Introdução a Python

Labic
Laboratório de estudos sobre
Imagem e Cibercultura

Sobre Python

- Criada em 1991 pelo neerlandês **Guido van Rossum** (1956-)
- Software livre
- Multiplataforma (Linux/Mac/Windows/…)
- Linguagem abstrata (interpretada)
- VHLL: Very High Level Language
- Nome inspirado nos filmes ingleses de **Mounty Python**

Características

- Fácil de aprender
- Linguagem dinâmica de sintaxe simples e elegante
- Rápido de desenvolver
- Vem instalado por padrão no Linux, Mac e agora também no Windows 10

Para que se usa Python?

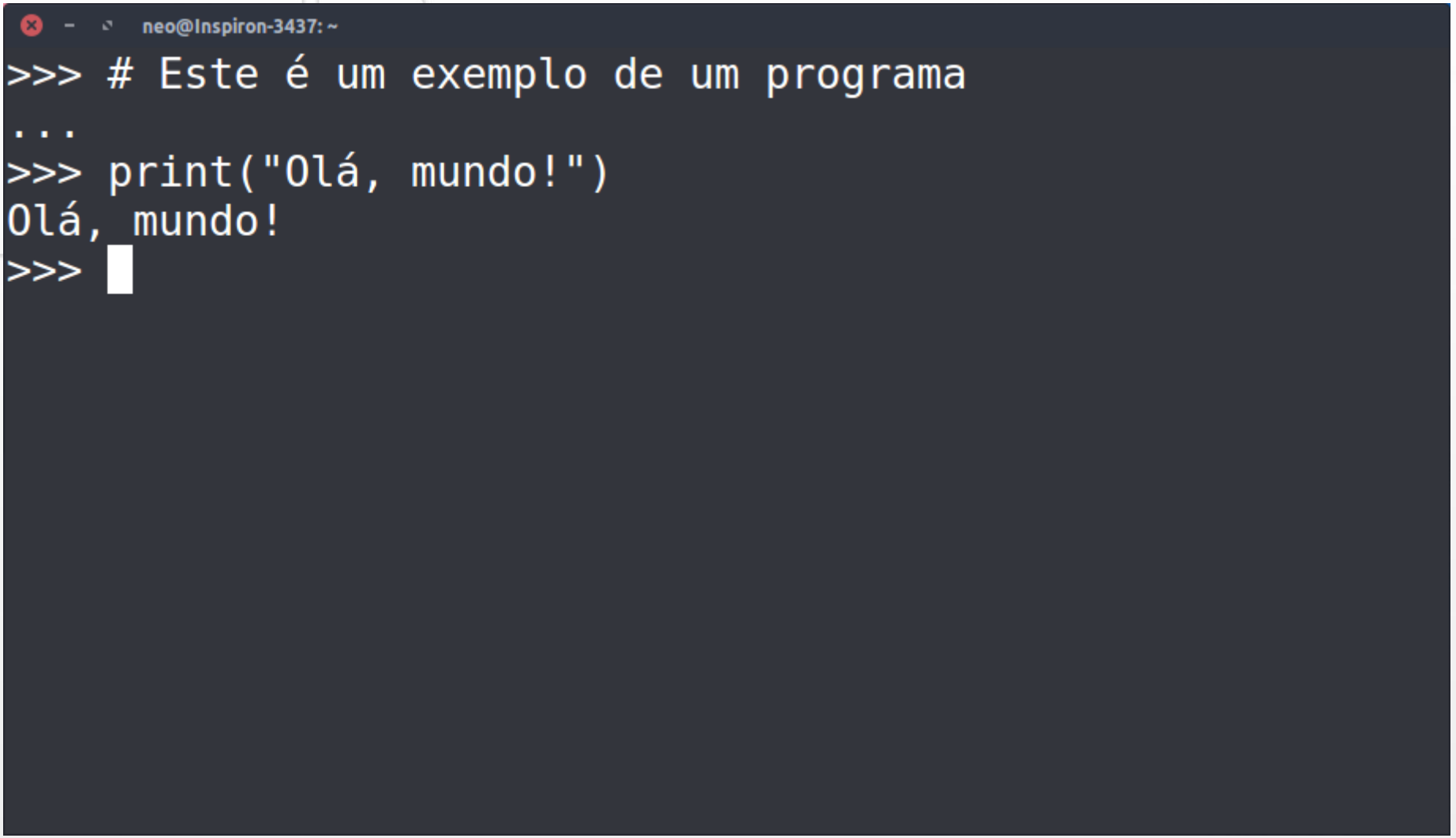
- Desenvolvimento Web;
- Programas educacionais;
- Análise de dados;
- Criação de softwares;
- Programação colaborativa;
- Jogos e gráficos 3D;
- **Entre outros vários usos.**

Quem usa Python?

O que vamos aprender hoje?

1) ~~O que é Python~~;
2) ***Exemplo de um programa;***
3) *Básicos da linguagem;*
4) **Exercício:** ‘Olá, nome!’
5) *Aprendendo a sintaxe;*
6) **Exercício:** criar um contador de palavras em uma frase especificada pelo usuário.
7) *Bônus: escrever numa tabela os resultados obtidos.*

Exemplo de um programa



Acima: comentários começando com #, funções terminando em parênteses. As três setas representam as linhas que são comandos para execução.

Conhecendo a linguagem

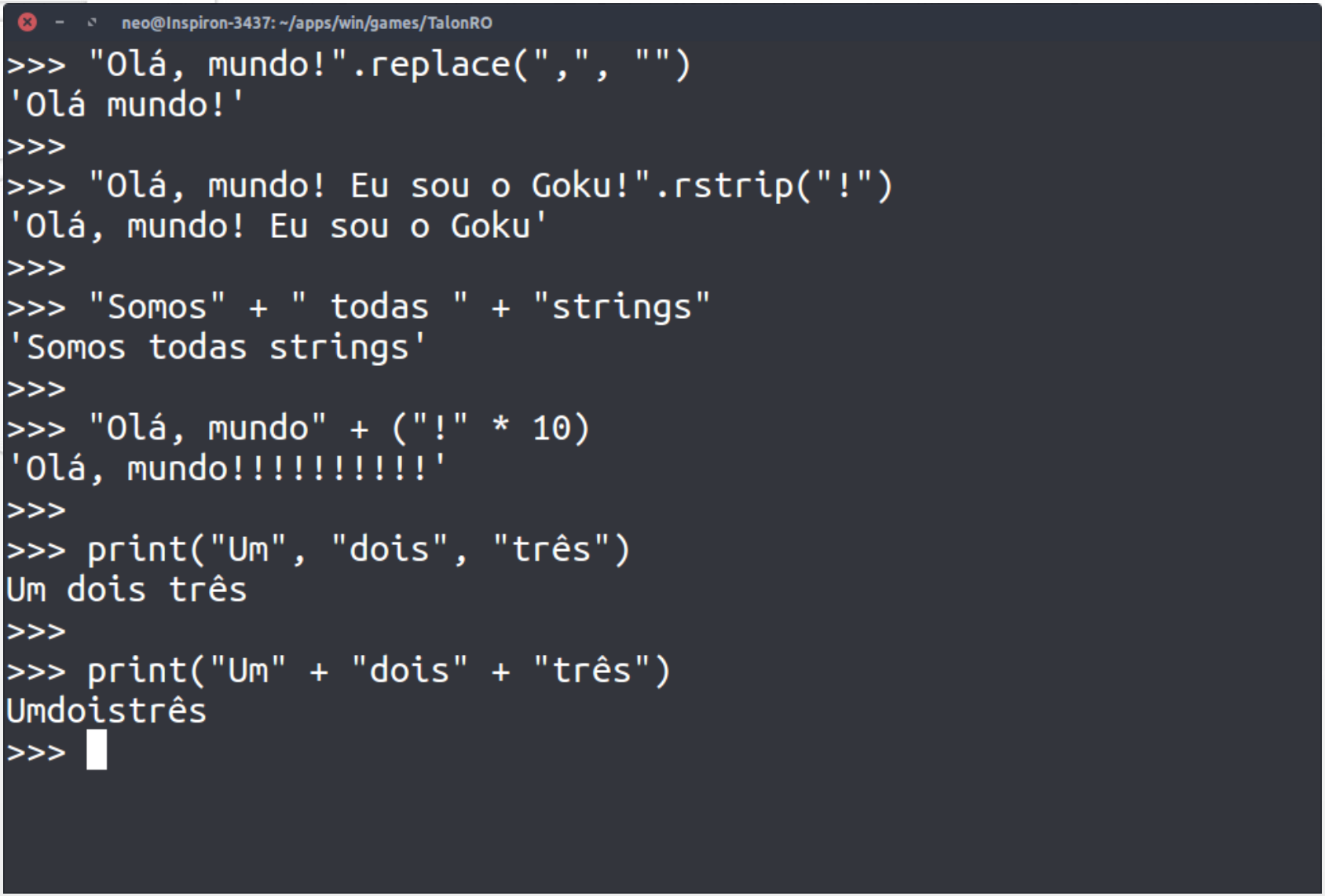
1) Strings;
2) Funções;
3) Erros;
4) Variáveis;
5) Listas e *tuples*;
6) Dicionários;
7) Comparando coisas;
8) Criando condições e *loops*;
9) Escrevendo funções.

