



Programando com Python

Diemeslino Souza Carvalho
diemeslino@simetric.com.br
www.simetric.com.br



Agenda



1 Por quê Python?

2 Quem utiliza Python?

3 Python Básico.

4 Aprofundando na linguagem.

5 Orientação a Objetos com Python.

6 Como seguir daqui para frente?

7 Conclusões.

Por quê Python?



- ★ É uma linguagem simples de aprender.
- ★ Permite focar no problema, sem perder tempo na sintaxe.
- ★ É interativa.
- ★ Alta produtividade.
- ★ Orientada a Objetos; Funcional; Estruturada.
- ★ Linguagem de uso geral.
- ★ ...

Quem utiliza Python?



1h video/s 4 billion views.
Python scales!

Quem utiliza Python?



Quem utiliza Python?



DISQUS

NATIONAL
GEOGRAPHIC



Quem utiliza Python?

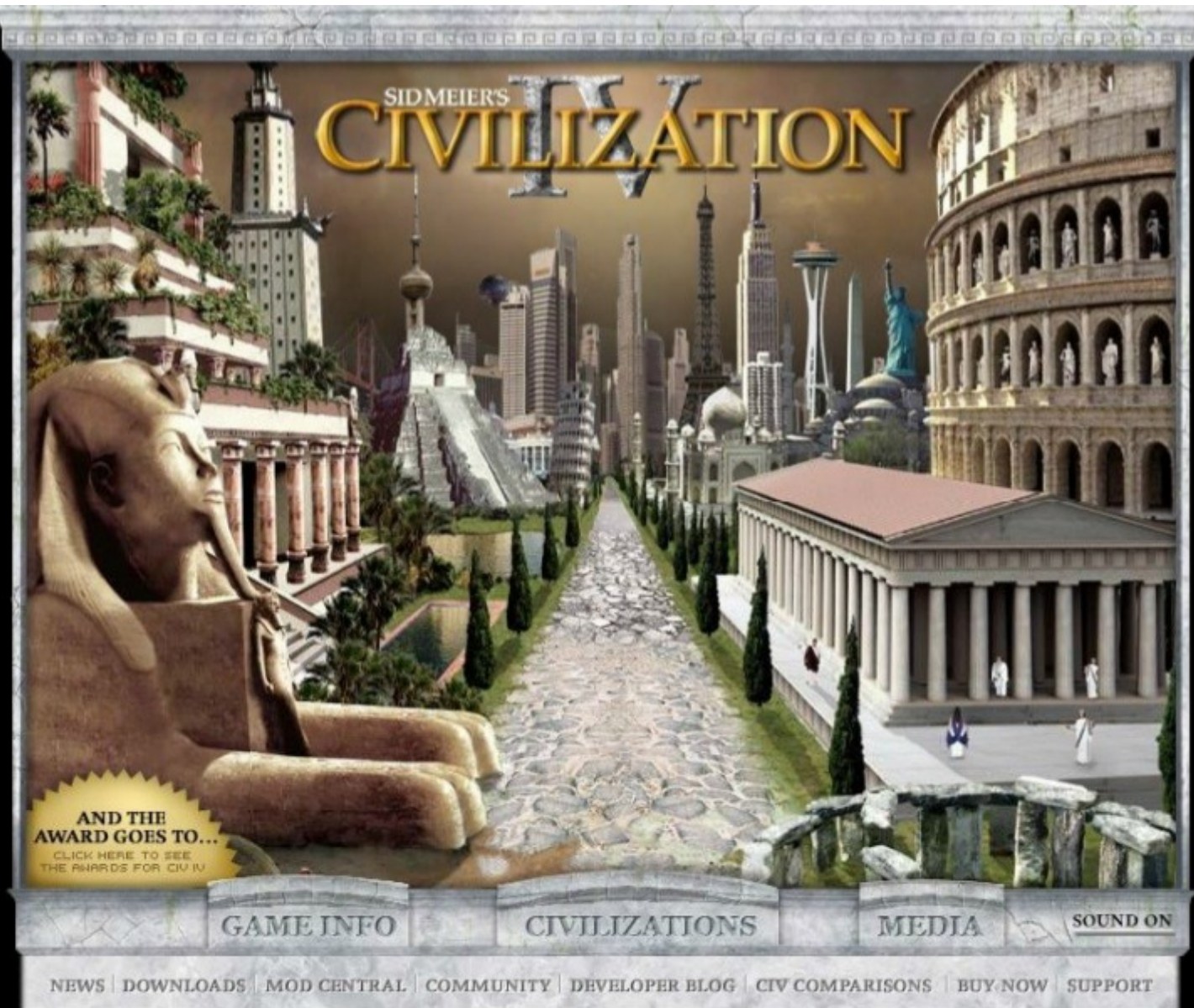


NTSC

The background of the slide is the cover art for the video game Call of Duty 4: Modern Warfare. It features a soldier in the foreground, wearing a helmet and goggles, holding a rifle. In the