



# Introdução à Programação para Sensoriamento Remoto

## **Aula 04 – Introdução à Programação com a Linguagem Python**

Gilberto Ribeiro de Queiroz  
Thales Sehn Körting  
Fabiano Morelli



20 de Março de 2019

# Tópicos

- Objetos e Tipos de Dados
- Operadores e Expressões
- Tipos Numéricos
- Variáveis
- Usando Funções

# Usando o Interpretador Python (Python shell)



Adicionados recentemente

Criar



gvSIG desktop



RStudio



R x64 3.4.3

Expandir ▾

A



Access 2016



Acessórios do Windows ▾



Acrobat Reader DC



Alarmes e Relógio  
Novo



Anaconda3 (64-bit) ^



Anaconda Navigator



Anaconda Prompt



Jupyter Notebook



Reset Spyder Settings



Spyder



Autodesk SketchBook  
Novo

domingo

11

Explorar



Ne  
Gra

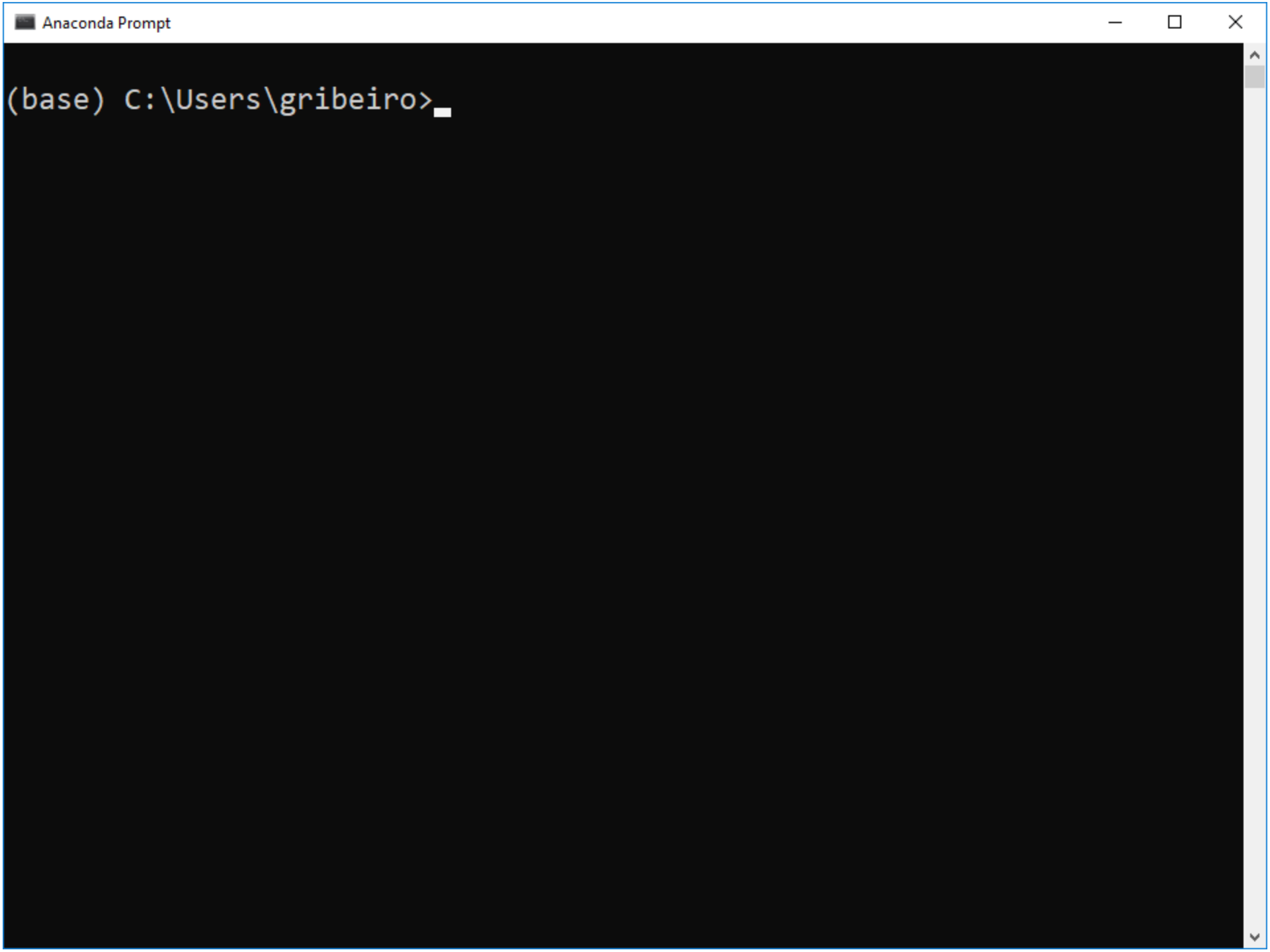
Microsoft Sto

Dell+

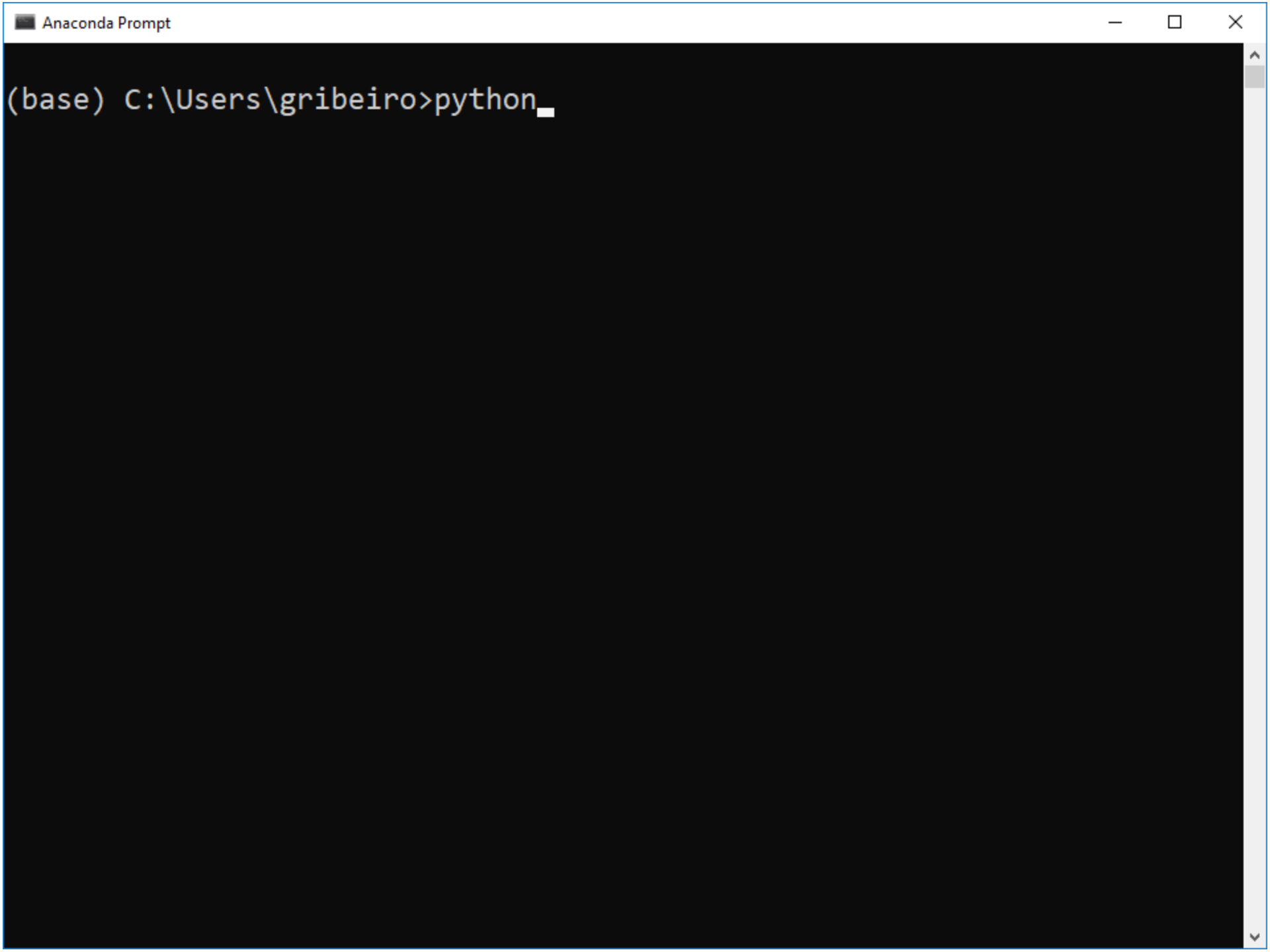


Digite aqui para pesquisar

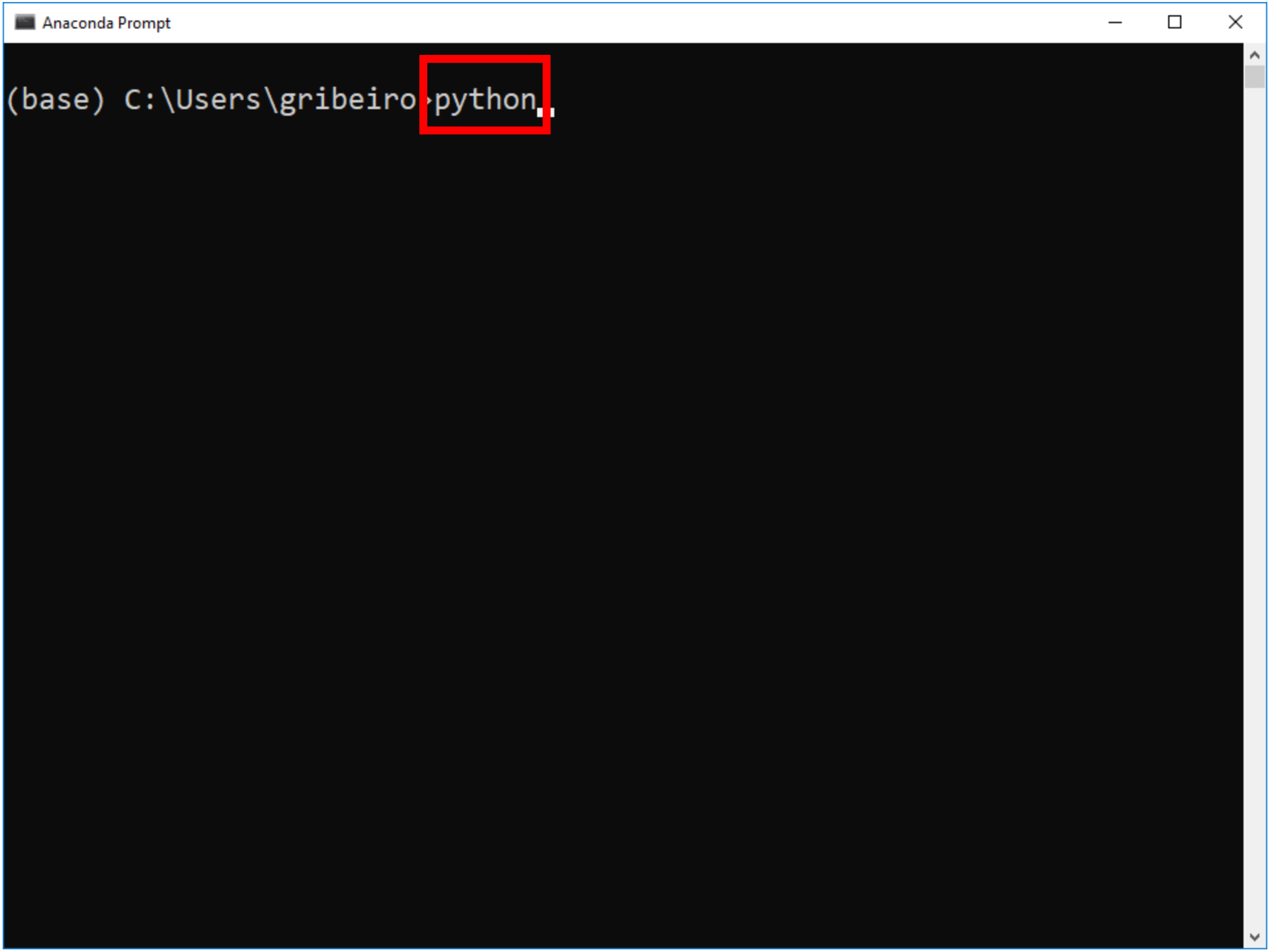




```
(base) C:\Users\gribeiro>
```



```
(base) C:\Users\gribeiro>python_
```



```
(base) C:\Users\gribeiro>python.
```

```
(base) C:\Users\gribeiro>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```



```
(base) C:\Users\gribeiro>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ^Z
```

```
(base) C:\Users\gribeiro>_
```

Para sair do modo interativo do Python:

- Windows: Ctrl-z
- Linux/Mac: Ctrl-D

# Objetos, Tipos de Dados, Operadores e Expressões

# Objetos

- Um programa Python manipula o que chamamos de objetos.

```
nome = "Gilberto Ribeiro de Queiroz"  
  
idade = 30  
  
print( nome.split(" "), idade + 11 )
```

- Todo objeto está associado a um **tipo**, que define as operações que podem ser realizadas com ele.

# Tipos de Dados: Definição

- Um **tipo de dado** (*data type*) é definido por um **conjunto de valores** e um **conjunto de operações** sobre esses valores.
- O **core da linguagem Python** contém um conjunto de tipos de dados chamados de **fundamentais** ou **primitivos**\*, para manipulação de valores numéricos, valores lógicos (ou booleanos), strings (cadeia de caracteres ou sequência de caracteres), listas, dicionários ente outros tipos.

\*Também chamados de *built-in types*.

# Tipos de Dados: Constantes ou Literais

- Os valores individuais de cada tipo são chamados de **literais** ou **literais constante**. Exemplo:
  - ✓ O número inteiro: `473`
  - ✓ O número real (ponto flutuante): `4.1`
  - ✓ O número complexo: `7 + 3j`
  - ✓ O valor lógico verdadeiro: `True`
  - ✓ A sequência de caracteres: `"Gilberto"`
  - ✓ A lista de números ímpares: `[ 1, 3, 5, 7 ]`
  - ✓ O conjunto: `{ "maçã", "banana", "goiaba" }`
  - ✓ O dicionário: `{ "latitude": -12, "longitude": -54 }`
  - ✓ O valor nulo: `None`

A lista completa dos tipos básicos de Python pode ser consultada em:

<https://docs.python.org/3/library/stdtypes.html>

# Tipos Numéricos: `int`

- O tipo `int` é capaz de representar **números inteiros**.
- Possui **precisão “infinita”**.
- Notação:  
`1003`  
`9223372036854775808`

# Tipos Numéricos: `float`

- O tipo `float` ou **ponto flutuante** é capaz de representar números reais com uma certa **precisão numérica** (64-bits).
- Um número em ponto flutuante pode ser expresso da seguinte maneira:
  - `5.1`
  - `5.`
  - `1.2e12`

# Outros Tipos Numéricos

- `complex`: números complexos.
- `fractions`: racionais.
- `decimal`: números em ponto flutuante com precisão definida pelo usuário.



# Tipos de Dados: Operações

- Para cada tipo de dado, existe um conjunto de operadores disponíveis:
