

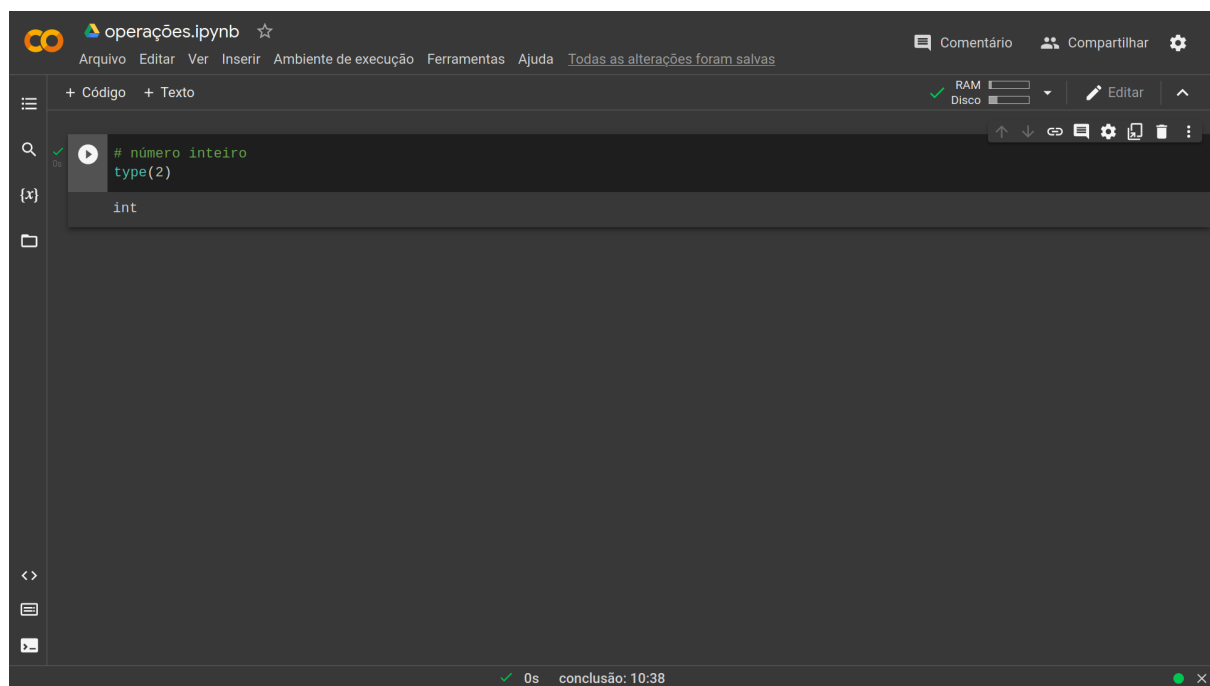
## Aula 27 - Operações matemáticas básicas com Python.

Olá a todos, sejam muito bem-vindos a mais uma aula, na aula de hoje nós vamos entender como podemos efetuar operações básicas em python, e entender como funciona os tipos de variáveis em python.

Na matemática temos o que chamamos de conjunto numérico, como por exemplo, os números Reais, números inteiros, negativos dentre outros. Na computação não é muito diferente, mas aqui temos uma maior generalização. Os números são divididos em inteiros e reais. Vamos entender como podemos representar isso em python.

No python, temos uma linha de código que nos permite verificar o tipo da nossa variável. Mas o que é necessariamente uma variável? Na programação uma variável é um espaço na memória do computador que nos permite guardar uma informação ou um dado. Um pouco diferente da matemática onde uma variável representa apenas um número que pode variar.

Por exemplo, o número 2, na matemática, é um número inteiro. Então se utilizarmos a função `type()` e dentro dos parênteses passarmos o nosso valor, o python irá nos retornar o tipo daquele valor.

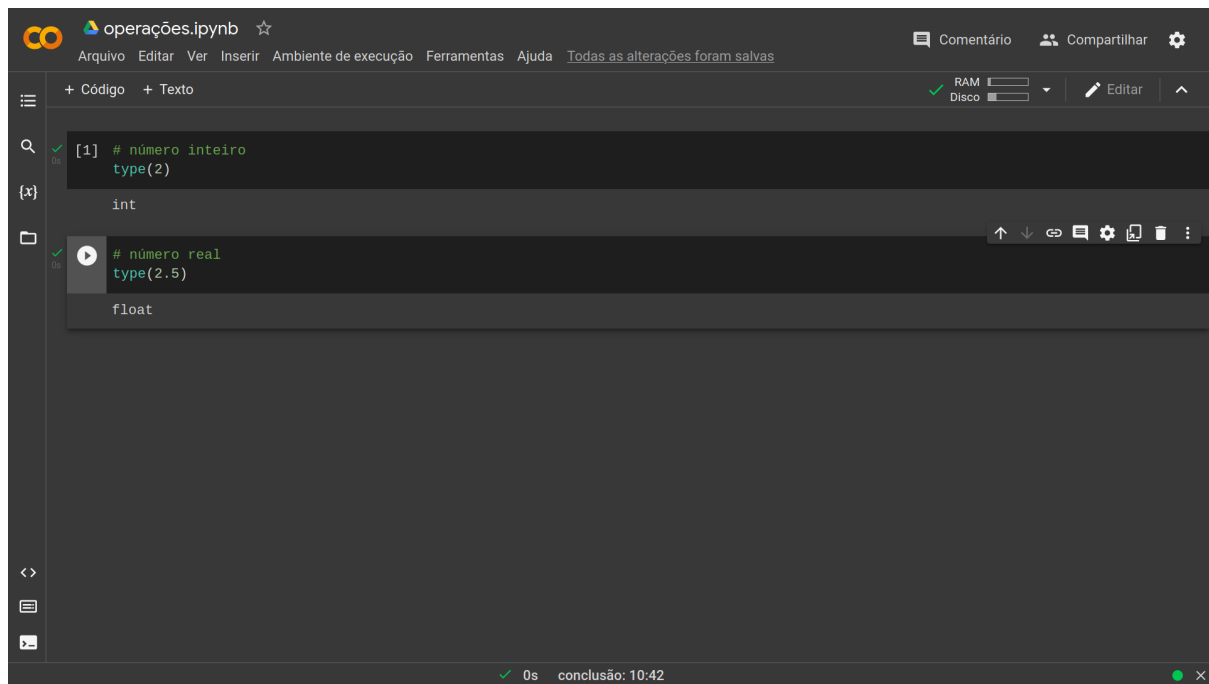


The screenshot shows a Jupyter Notebook window titled "operações.ipynb". The interface includes a top menu bar with options like "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", and "Ajuda". Below the menu, there are tabs for "+ Código" and "+ Texto". The main area displays a code cell with the following content:

```
# número inteiro  
type(2)  
  
int
```

The code cell is executed, as indicated by the green play button icon and the output "int" shown below the code. The status bar at the bottom indicates "0s" and "conclusão: 10:38".

