



Linguagem de programação II

Aula 02: Testes





Testes automatizados



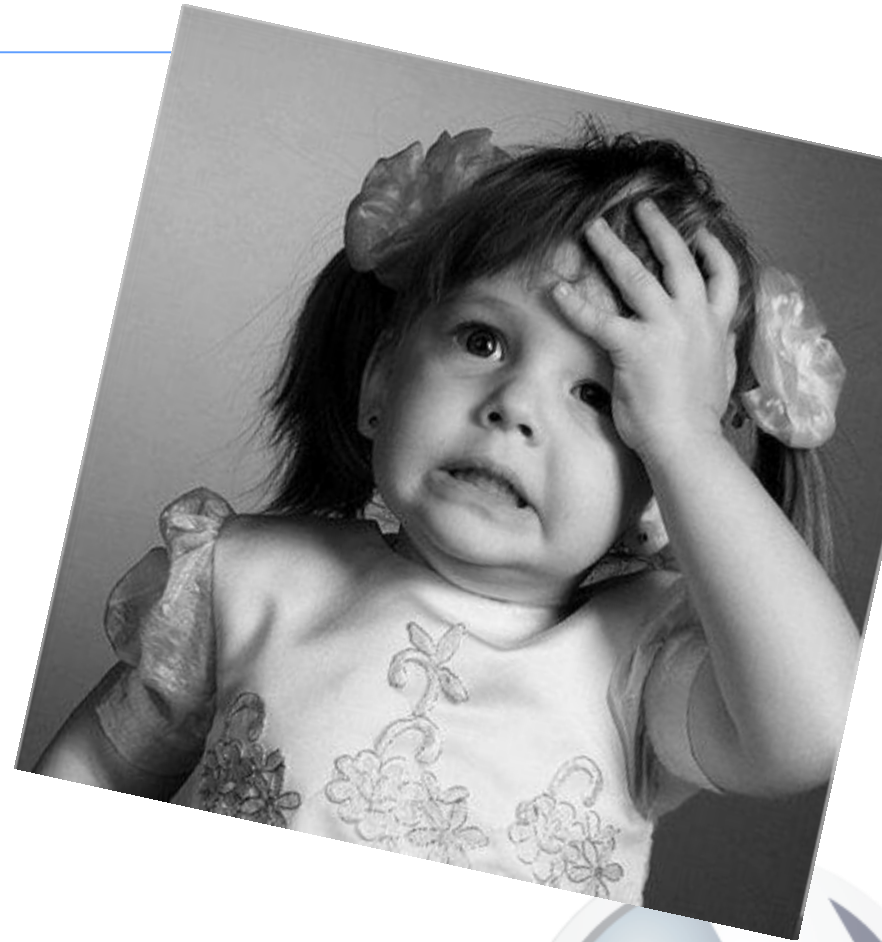


Por que testar ?



Faculdade
IMPACTA
TECNOLOGIA

Seres humanos erram !





Programas contém
erros!



Sintaxe

Erros em programas



Lógica



Software robusto deve conter o
mínimo possível de erros.





Bugs podem causar catástrofes ou
prejuízos financeiros.





Como testar?



Testes manuais


- cansativo
- você vai testar alguns casos
- você vai testar apenas algumas vezes.



QUALiTY

A hand holding a blue eraser is positioned in the center. Two blue curved lines, one above and one below the hand, form a partial frame. Black arrows indicate a clockwise cycle: one arrow points from the top right towards the word 'QUALiTY', another points from the top right towards the bottom right, and a third points from the bottom right towards the bottom left.

Testes automatizados

- Bateria de testes que cobre o máximo possível do seu código
 - Executada rotineiramente várias vezes por dia
- 
- A circular arrow icon, consisting of a grey circle with a white arrow pointing clockwise, is located in the bottom right corner of the slide.



Arcabouço de Testes





- Vamos utilizar o **pytest**:
docs.pytest.org

Installation and Getting Started — pytest documentation - Mozilla Firefox

Installation and Getting Started

Pythons: Python 2.6, 2.7, 3.3, 3.4, 3.5, Jython, PyPy-2.3

Platforms: Unix/Posix and Windows

PyPI package name: [pytest](#)

dependencies: [py](#), [colorama](#) (Windows), [argparse](#) (py26), [ordereddict](#) (py26).

documentation as PDF: [download latest](#)

Installation

Installation:

```
pip install -U pytest
```

To check your installation has installed the correct version:

Table Of Contents

- [Home](#)
- [Contents](#)
- [Install](#)
- [Examples](#)
- [Customize](#)
- [Contact](#)
- [Talks/Posts](#)
- [Changelog](#)
- [Backwards Compatibility](#)
- [License](#)



Instalando o pytest no virtualEnv

1. Abrir um terminal dentro da pasta Scripts do virtualEnv
2. Ativar o virtualEnv com o seguinte comando:
 - a. `<pasta-ambiente-virtual>\Scripts\activate`
O caminho exibido no terminal terá o nome da pasta do seu ambiente virtual
1. Instalar o pytest
 - a. `pip install pytest`





Instalando o pytest no virtualEnv

Suponha que a pasta do meu ambiente virtual seja:

C:\LP11\myPython

```
LP11 - "C:\LP11"
C:\LP11 13/08/2017 8:31:39,82
λ .\myPython\Scripts\activate
(myPython) C:\LP11 13/08/2017 8:32:13,25
λ pip install pytest
Collecting pytest
  Using cached pytest-3.2.1-py2.py3-none-any.whl
Collecting py>=1.4.33 (from pytest)
  Using cached py-1.4.34-py2.py3-none-any.whl
Requirement already satisfied: setuptools in c:\lp11\mypython\lib\site-packages (from pytest)
Collecting colorama; sys_platform == "win32" (from pytest)
  Using cached colorama-0.3.9-py2.py3-none-any.whl
Installing collected packages: py, colorama, pytest
Successfully installed colorama-0.3.9 py-1.4.34 pytest-3.2.1

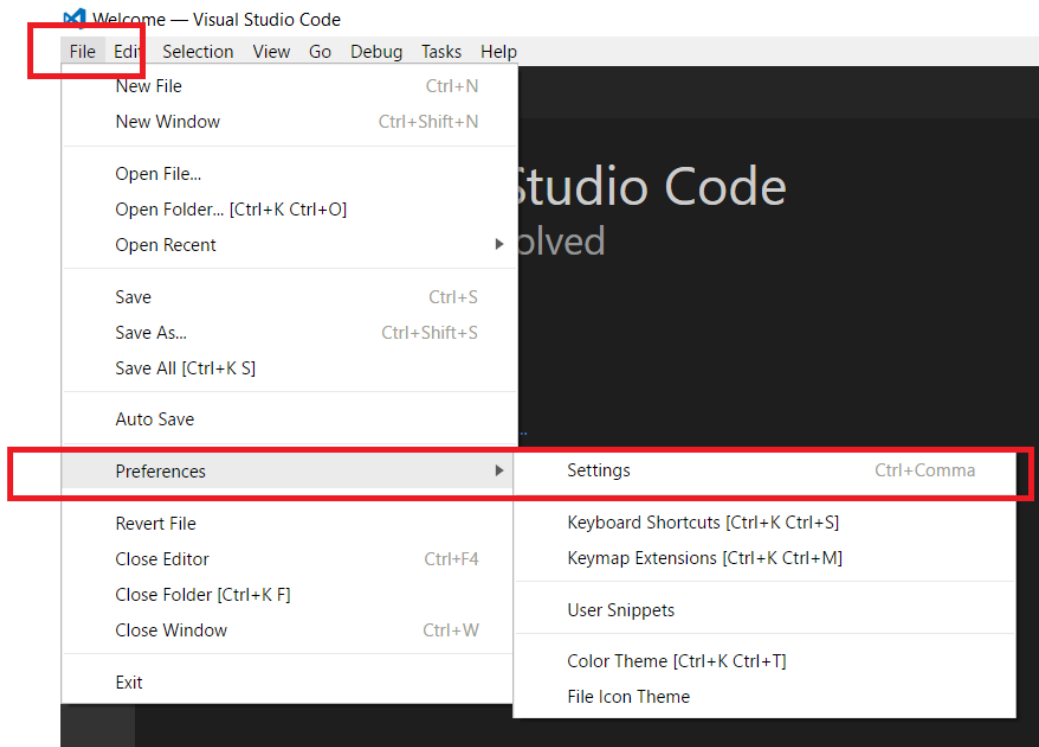
(myPython) C:\LP11 13/08/2017 8:32:36,81
λ pytest --version
This is pytest version 3.2.1, imported from c:\lp11\mypython\lib\site-packages\pytest.py

(myPython) C:\LP11 13/08/2017 8:32:44,98
λ
```