  - Ex: para os tipos numéricos, temos os operadores aritméticos básicos:
    - adição, subtração, divisão e multiplicação.
- **Cada operador possui uma notação própria:**
  - Multiplicação: 3 \* 7
  - Divisão: 4 / 2

# Tipos Numéricos:

## Operadores Aritméticos

- As operações aritméticas disponíveis para os tipos numéricos são muito semelhantes às que usamos na matemática.

Operador	Nome	Expressão	Valor
<b>+</b>	soma	5 <b>+</b> 8	13
<b>-</b>	subtração	3 <b>-</b> 2	1
<b>*</b>	multiplicação	3 <b>*</b> 4	12
<b>/</b>	divisão	6 <b>/</b> 4	1.5
<b>%</b>	resto da divisão	6 <b>%</b> 4	2
<b>**</b>	potenciação	2 <b>**</b> 4	16

# Tipos Numéricos:

## Operadores Aritméticos

- Abra o interpretador Python e digite a sequência de comandos a seguir:

```
>>> 5 + 2
```

```
>>> 5 - 2
```

```
>>> 5 * 2
```

```
>>> 5 / 2
```

```
>>> 5 // 2
```

```
>>> 5 % 2
```

```
>>> 5 ** 2
```

# Expressões

- Através da combinação das operações e operandos, podemos criar **expressões**, como as expressões matemáticas convencionais:

$$2 + 3 * 4 / 2$$

- Uma expressão deve resultar em um valor de um certo tipo:

$$\underbrace{2 + 3 * 4 / 2}_{\text{Expressão}} \Rightarrow \underbrace{8}_{\text{Resultado}} \Rightarrow \underbrace{\text{tipo inteiro}}_{\text{Tipo Expressão ou Tipo Resultado}}$$

Expressão

Resultado

Tipo Expressão

ou

Tipo Resultado

# Função: `type(objeto)`

- Informa o tipo de um objeto (ou valor).
- Abra o interpretador Python e digite a sequência de comandos a seguir:

```
>>> type( "Gilberto" )  
>>> type( 30 )  
>>> type( 22.5 )  
>>> type( 5 / 2 + 4 * 5 )  
>>> type( [1, 3, 5, 7] )
```

# Ordem de Avaliação de Expressões

- Considere a seguinte expressão:  $5.0 * 2.0 + 3.0 / 4.0$

- Qual o resultado dessa expressão? 3.25? 10.75? Por quê?

$$(5.0 * 2.0 + 3.0) / 4.0 \rightarrow 3.25$$

$$5.0 * 2.0 + 3.0 / 4.0 \rightarrow 10.75$$

- Quando uma expressão contém mais de dois operadores, a ordem em que eles são avaliados é significativa. Por isso, existe uma **convenção bem definida** da **precedência de cada operador**.
- No caso das operações aritméticas, a prioridade é a seguinte:
  - multiplicação, divisão, potenciação e resto da divisão;
  - adição e subtração.
- Assim como na matemática, podemos usar parênteses para controlar essa prioridade.

# Ordem de Avaliação de Expressões

- Além das regras de precedência, temos também a ordem de aplicação.
- Vários operadores são **infixos**, isto é, temos um literal ou variável ou expressão, seguido do operador, seguido por outro literal ou variável ou expressão.
- Nas linguagens de programação, vários operadores possuem uma **associatividade** da esquerda para direita.

# Funções e Chamada de Funções

- Apenas com o conjunto de operações básicas seria muito difícil expressarmos nossos programas.
- Por isso, várias funcionalidades que iremos incluir na escrita dos nossos programas pressupõe a existência de algumas funções e procedimentos auxiliares, como as funções matemáticas.
- Estas funcionalidades podem ser incluídas no nosso programa na forma de **chamada de uma função (*function call*)**, que é uma forma de desviar o fluxo de controle do nosso programa para uma outra parte que irá realizar uma determinada computação e depois irá retornar o fluxo de controle ao ponto onde foi chamada (ou invocada).
- A chamada de uma função é feita colocando-se o nome da função e a lista de argumentos que será passada à função, para que ela realize sua computação.



# Funções e Chamada de Funções