background, there are several helicopters flying in a hazy, war-torn environment. The title 'CALL OF DUTY 4' is prominently displayed in large, metallic, 3D letters, with 'MODERN WARFARE' written in smaller, green, block letters below it.

CALL OF DUTY 4
MODERN WARFARE

Quem utiliza Python?



Quem utiliza Python?



Home > Products > NUKE

(empty) 

NUKE | Advancing the art of digital compositing



Rio © 2011 Fox. All rights reserved. Images courtesy of Blue Sky Studios.

"NUKE allows us to leverage our 3D pipeline while remaining in the compositing environment which puts a great deal of power into the hands of the compositors. We work on very complex shots and the remarkable speed at which Nuke operates means our artists can focus on the art of visual effects and not have their creativity impeded



NUKE

6.3v5

[» Free 15 day trial](#)

[» Buy](#)

[» Rent](#)

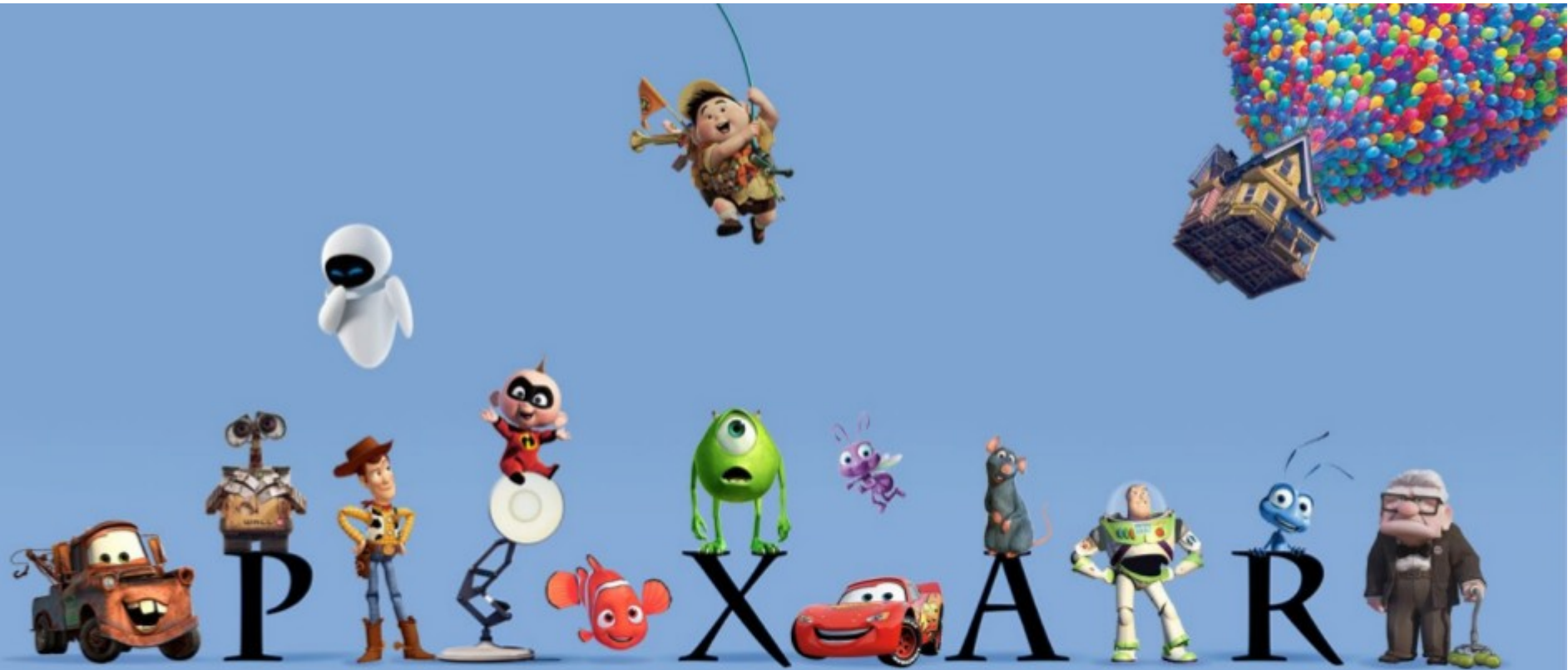
筒体
繁體



Quem utiliza Python?