Configurando o pytest no VSCode

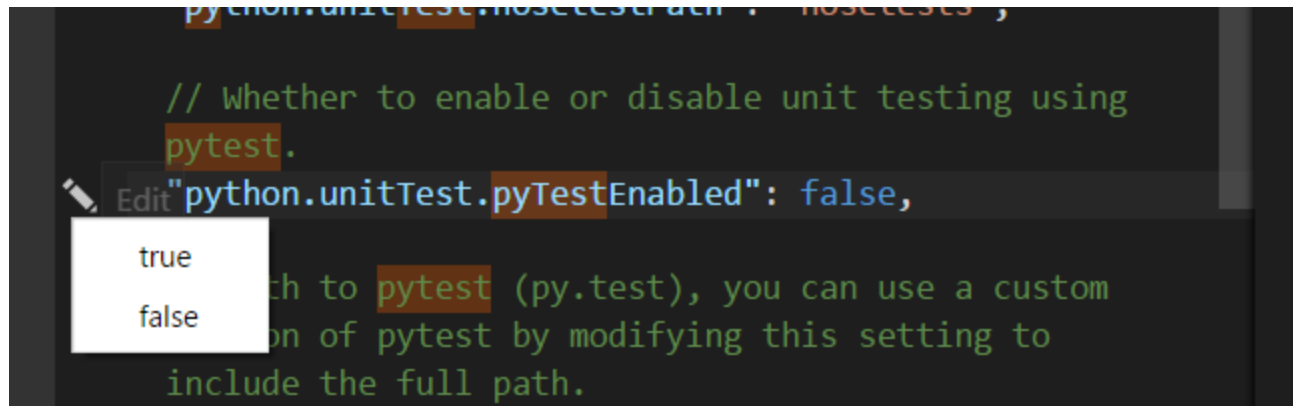
1. Abri o VSCode
2. Selezione o menu File>Preference>Settings





Configurando o pytest no VSCode

3. Em Search Settings digite pytest
4. Selecione python.unitTest.pytestEnabled
 - a. mude para True



```
python.unitTest.noseTestRunner : noseTestRunner ;  
  
// Whether to enable or disable unit testing using  
pytest.  
"python.unitTest.pytestEnabled": false,  
  true  
  false  
  with to pytest (py.test), you can use a custom  
  on of pytest by modifying this setting to  
  include the full path.
```





Configurando o pytest no VSCode

```
// Where to prompt to configure a test framework if
potential tests directories are discovered.
"python.unittest.promptToConfigure": true,

// Port number used for debugging of unittests.
"python.unittest.debugPort": 3000,

// Whether to enable or disable unit testing using
nosetests.
"python.unittest.nosetestsEnabled": false,

// Path to nosetests, you can use a custom version of
nosetests by modifying this setting to include the
full path.
"python.unittest.nosetestPath": "nosetests",

// Whether to enable or disable unit testing using
pytest.
"python.unittest.pyTestEnabled": false,

// Path to pytest (py.test), you can use a custom
version of pytest by modifying this setting to
include the full path.
"python.unittest.pyTestPath": "py.test",

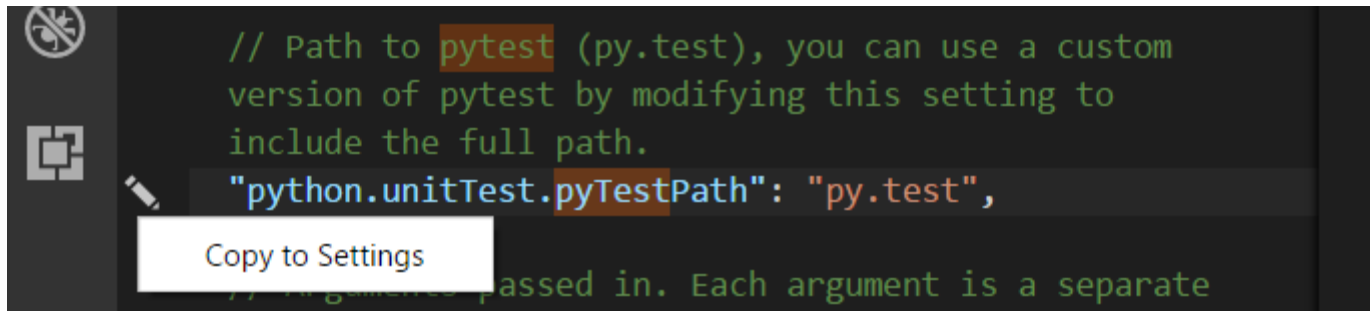
// Arguments passed in. Each argument is a separate
item in the array.
"python.unittest.nosetestArgs": [],
```

```
1 // Place your settings in this file to override
2 {
3     "python.pythonPath": "C:/LPII/myPython/Script
4     "python.unittest.pyTestEnabled": true
5 }
```