```
>>> abs( -22.5 )  
>>> print( "Gilberto", "Ribeiro", sep=";")  
>>> int( 22.5 )  
>>> float( 5 )  
>>> type( [1, 3, 5, 7 ] )
```

A lista completa das funções básicas de Python pode ser consultada em:

<https://docs.python.org/3/library/functions.html>

# Tipos Numéricos: Funções Matemáticas

- Além dos operadores básicos, temos diversas funções matemáticas disponíveis:

Função	Descrição	Expressão	Valor
<code>abs(x)</code>	Valor absoluto de x	<code>abs(-2)</code>	2
<code>ceil(x)</code>	Teto de x, isto é, o maior inteiro que não seja menor do que x	<code>ceil(1.2)</code>	2
<code>floor(x)</code>	O piso de x, isto é, o maior inteiro que não seja maior do que x	<code>floor(1.2)</code>	1
<code>exp(x)</code>	Exponencial: $e^x$	<code>exp(2)</code>	7.38
<code>pow(x,y)</code>	Potenciação: $x^y$	<code>pow(2, 6)</code>	64
<code>log(x)</code>	Logaritmo natural: $\log_e x$	<code>log(10)</code>	2.30
<code>log10(x)</code>	Logaritmo na base 10: $\log_{10} x$	<code>log10(10)</code>	1.0
...	...	...	...

# Tipos Numéricos: Funções Matemáticas

- Se tentarmos usar algumas das funções matemáticas diretamente, teremos uma surpresa:

```
>>> log10(10)
```

```
NameError: name 'log10' is not defined
```

- Para utilizar algumas funções matemáticas, iremos precisar importar a biblioteca **math**:

```
>>> import math
```

```
>>> math.log10(10)
```

```
1.0
```

A lista completa das funções matemáticas de Python pode ser consultada em:

<https://docs.python.org/3/library/math.html>

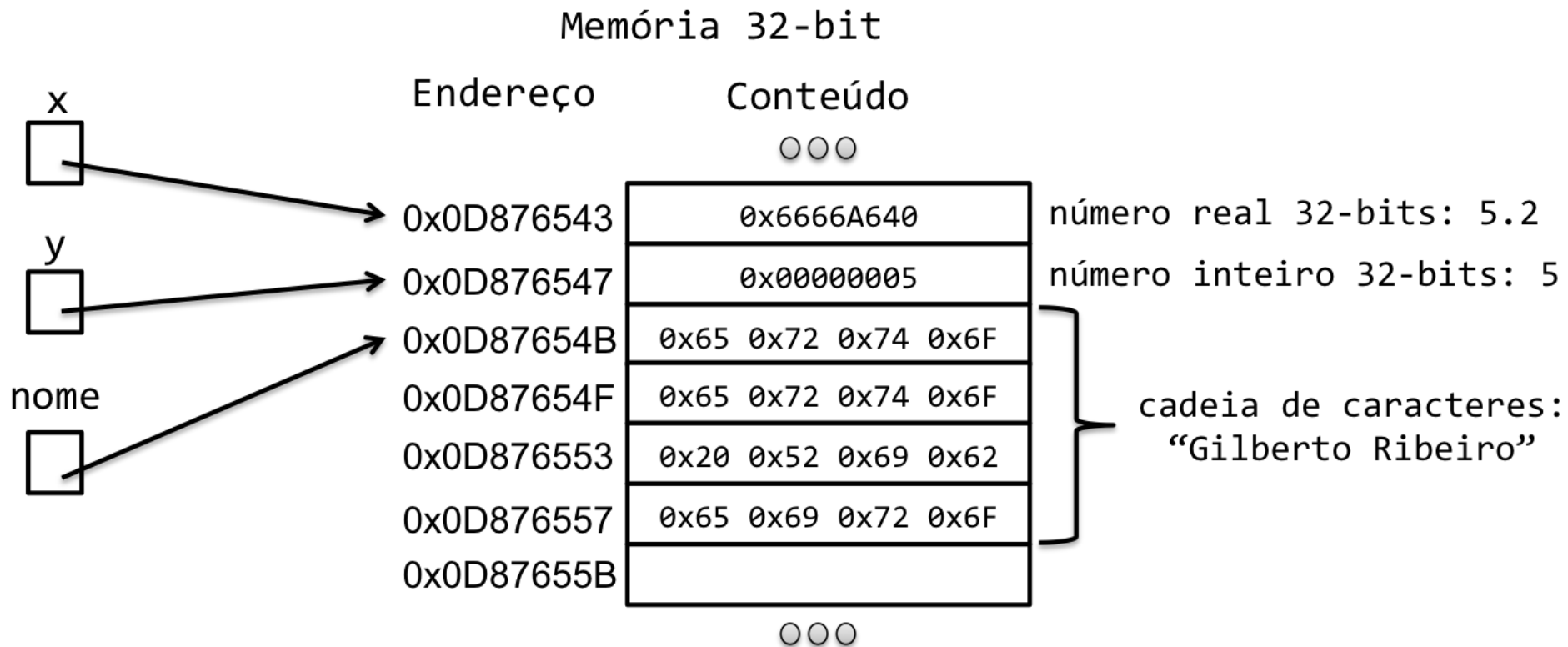
# Variáveis e Atribuição

# Variáveis

- Um programa, além de manipular **valores constantes** ou **literais**, também manipula o que chamamos de **variáveis**.
- Cada **variável** corresponde a uma posição de memória cujo conteúdo pode variar ao longo do tempo de execução de um programa.
- Uma **variável** possui **um nome** usado como **identificador** e, em geral, é **associada** com um **tipo de dado**.

# Variáveis

```
>>> x = 5.2
>>> y = 5
>>> nome = "Gilberto Ribeiro"
```




# Atribuição

- A atribuição é um comando que associa um valor de um determinado tipo de dados a uma variável.
- Essa associação pode também ser o resultado de uma expressão.
- Portanto, este comando possui a seguinte forma:

**identificador** = **expressão**

  
variável

  
valor, variável, expressão,  
resultado de um comando, ...

# Variáveis: Considerações

- Os **tipos de dados** nos **abstraem** da **representação interna**, destacada na figura anterior pela forma de representação dos valores na memória do computador.
- O conceito de **variável**, nos **abstrai** da necessidade de lembrarmos das **posições de memória** onde armazenamos **valores**.



# Regra para Nomes de Variáveis

- Cada linguagem de programação possui suas regras para nomenclatura das variáveis, inclusive para dizer se há diferenças entre nomes de variáveis com letras **maiúsculas** e **minúsculas**.
- Em Python, os identificadores de variáveis podem ser qualquer cadeia de caracteres formadas por letras, dígitos e *underscore*, desde que não comece com um dígito.
- Além disso existe a distinção entre caracteres maiúsculos e minúsculos.

# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

07 a = 18

08 print(a)
09 print(b)
10 print(A)
```

Variável Valor


Memória  
do  
Computador



# Variáveis e Atribuições



```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

07 a = 18

08 print(a)
09 print(b)
10 print(A)
```

Variável Valor

Variável	Valor
a	2

Memória  
do  
Computador



# Variáveis e Atribuições



```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

07 a = 18

08 print(a)
09 print(b)
10 print(A)
```

Variável Valor

a	2
b	6

Memória  
do  
Computador



# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5
04 print(a)
05 print(b)
06 print(A)
07
08 a = 18
09 print(a)
10 print(b)
11 print(A)
```



Variável Valor

a	2
b	6
A	5

Memória  
do  
Computador



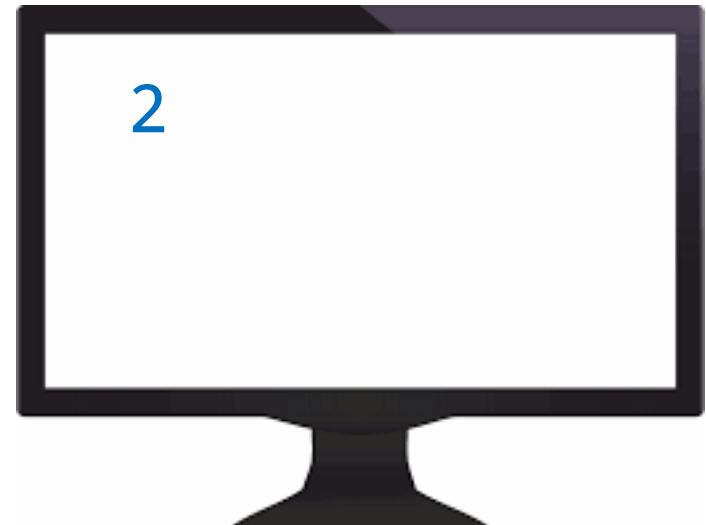
# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5
04 print(a)
05 print(b)
06 print(A)
07
08 a = 18
09
10 print(a)
    print(b)
    print(A)
```

