

A grayscale photograph of a person's hands typing on a laptop keyboard. A cup of tea with a tea bag is on the desk next to the laptop. The image is split vertically, with the left side being lighter and the right side being darker.

Estruturas de dados e json

Prof. Tiago Martins de Alexandre

Faculdade
IMPACTA

Tópicos

Nesta Unidade iremos abordar:

- Estrutura de dados em python
 - String
 - Int
 - Float
 - Boolean
 - List
 - Tuple
 - Dictionary
- Conversão de python para json
- Conversão de json para python

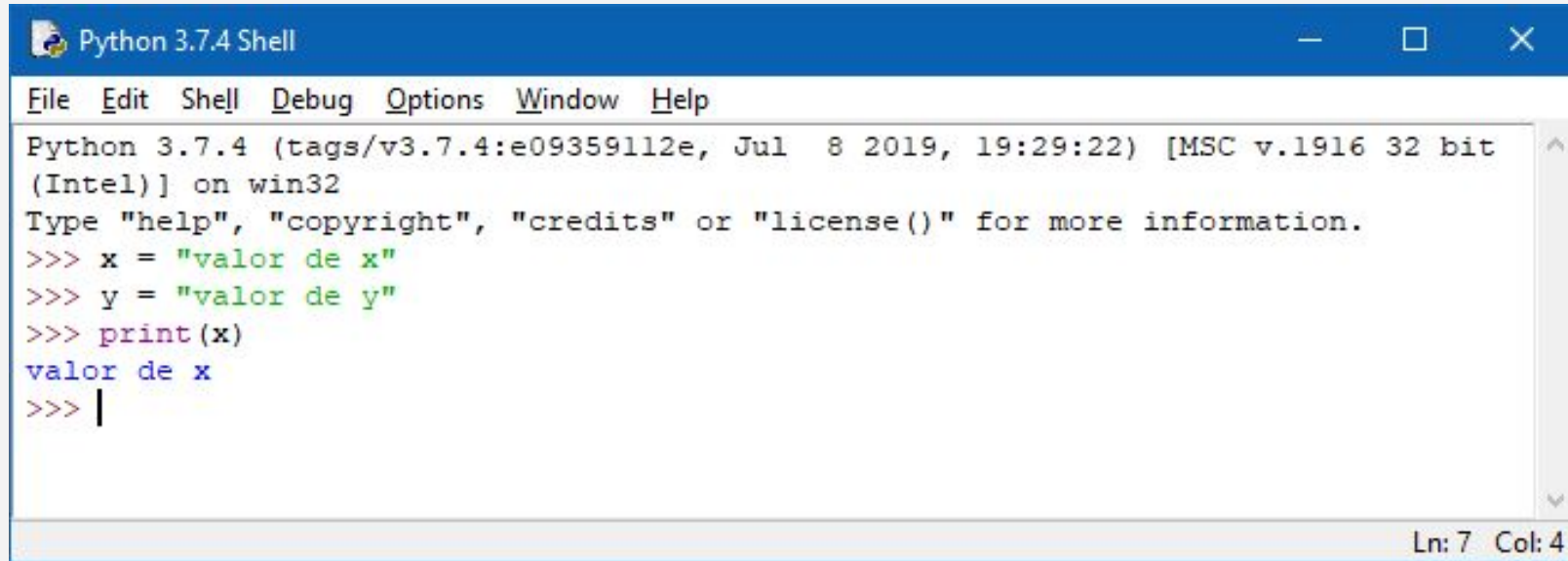
Estruturas de dados em python

String

x = "valor de x"

y = "valor de y"

print(x)