Quem utiliza Python?



Quem utiliza Python?



Massachusetts
Institute of
Technology



coursera

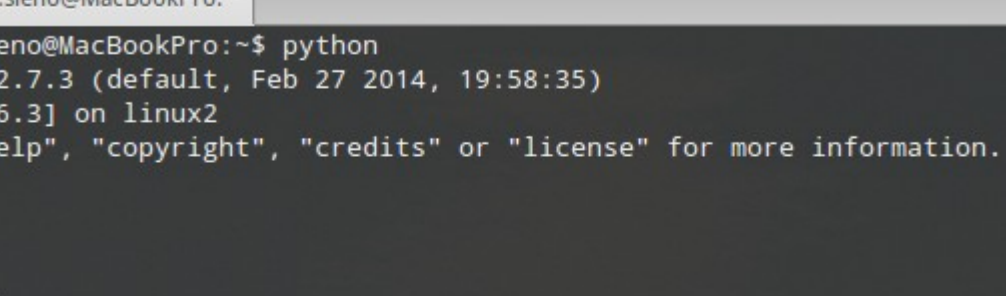


Python Básico



```
print("Hello, world!")
```

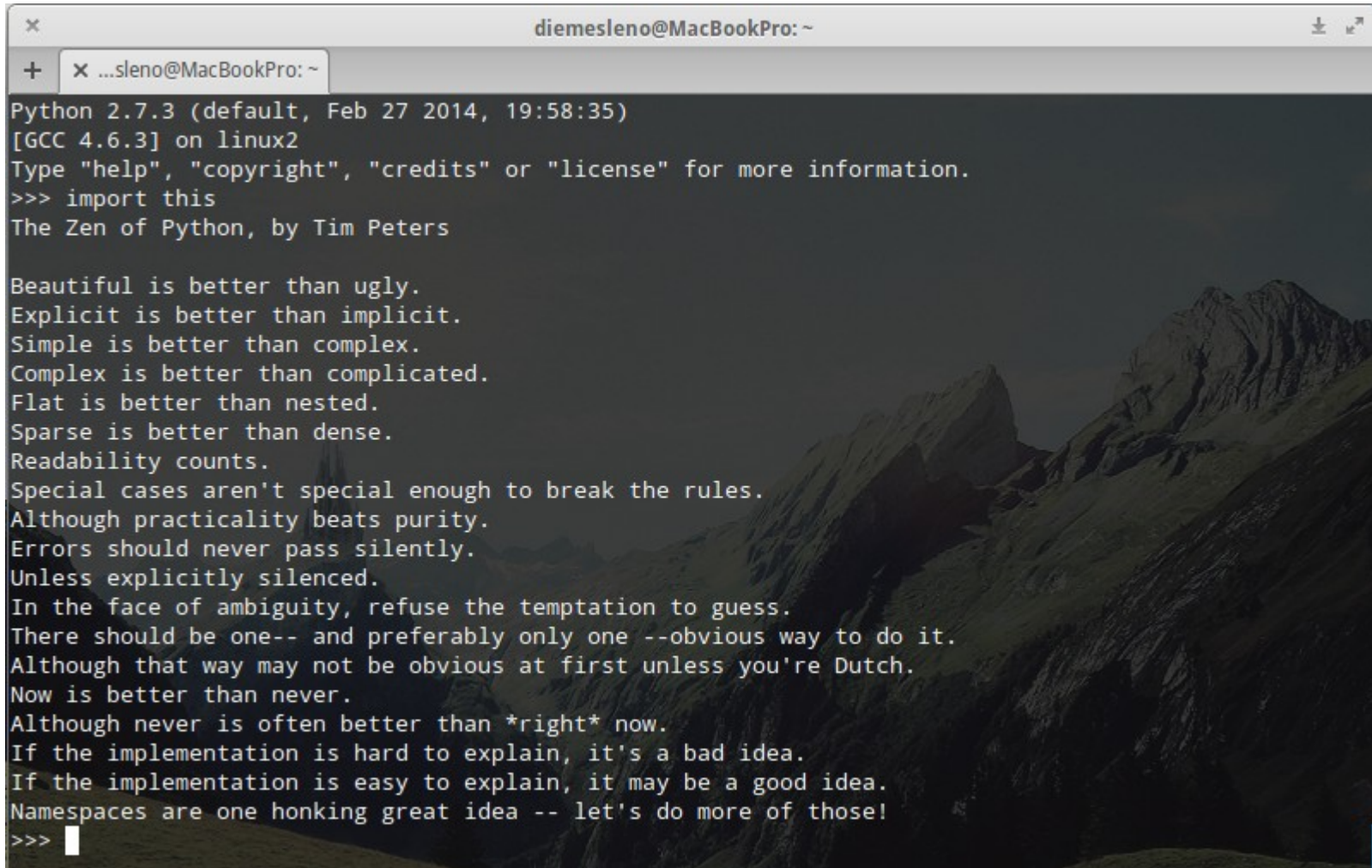
- Abra o terminal e digite: `python` (aperte enter)



```
diemesleno@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python Básico