2.5 é um valor real, então se passarmos este valor como parâmetro desta função, o programa irá nos retornar float, que no python significa real.



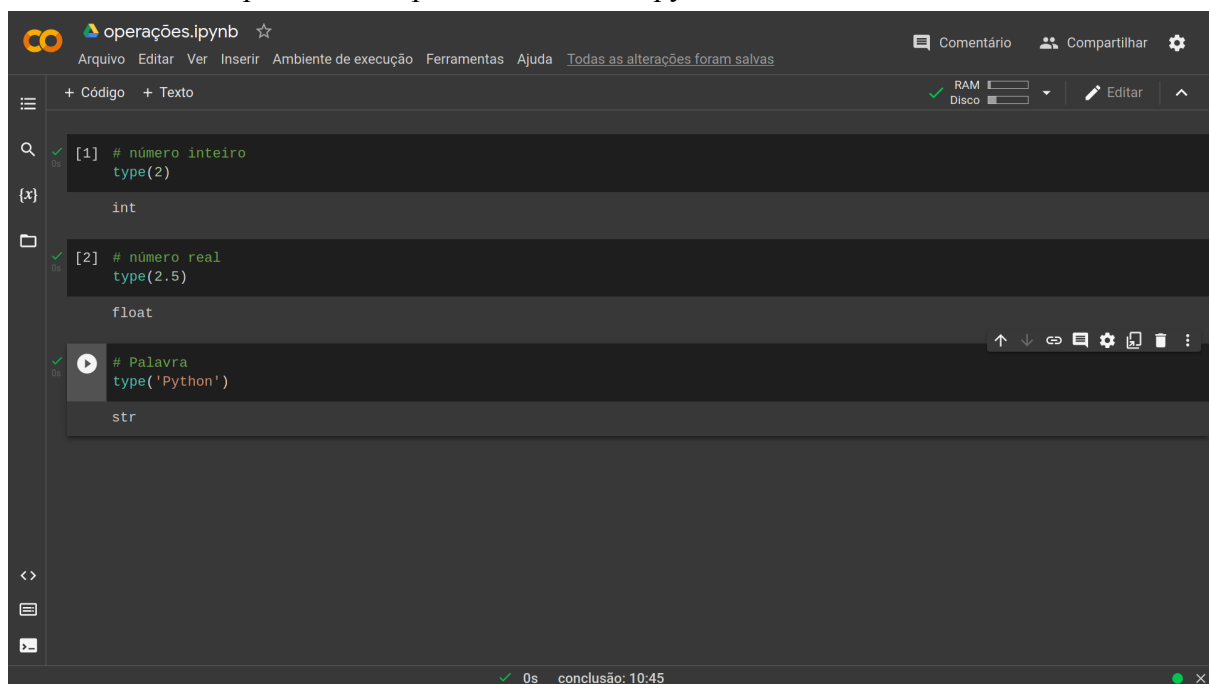
```
[1] # número inteiro
type(2)

int

# número real
type(2.5)

float
```

Mas não existem apenas estes tipos de variável no python.



```
[1] # número inteiro
type(2)

int

[2] # número real
type(2.5)

float

# Palavra
type('Python')

str
```

Se usarmos aspas simples e dentro passarmos uma palavra, vamos notar que o resultado será str. Mas o que isso significa? Significa String, que em inglês serve para representar palavras ou até mesmo frases.

```
[1] # número inteiro
type(2)

int

[2] # número real
type(2.5)

float

[3] # Palavra
type('Python')

str

# Frase
type('Inteligência artificial')

str
```