Variável Valor

a	2
b	6
A	5

Memória  
do  
Computador



# Variáveis e Atribuições

```
01  a = 2
02  b = a * 3
03  A = 5

04  print(a)
05  print(b)
06  print(A)

07  a = 18

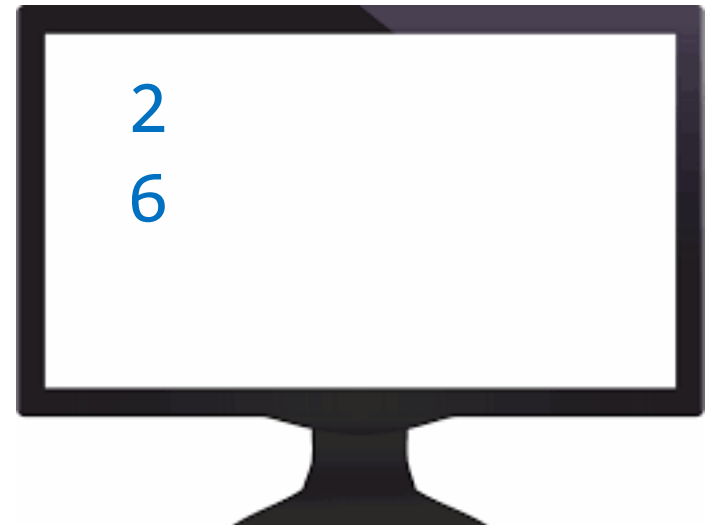
08  print(a)
09  print(b)
10  print(A)
```



Variável Valor

a	2
b	6
A	5

Memória  
do  
Computador



# Variáveis e Atribuições

```
01  a = 2
02  b = a * 3
03  A = 5

04  print(a)
05  print(b)
06  print(A)

07  a = 18

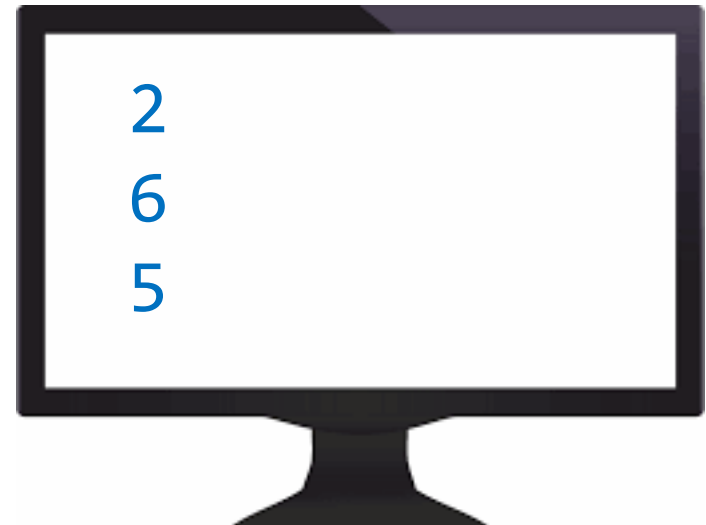
08  print(a)
09  print(b)
10  print(A)
```



Variável Valor

a	2
b	6
A	5

Memória  
do  
Computador





# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

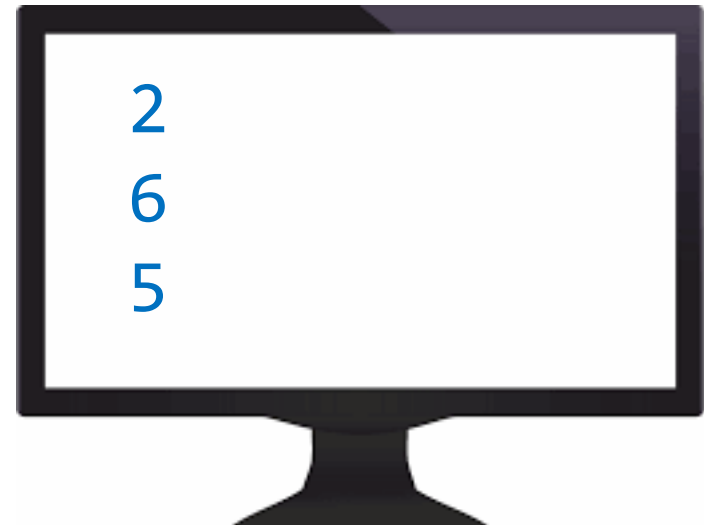
→ 07 a = 18

08 print(a)
09 print(b)
10 print(A)
```

Variável Valor

a	18
b	6
A	5

Memória  
do  
Computador



# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

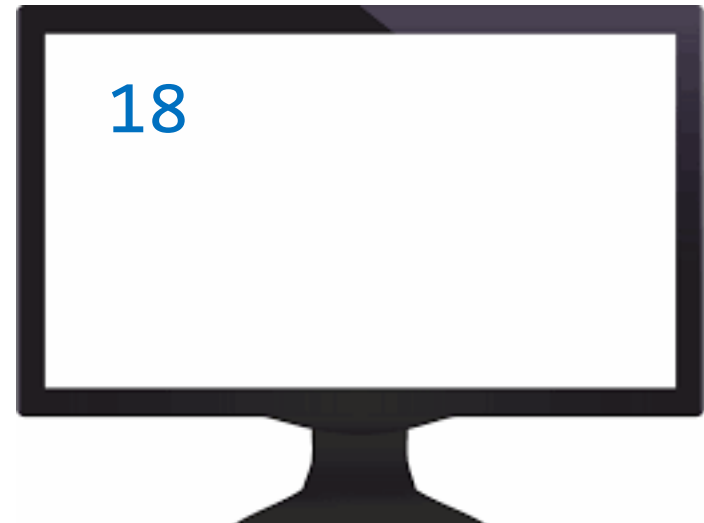
07 a = 18

→ 08 print(a)
09 print(b)
10 print(A)
```

Variável Valor

a	18
b	6
A	5

Memória  
do  
Computador



# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

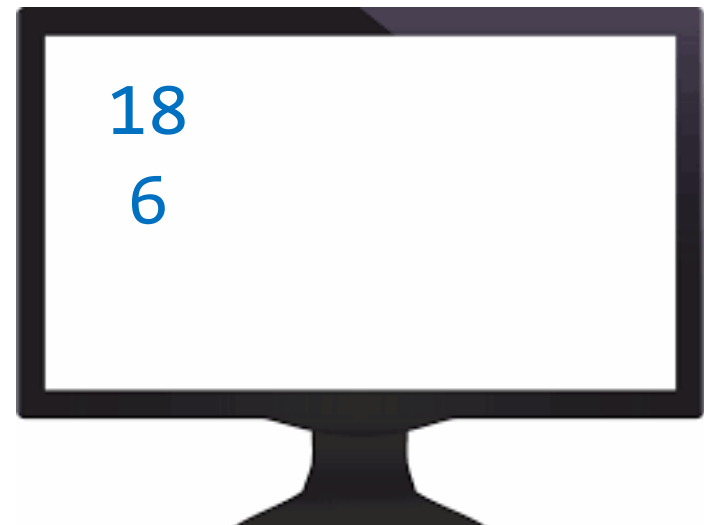
07 a = 18

08 print(a)
09 print(b)
10 print(A)
```

Variável Valor

a	18
b	6
A	5

Memória  
do  
Computador



# Variáveis e Atribuições

```
01 a = 2
02 b = a * 3
03 A = 5

04 print(a)
05 print(b)
06 print(A)

07 a = 18

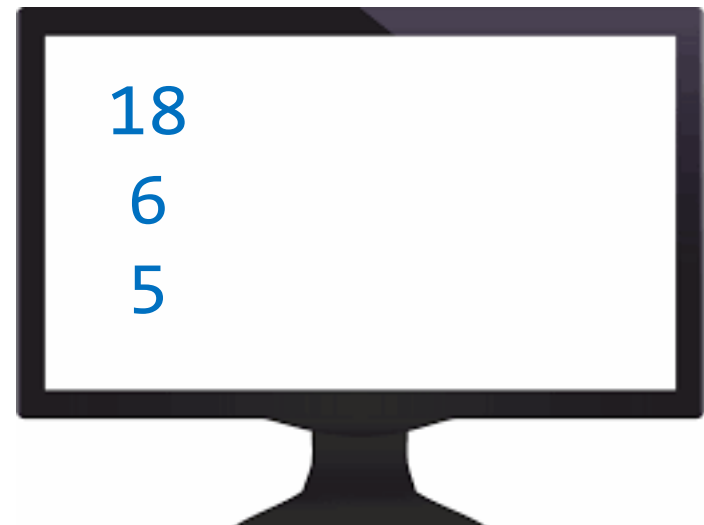
08 print(a)
09 print(b)
10 print(A)
```



Variável Valor

a	18
b	6
A	5

Memória  
do  
Computador



# Comentários

# Comentários

- Comentários são parte importante de qualquer programa, não sendo considerados instruções a serem executadas.
- Servem apenas ao propósito de documentar o código do programa.
- Temos uma notação especial para documentação de funções e classes.

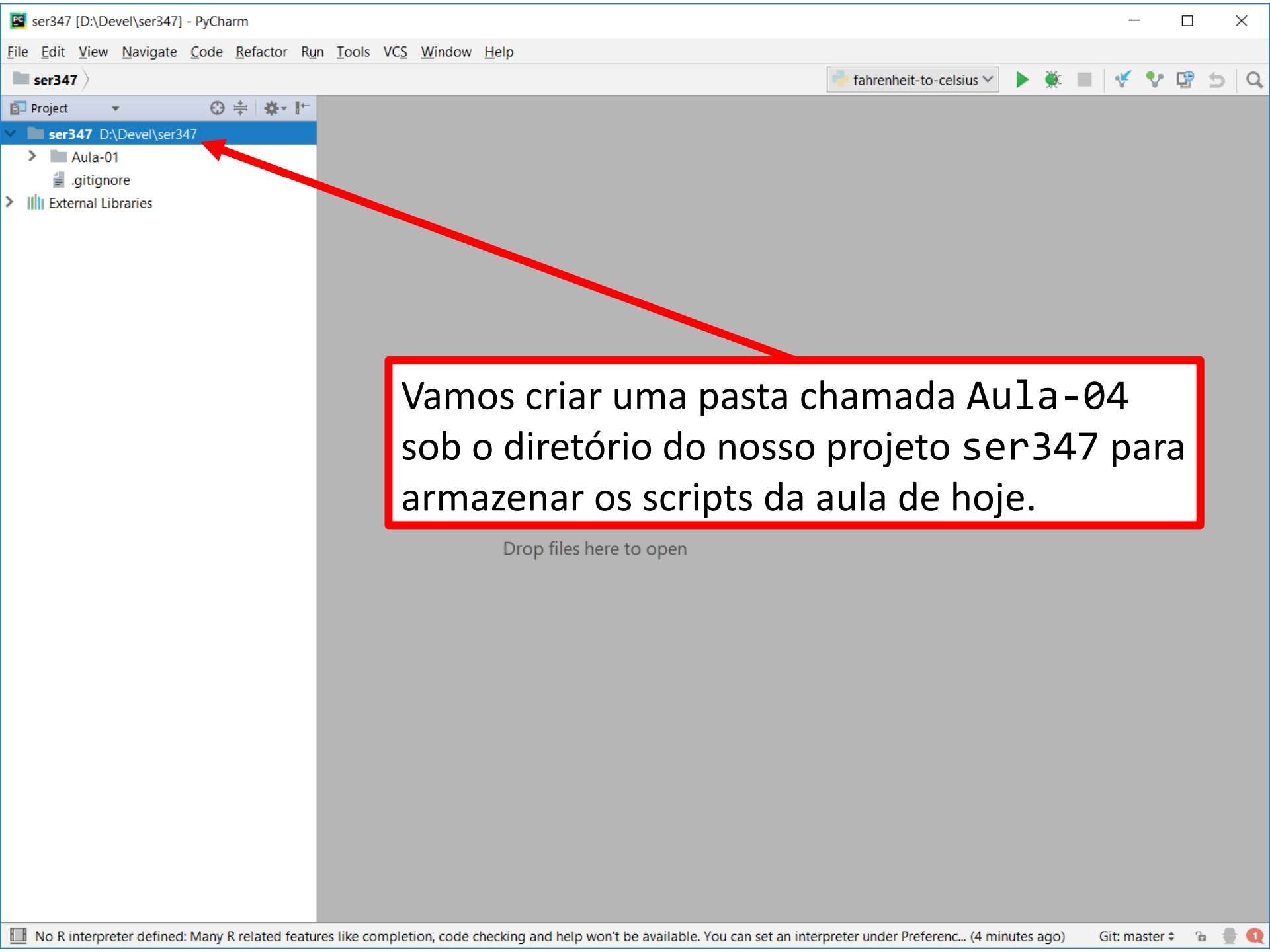
# Exemplo: Calculando o NDVI