- Digite: import this (enter)

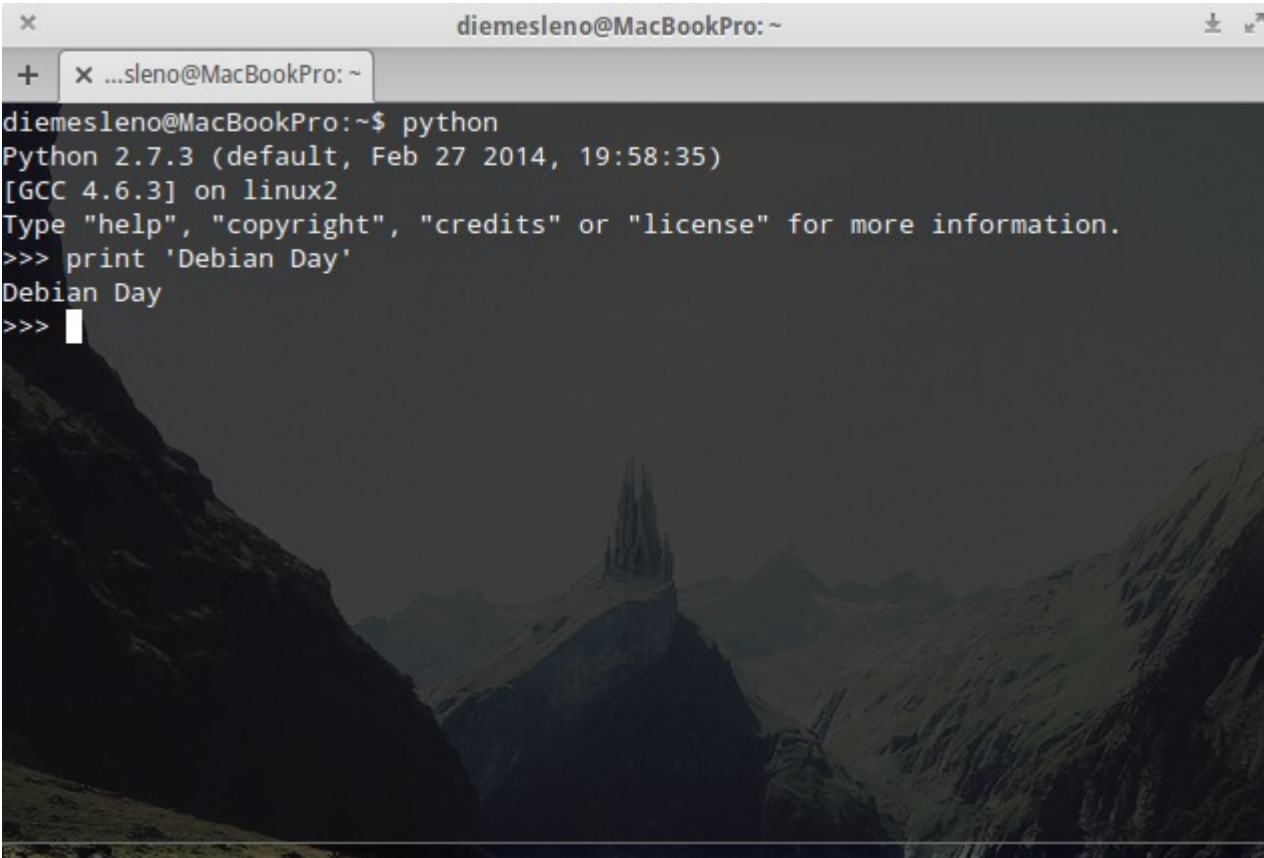
A screenshot of a terminal window on a Mac. The window title is 'diemeslino@MacBookPro: ~'. The terminal shows the Python 2.7.3 prompt and the output of the 'import this' command, which displays 'The Zen of Python' by Tim Peters. The background of the terminal window features a dark, mountainous landscape.