No python existem diversas vantagens de se usar a linguagem para programar, uma delas é a simplicidade. E o python é o que nós chamamos de linguagem não tipada. Mas o que isso quer dizer? Vamos ver na prática: Vamos criar uma valor variável chamada valor.

```
[4] type('Inteligência artificial')

str

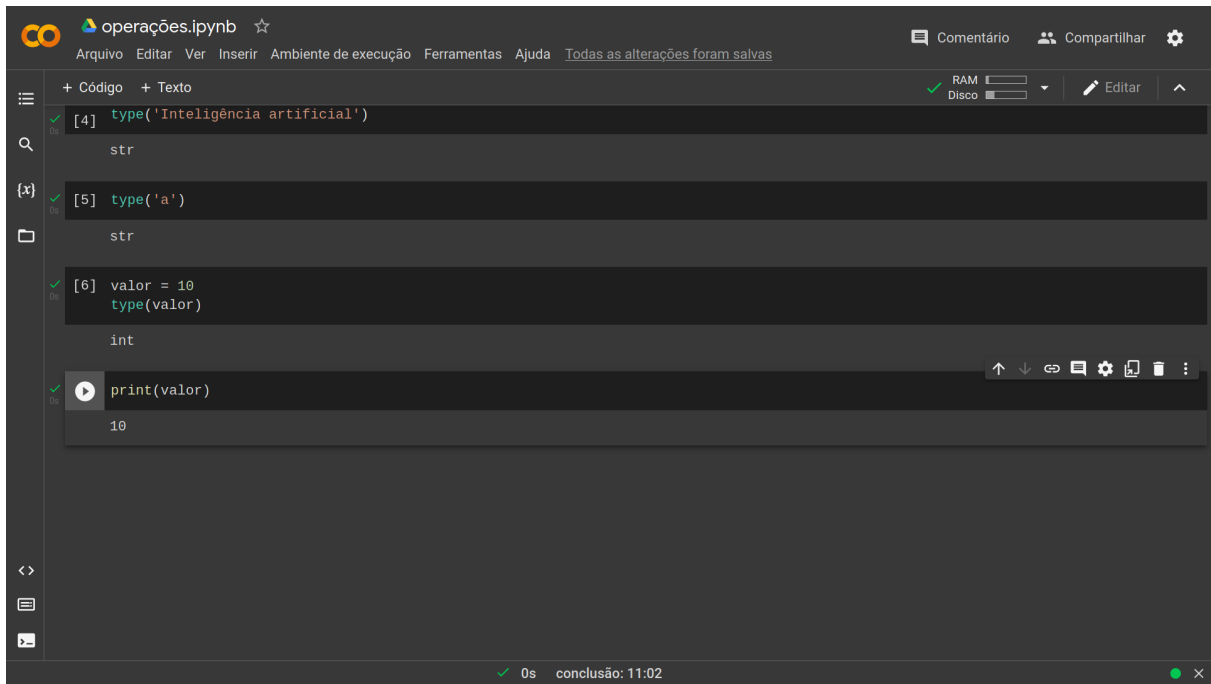
[5] type('a')

str

valor = 10
type(valor)

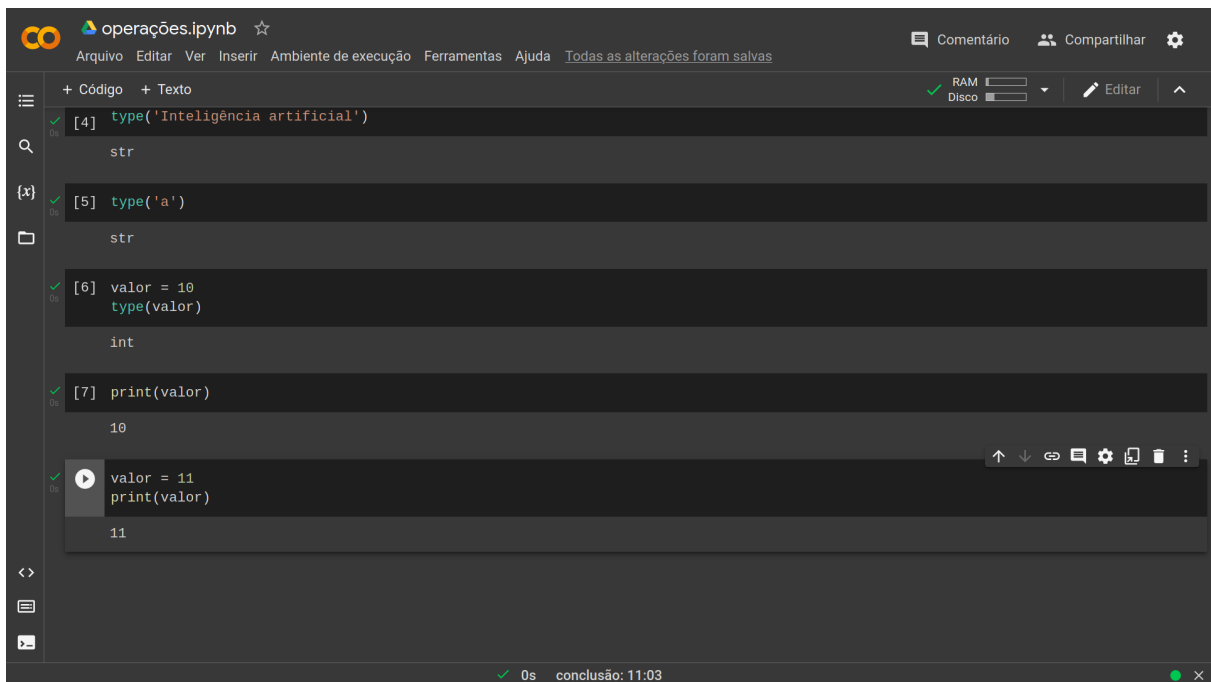
int
```

Agora esta variável está salva na memória, se tentarmos acessar podemos perceber isto.



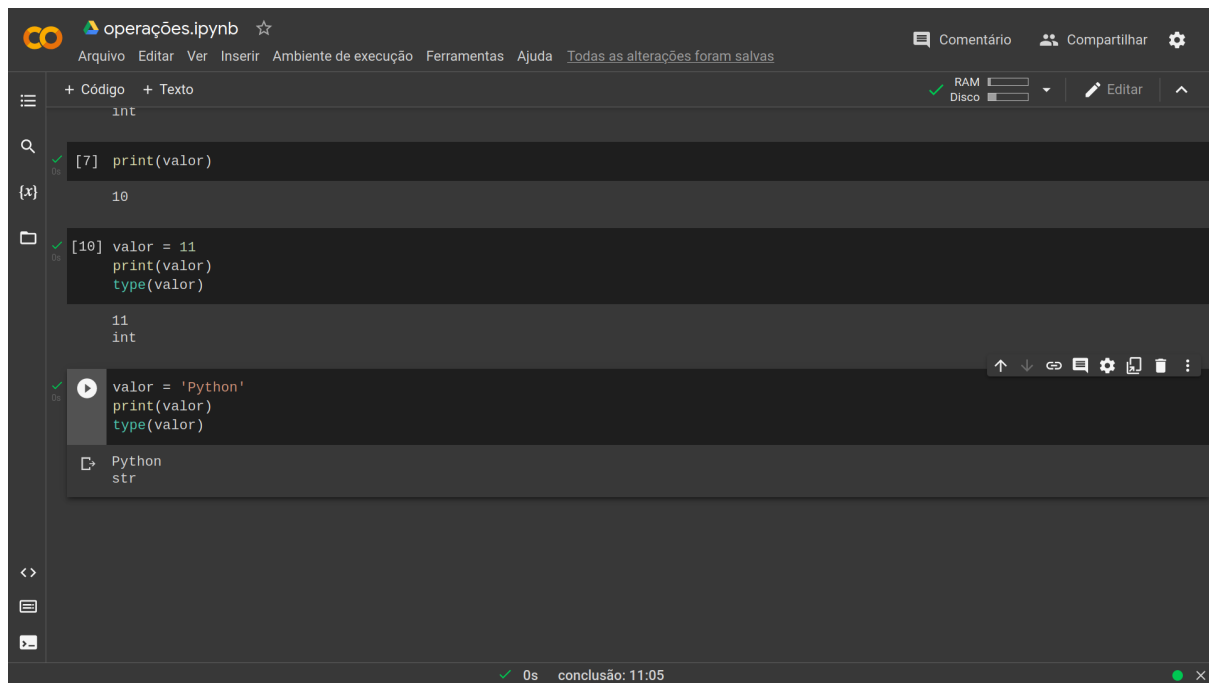
The screenshot shows a Jupyter Notebook titled "operações.ipynb". The interface includes a menu bar with options like "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", "Ajuda", and "Todas as alterações foram salvas". On the left, there are icons for file operations and a search bar. The main area displays four code cells, each with a green checkmark indicating successful execution. The first cell (index 4) defines a string variable: `type('Inteligência artificial')` followed by `str`. The second cell (index 5) defines a character variable: `type('a')` followed by `str`. The third cell (index 6) defines an integer variable: `valor = 10` followed by `type(valor)` and `int`. The fourth cell (index 7) prints the value of `valor`: `print(valor)`, with the output `10` displayed below. The bottom status bar shows "0s" and "conclusão: 11:02".