Configurando o pytest no VSCode

5. Selecione python.unitTest.pytestPath



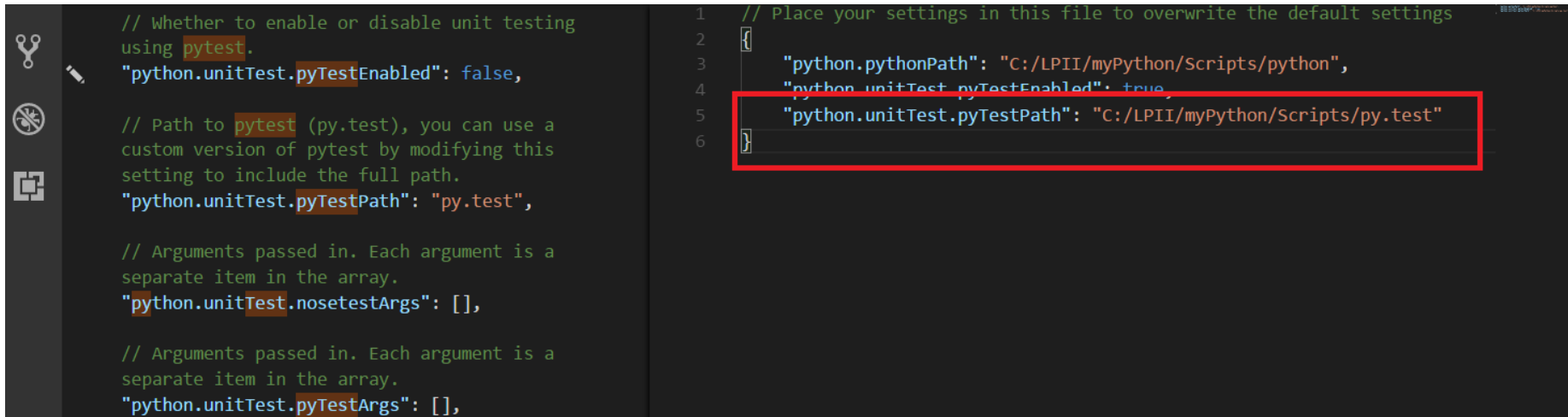
```
// Path to pytest (py.test), you can use a custom  
version of pytest by modifying this setting to  
include the full path.  
"python.unitTest.pyTestPath": "py.test",  
// Arguments passed in. Each argument is a separate
```





Configurando o pytest no VSCode

5. Selecione python.unitTest.pytestPath
6. Inclua o path do virtuaenv



The screenshot shows the VS Code settings editor with the 'python.unitTest.pytestPath' setting highlighted in a red box. The left pane shows the 'python.unitTest.pytestPath' setting set to 'py.test'. The right pane shows the 'python.unitTest.pytestPath' setting set to 'C:/LPII/myPython/Scripts/py.test'.

```
// Whether to enable or disable unit testing using pytest.
"python.unitTest.pytestEnabled": false,

// Path to pytest (py.test), you can use a custom version of pytest by modifying this setting to include the full path.
"python.unitTest.pytestPath": "py.test",

// Arguments passed in. Each argument is a separate item in the array.
"python.unitTest.nosetestArgs": [],

// Arguments passed in. Each argument is a separate item in the array.
"python.unitTest.pytestArgs": [],

1 // Place your settings in this file to overwrite the default settings
2 {
3     "python.pythonPath": "C:/LPII/myPython/Scripts/python",
4     "python.unitTest.pytestEnabled": true,
5     "python.unitTest.pytestPath": "C:/LPII/myPython/Scripts/py.test"
6 }
```




Primeiro Testes





Primeiro teste

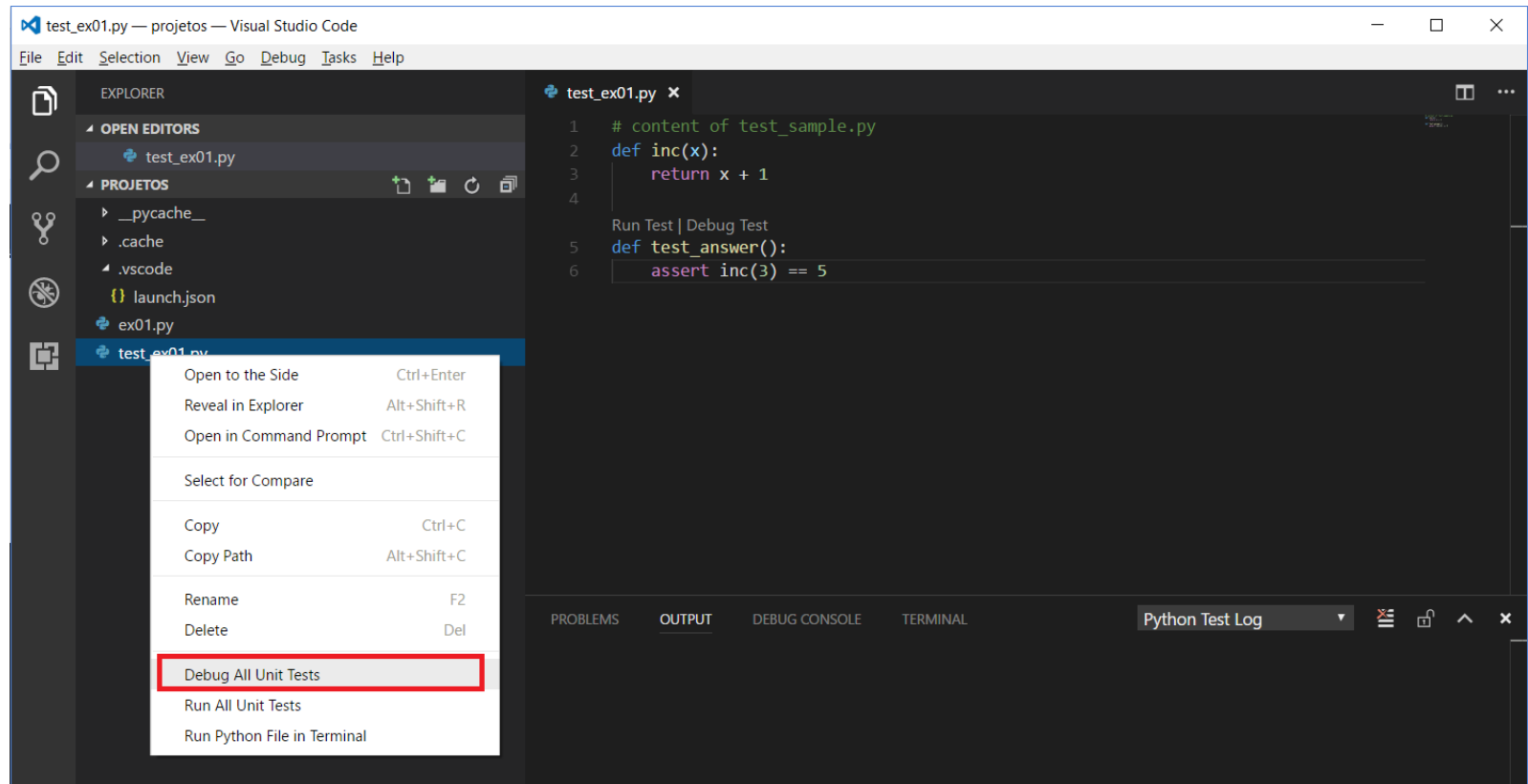
1. Crie um arquivo de teste chamado test_ex01.py
2. Escreva o seguinte código de teste

```
test_ex01.py x
1  # content of test_sample.py
2  def inc(x):
3      return x + 1
4
   Run Test | Debug Test
5  def test_answer():
6      assert inc(3) == 5
```





3. Salve o arquivo
4. Na janela Explorer, clique com o botão direito sobre o arquivo `test_ex01.py` e selecione a opção “Debug All Unit Tests”





5. Verifique pelo output que ocorreu falhas

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  Python Test Log  [Icons]

generated xml file: C:\Users\leo\AppData\Local\Temp\tmp-3428oh3DWY2aXONX.xml -
===== FAILURES =====
test_answer

def test_answer():
>     assert inc(3) == 5
E       assert 4 == 5
E       + where 4 = inc(3)

test_ex01.py:6: AssertionError
===== 1 failed in 0.37 seconds =====

Ln 6, Col 23  Spaces: 4  UTF-8  CRLF  Python  [Icon]
```



Primeiro teste

6. Corrija os erros e execute até conseguir resolver todos os problemas.

The screenshot shows the Visual Studio Code interface with a file named `test_ex01.py` open. The Explorer sidebar on the left shows the project structure, including `test_ex01.py` under the `PROJETOS` folder. The main editor displays the following code:

```
1 # content of test_sample.py
2 def inc(x):
3     return x + 1
4
5 def test_answer():
6     assert inc(3) == 4
```

Below the code editor, the `Python Test Log` panel shows the execution output:

```
===== test session starts =====
platform win32 -- Python 3.6.2, pytest-3.2.1, py-1.4.34, pluggy-0.4.0
rootdir: c:\LPII\projetos, inifile:

collecting 0 items
collecting 1 item
collected 1 item

test_ex01.py .

generated xml file: C:\Users\leo\AppData\Local\Temp\tmp-34283wtym\4ApYba.xml -
===== 1 passed in 0.18 seconds =====
```

The status bar at the bottom indicates the current position is `Ln 6, Col 23`, with `Spaces: 4`, `UTF-8`, `CRLF`, and `Python` selected.

O pytest:

- Considera que arquivos do tipo **test_*.py** e ***_test.py** são arquivos de teste
- Executa funções do tipo **test_***
- Considera classes do tipo **Test***





- Pense cuidadosamente nos casos em que seu programa pode falhar.





- Pense cuidadosamente nos casos em que seu programa pode falhar.
- Pense nos diferentes tipos de entrada que exercitam caminhos diferentes no seu programa.





- Pense cuidadosamente nos casos em que seu programa pode falhar.
- Pense nos diferentes tipos de entrada que exercitam caminhos diferentes no seu programa.
- Pense nos casos diferentes no seu código.





- Pense cuidadosamente nos casos em que seu programa pode falhar.
- Pense nos diferentes tipos de entrada que exercitam caminhos diferentes no seu programa.
- Pense nos casos diferentes no seu código.
- Escreva testes automatizados para todos os casos!





Como utilizar o debugger?