```
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Python Básico

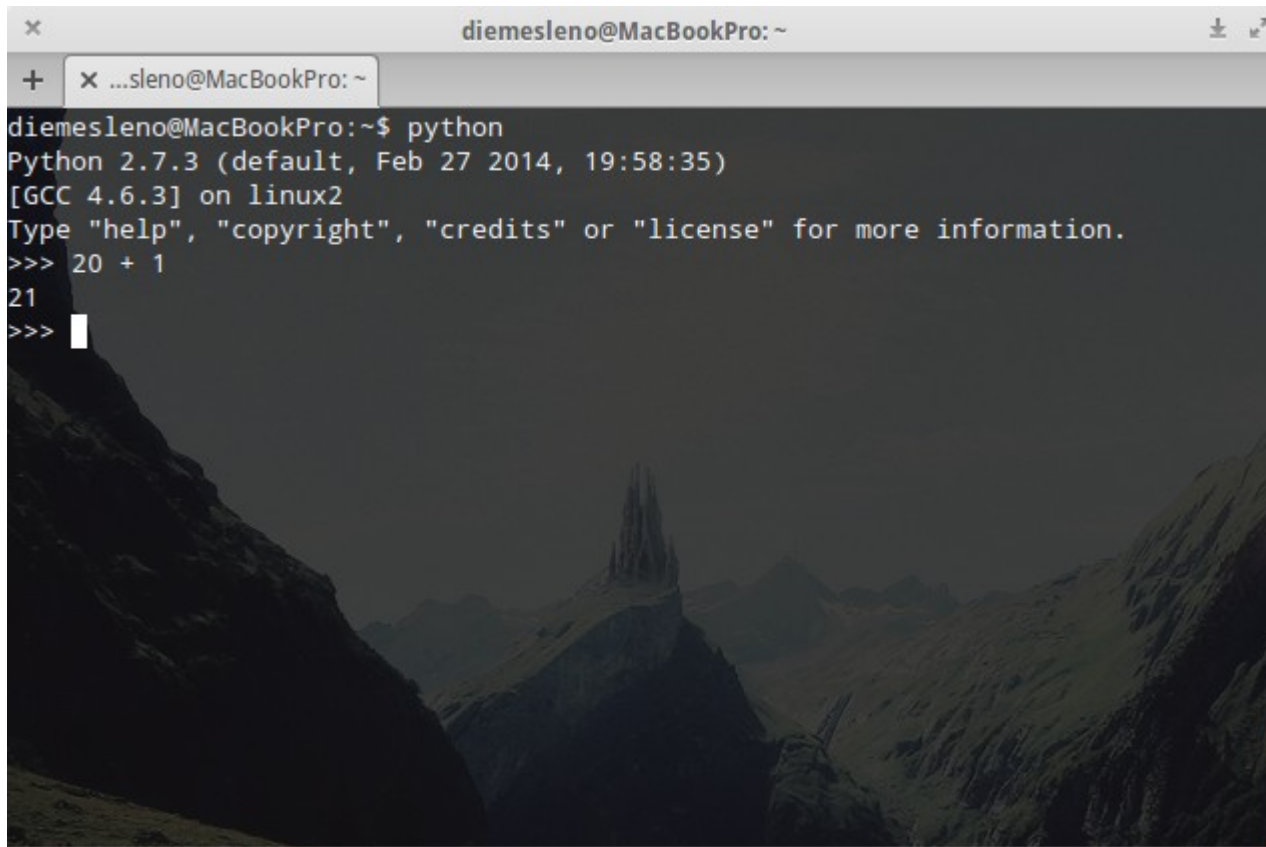
- Digite: `print 'Debian Day'` (enter)