Mas como dito, uma variável serve para guardar um valor na memória, então nós podemos alterar o valor desta variável.



This screenshot shows the same Jupyter Notebook after an additional code cell has been added. The first three cells are identical to the previous screenshot. The fourth cell (index 7) now contains `print(valor)` followed by `10`. A new fifth cell (index 8) has been added, containing `valor = 11` followed by `print(valor)` and the output `11`. The bottom status bar now shows "0s" and "conclusão: 11:03".

Mas o que isso tem haver com tipagem da linguagem? Bem oque o python nos permite é não apenas mudar o valor da variável, mas também o tipo:



The screenshot shows a Jupyter Notebook interface with the title 'operações.ipynb'. The menu bar includes 'Arquivo', 'Editar', 'Ver', 'Inserir', 'Ambiente de execução', 'Ferramentas', 'Ajuda', and 'Todas as alterações foram salvas'. The left sidebar has icons for file explorer, search, and code execution. The main area displays three code cells. The first cell contains 'int'. The second cell contains '[7] print(valor)' followed by '10'. The third cell contains '[10] valor = 11', 'print(valor)', and 'type(valor)', with the output '11' and 'int' shown below. A fourth cell contains a play button icon, 'valor = 'Python'', 'print(valor)', and 'type(valor)', with the output 'Python' and 'str' shown below. The status bar at the bottom indicates '0s' and 'conclusão: 11:05'.

```
int
```

```
[7] print(valor)
```

```
10
```

```
[10] valor = 11
      print(valor)
      type(valor)
```

```
11
int
```

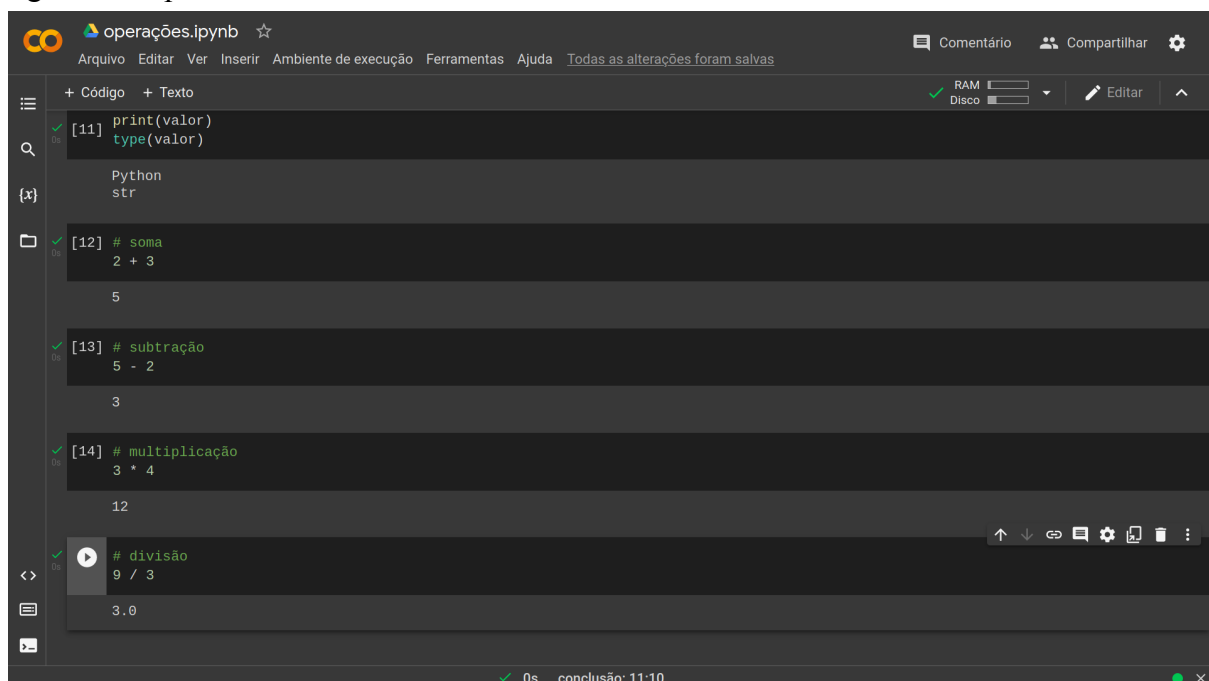
```
valor = 'Python'
print(valor)
type(valor)
```

```
Python
str
```

0s conclusão: 11:05

Ao mudar a variável para 'Python' a linguagem identifica que é uma String. Em outras linguagens isso não é possível, em java por exemplo, uma vez que uma variável está declarada ela somente pode receber valores do seu tipo declarado. Isso se chama tipagem da linguagem.

Além disso, podemos efetuar diversas operações matemáticas com o python. Vamos ver alguns exemplos.