Strings e funções

```
neo@Inspiron-3437: ~/apps/win/games/TalonRO
>>> # O que são strings?
...
>>> "Olá, mundo! Eu sou uma string!"
'Olá, mundo! Eu sou uma string!'
>>>
>>> "Olá, mundo!".lower()
'olá, mundo!'
>>>
>>> "Olá, mundo!".upper()
'OLÁ, MUNDO!'
>>>
>>> "Olá, mundo!"[0:3]
'Olá'
>>>
>>> "Olá, mundo!"[5:].capitalize()
'Mundo!'
>>>
>>> print("Olá, mundo!".upper())
OLÁ, MUNDO!
>>>
```

Acima: exemplo de uma string (“Olá mundo!”) e algumas de suas funções..

Strings e funções



Acima: mais exemplos de possíveis operações com strings.
Nota-se como a função print() ***sempre*** *retorna uma string, portanto sem aspas.*

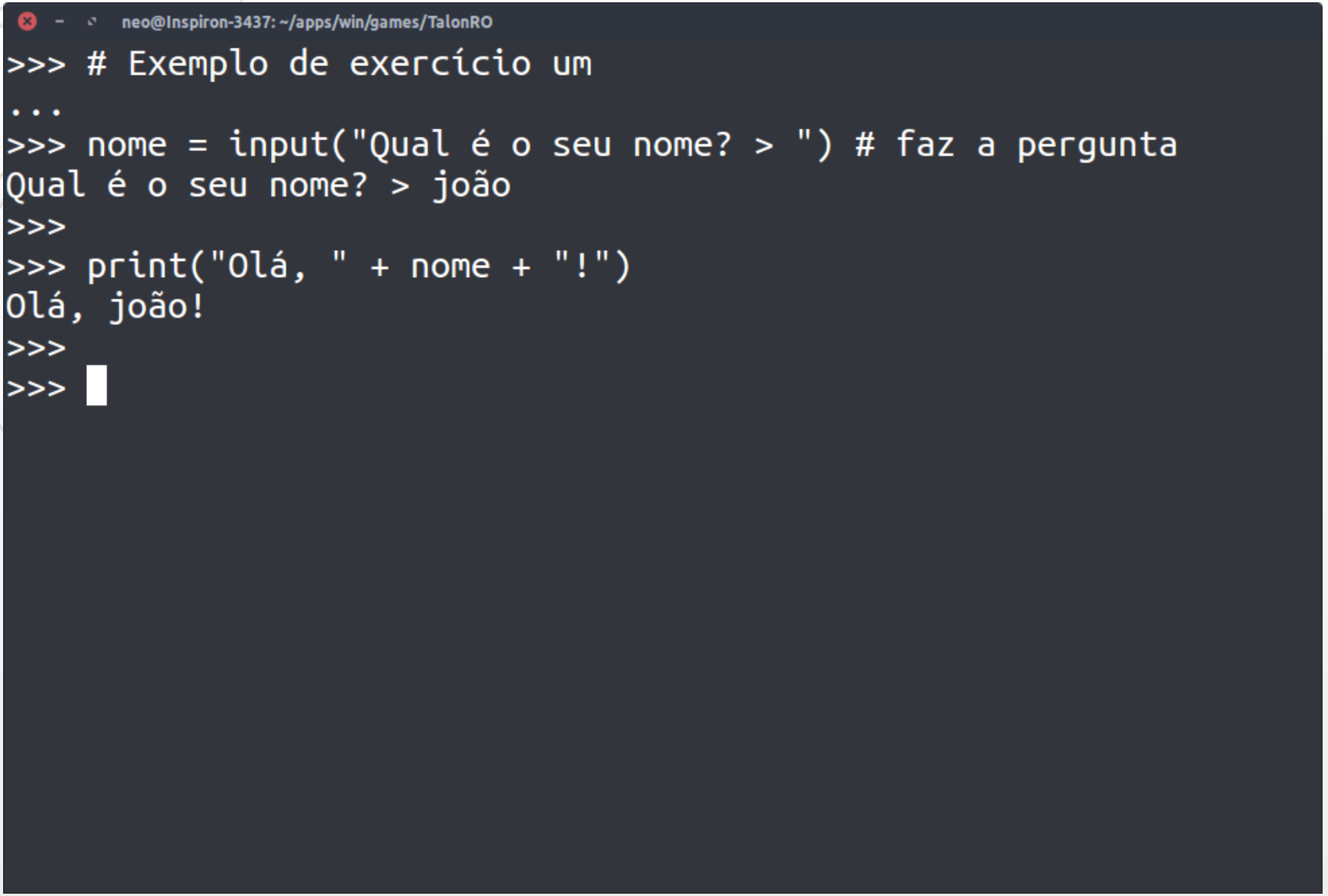
Exercício 1

Crie um programa que **pergunte** o nome do usuário e responda-o com uma mensagem.

Funções para utilizar:

print() e **input()**

Exercício 1: solução



Acima: exemplo de solução para o exercício um. As linhas a serem salvas em arquivo são os comandos de execução (começando com >>>).

Conhecendo a linguagem

1) Strings;

2) Funções;

3) Erros;

4) Variáveis;

5) Listas e *tuples*;