Depurando o programa

1. Crie um arquivo ex02.py com o seguinte código

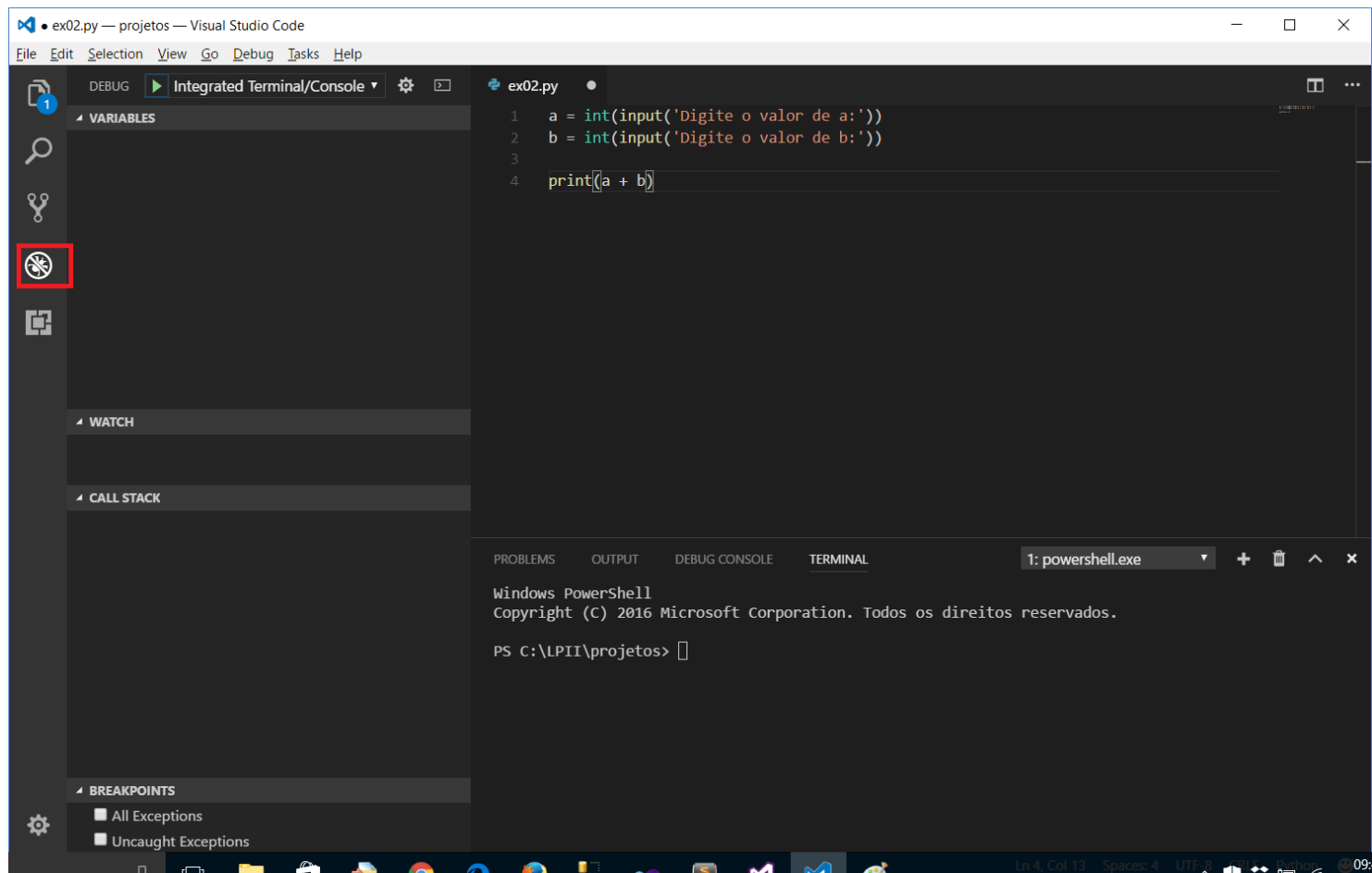
```
ex02.py
1  a = int(input('Digite o valor de a:'))
2  b = int(input('Digite o valor de b:'))
3
4  print(a + b)
```





Depurando o programa

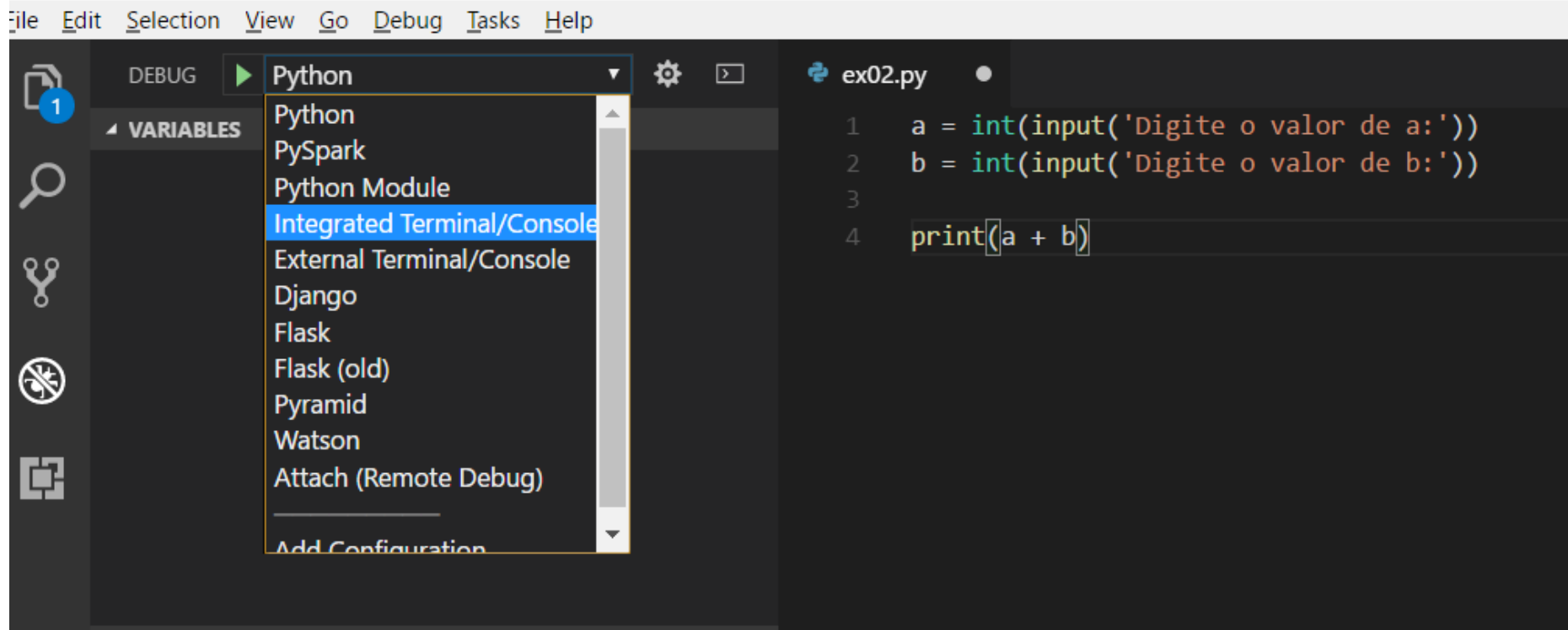
2. Nos ícones laterais selecione a opção debug



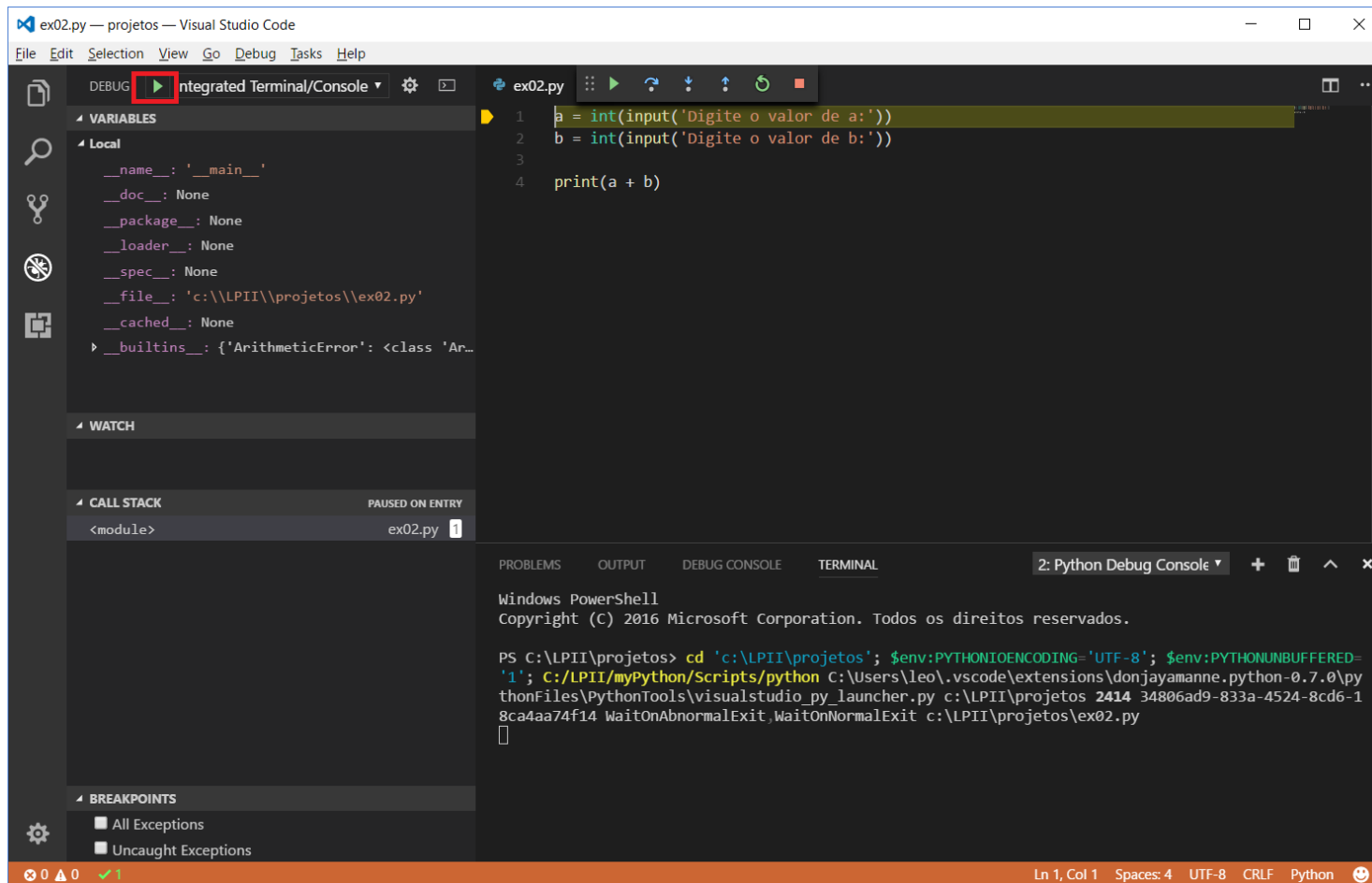


Depurando o programa

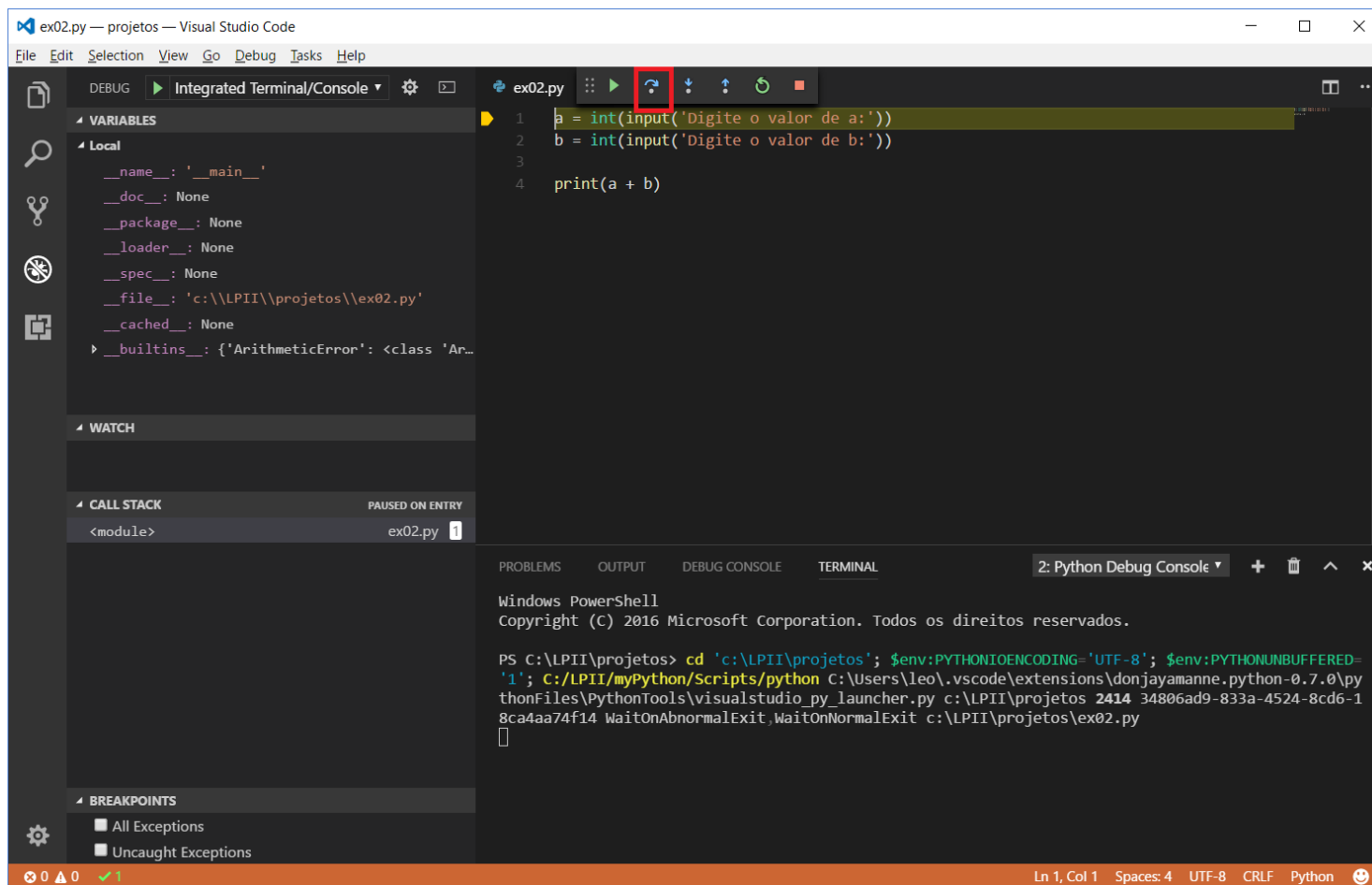
3. Na lista de debugger selecione Integrated Terminal/Console



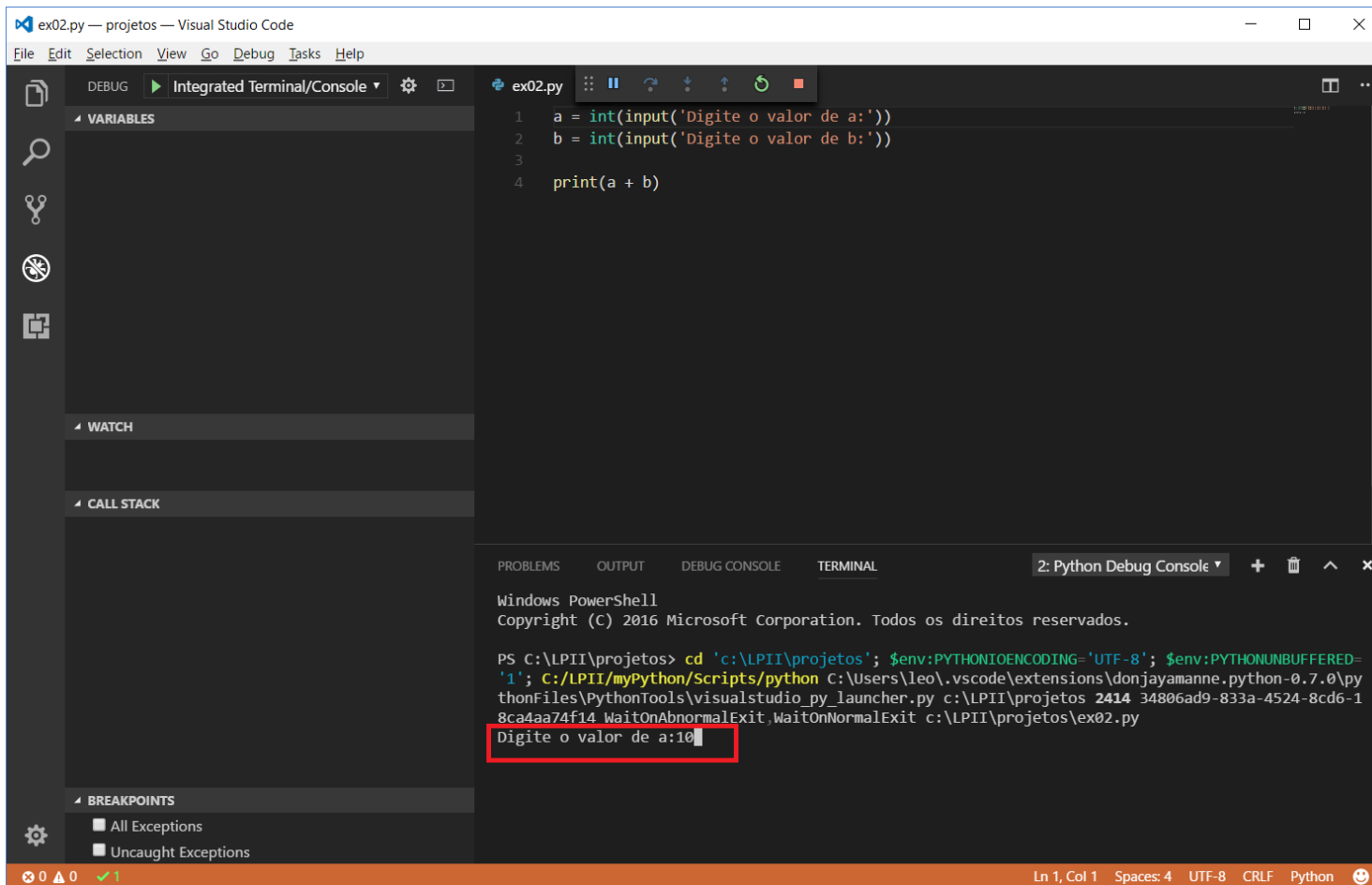
4. Clique em “Start Debugging”



5. Clique em “Step over (F10)”



6. Digite o valor e aperte ENTER



The screenshot shows the Visual Studio Code interface with a Python file named `ex02.py` open. The code is as follows:

