

# Atividade S3 A1

**Nome:** Adalberto Caldeira Brant Filho

**Repositório GitHub:** <https://github.com/adalbertobrant/lipai>

## Código das Videoaulas

### Depuração em Python com PDB e VS Code

Utilizando o Pdb

```
""" Aula 01 - Debug """

def somar(n1, n2, n3):
    """ retorna a soma das notas """
    soma = n1 + n2 + n3
    return soma

def calcular_media(n1, n2, n3):
    """ calcula a média das notas """
    soma = somar(n1, n2, n3)
    mean = soma / 3
    return mean

breakpoint() // cria um ponto de parada

NOTA1 = 10.0
NOTA2 = 3.0
NOTA3 = 5.5

media = calcular_media(NOTA1, NOTA2, NOTA3)

print(media)
```

Ao criar o comando breakpoint() no Python, abre-se a possibilidade de conseguir através do Python debug, usar uma série de comandos que podem ser visualizados no terminal pdb:

```
# comando help
# faz a listagem de comandos possíveis de se utilizar no pdb

# comando next
# vai para o próximo comando

# chamada de variável
# pode-se chamar uma variável após um comando como o next, pois ela já está
# dentro da memória, bastando usar o nome da variável no pdb, ajuda a inspecionar
# valores
```

```
# comando step
# serve para entrar dentro de uma chamada de função

# comando where
# mostra aonde o cursor está no momento e a pilha de execução callstack
```

The screenshot shows the VS Code interface with the Python extension installed. The Explorer sidebar shows a project structure with 'src' and 'aula01.py'. The code editor has 'aula01.py' open, with a breakpoint at line 17. The terminal window shows the help command for the Pdb debugger:

```
(base) alberto@alberto-X751L:~/github/lipai/atividades/estudo-python/src/05-debug$ python3 aula01.py
EOF  cl   disable  ignore  n    return  u    where
a    clear  display  interact  next  retval  unalias
alias commands  down  j    p    run  undisplay
args condition  enable  jump  pp  rv  unt
b    cont  exceptions  i    q    s    until
break continue  exit  list  quit  source  up
bt  d    h    ll  r    step  w
c    debug  help  longlist  restart  tbreak  whatis
Miscellaneous help topics:
=====
exec  pdb
```

O comando breakpoint é uma função em python que ajuda a fazer o debug do código , na figura vemos o comando breakpoint() na linha 17 e a chamada do comando help no terminal pdb

The screenshot shows the VS Code interface with the Python extension installed. The Explorer sidebar shows a project structure with 'src' and 'aula01.py'. The code editor has 'aula01.py' open, with a breakpoint at line 17. The terminal window shows the execution of the script and the pdb help command:

```
(base) alberto@alberto-X751L:~/github/lipai/atividades/estudo-python/src/05-debug$ python3 aula01.py
EOF  cl   disable  ignore  n    return  u    where
a    clear  display  interact  next  retval  unalias
alias commands  down  j    p    run  undisplay
args condition  enable  jump  pp  rv  unt
b    cont  exceptions  i    q    s    until
break continue  exit  list  quit  source  up
bt  d    h    ll  r    step  w
c    debug  help  longlist  restart  tbreak  whatis
Miscellaneous help topics:
=====
exec  pdb
```

Then, the next command is executed:

```
(Pdb) next
> /home/alberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(19)<module>()
-> NOTA1 = 10.0
(Pdb) next
> /home/alberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(20)<module>()
-> NOTA2 = 3.0
(Pdb) next
> /home/alberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(21)<module>()
-> NOTA3 = 5.5
```

O comando next do pdb ajuda a ver linha a linha a execução do programa , pois o mesmo é executado linha a linha, após ser executada uma variável pode-se chamar a mesma para ver o seu valor

```

(base) adalberto@adalberto-X751L1:~/github/lipai/atividades/estudo-python/src/05-debug$ python3 aula01.py
> /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(23)<module>()
-> breakpoint()
(Pdb) next
> /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(24)<module>()
-> media = calcular_media(NOTA1, NOTA2, NOTA3)
(Pdb) step
--Call--
> /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(10)calcular_media()
-> def calcular_media(n1, n2, n3):
(Pdb) where
/home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(24)<module>()
-> media = calcular_media(NOTA1, NOTA2, NOTA3)
> /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula01.py(10)calcular_media()
-> def calcular_media(n1, n2, n3):
(Pdb) 

```

O comando step entra dentro de uma função e é usado em conjunto com o comando next , o comando where mostra a pilha de execução apontando a linha do programa principal, e depois a sua chamada de função como mostra a figura acima

O comando step dá sempre um passo enquanto que o comando next vai seguindo a pilha de execução principal do programa ignorando subrotinas.