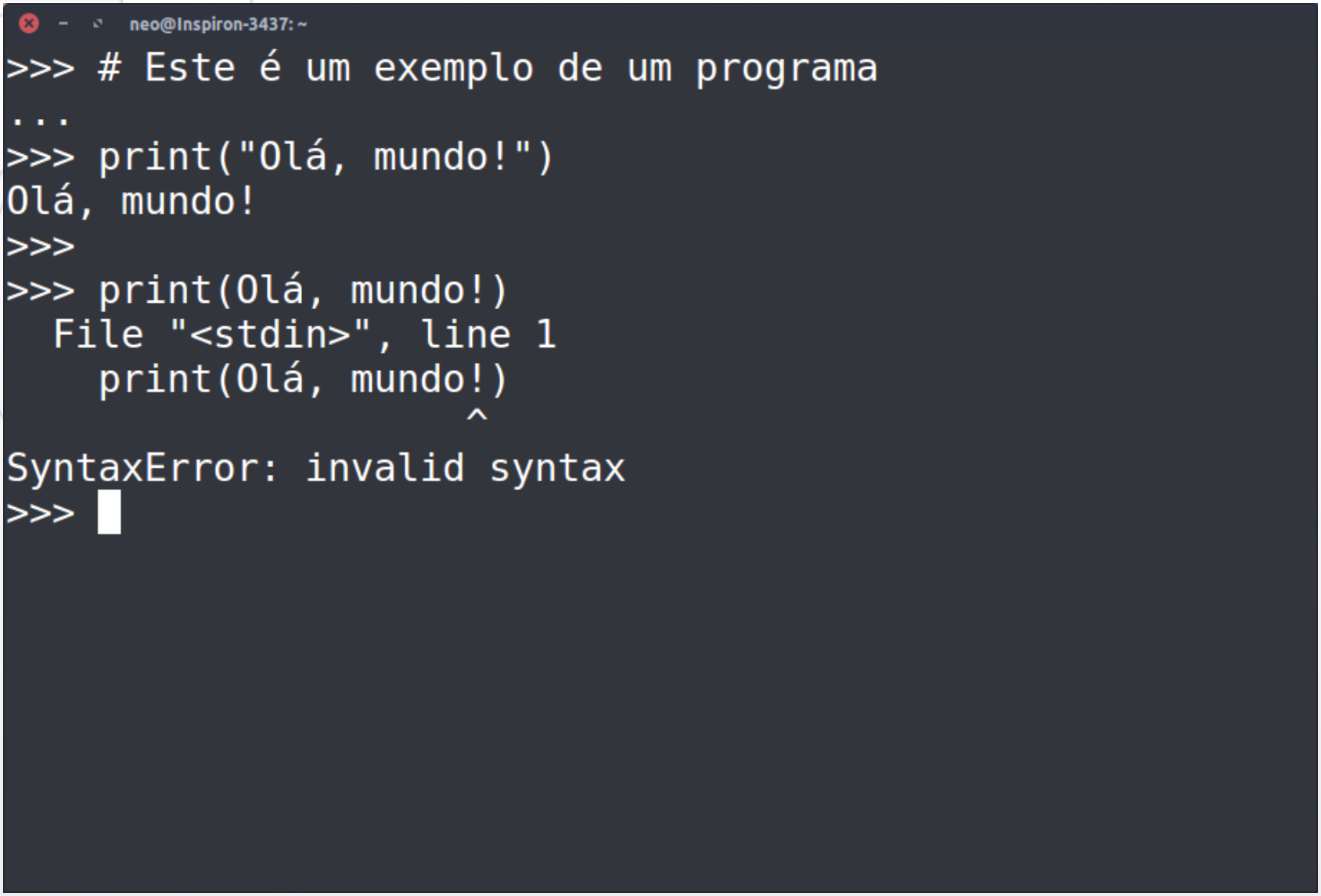
6) Dicionários;

7) Comparando coisas;

8) Criando condições e *loops*;

9) Escrevendo funções.

Strings, funções, **erros**



Acima: exemplo de erro retornado por uma função print()
cujo argumento não foi dado como uma string (portanto, sem aspas).

Strings, funções, erros, **variáveis**

```
neo@Inspiron-3437: ~
>>> # Este é um exemplo de um programa
...
>>> print("Olá, mundo!")
Olá, mundo!
>>>
>>> print(Olá, mundo!)
  File "<stdin>", line 1
    print(Olá, mundo!)
                     ^
SyntaxError: invalid syntax
>>>
>>> variavel = "Olá, mundo!"
>>>
>>> print(variavel)
Olá, mundo!
>>>
```

Acima: exemplo de atribuição de uma string (“Olá, mundo!”) a uma variável. Variáveis devem ser passadas para a função print() sem aspas, que a retornará como strings.

Strings, funções, erros, variáveis

```
neo@Inspiron-3437: ~/apps/win/games/TalonRO
>>> numero_sem_aspas = 123
>>> numero_com_aspas = "123"
>>>
>>> print(numero_sem_aspas, numero_com_aspas)
123 123
>>>
>>> numero_sem_aspas * 2
246
>>> numero_com_aspas * 2
'123123'
>>>
>>> numero_sem_aspas + numero_com_aspas
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

*Acima: atribuição de variáveis com aspas (texto) e sem aspas (número), resultando Em erro ao tentar-se somar um valor int (número **int**eiro) com uma **str**ing.*

Strings, funções, erros, variáveis

neo@Inspiron-3437: ~/apps/win/games/TalonRO