```
01  # definir valores de NIR e Red
02  NIR = 0.5
03  Red = 0.3

04  # mostrar dados de entrada na tela
05  print("NIR:", NIR)
06  print("Red:", Red)

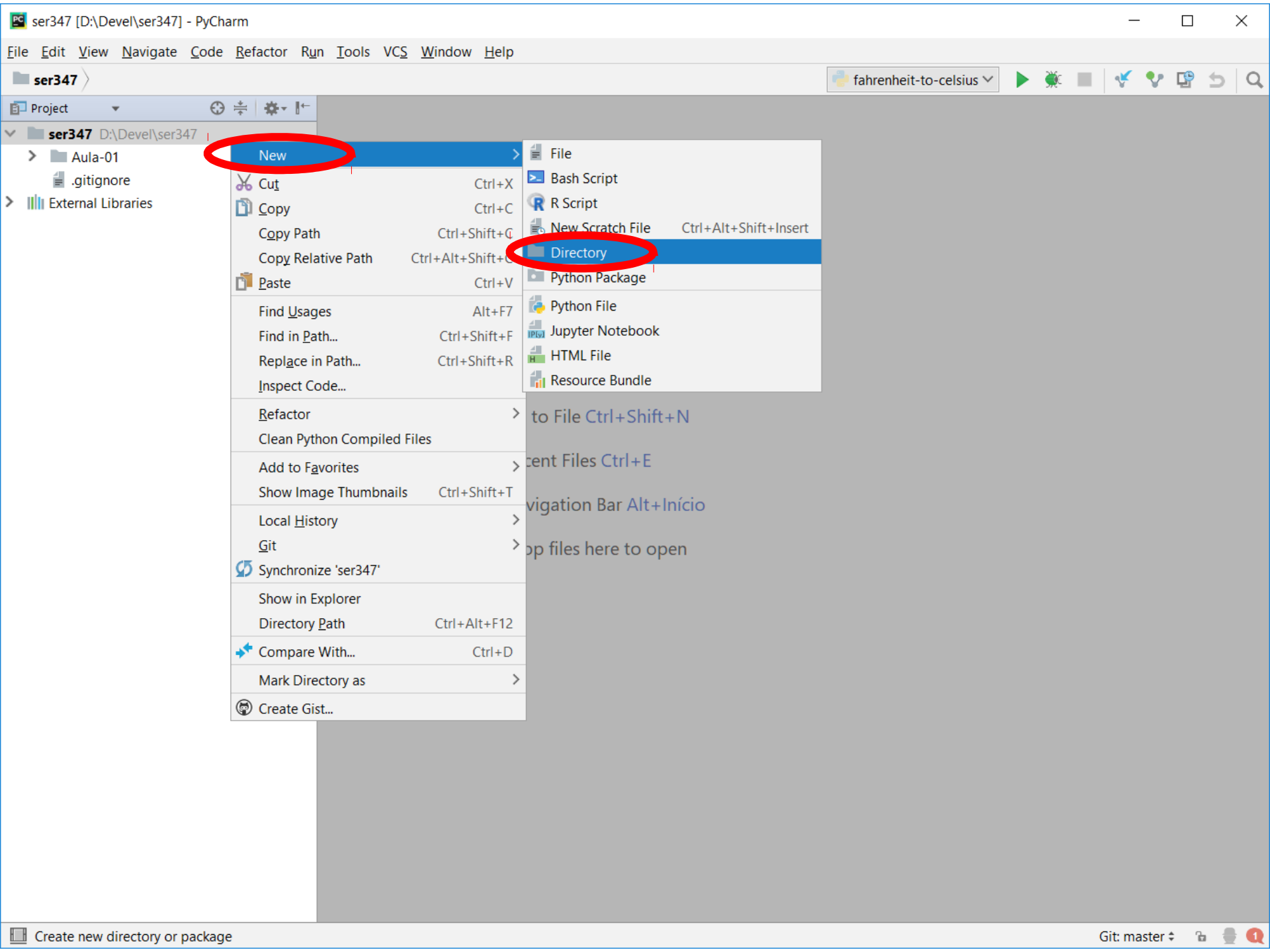
07  # calcular NDVI
08  NDVI = (NIR - Red) / (NIR + Red)

