       condition  down     j         next      return   tbreak   w
break   cont       enable   jump      p         retval   u        whatis
bt      continue   exit     l         pp        run      unalias  where

Miscellaneous help topics:
==========================
exec  pdb

(Pdb)
```

Os comandos básicos do debugger

- **l(ist)**
  - mostra 10 linhas de código para entendermos o contexto do breakpoint, as 5 anteriores e as 5 posteriores
  - Se usado com parâmetros pode nos mostrar outras partes do arquivo
    - **l 10**: coloca a linha 10 no centro
    - **l 55,99**: Mostra as linhas de 55 a 99
- **ll (longlist)**
  - Mostra todo o contexto que estamos, por exemplo, o corpo completo de uma função

Os comandos básicos do debugger

- **n(ext)**
  - Avança o debugger para a próxima linha e a executa.
- **s(tep)**
  - Entra no bloco, caso seja um bloco
    - Uma chamada de função, execução de método, etc...
- **whatis**
  - Diz o tipo de algum objeto
- **c(cont(inue))**
  - Avança o debuger até o próximo breakpoint
- **q(uit) / exit**
  - Saí do debugger

Não são do debuger, ma salvam a nossa pele direto:

- vars()
- globals()
- type()

Outros comandos

Evite usar variáveis de 1 único caracter

use p ou pp quando isso acontecer

Mais algumas dicas

Locomoção na pilha de execução [exemplo_02.py]

- **w(here)**
  - Diz onde estamos, em qual arquivo, de qual módulo em qual linha, etc...

Locomoção na pilha de execução [exemplo_02.py]

```
1   def função_a():
2     return ...
3
4
5   def função_b():
6     return função_a()
7
8
9   def função_c():
10    return função_b()
11
12
13  def função_d():
14    return função_c()
15
16
17  breakpoint()
18  função_d()
```

Locomoção na pilha de execução [exemplo_02.py]

- **w(here)**
  - Diz onde estamos, em qual arquivo, de qual módulo em qual linha, etc...
- **u(p)**
  - Sobe um nível na pilha
  - Se estivermos em uma função, ele vai na linha onde ela foi chamada
- **d(own)**
  - Desce um nível na pilha
  - Caso tenhamos subido pra ver que chamou, mas voltar para o corpo da função

Meu deus, não consigo ler isso

pip install ipdb

PYTHONBREAKPOINT

A variável de ambiente `PYTHONBREAKPOINT` é responsável por alterar o comportamento do python em relação ao debuger.

- PYTHONBREAKPOINT=0
  - Desativa os breakpoints, caso alguém esqueça em produção
- PYTHONBREAKPOINT=**seu_debuger**.set_trace
  - Troca o debuger para outra opção.

Uso

```
$ PYTHONBREAKPOINT=seu_debuger.set_trace python seu_script.py

# ipdb
$ PYTHONBREAKPOINT=ipdb.set_trace python seu_script.py

# rpdb
$ PYTHONBREAKPOINT=rpdb.set_trace python seu_script.py

# WEB
$ PYTHONBREAKPOINT=web_pdb.set_trace python seu_script.py

# Desativar
$ PYTHONBREAKPOINT=0 python seu_script.py
```

O que fazer em momentos de desespero!

Deu ruim

O caso Jupyter [jupyter notebook]

Python debuger

In [ ]:
```
def formatação(quem, prog, n):
    return '{quem} está apresentando a {prog} #{n}'


nome = 'Eduardo'
programa = 'Live de Python'
numero = '197'