```
>>> numero_sem_aspas = 123
>>> numero_com_aspas = "123"
>>>
>>> type(numero_sem_aspas)
<class 'int'>
>>>
>>> type(numero_com_aspas)
<class 'str'>
>>>
>>> try:
...   print(numero_sem_aspas + numero_com_aspas)
... except:
...   print("Erro: impossível somar int e str!")
...   print("Convertendo tudo para int...")
...   numero_com_aspas = int(numero_com_aspas)
...   print(numero_sem_aspas + numero_com_aspas)
...
Erro: impossível somar int e str!
Convertendo tudo para int...
246
>>>
```

*Acima: função type() para descobrir o tipo de uma variável e exemplo de uma cláusula **try/except** para lidar com erros. É importante notar a **indentação**, caracteristicamente composta por ESPAÇOS ou TABS no Python.*

Conhecendo a linguagem

1) ~~Strings;~~

2) ~~Funções;~~

3) ~~Erros;~~

4) ~~Variáveis;~~

5) Listas e *tuples*;

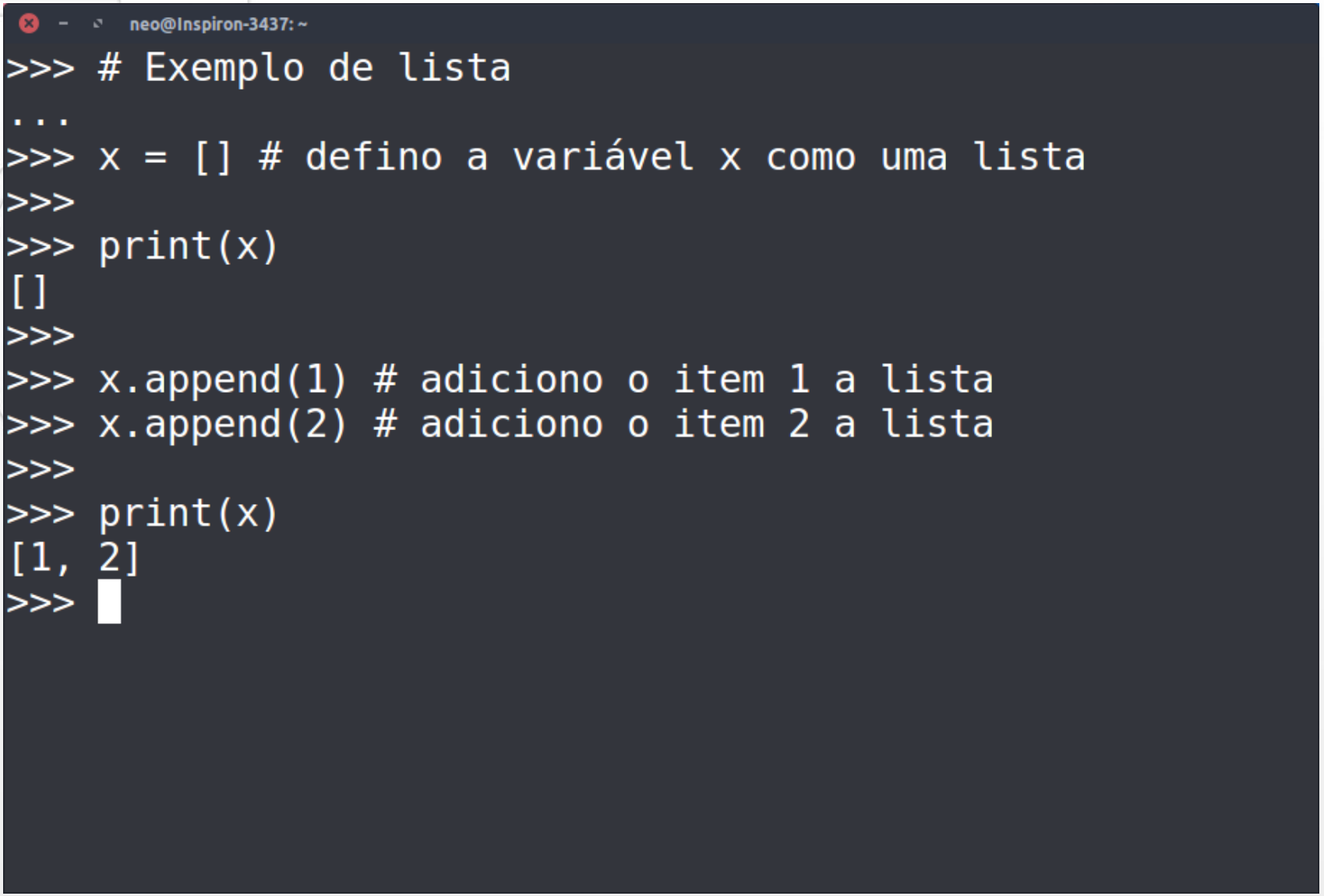
6) Dicionários;

7) Comparando coisas;

8) Criando condições e *loops*;

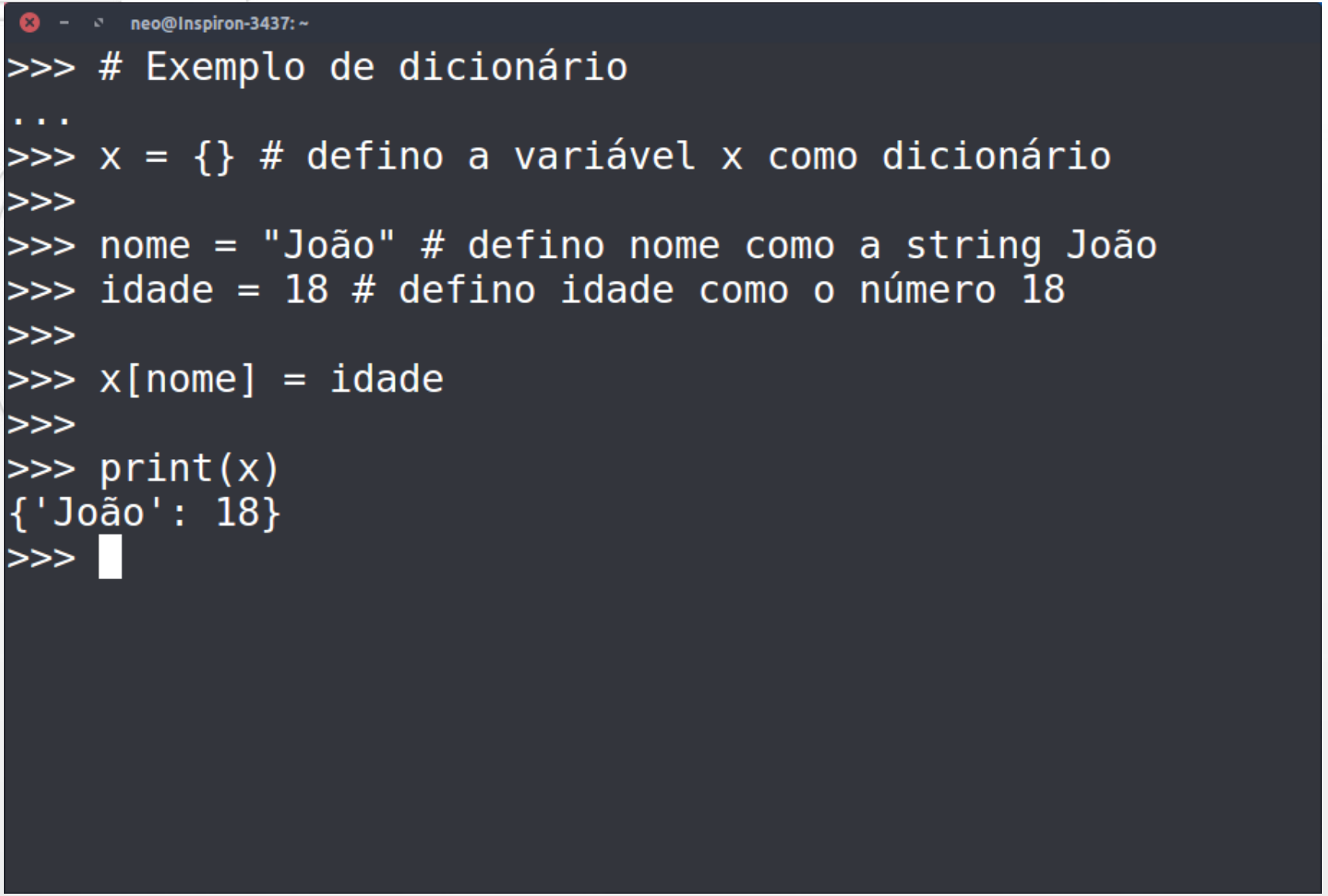
9) Escrevendo funções.

Listas: [] ou list()



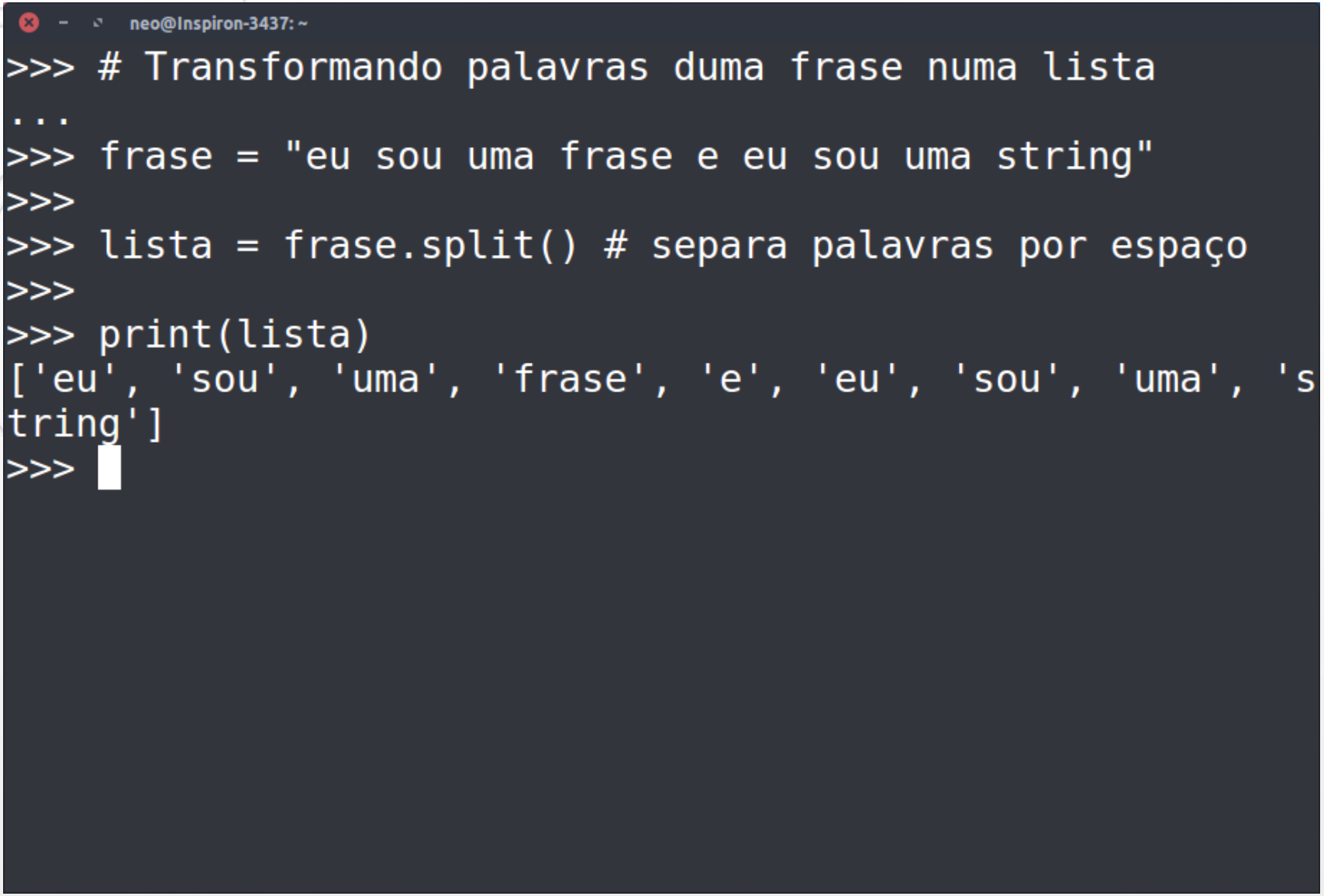
*Acima: exemplo de uma lista (definida como variável **x**) e sua função **append**.*

Dicionários: {} ou dict()



*Acima: exemplo de um dicionário (variável **x**), seguido por atribuição de um valor **idade (18)** a uma chave **nome** (João) do dicionário.*

Transformando string em lista



*Acima: exemplo de uma string (**frase**) transformada em **lista** com a função split().*

Removendo duplicatas numa lista