The screenshot shows a Jupyter Notebook interface with the title 'operações.ipynb'. The menu bar includes 'Arquivo', 'Editar', 'Ver', 'Inserir', 'Ambiente de execução', 'Ferramentas', 'Ajuda', and 'Todas as alterações foram salvas'. The left sidebar has icons for file explorer, search, and code execution. The main area displays five code cells. The first cell contains '[11] print(valor)' and 'type(valor)', with the output 'Python' and 'str' shown below. The second cell contains '[12] # soma' and '2 + 3', with the output '5' shown below. The third cell contains '[13] # subtração' and '5 - 2', with the output '3' shown below. The fourth cell contains '[14] # multiplicação' and '3 \* 4', with the output '12' shown below. The fifth cell contains a play button icon, '# divisão', and '9 / 3', with the output '3.0' shown below. The status bar at the bottom indicates '0s' and 'conclusão: 11:10'.

```
[11] print(valor)
      type(valor)
```

```
Python
str
```

```
[12] # soma
      2 + 3
```

```
5
```

```
[13] # subtração
      5 - 2
```

```
3
```

```
[14] # multiplicação
      3 * 4
```

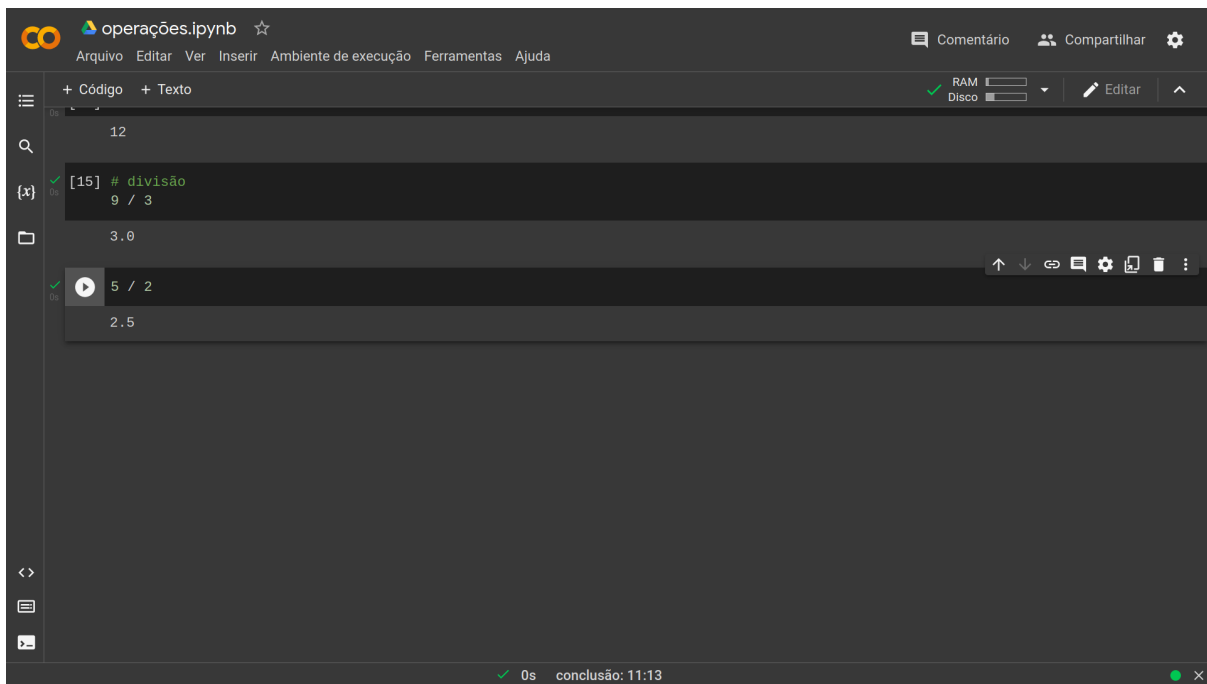
```
12
```

```
# divisão
9 / 3
```

```
3.0
```

0s conclusão: 11:10

Notem que nas operações soma, subtração e multiplicação o valor retornado é inteiro, mas na parte da divisão o valor é um número real. Isso acontece porque uma divisão pode resultar em um valor que não é inteiro, e o python já entende isso.



The screenshot shows a Jupyter Notebook titled "operações.ipynb". The interface includes a menu bar with options like "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", and "Ajuda". On the right, there are buttons for "Comentário", "Compartilhar", and a settings icon. Below the menu, there are tabs for "+ Código" and "+ Texto". The main area contains a code cell with the following content:

```
12
```

```
[15] # divisão
      9 / 3
```

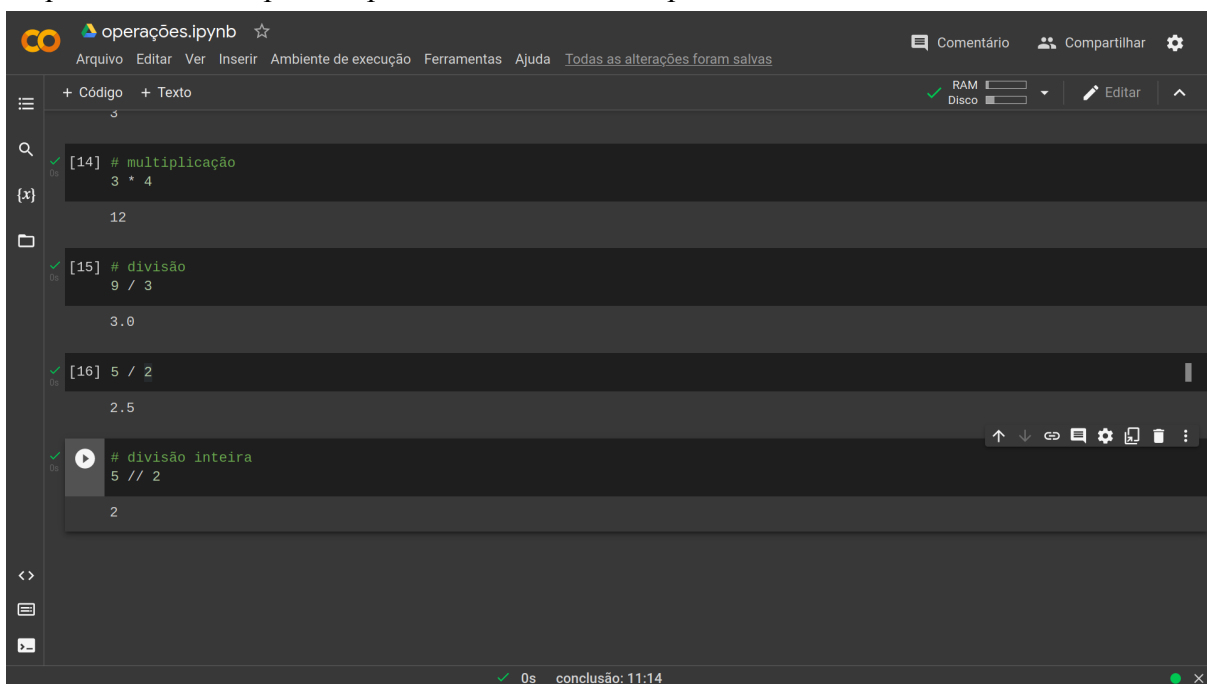
```
3.0
```

```
5 / 2
```

```
2.5
```

The status bar at the bottom indicates "0s" and "conclusão: 11:13".