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> x = "valor de x"
>>> y = "valor de y"
>>> print(x)
valor de x
>>> |
```

Ln: 7 Col: 4

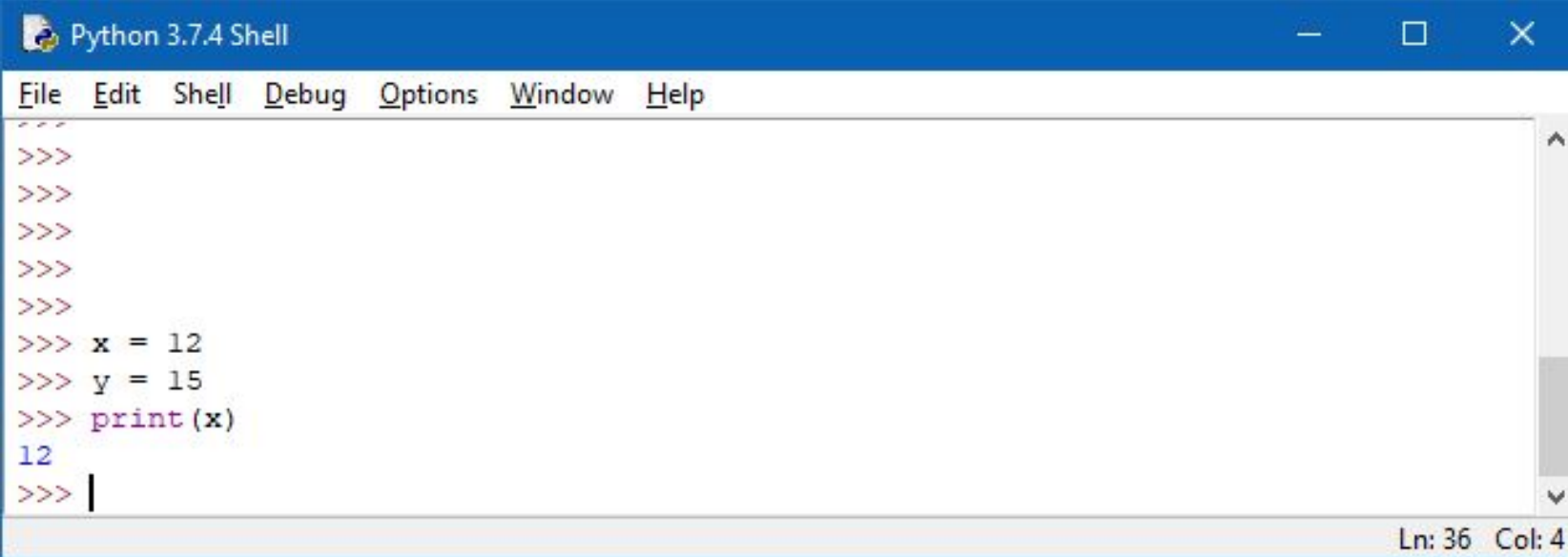
Estruturas de dados em python

Int

x = 12

y = 15

print(x)

A screenshot of a Python 3.7.4 Shell window. The window has a blue title bar with the text "Python 3.7.4 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window is a text editor with a white background. It contains the following code:

```
>>>  
>>>  
>>>  
>>>  
>>> x = 12  
>>> y = 15  
>>> print(x)  
12  
>>> |
```

The code is color-coded: the prompt is red, keywords like 'print' are purple, and the output '12' is blue. A vertical scrollbar is on the right side of the text area. At the bottom right of the window, the status bar shows "Ln: 36 Col: 4".

Estruturas de dados em python

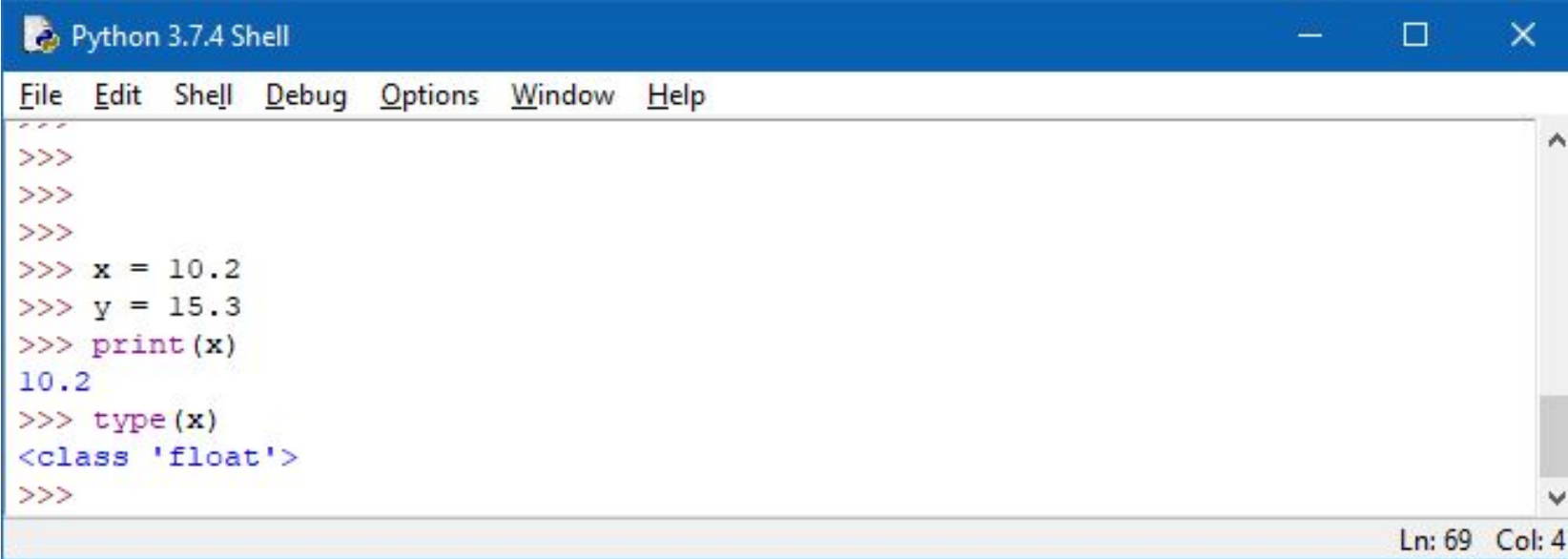
Float

x = 10.2

y = 15.3

print(x)

type(x)



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>> x = 10.2
>>> y = 15.3
>>> print(x)
10.2
>>> type(x)
<class 'float'>
>>>
```