```
neo@Inspiron-3437: ~
>>> # Removendo duplicatas de uma lista com set()
...
>>> lista = ["Hugo", "Zé", "Luís", "Zé"]
>>>
>>> lista.count("Zé") # conta quantos itens há na lista
2
>>> x = set(lista) # remove duplicatas
>>> y = list(x) # transforma x em uma lista
>>>
>>> print(x)
{'Zé', 'Luís', 'Hugo'}
>>>
>>> print(y)
['Zé', 'Luís', 'Hugo']
>>>
>>> y.count("Zé")
1
>>>
```

*Acima: exemplo da função **count()** de listas, seguido por exemplo de como retirar duplicatas de uma lista transformando-a em uma **set**. Ao contrário de listas, **sets** não aceitam valores duplicados em si.*

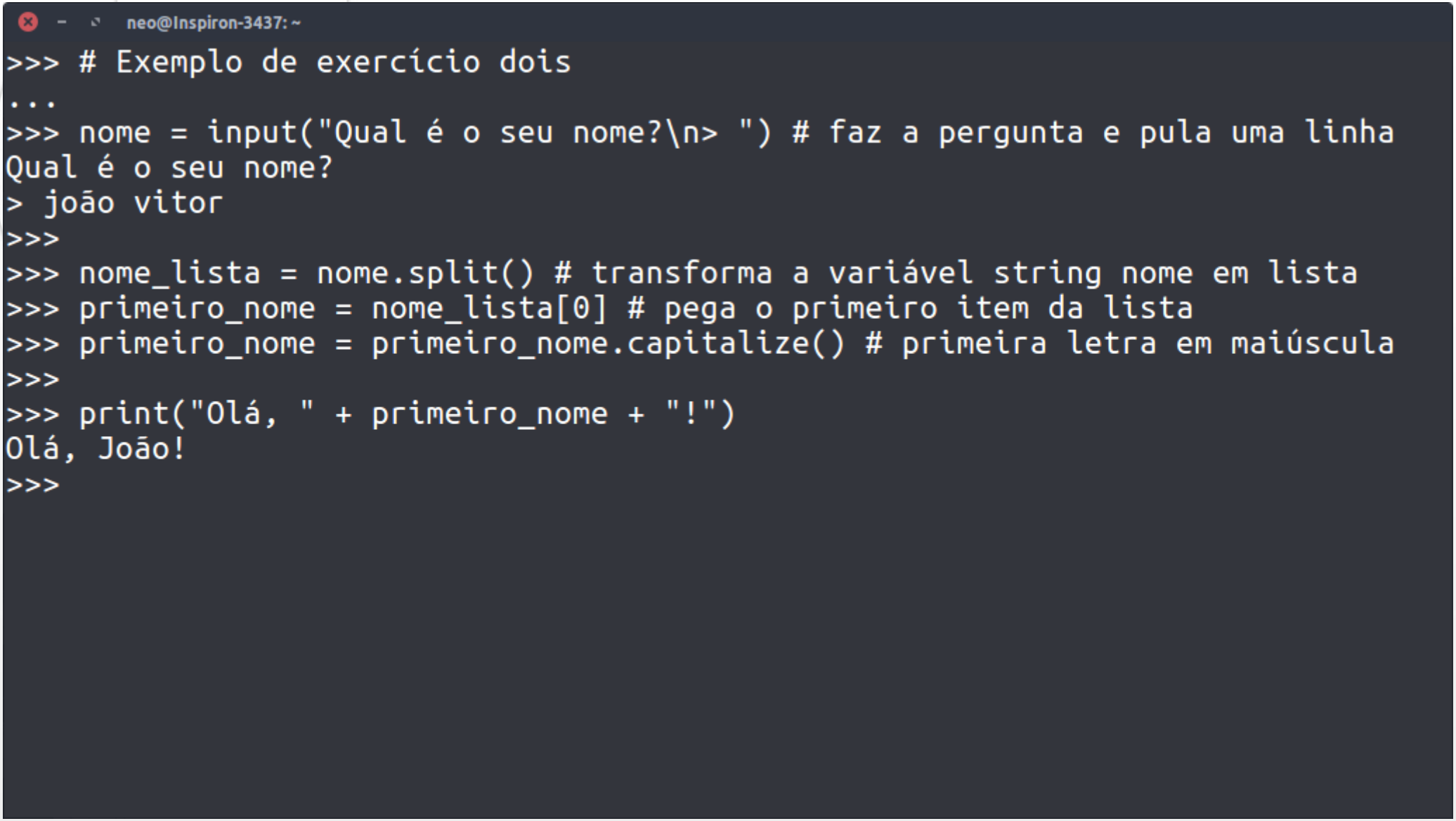
Exercício 2

Agora **refine** o programa criado no exercício um, retornando porém **apenas o primeiro nome** em caso de **nomes compostos**, com as iniciais sempre em maiúsculas.

~~**print()** e **input()**~~

capitalize() e split()

Exercício 2: solução



*Acima: possível solução para o exercício dois utilizando as funções **input()**, **split()**, **capitalize()** e **print()**.*

Conhecendo a linguagem

1) ~~Strings;~~

2) ~~Funções;~~