09  # mostrar resultado na tela
10  print("NDVI:", NDVI)
```




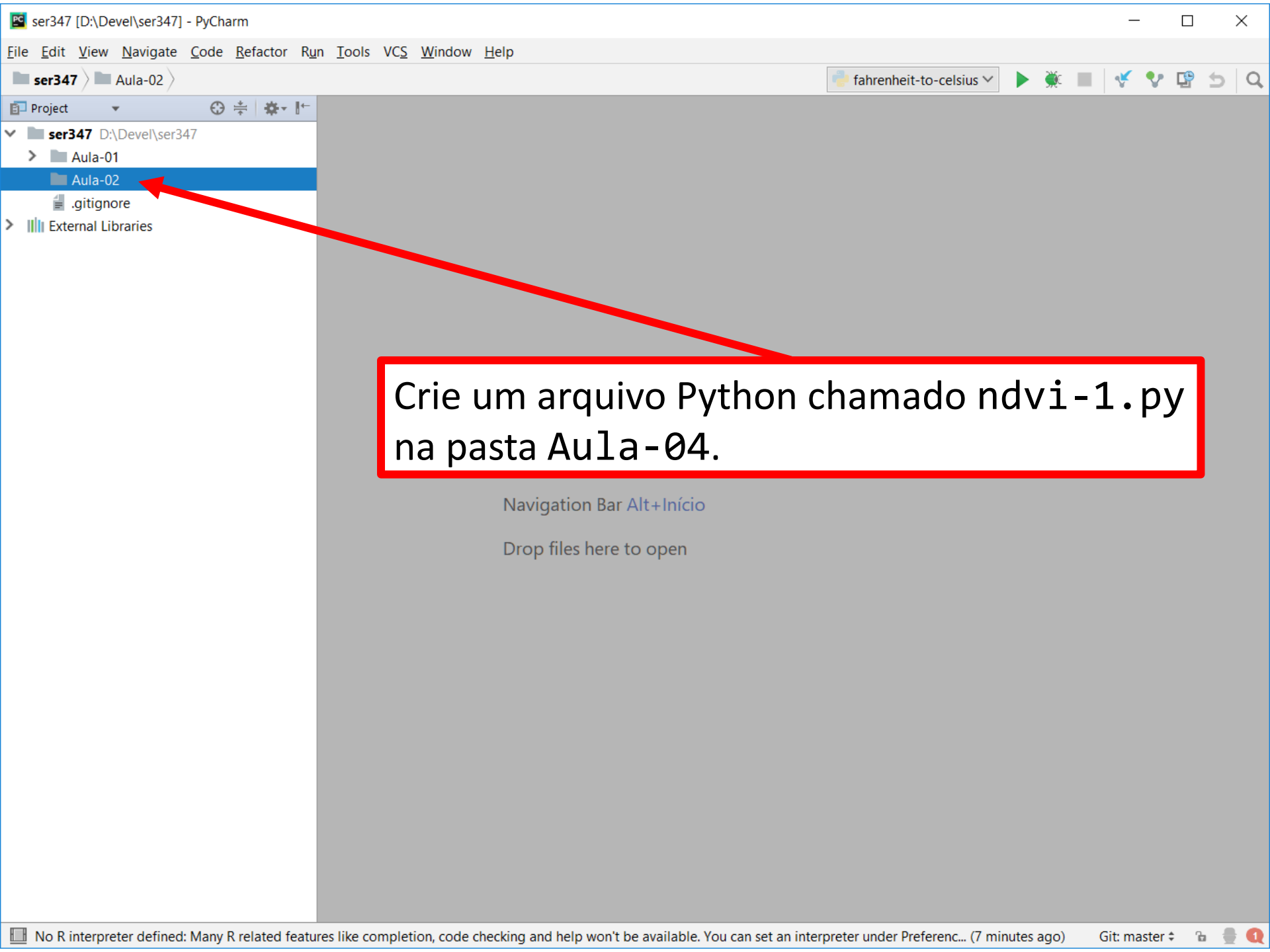
Vamos criar uma pasta chamada Aula-04 sob o diretório do nosso projeto ser347 para armazenar os scripts da aula de hoje.



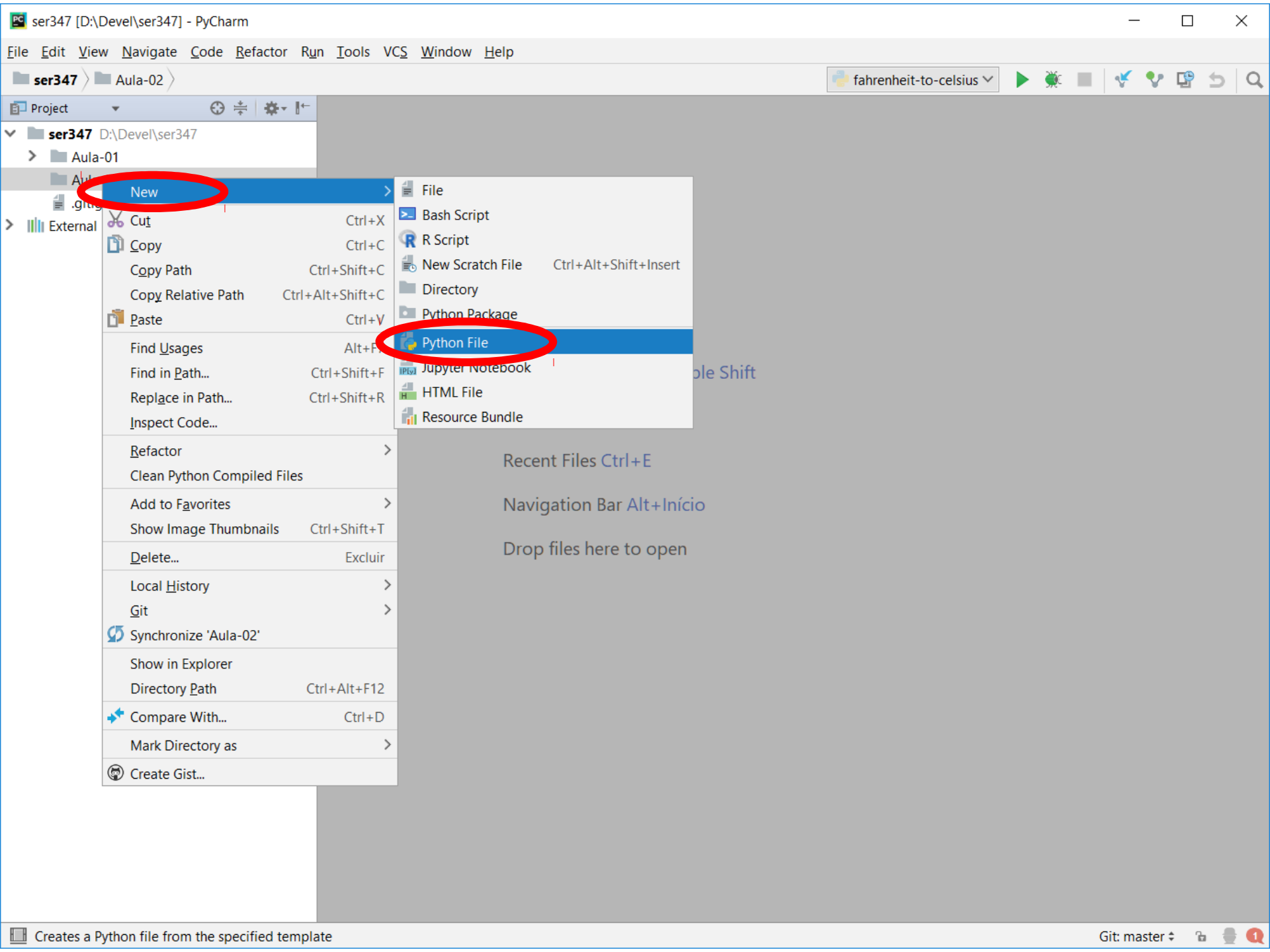



PC New Directory ×

 Enter new directory name:




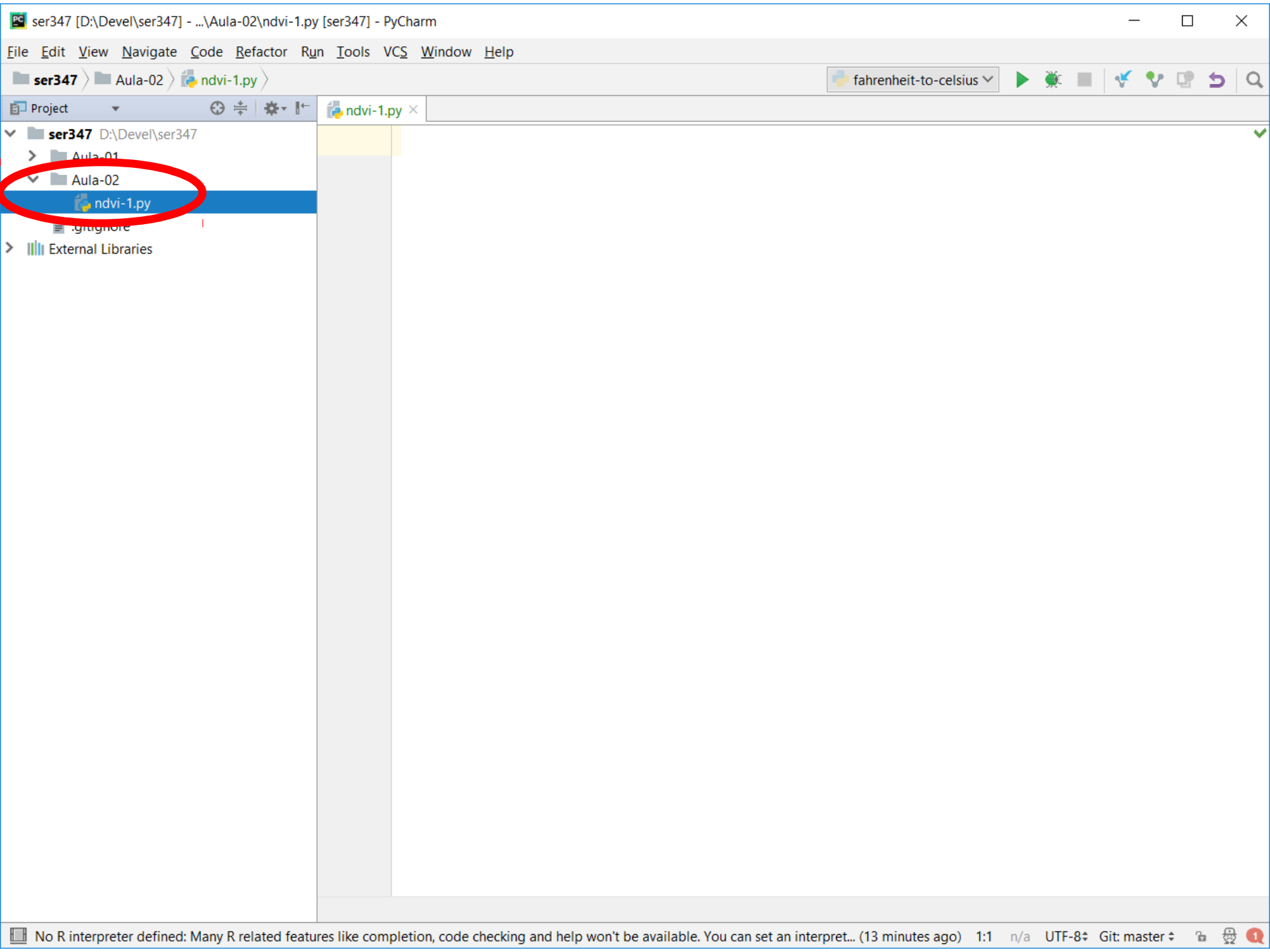
Crie um arquivo Python chamado ndvi-1.py na pasta Aula-04.

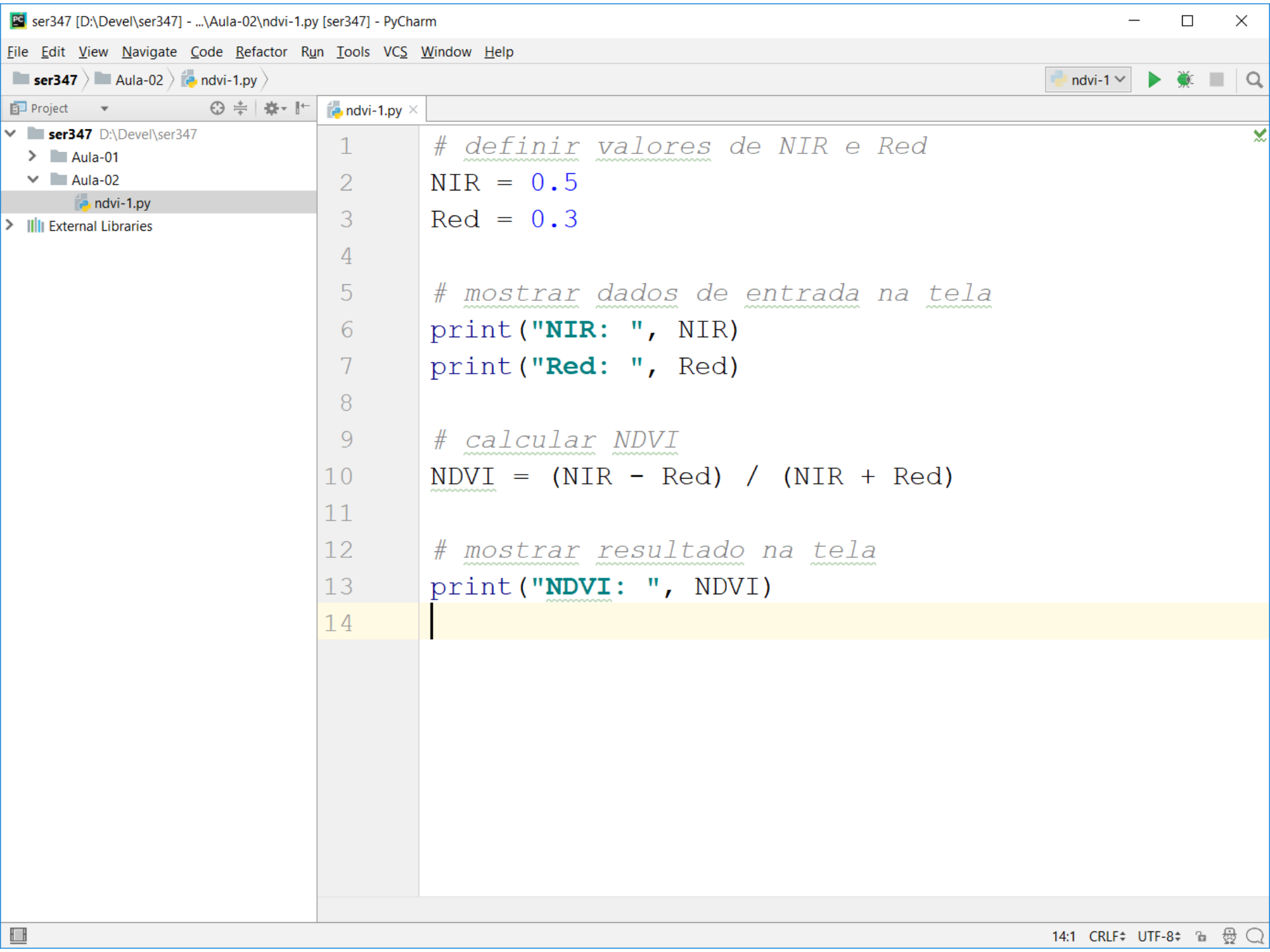