Ln: 69 Col: 4

Estruturas de dados em python

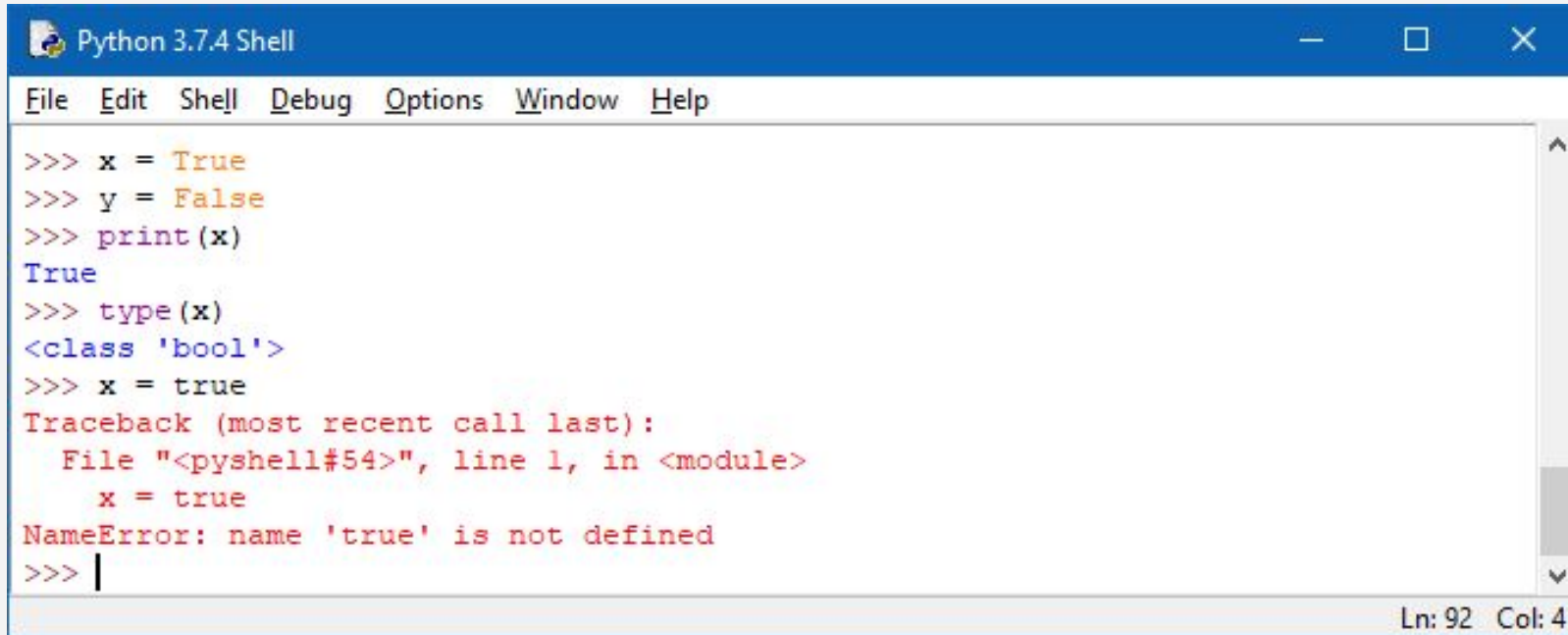
- Boolean

x = True

y = False

print(x)

type(x)



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> x = True
>>> y = False
>>> print(x)
True
>>> type(x)
<class 'bool'>
>>> x = true
Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    x = true
NameError: name 'true' is not defined
>>> |
```

Ln: 92 Col: 4

Estruturas de dados em python

- List

x = ['verde', 'roxo']

y = [1,2,3]

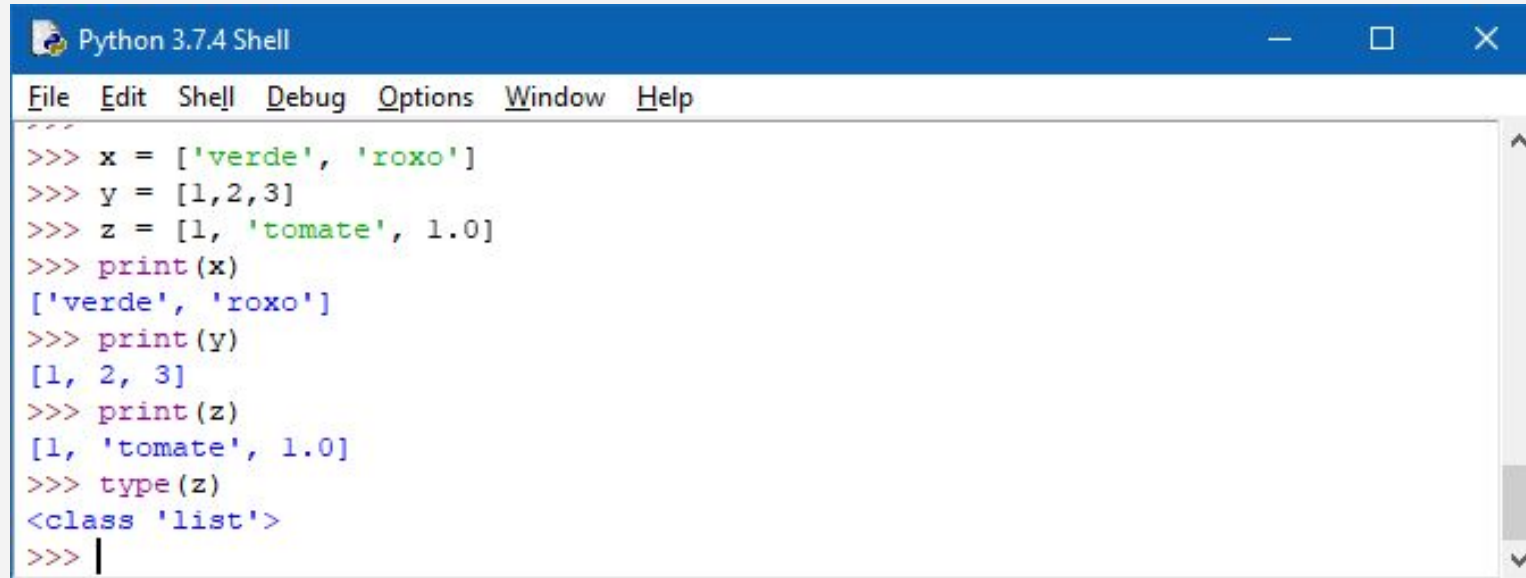
z = [1, 'tomate', 1.0]

print(x)

print(y)

print(z)

type(x)

A screenshot of a Python 3.7.4 Shell window. The window has a blue title bar with the text "Python 3.7.4 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a Python REPL session. The user has entered several lines of code: three list assignments (x, y, and z), followed by print statements for each list, and a type check for list z. The output shows the lists as they are and the type as <class 'list'>. The cursor is at the end of the last prompt line.

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> x = ['verde', 'roxo']
>>> y = [1,2,3]
>>> z = [1, 'tomate', 1.0]
>>> print(x)
['verde', 'roxo']
>>> print(y)
[1, 2, 3]
>>> print(z)
[1, 'tomate', 1.0]
>>> type(z)
<class 'list'>
>>> |
```