se quisermos obter apenas a parte inteira da divisão podemos usar duas barras:



The screenshot shows the same Jupyter Notebook interface, but with additional code cells. The code cells are as follows:

```
[14] # multiplicação
      3 * 4
```

```
12
```

```
[15] # divisão
      9 / 3
```

```
3.0
```

```
[16] 5 / 2
```

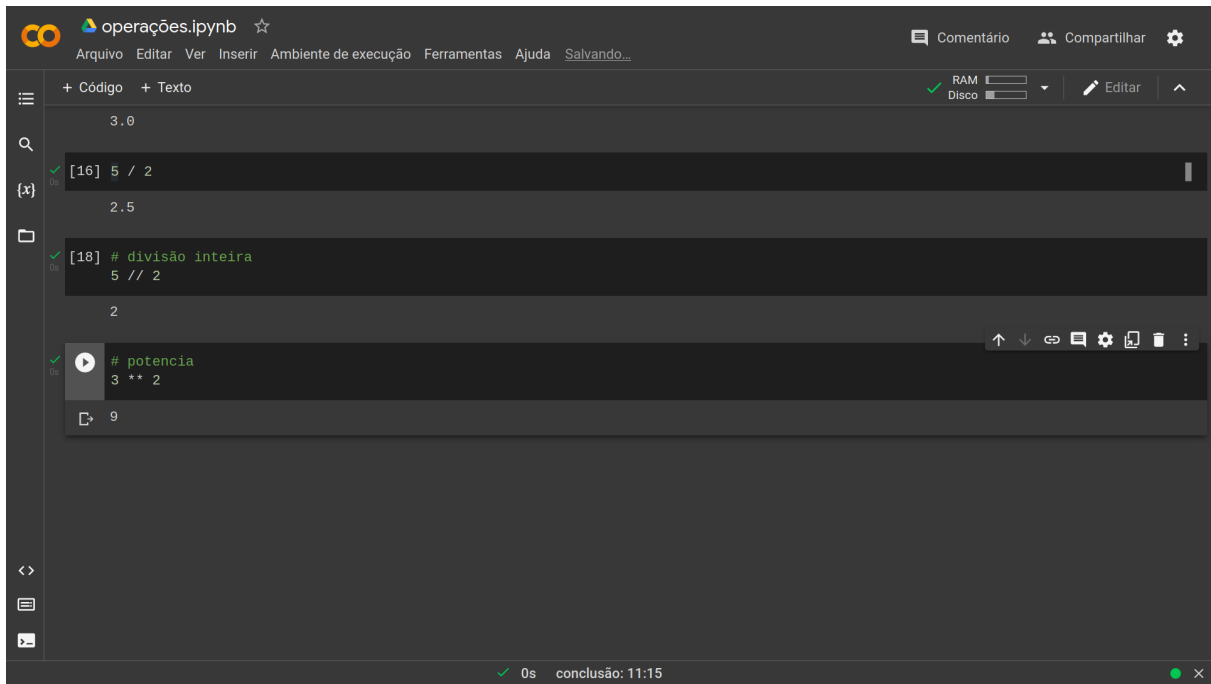
```
2.5
```

```
# divisão inteira
      5 // 2
```

```
2
```

The status bar at the bottom indicates "0s" and "conclusão: 11:14".

podemos efetuar potenciação também representando potência com dois asteriscos.



The image shows a Jupyter Notebook interface with a dark theme. The title bar reads "operações.ipynb" with a star icon. The menu bar includes "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", "Ajuda", and "Salvando...". The left sidebar has icons for a list, search, a code cell, a file explorer, and a console. The main area contains a code cell with the following content: 

```
3.0
```

```
[16] 5 / 2
```

```
2.5
```

```
[18] # divisão inteira
```

```
5 // 2
```

```
2
```

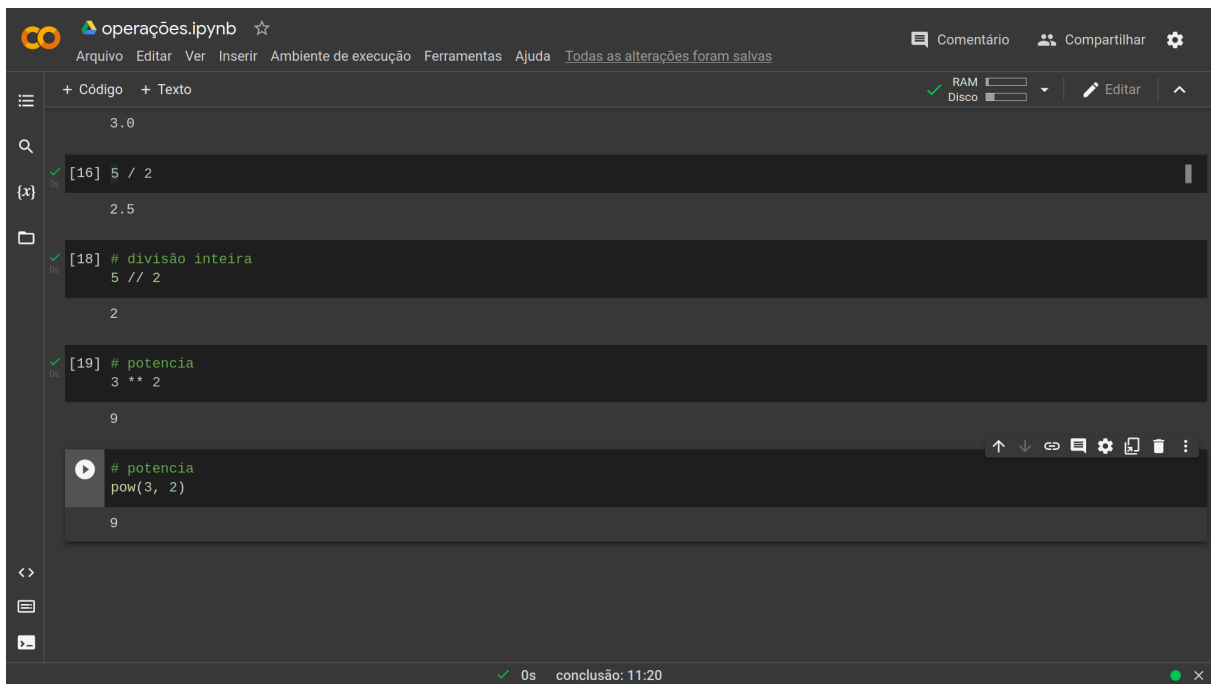
```
# potencia
```

```
3 ** 2
```

```
9
```

The cell is marked as successful with a green checkmark and "0s". The bottom status bar shows "conclusão: 11:15".

ou ainda utilizando uma função chamada `pow(base, expoente)` e passando os parâmetros base e o expoente que desejamos.