3) ~~Erros;~~

4) ~~Variáveis;~~

5) ~~Listas e tuples;~~

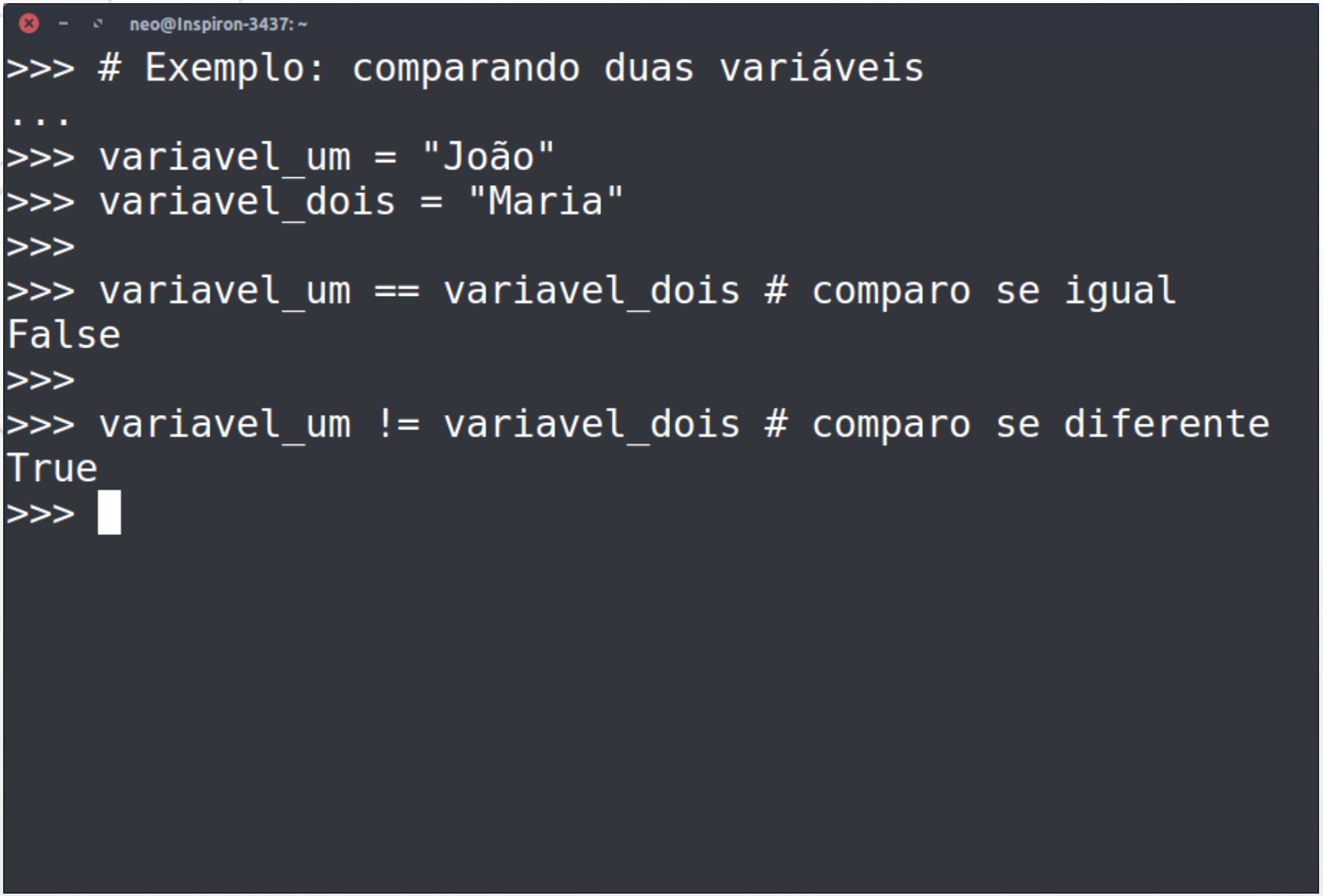
6) ~~Dicionários;~~

7) Comparando coisas;

8) Criando condições e *loops*;

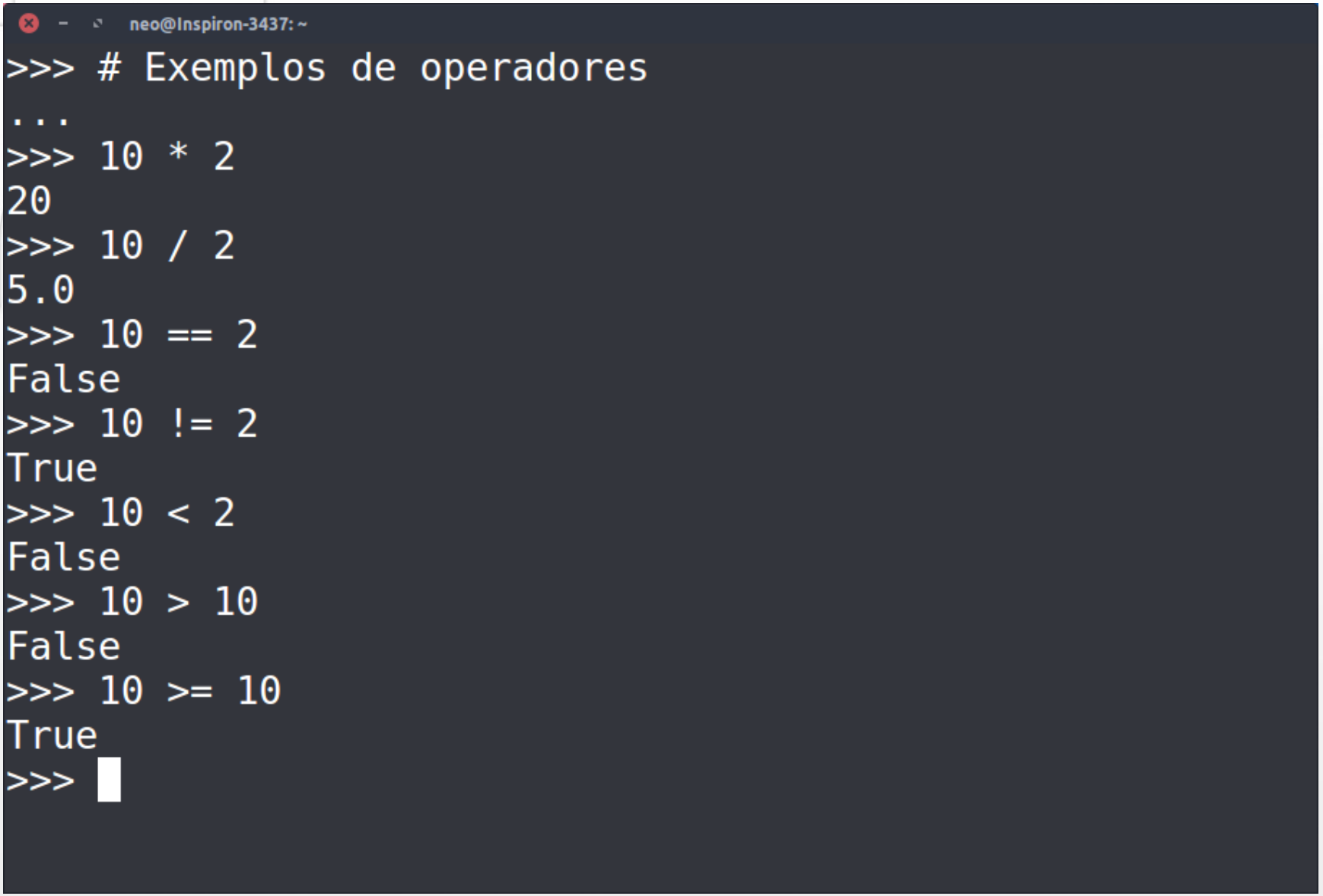
9) Escrevendo funções.

Comparando coisas



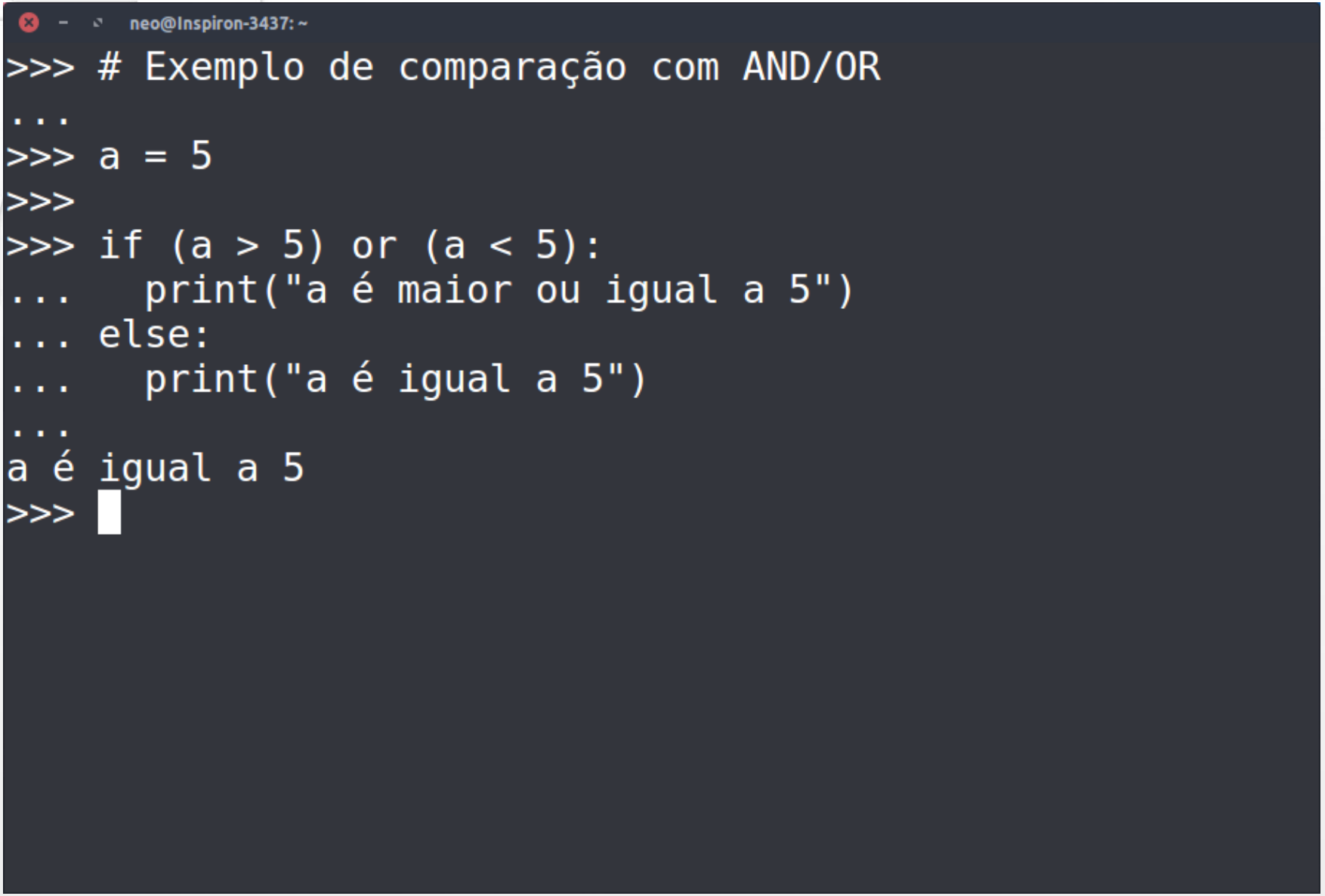
Acima: comparações realizadas com os operadores ***==*** *(igual a) e* ***!=*** *(diferente de).*

Comparando coisas



Acima: outros exemplos de comparações utilizando operadores.

Comparando coisas



*Acima: exemplo de comparações múltiplas com os operadores **AND** e **OR**.*

Conhecendo a linguagem

1) ~~Strings;~~

2) ~~Funções;~~

3) ~~Erros;~~

4) ~~Variáveis;~~

5) ~~Listas e tuples;~~

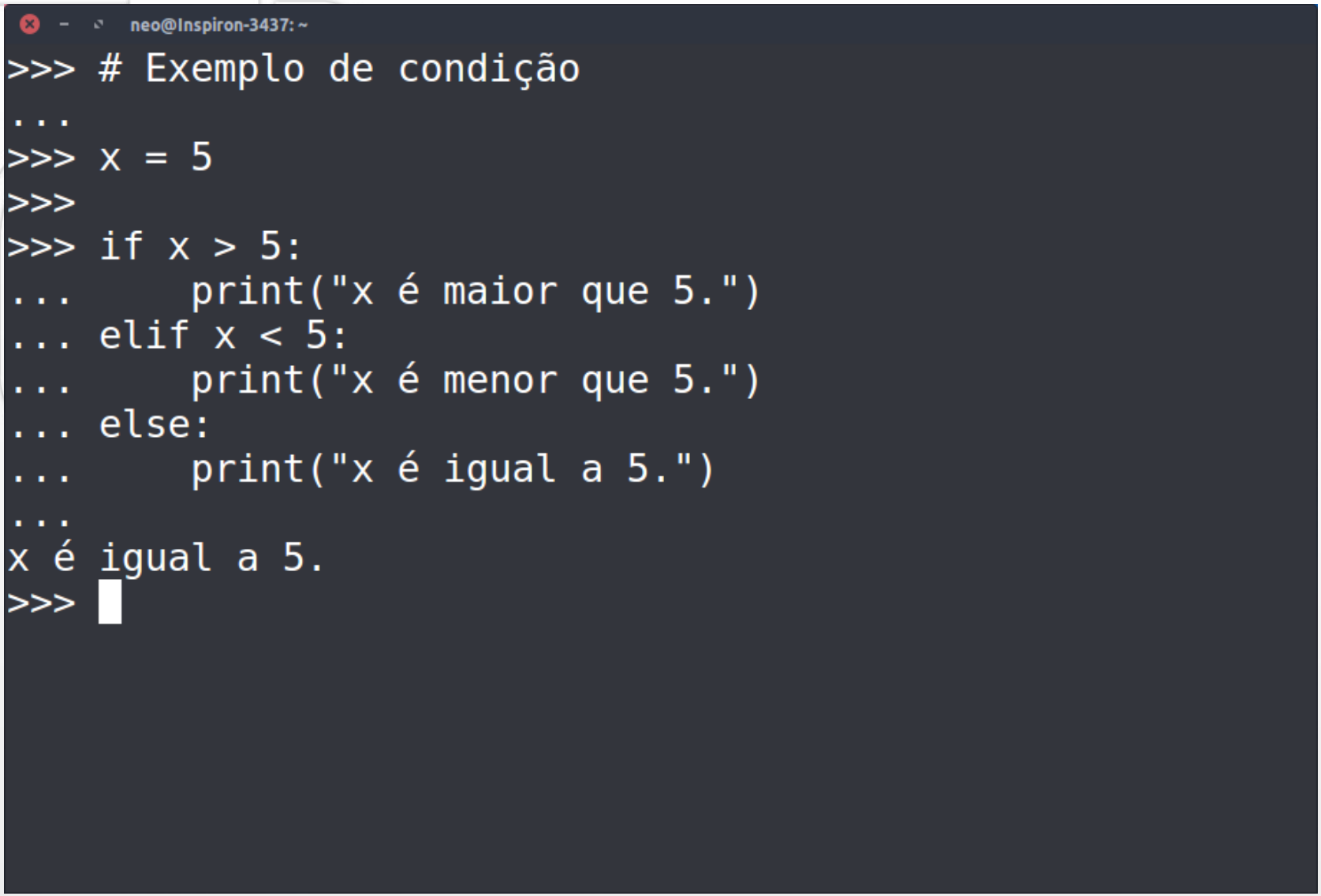
6) ~~Dicionários;~~

7) ~~Comparando coisas;~~

8) Criando condições e *loops*;

9) Escrevendo funções.

Criando **condições**



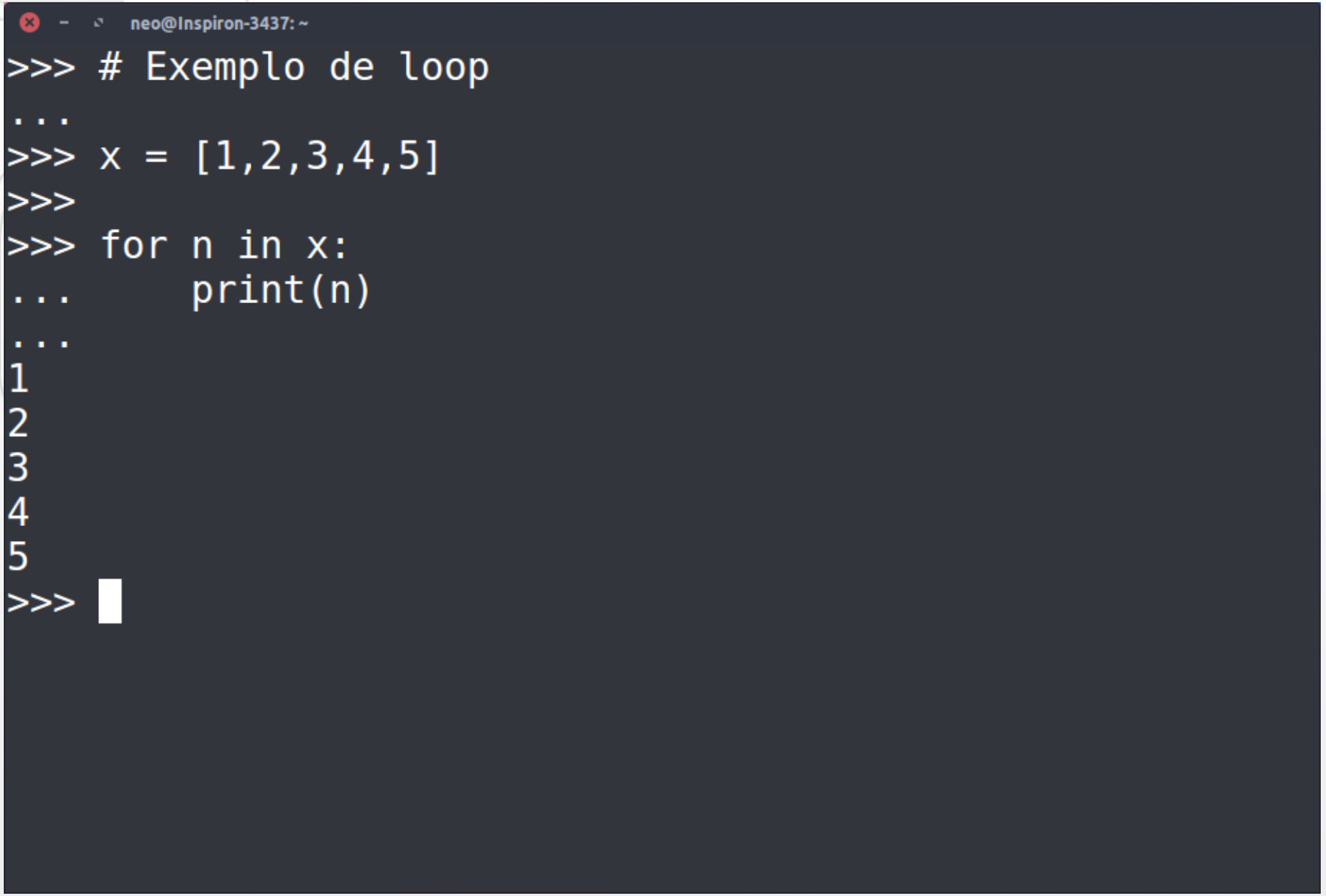
*Acima: criando condições com uma cláusula **if/elif/else**.*
*Atenção para a **indentação** com espaços após os sinais de dois pontos.*

Criando **condições**