Estruturas de dados em python

Tuple

x = ('verde', 'roxo')

y = (1,2,3)

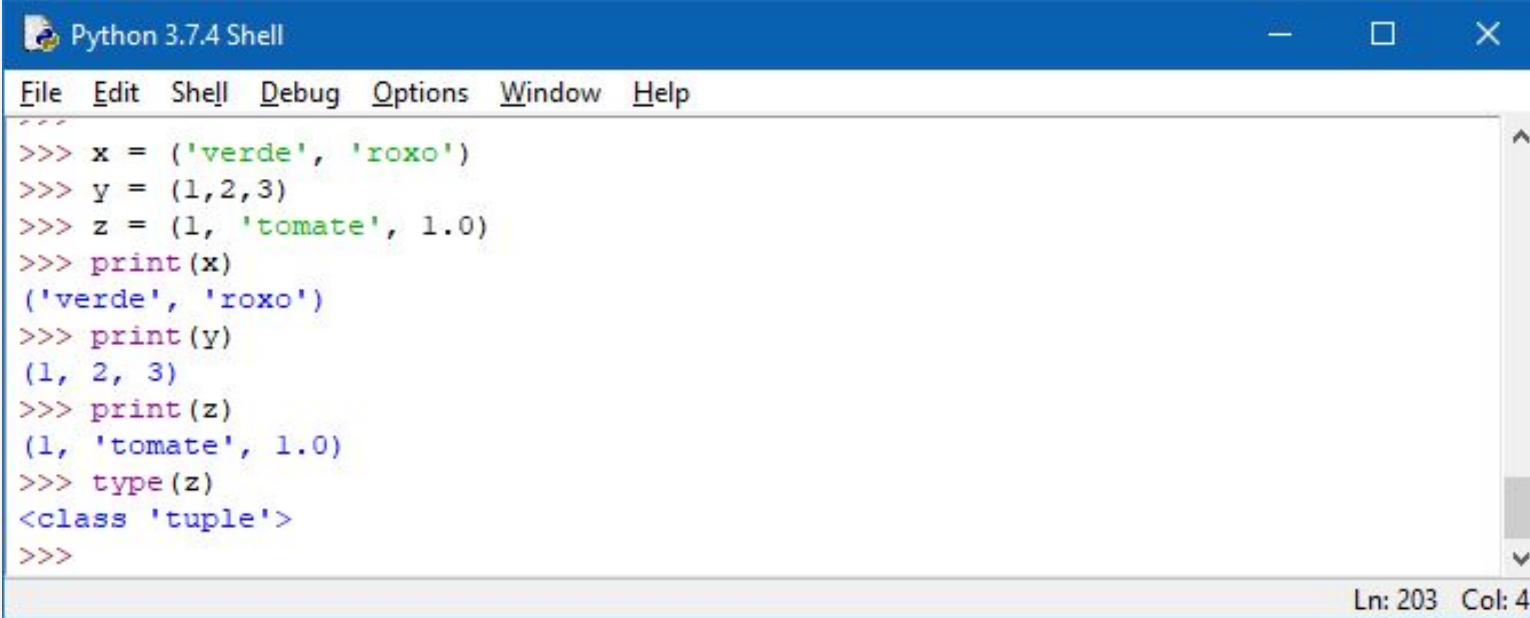
z = (1, 'tomate', 1.0)

print(x)

print(y)

print(z)

type(x)