```
1 a = int(input('Digite o valor de a:'))
2 b = int(input('Digite o valor de b:'))
3
4 print(a + b)
```

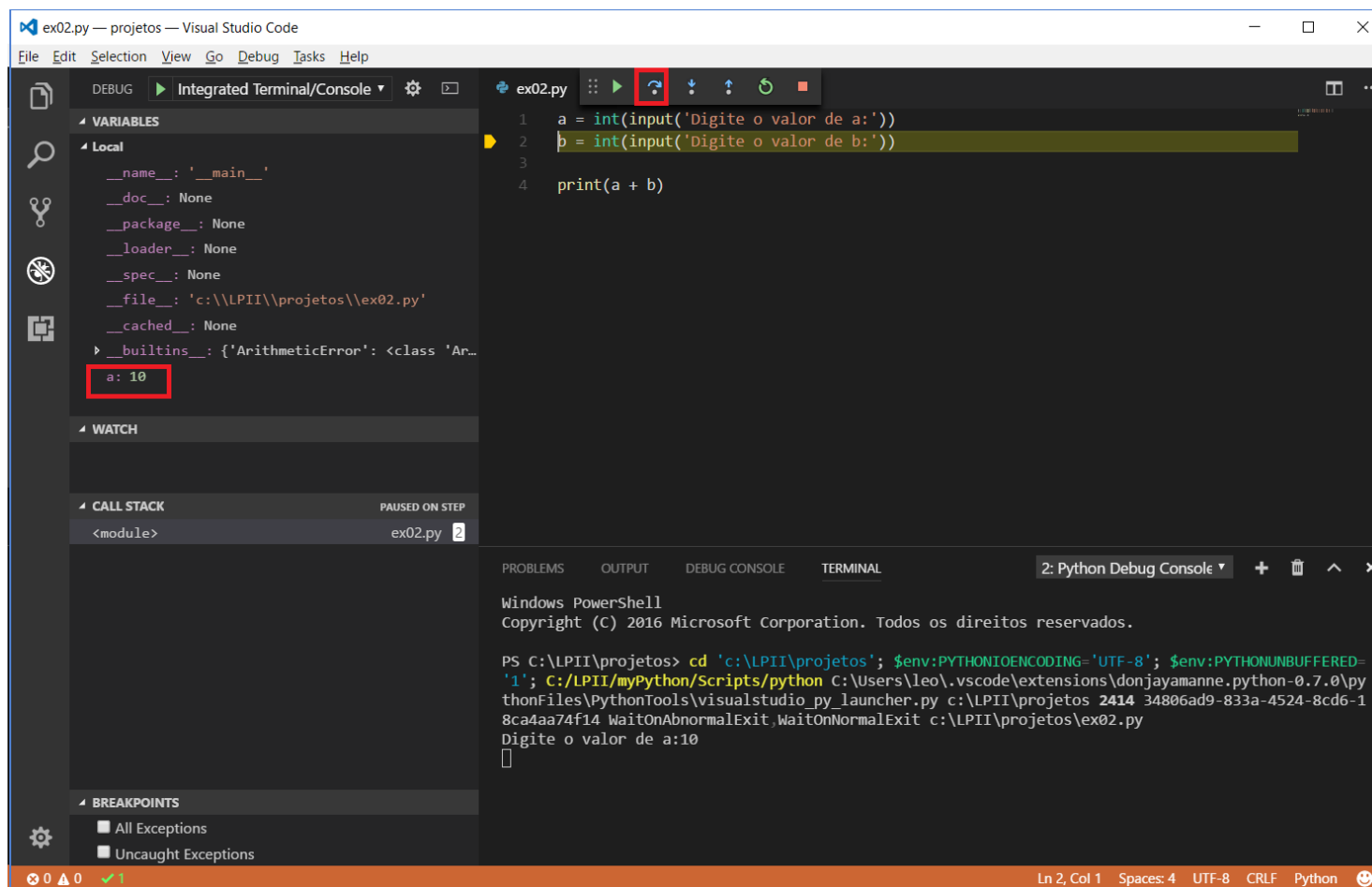
The left sidebar shows the **DEBUG** tab selected, with sections for **VARIABLES**, **WATCH**, **CALL STACK**, and **BREAKPOINTS**. The **TERMINAL** tab at the bottom shows the command prompt output:

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.

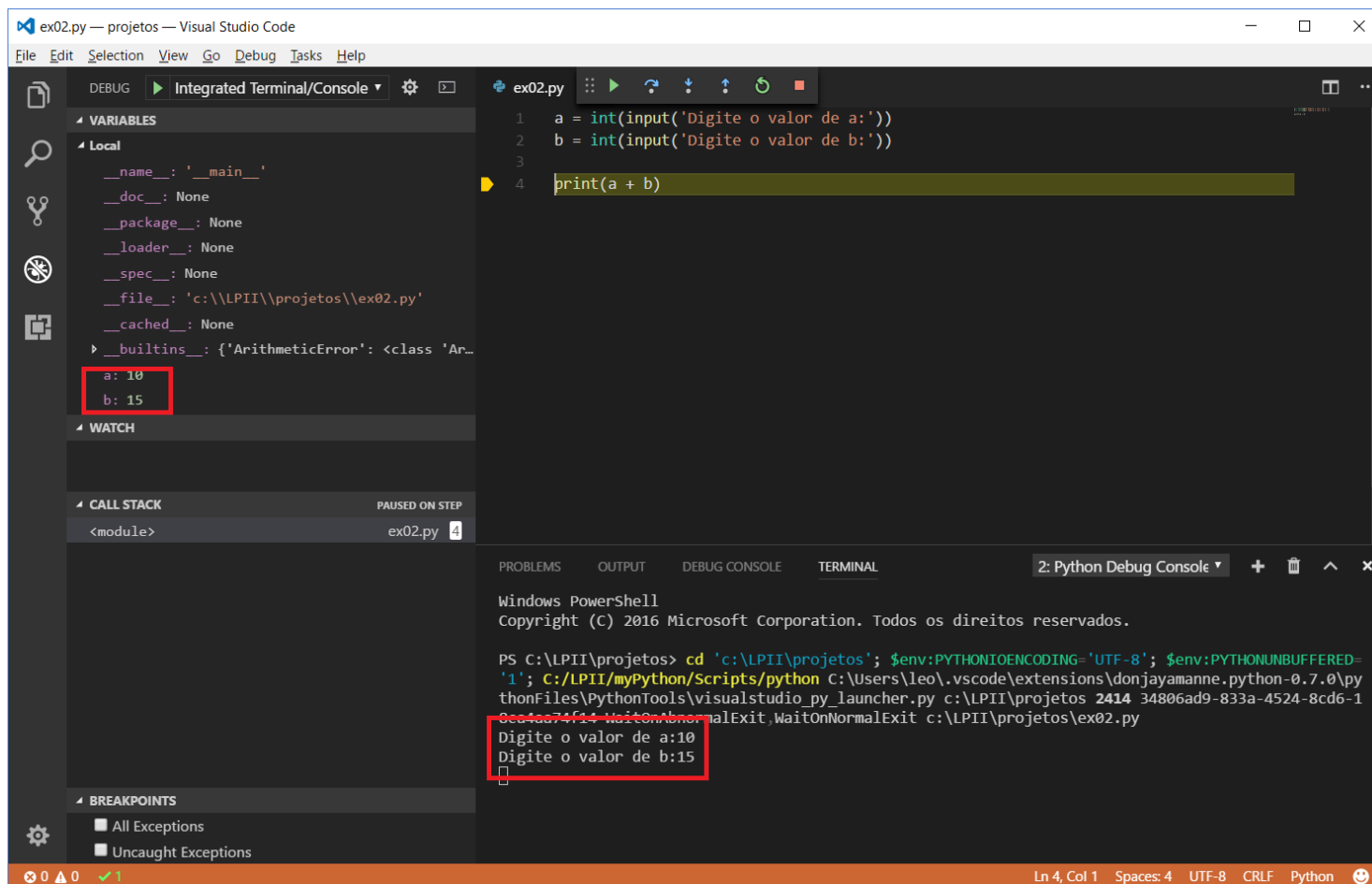
PS C:\LPII\projetos> cd 'c:\LPII\projetos'; $env:PYTHONIOENCODING='UTF-8'; $env:PYTHONUNBUFFERED='1'; C:\LPII\myPython\Scripts\python C:\Users\leo\.vscode\extensions\donjayamanne.python-0.7.0\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\LPII\projetos 2414 34806ad9-833a-4524-8cd6-18ca4aa74f14 WaitOnAbnormalExit,WaitOnNormalExit c:\LPII\projetos\ex02.py
Digite o valor de a:10
```