import pdb; pdb.set_trace() # pode ser feito em duas linhas tbm

formatado = formatação(nome, programa, numero)
```

Ipython debuger

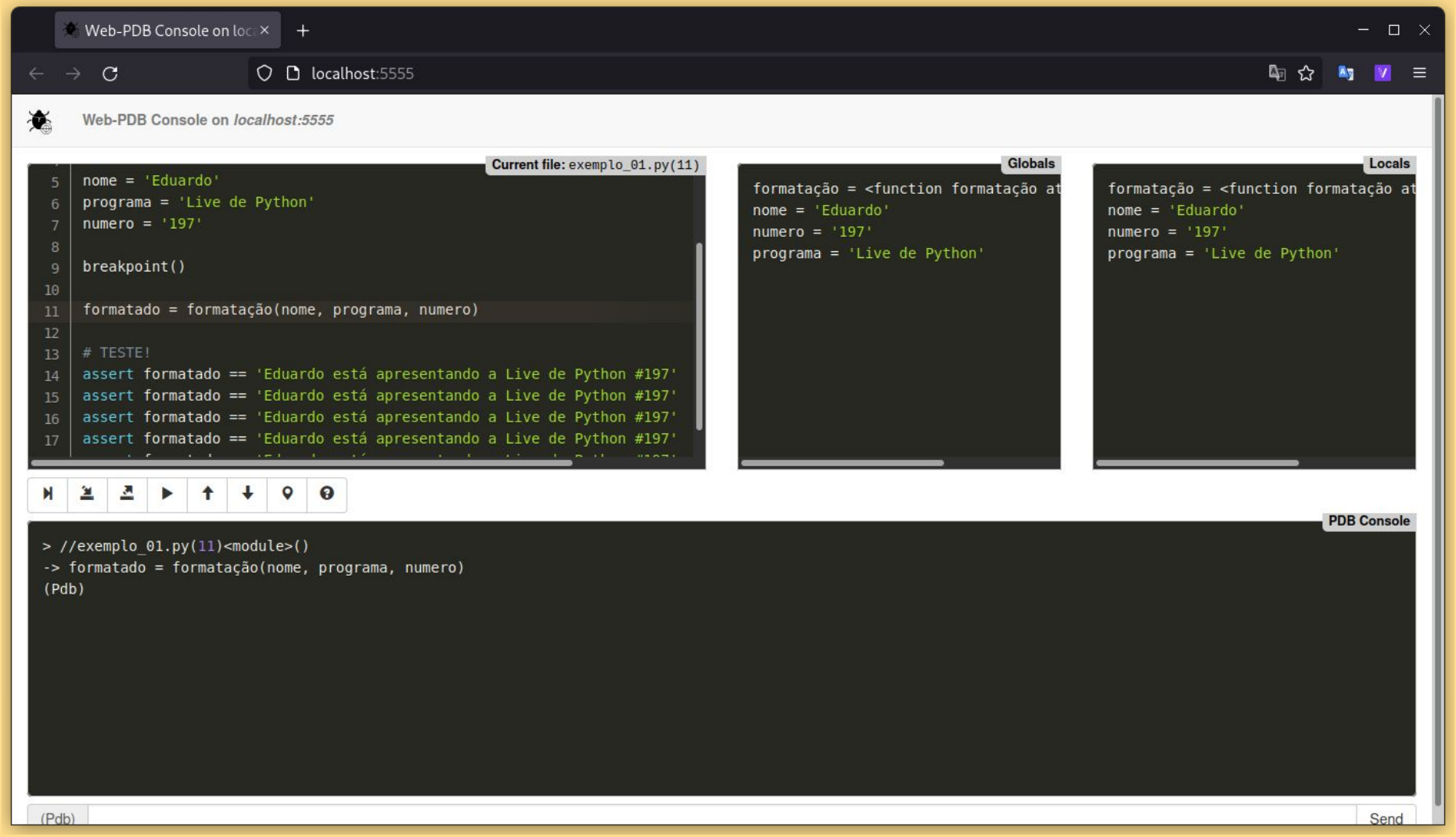
In [ ]:
```
def formatação(quem, prog, n):
    return '{quem} está apresentando a {prog} #{n}'


nome = 'Eduardo'
programa = 'Live de Python'
numero = '197'

from IPython.core.debugger import set_trace
set_trace()

formatado = formatação(nome, programa, numero)
```

O caso dos containers

```
FROM python:3.10

COPY ./exemplo_01.py .
RUN pip install rpdb web_pdb

# Web
EXPOSE 5555
ENV PYTHONBREAKPOINT=web_pdb.set_trace

# Remoto
EXPOSE 4444
ENV PYTHONBREAKPOINT=rpdb.set_trace

CMD [ "python", "exemplo_01.py" ]
```

Bora testar

```
buildah bud -t live197 .
podman run -p 5555:5555 live197
```



http://localhost:5555/

Stack

E os traces

Voltando a stack [exemplo_03.py]

```python
import sys

def my_trace_function(frame, event, arg):
  print(
    frame.f_lineno,
    frame.f_code.co_name,
    frame.f_locals,
    event,
  )
  return my_trace_function

sys.settrace(my_trace_function)
... # O fluxo que vai entrar em trace
```

Voltando a stack [exemplo_03.py]

```python
import sys

def my_trace_function(frame, event, arg):
  print(
    frame.f_lineno,
    frame.f_code.co_name,
    frame.f_locals,
    event,
  )
  return my_trace_function

sys.settrace(my_trace_function)
... # O fluxo que vai entrar em trace
```

A saída

```
16 primeira_função {} call
17 primeira_função {} line
18 primeira_função {'lista_original': [1, 2, 3, 4]} line
20 primeira_função {'lista_original': [1, 2, 3, 4], 'nova_lista': []} line
21 primeira_função {'lista_original': [1, 2, 3, 4], 'nova_lista': [], 'valor': 1} line
24 segunda_função {'valor': 1} call
25 segunda_função {'valor': 1} line
26 segunda_função {'valor': 1, 'retorno': 2} line
26 segunda_função {'valor': 1, 'retorno': 2} return
20 primeira_função {'lista_original': [1, 2, 3, 4], 'nova_lista': [2], 'valor': 1} line
```

picpay.me/dunossauro

apoia.se/livedepython

pix.dunossauro@gmail.com

Ajude o projeto <3