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> x = ('verde', 'roxo')
>>> y = (1,2,3)
>>> z = (1, 'tomate', 1.0)
>>> print(x)
('verde', 'roxo')
>>> print(y)
(1, 2, 3)
>>> print(z)
(1, 'tomate', 1.0)
>>> type(z)
<class 'tuple'>
>>>
```

Ln: 203 Col: 4

Estruturas de dados em python

Dictionary

```
x = {'cor1':'verde', 'cor2': 'roxo'}
```

```
y = {"1":1,"2":2,"3":3}
```

```
z = {"1":"um","frutas": ['tomate','uva'],  
"itens": {1:"item 1",2: "item 2", 3: "item 3"}}
```

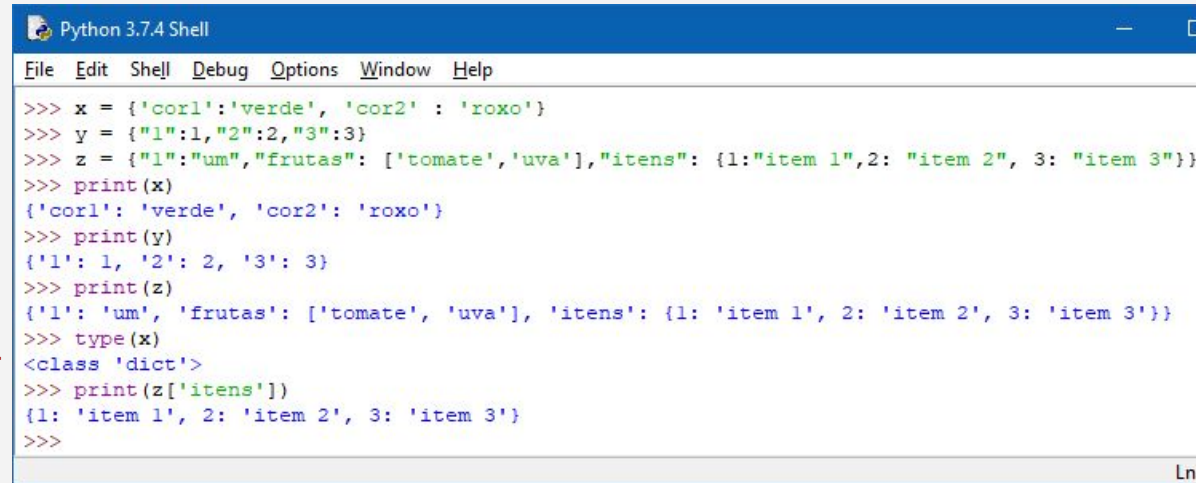
```
print(x)
```