## Aula 02 - Depuração com o VSCode

No VSCode para utilizar o breakpoint é necessário clicar com o botão direito do mouse do lado esquerdo da numeração da linha do código.

```

"" Aula - 02 Uso do VSCode para Debug do código """
def somar(n1, n2, n3):
    """ retorna a soma das notas """
    soma = n1 + n2 + n3
    return soma

def calcular_media(n1, n2, n3):
    """ calcula a média das notas """
    soma = somar(n1, n2, n3)
    mean = soma / 3
    return mean

NOTA1 = 10.0
NOTA2 = 3.0
NOTA3 = 5.5

media = calcular_media(NOTA1, NOTA2, NOTA3)
print(media)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Leaving functions creating breakpoints in production code is not recommended Pylint[W1515:forgotten-debug-statement] [Ln 23, Col 1]

Line too long (117/100) Pylint[C0301:line-too-long] [Ln 17, Col 1]

## Deve-se clicar em Run and Debug

```
dez 16 10:00
File Edit Selection View Go Run Terminal Help
src > 05-debug > aula02.py > calcular_media
1 """ Aula - 02 Uso do VSCode para Debug do código """
2
3
4 def somar(n1, n2, n3):
5     """ retorna a soma das notas """
6     soma = n1 + n2 + n3
7     return soma
8
9
10 def calcular_media(n1, n2, n3):
11     """ calcula a média das notas """
12     soma = somar(n1, n2, n3)
13     mean = soma / 3
14     return mean
15
16
17 NOTA1 = 10.0
18 NOTA2 = 3.0
19 NOTA3 = 5.5
20
21 media = calcular_media(NOTA1, NOTA2, NOTA3)
22
23 print(media)
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

BREAKPOINTS  
Raised Exceptions  
Uncought Exceptions  
User Uncaught Exceptions  
aula02.py src/05-debug

Ln 14, Col 16 Spaces: 4 UTF-8 {} Python 3.12.3 64-bit

Para ir para a próxima execução de código usa-se a tecla de atalho F10 ou então clica na seta que se chama step over como indica a figura abaixo:

```
dez 16 10:08
File Edit Selection View Go Run Terminal Help
src > 05-debug > aula02.py > NOTA1
10 def calcular_media(n1, n2, n3):
11     """ calcula a média das notas """
12     soma = somar(n1, n2, n3)
13     mean = soma / 3
14     return mean
15
16
17 NOTA1 = 10.0
18 NOTA2 = 3.0
19 NOTA3 = 5.5
20
21 media = calcular_media(NOTA1, NOTA2, NOTA3)
22
23 print(media)
24
```

VARIABLES  
Locals  
special variables  
function variables  
Globals  
special variables  
function variables

WATCH

CALL STACK  
<modul> aula02.py [17:1]

BREAKPOINTS  
Raised Exceptions  
Uncought Exceptions  
User Uncaught Exceptions  
aula02.py src/05-debug

Python Debug Console

Ln 17, Col 1 Spaces: 4 UTF-8 {} Python 3.12.3 64-bit

Ao clicar em step over notamos que a NOTA1=10 aparece como variável local e o debug indica que está na próxima linha como podemos notar pela figura abaixo:

```

10 def calcular_media(n1, n2, n3):
11     """ calcula a média das notas """
12     soma = somar(n1, n2, n3)
13     mean = soma / 3
14     return mean
15
16
17 NOTA1 = 10.0
18 NOTA2 = 3.0
19 NOTA3 = 5.5
20
21
22 media = calcular_media(NOTA1, NOTA2, NOTA3)
23
24 print(media)
25