```
neo@Inspiron-3437: ~
>>> # Exemplo de condição em uma lista
...
>>> x = [1,2,3,4,5] # lista de números
>>>
>>> if "3" in x: # str(3)
...   print("string 3 está contido em x")
... elif 3 in x: # int(3)
...   print("int 3 está contido em x")
...
int 3 está contido em x
>>>
>>> str(3) == "3"
True
>>> int(3) == 3
True
>>>
```

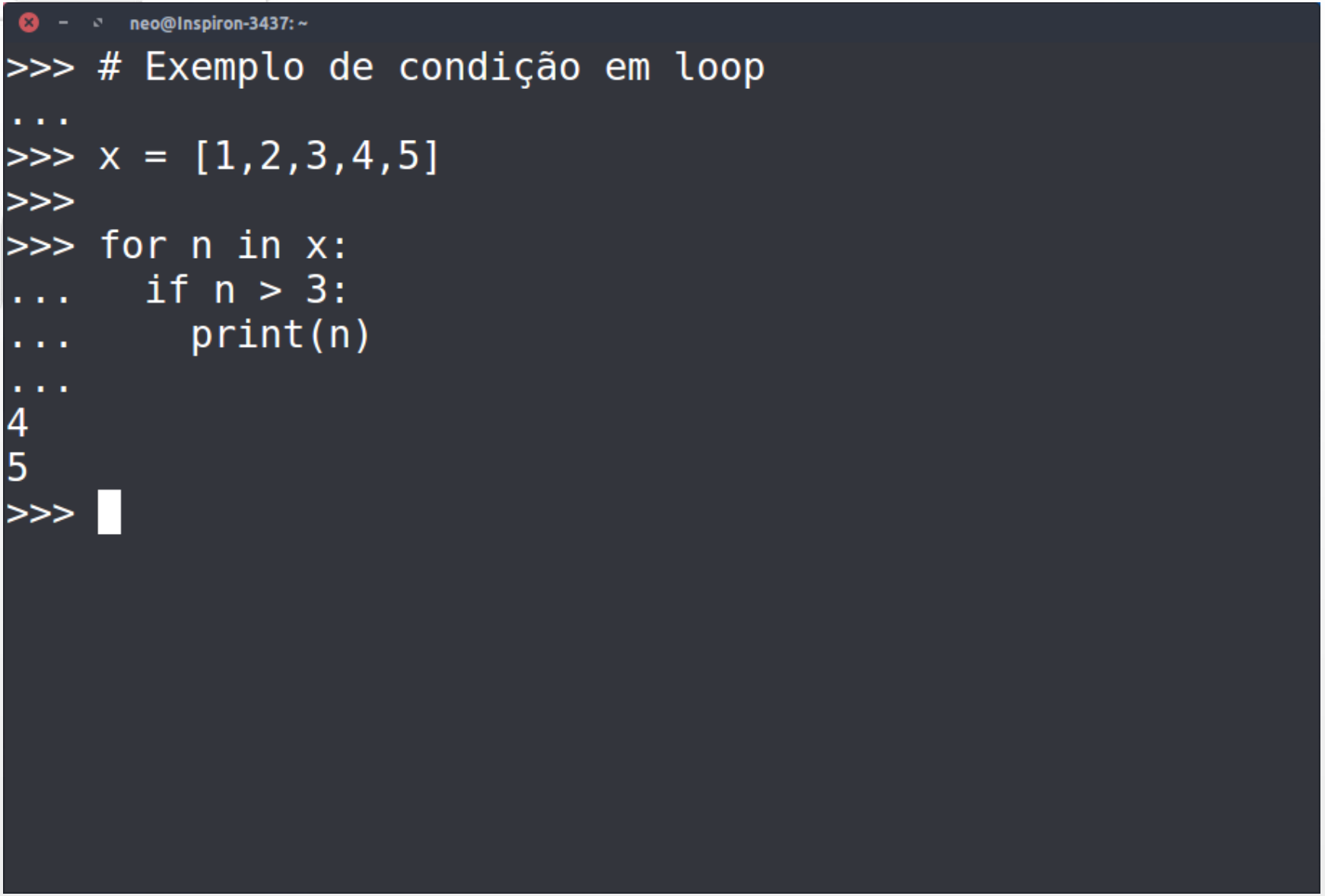
Acima: exemplo de condições realizadas em lista.

Criando ***loops***



Acima: exemplo de uma lista utilizada para várias interações.

Criando **condições e *loops***



Acima: exemplo de uma condição aplicada em uma lista.

Criando **condições e *loops***