A screenshot of a terminal window on a Mac. The window title is 'diemeslano@MacBookPro: ~'. The terminal shows the command 'python' being executed, which starts the Python 2.7.3 interpreter. The prompt is 'Python 2.7.3 (default, Feb 27 2014, 19:58:35) [GCC 4.6.3] on linux2'. The user enters 'Type "help", "copyright", "credits" or "license" for more information.' followed by the command '>>> print \'Debian Day\''. The output is 'Debian Day'. The prompt '>>>' is shown again with a cursor. The background of the terminal window is a dark, mountainous landscape.

```
diemeslano@MacBookPro: ~  
+ x ...slano@MacBookPro: ~  
diemeslano@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print 'Debian Day'  
Debian Day  
>>> 
```


Python Básico

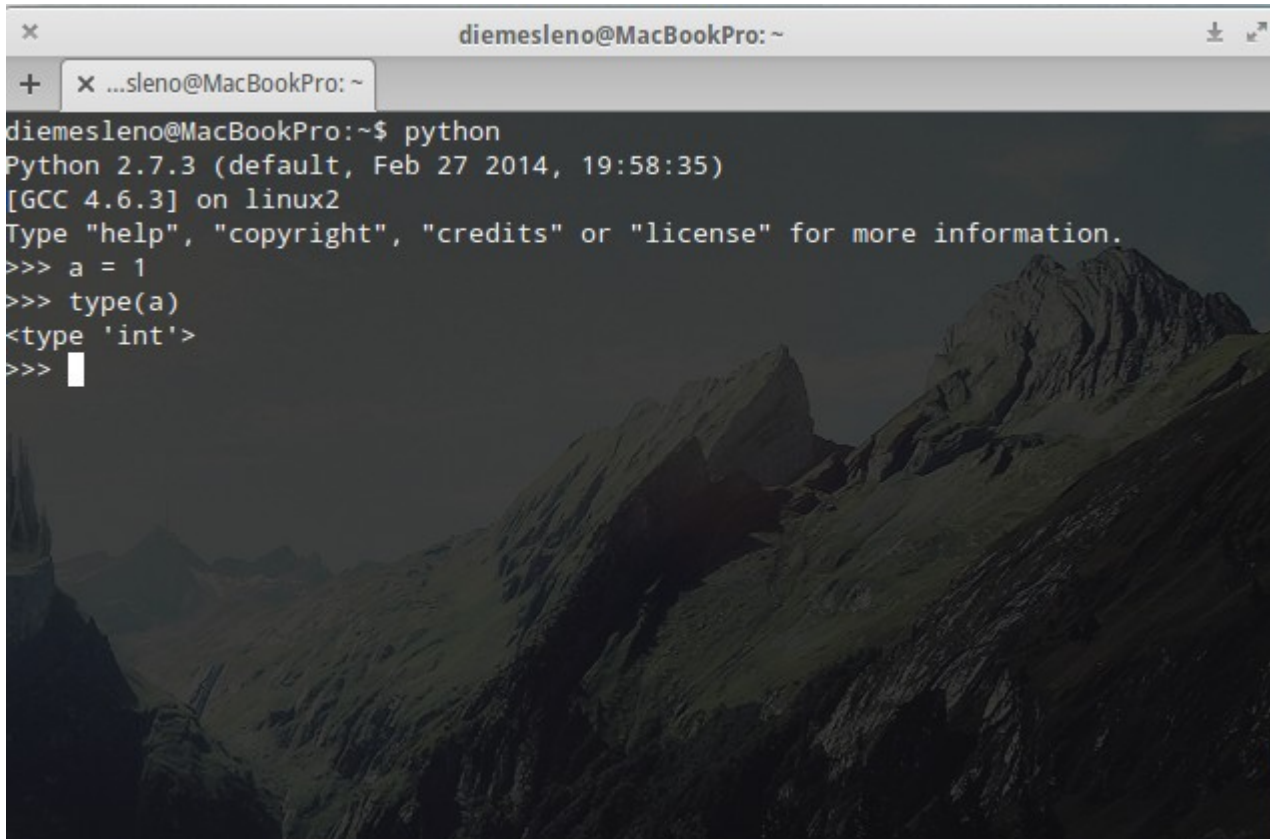
- Digite: $20 + 1$ (enter)

A screenshot of a terminal window titled "diemeslino@MacBookPro: ~". The window shows a Python 2.7.3 shell session. The user has entered "python" at the prompt, which has started the interpreter. The interpreter displays version information and a prompt "Type 'help', 'copyright', 'credits' or 'license' for more information.". The user then enters ">>> 20 + 1", and the interpreter outputs "21". The prompt ">>>" is followed by a cursor. The background of the terminal window is a dark, mountainous landscape.

```
diemeslino@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 20 + 1
21
>>> 
```

Python Básico: Variáveis e Tipos

★ Python utiliza tipagem dinâmica.