```
print(y)
```

```
print(z)
```

```
type(x)
```

```
print(z['itens'])
```

A screenshot of a Python 3.7.4 Shell window. The window has a blue title bar with the text "Python 3.7.4 Shell" and standard window controls. Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main area of the window contains a Python script. The script defines three dictionaries: x, y, and z. x is {'cor1':'verde', 'cor2': 'roxo'}. y is {"1":1,"2":2,"3":3}. z is {"1":"um","frutas": ['tomate','uva'], "itens": {1:"item 1",2: "item 2", 3: "item 3"}}. The script then prints each dictionary, prints the type of x (which is <class 'dict'>), and prints the value of z['itens'] (which is {1: 'item 1', 2: 'item 2', 3: 'item 3'}). The output of the script is shown in the same color as the code. At the bottom right of the window, there is a status bar with the text "Ln:".

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> x = {'cor1':'verde', 'cor2': 'roxo'}
>>> y = {"1":1,"2":2,"3":3}
>>> z = {"1":"um","frutas": ['tomate','uva'], "itens": {1:"item 1",2: "item 2", 3: "item 3"}}
>>> print(x)
{'cor1': 'verde', 'cor2': 'roxo'}
>>> print(y)
{'1': 1, '2': 2, '3': 3}
>>> print(z)
{'1': 'um', 'frutas': ['tomate', 'uva'], 'itens': {1: 'item 1', 2: 'item 2', 3: 'item 3'}}
>>> type(x)
<class 'dict'>
>>> print(z['itens'])
{1: 'item 1', 2: 'item 2', 3: 'item 3'}
>>>
```

Conversão de python para json

```
import json
```

Considere o seguinte objeto (dict):

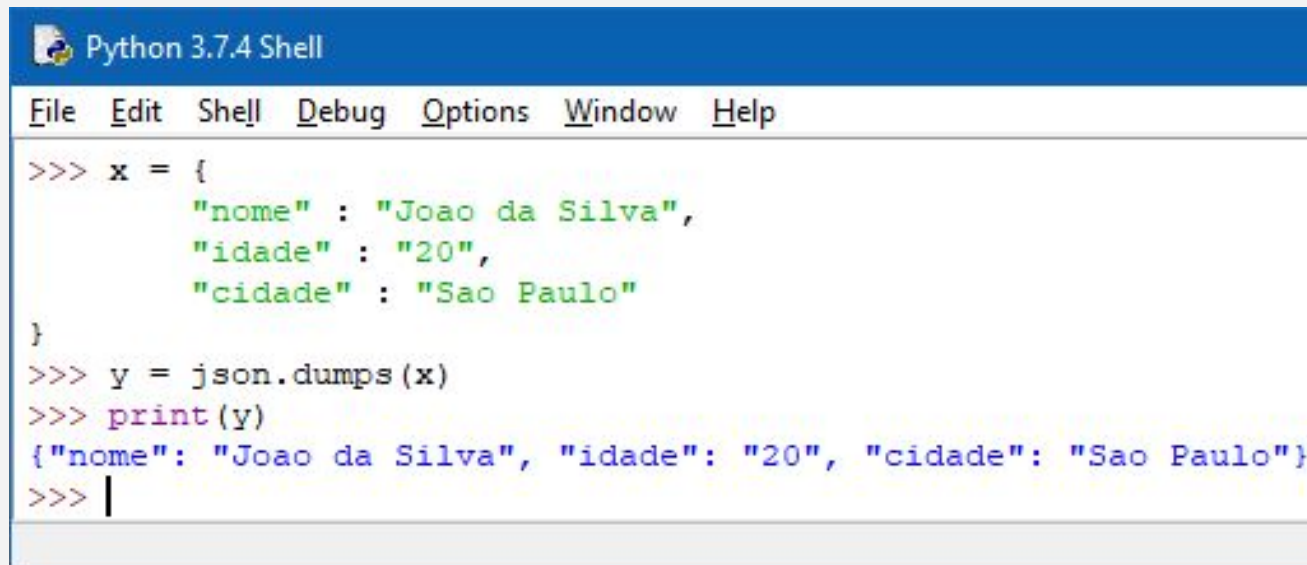
```
x = {  
    "nome": "Joao da Silva",  
    "idade": "20",  
    "cidade": "Sao Paulo"  
}
```

Converter para json:

```
y = json.dumps(x)
```

Ver o resultado:

```
print(y)
```



```
Python 3.7.4 Shell  
File Edit Shell Debug Options Window Help  
>>> x = {  
    "nome" : "Joao da Silva",  
    "idade" : "20",  
    "cidade" : "Sao Paulo"  
}  
>>> y = json.dumps(x)  
>>> print(y)  
{"nome": "Joao da Silva", "idade": "20", "cidade": "Sao Paulo"}  
>>> |
```