```
neo@Inspiron-3437: ~
>>> # Outro exemplo de loop
...
>>> x = 0
>>>
>>> while True:
...     x += 1 # adiciono 1 a x
...     print(x)
...     if x == 5:
...         break
...
1
2
3
4
5
>>>
```

*Acima: exemplo de um **loop** utilizando-se **while True**. O comando **break** interrompe o loop criado, enquanto **continue** interromperia uma interação apenas.*

Conhecendo a linguagem

1) ~~Strings;~~

2) ~~Funções;~~

3) ~~Erros;~~

4) ~~Variáveis;~~

5) ~~Listas e tuples;~~

6) ~~Dicionários;~~

7) ~~Comparando coisas;~~

8) ~~Criando condições e loops;~~

9) Escrevendo funções.

Escrevendo uma função

```
neo@Inspiron-3437: ~
>>> # Escrevendo uma função
...
>>> def dobro_da_soma(x, y):
...     z = 2 * (x+y)
...     return z
...
>>> numero = dobro_da_soma(2,4)
>>>
>>> print(numero)
12
```

*Acima: exemplo de uma função que aceita dois argumentos **(x, y)** na execução. Para um deles ser opcional, deve ser definido com um valor padrão como a seguir: **(x, y=2)**.*

O que aprendemos?

1) Strings: **“João Victor”**, **“123”**, **str(123)...**
2) Funções: **print()**, **input(), list(), split(), count()**...
3) Erros: **try/except/else**.
4) Variáveis: **x, y, nome_do_aluno** …
5) Listas, tuples e sets: **[], ()** e **set()**;
6) Dicionários: **{}** ou **dict()**;
7) Comparando coisas com operadores: **==, !=, >=** ...
8) Criando condições e loops: **if/elif/else** e **while True**;
9) Escrevendo funções: **def()**

Exercício 3

Crie uma **função** que **adicione num dicionário** o número de ocorrências por palavra numa dada frase **definida pelo usuário** na execução.

Funções necessárias:

split() **lower()**

input() **replace()**

Exercício 3: solução

neo@Inspiron-3437: ~/apps/win/games/TalonRO

```
>>> dicionario = {} # define dicionario vazio
>>> pontuacao = [',', ';', '.', '!', '?', '"'] # lista
>>>
>>> for palavra in input("TEXTO > ").split():
...   palavra = palavra.lower()
...   for x in pontuacao: # remove pontuação da palavra
...     palavra = palavra.replace(x, "")
...   if palavra in dicionario: # procura se está no dicionário
...     dicionario[palavra] += 1 # adiciona um
...   else:
...     dicionario[palavra] = 1 # primeira ocorrência
...
```

*Acima: exemplo de solução para o exercício 3. A cláusula **if/else** poderia também ser substituída por uma **try/except** sem grandes prejuízos, visto que é necessário fazer existir a **palavra** no dicionário antes de adicioná-la +=1 para não resultar em erro.*