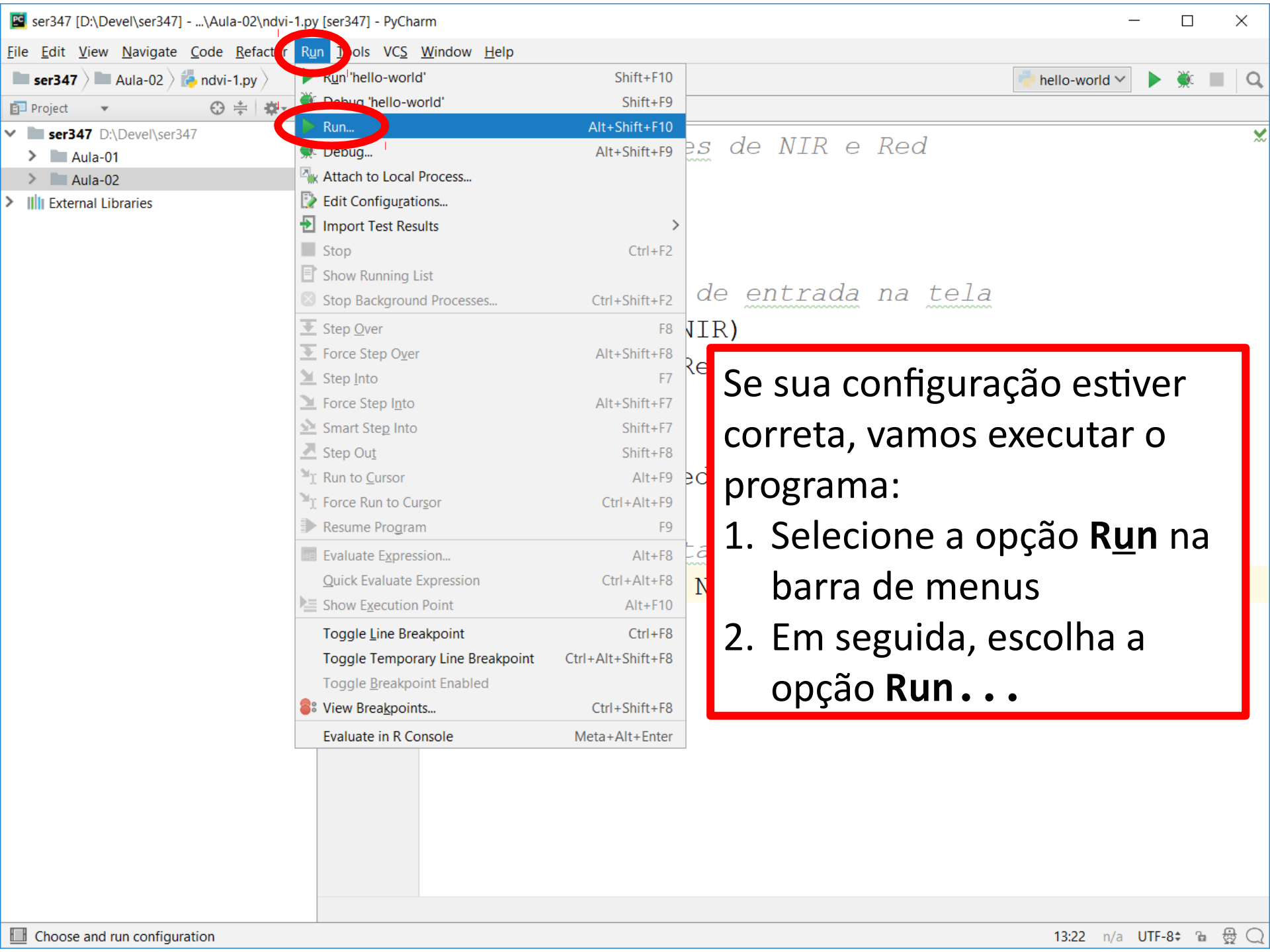
 New Python file ✕

Name:  ↑↓

Kind:  Python file ▼



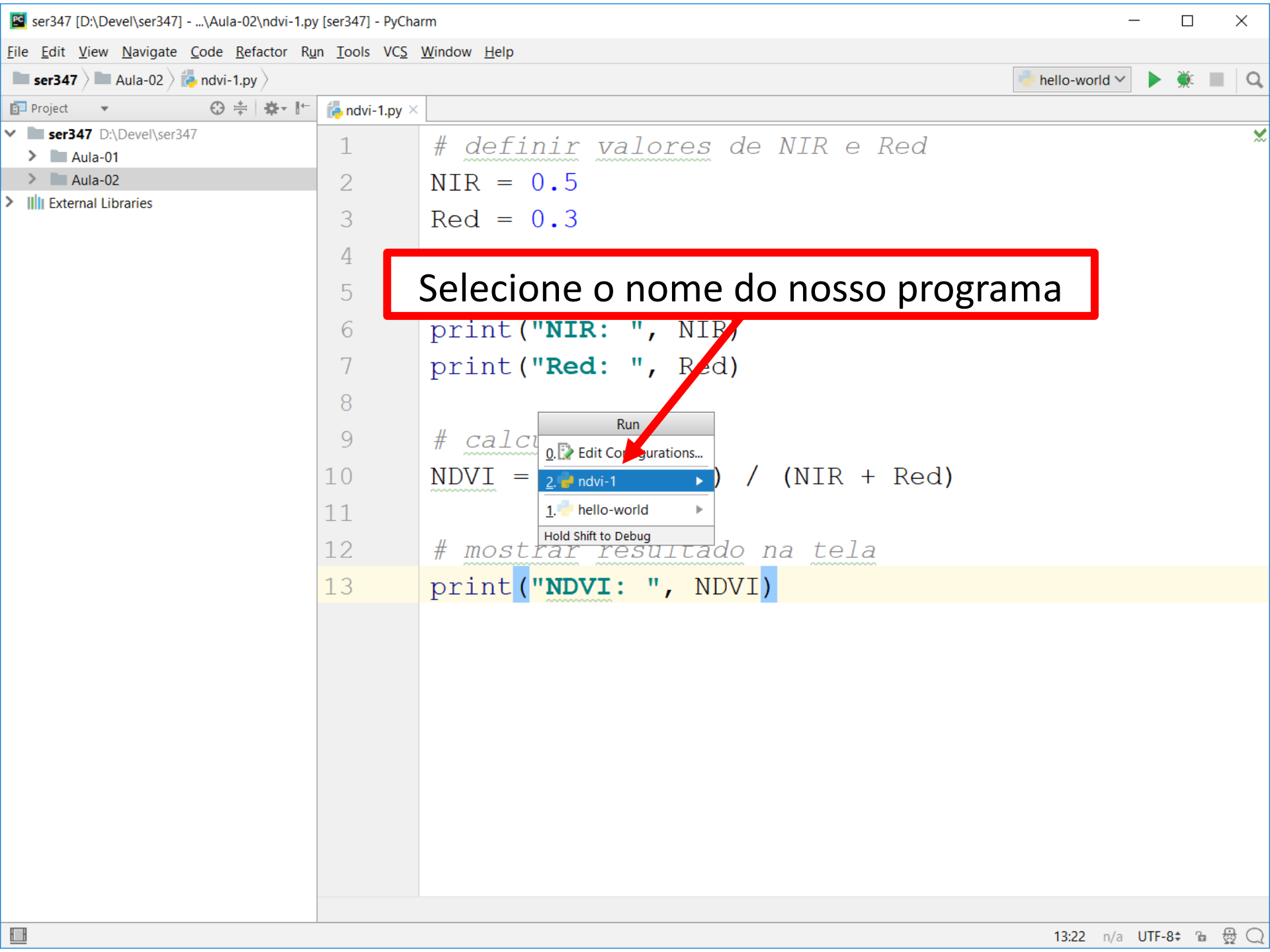


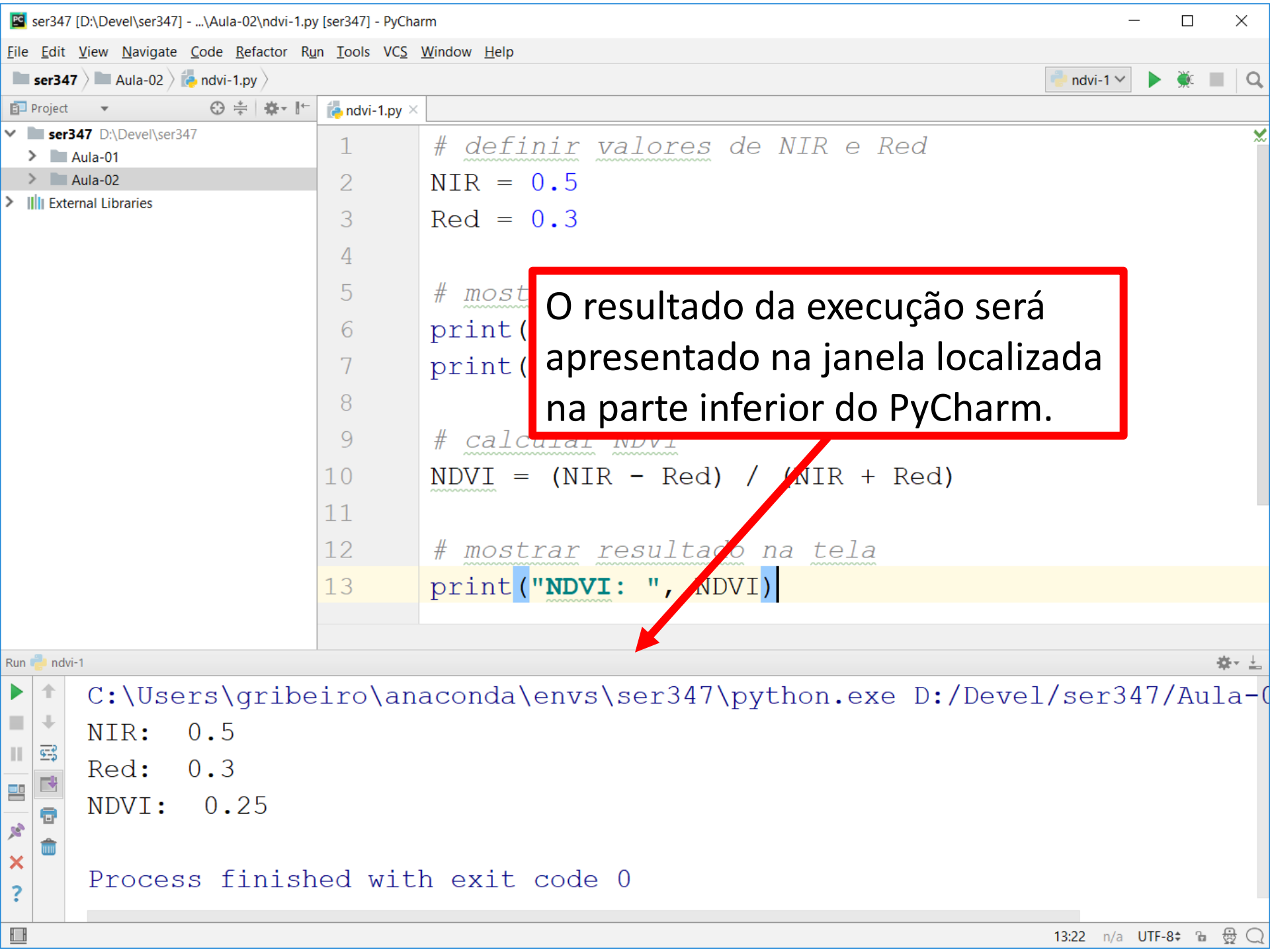


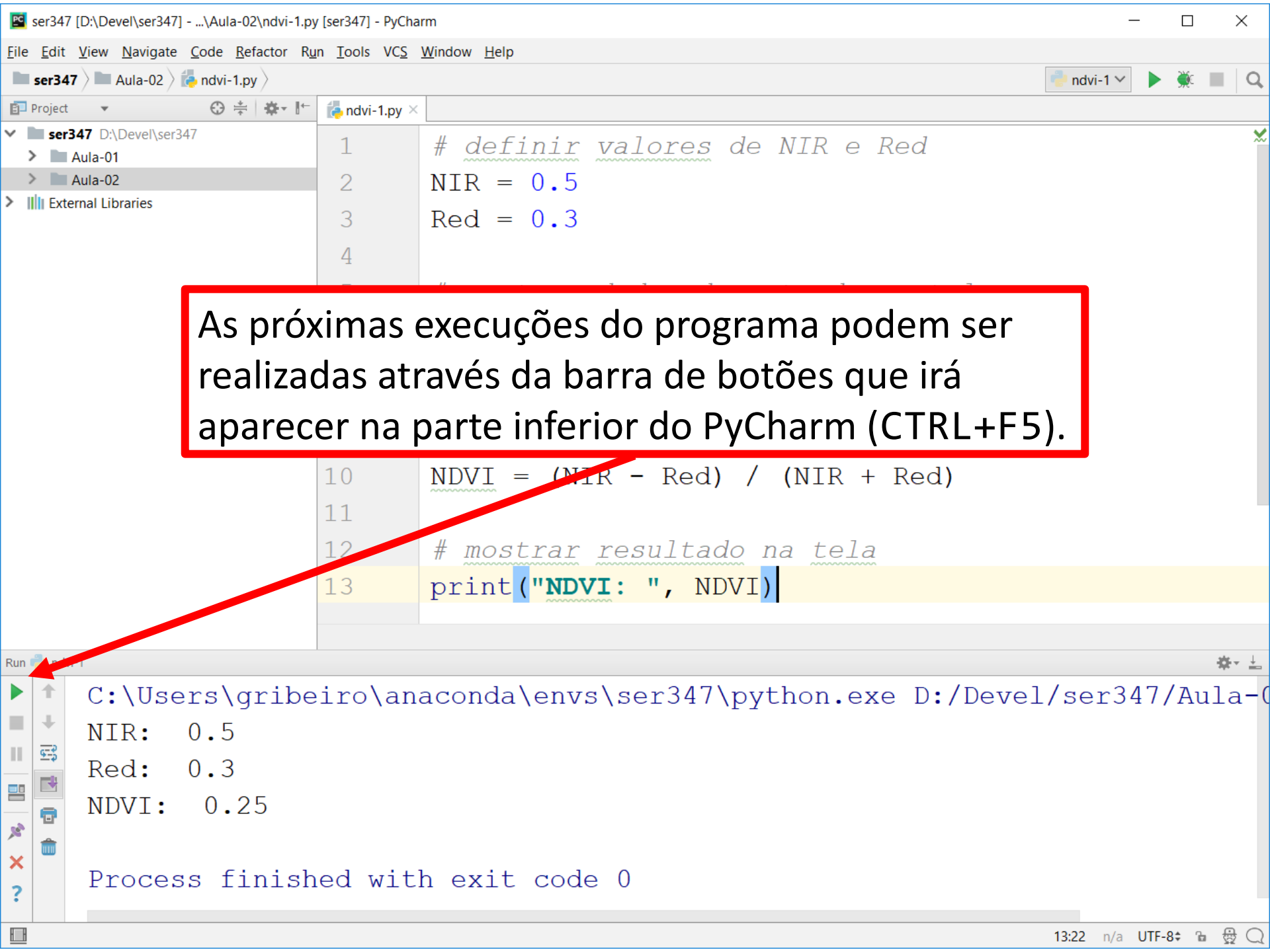
Se sua configuração estiver correta, vamos executar o programa:

1. Selecione a opção **Run** na barra de menus
2. Em seguida, escolha a opção **Run...**









As próximas execuções do programa podem ser realizadas através da barra de botões que irá aparecer na parte inferior do PyCharm (CTRL+F5).

# Considerações Finais

# Considerações Finais

- Na aula de hoje, aprendemos diversos conceitos sobre linguagens de programação:
  - Tipos de Dados
  - Valores Literais ou Constantes
  - Operadores
  - Expressões
  - Ordem de avaliação das expressões
  - Funções e Chamada de Funções
  - Variáveis
  - Comentários

# Referências Bibliográficas

# Python

- [The Python Standard Library](#). Acesso: Março de 2019.