The image shows the same Jupyter Notebook interface as above, but with an additional cell. The new cell contains the following content: 

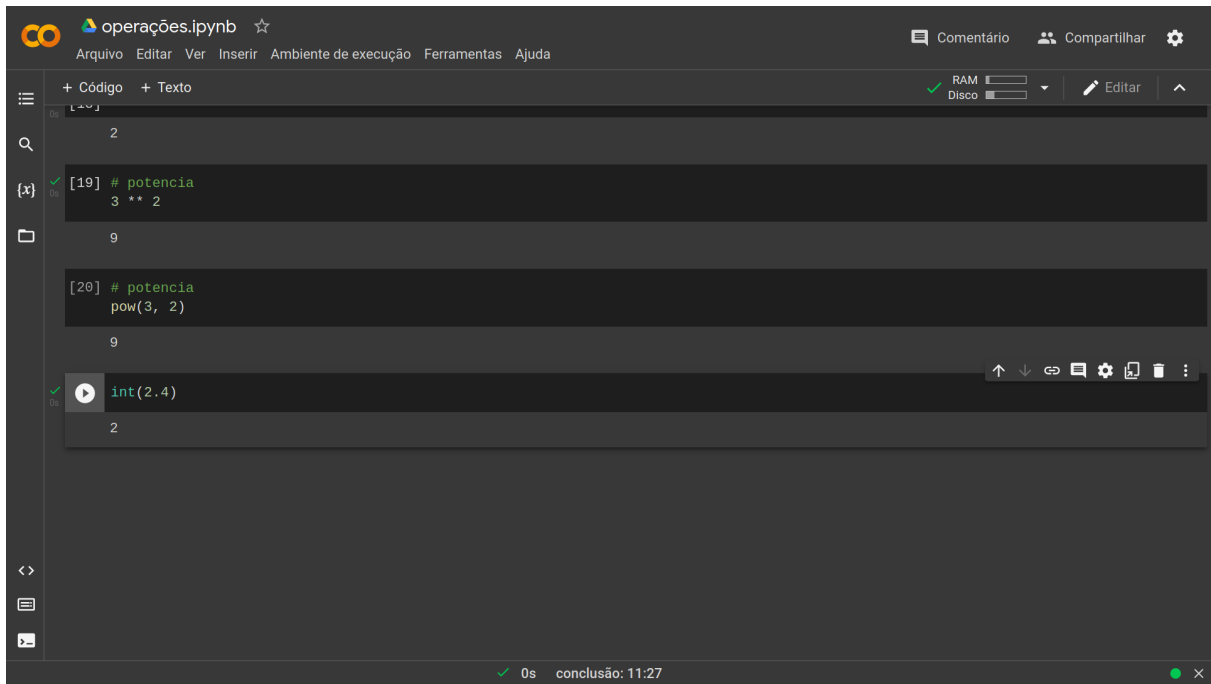
```
# potencia
```

```
pow(3, 2)
```

```
9
```

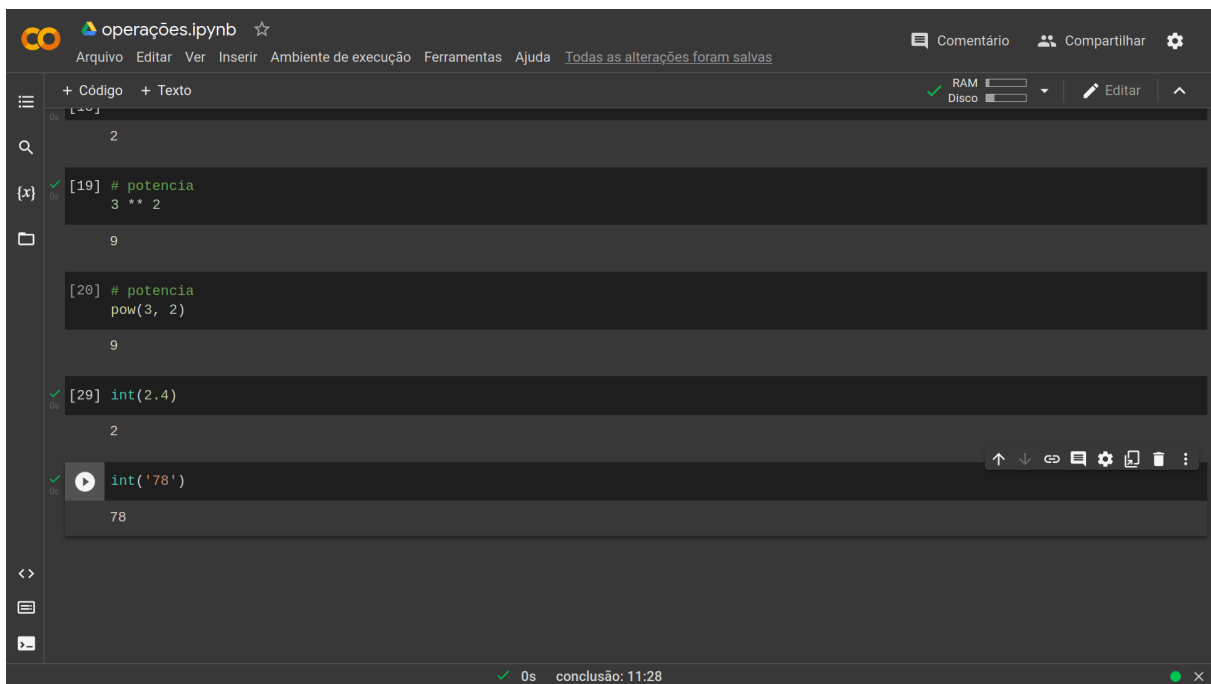
This cell is also marked as successful with a green checkmark and "0s". The bottom status bar now shows "conclusão: 11:20".