The input `10` is highlighted with a red box. The status bar at the bottom indicates the current line is `Ln 1, Col 1`, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python syntax.

7. Verifique a variável local e execute a próxima instrução



8. Verifique as variáveis locais e execute a próxima instrução



The screenshot shows the Visual Studio Code interface with a Python file named `ex02.py` open. The code is as follows:

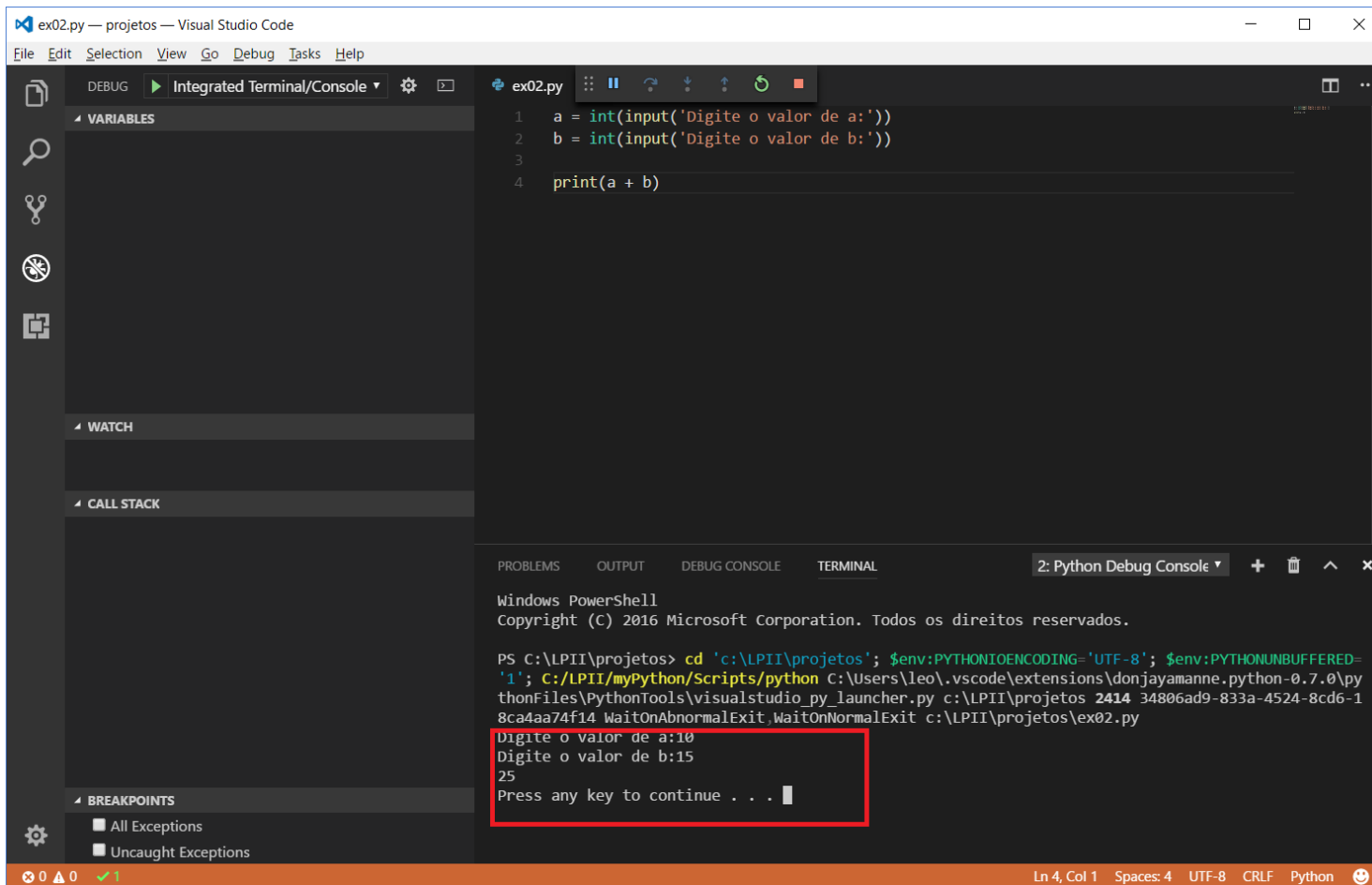
```
1 a = int(input('Digite o valor de a:'))
2 b = int(input('Digite o valor de b:'))
3
4 print(a + b)
```

The debugger is paused at line 4. The **VARIABLES** pane on the left shows the local variables `a` and `b` with values 10 and 15, respectively. The **WATCH** pane is empty. The **CALL STACK** pane shows the current frame as `<module>` in `ex02.py`. The **TERMINAL** pane at the bottom shows the command prompt output:

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.

PS C:\LPII\projetos> cd 'c:\LPII\projetos'; $env:PYTHONIOENCODING='UTF-8'; $env:PYTHONUNBUFFERED='1'; C:\LPII\myPython\Scripts\python C:\Users\leo\.vscode\extensions\donjayamanne.python-0.7.0\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\LPII\projetos 2414 34806ad9-833a-4524-8cd6-10c0c0c74114 WaitOnNormalExit,WaitOnNormalExit c:\LPII\projetos\ex02.py
Digite o valor de a:10
Digite o valor de b:15
```

9. Verifique o resultado no console.



The screenshot shows the Visual Studio Code interface with a Python file named `ex02.py` open. The code is as follows:

```
1 a = int(input('Digite o valor de a:'))
2 b = int(input('Digite o valor de b:'))
3
4 print(a + b)
```

The left sidebar shows the **DEBUG** tab selected, with sections for **VARIABLES**, **WATCH**, **CALL STACK**, and **BREAKPOINTS**. The **TERMINAL** tab is active at the bottom, showing the execution of the program in a Windows PowerShell environment. The terminal output is:

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos os direitos reservados.

PS C:\LPII\projetos> cd 'c:\LPII\projetos'; $env:PYTHONIOENCODING='UTF-8'; $env:PYTHONUNBUFFERED='1'; C:\LPII\myPython\Scripts\python C:\Users\leo\.vscode\extensions\donjayamanne.python-0.7.0\pythonFiles\PythonTools\visualstudio_py_launcher.py c:\LPII\projetos 2414 34806ad9-833a-4524-8cd6-18ca4aa74f14 WaitOnAbnormalExit,WaitOnNormalExit c:\LPII\projetos\ex02.py
Digite o valor de a:10
Digite o valor de b:15
25
Press any key to continue . . .
```

The output `25` is highlighted with a red box, indicating the result of the program's execution.



Exercícios guiados por testes



Abra a AC 3, leia os exercícios com atenção e escreva as funções de acordo com o exercício. Execute os testes e verifique se a função que você criou passa em todos os testes.

Boa sorte





Faculdade
IMPACTA
TECNOLOGIA

Fim