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

Conversão de json para python

```
import json
```

Considere o seguinte json:

```
x = "{ \"nome\" : \"Joao da Silva\", \"idade\" : \"20\", \"cidade\" : \"Sao Paulo\" }"
```

'Parse':

```
y = json.loads(x)
```

Conferir o tipo

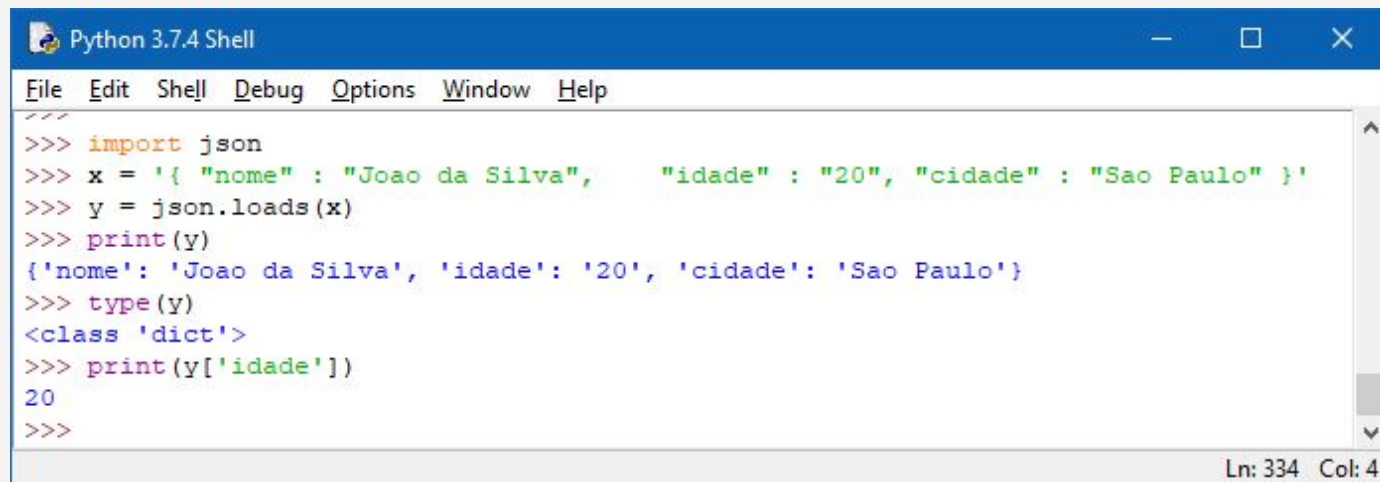
```
type(y)
```

O resultado, neste caso,
será um dicionário:

```
print(y)
```

É possível acessar os objetos do
dicionário normalmente

```
print(y["idade"])
```

A screenshot of a Python 3.7.4 Shell window. The window has a blue title bar with the text "Python 3.7.4 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main area of the window contains a Python script being executed in a REPL. The script starts with three blank lines, followed by "import json", then "x = '{ \"nome\" : \"Joao da Silva\", \"idade\" : \"20\", \"cidade\" : \"Sao Paulo\" }'", then "y = json.loads(x)", then "print(y)", which outputs a dictionary: {"nome": "Joao da Silva", "idade": "20", "cidade": "Sao Paulo"}. This is followed by "type(y)", which outputs "<class 'dict'>", then "print(y['idade'])", which outputs "20", and finally another blank line. The bottom right corner of the window shows "Ln: 334 Col: 4".

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
>>> import json
>>> x = '{ \"nome\" : \"Joao da Silva\", \"idade\" : \"20\", \"cidade\" : \"Sao Paulo\" }'
>>> y = json.loads(x)
>>> print(y)
{'nome': 'Joao da Silva', 'idade': '20', 'cidade': 'Sao Paulo'}
>>> type(y)
<class 'dict'>
>>> print(y['idade'])
20
>>>
```

Resumo

Nesta Unidade abordamos:

- Estruturas de dados em python
 - String
 - Int
 - Float
 - Boolean
 - List
 - Tuple
 - Dictionary
- Conversão de python para json
- Conversão de json para python