Além disso, temos algumas funções que podem nos ajudar: `int(2.4)` converte um número real para um número inteiro.



The screenshot shows a Jupyter Notebook titled "operações.ipynb". The interface includes a top menu bar with options like "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", and "Ajuda". On the right, there are buttons for "Comentário", "Compartilhar", and a settings icon. Below the menu, there are tabs for "+ Código" and "+ Texto". The notebook contains three code cells. The first cell has the number "2". The second cell contains the code `[19] # potencia` and `3 ** 2`, with the output "9". The third cell contains the code `[20] # potencia` and `pow(3, 2)`, also with the output "9". A fourth cell is currently being executed, showing the code `int(2.4)` and the output "2". The status bar at the bottom indicates "0s" and "conclusão: 11:27".

também funciona para caso o número esteja dentro de uma String



This screenshot shows the same Jupyter Notebook interface as the previous one, but with an additional code cell. The fourth cell, which was previously being executed, now shows the code `int(2.4)` and the output "2". A new fifth cell is being executed, containing the code `int('78')` and the output "78". The status bar at the bottom now shows "0s" and "conclusão: 11:28".

mas perceba que se colocar um valor que não seja um número dentro da String o python irá acusar um erro, já que não se pode converter este número.



The image shows a Jupyter Notebook interface with the title "operações.ipynb". The top bar includes a menu with "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", "Ajuda", and "Todas as alterações foram salvas". On the right, there are buttons for "Comentário", "Compartilhar", and a settings icon. The left sidebar has icons for "Código", "Texto", "RAM", "Disco", and "Editar". The main area displays three code cells. The first cell contains `[19] 3 ** 2` and outputs `9`. The second cell contains `[20] # potencia` and `pow(3, 2)`, also outputting `9`. The third cell contains `[29] int(2.4)` and outputs `2`. Below this, a fourth cell shows a red error icon and a traceback for `int('Python')`. The traceback indicates a `ValueError: invalid literal for int() with base 10: 'Python'`. At the bottom, a status bar shows "0s" and "conclusão: 11:30".

```
[19] 3 ** 2
9

[20] # potencia
pow(3, 2)
9

[29] int(2.4)
2

int('Python')
Traceback (most recent call last)
<ipython-input-33-6d5e674a3161> in <module>
----> 1 int('Python')

ValueError: invalid literal for int() with base 10: 'Python'
```

podemos também converter um número inteiro para um real através da função `float()`

The image shows the same Jupyter Notebook interface as before, but with a fourth code cell added. This cell contains `float(10)` and outputs `10.0`. The status bar at the bottom now shows "0s" and "conclusão: 11:32".

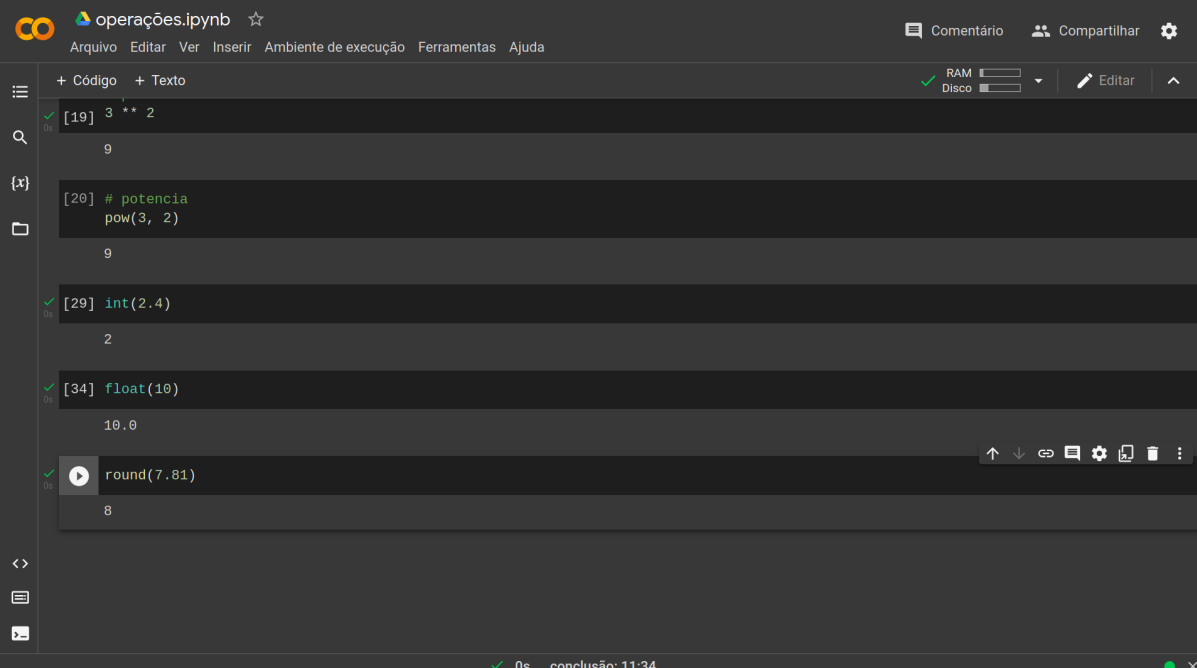
```
[19] 3 ** 2
9

[20] # potencia
pow(3, 2)
9

[29] int(2.4)
2

float(10)
10.0
```

Podemos também arredondar um valor com o `round()`



The screenshot shows a Jupyter Notebook titled "operações.ipynb". The interface includes a top menu bar with options like "Arquivo", "Editar", "Ver", "Inserir", "Ambiente de execução", "Ferramentas", and "Ajuda". On the left, there is a sidebar with icons for file explorer, search, and other functions. The main area displays five code cells, each with a prompt (e.g., [19], [20]) and its output. The code cells contain the following Python code:

```
[19] 3 ** 2
9

[20] # potencia
pow(3, 2)
9

[29] int(2.4)
2

[34] float(10)
10.0

round(7.81)
8
```

At the bottom of the notebook, a status bar indicates "0s" and "conclusão: 11:34".

Bom esta foi uma aula com os operadores básicos no python, muito obrigado por acompanhar até aqui, nos vemos na próxima aula.