```

Existe também um espaço no VSCode chamado debug console, onde podemos acessar as variáveis já em memória ou que já foram executadas pelo debug , como indica a figura a seguir:

```

14     return mean
15
16
17 NOTA1 = 10.0
18 NOTA2 = 3.0
19 NOTA3 = 5.5
20
21
22 media = calcular_media(NOTA1, NOTA2, NOTA3)
23
24 print(media)
25

```

DEBUG CONSOLE

- NOTA1: 10.0
- NOTA2: 3.0
- NOTA3: 5.5

Ao executarmos o step over na próxima linha que será o cálculo da média notamos que é retornado o valor do cálculo da média e também o retorno da função calcular\_media(NOTA1, NOTA2, NOTA3), isso indica que o step over não entra dentro da função e teremos que usar outra ferramenta visual do debug para ver o que está se passando dentro da função calcular\_media(), observe a figura abaixo:

```

    dez 16 10:25
    Welcome aula01.py :: I breakpoint
    src > 05-debug > aula02.py > ...
    14 |     return mean
    15 |
    16 |● 17 NOTA1 = 10.0
    17 | 18 NOTA2 = 3.0
    18 | 19 NOTA3 = 5.5
    19 |
    20 |
    21 |
    22 media = calcular_media(NOTA1, NOTA2, NOTA3)
    23 |
    24 print(media)
    25

    PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
    NOTA1 18.0
    NOTA2 3.0
    NOTA3 5.5

```

Indica o retorno da função calcular\_media

Para conseguir acessar o que acontece dentro da função calcular\_media() usamos o step into que é a tecla F11 do VSCode, essa tecla possibilita o acesso a função calcular\_media(), veja a figura:

```

    dez 16 10:29
    Welcome aula01.py :: I breakpoint
    src > 05-debug > aula02.py > media
    14 |     return mean
    15 |
    16 |
    17 |● 17 NOTA1 = 10.0
    18 | 18 NOTA2 = 3.0
    19 | 19 NOTA3 = 5.5
    20 |
    21 |
    22 media = calcular_media(NOTA1, NOTA2, NOTA3)
    23 |
    24 print(media)
    25

    PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
    /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy/2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 60899 -- /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula02.py
    (base) adalberto@adalberto-X51LJ:~$ /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy/2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 60899 -- /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula02.py
    (base) adalberto@adalberto-X51LJ:~$ cd /home/adalberto/github/lipai/atividades/estudo-python ; /usr/bin/env /bin/python3 /home/adalberto/.vscode-oss/extensions/ms-python.debugpy/2025.14.1-linux-x64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 37293 -- /home/adalberto/github/lipai/atividades/estudo-python/src/05-debug/aula02.py

```

Step into

Paused on step

O call stack então nos mostra a pilha de execução do nosso programa e também mostra as linhas que estão sendo executadas.

The screenshot shows the VS Code interface with a Python debugger session. The code editor displays two files: 'aula01.py' and 'aula02.py'. In 'aula02.py', line 12 is highlighted. The 'CALL STACK' section in the sidebar shows the execution path: 'aula01.py' at line 12, which calls 'aula02.py' at line 22. The terminal window shows the command line and the current directory. A red arrow points from the 'CALL STACK' section to the word 'CALLSTACK' written in red ink below the terminal window.

O Debug do python tanto na parte do terminal que é mais direta quanto na parte visual é um auxiliar importante para a depuração e melhoria do código, evitando utilizar prints

## Reflexão\_debug.md

- O que você conseguiu enxergar no debugger que não ficava tão claro apenas executando o programa normalmente (sem depuração)?

R: O pdb ajuda a entender o fluxo e estrutura do código melhor do que apenas ver onde está o erro, como normalmente se utiliza ao fazer o ato de degub usando o comando print

- Em quais situações você acha mais prático usar o pdb pela linha de comando e em quais prefere o debugger visual do VS Code?

R: Na linha de comando é mais fácil senão tivermos acesso a um sistema visual, não notei até o momento muitas diferenças entre um e outro, talvez no VS Code seja relativo a questões onde eu tenha que ver a callstack completa do programa

- Houve algum erro ou comportamento inesperado nos seus exercícios que você só percebeu por causa da depuração? Descreva brevemente.

R: Sim, chamada de pilhas muitas vezes podem ser complexas e podendo ver a questão da depuração passo a passo e o desvio do fluxo da execução me ajuda a entender melhor onde está o bug.