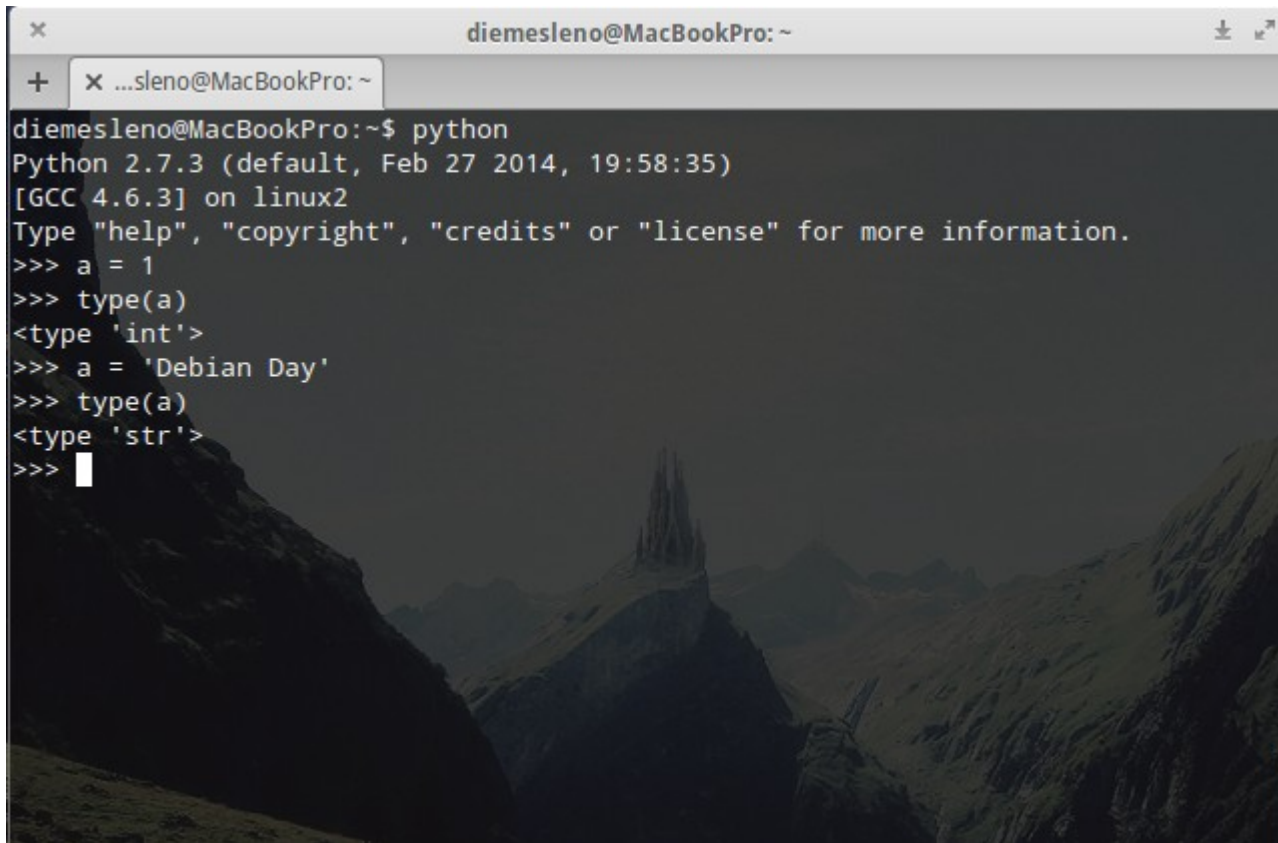


```
diemeslino@MacBookPro: ~  
+ x ...sleno@MacBookPro: ~  
diemeslino@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> a = 1  
>>> type(a)  
<type 'int'>  
>>> 
```

A variável 'a' é do tipo int

Python Básico: Variáveis e Tipos

★ Python utiliza tipagem dinâmica.

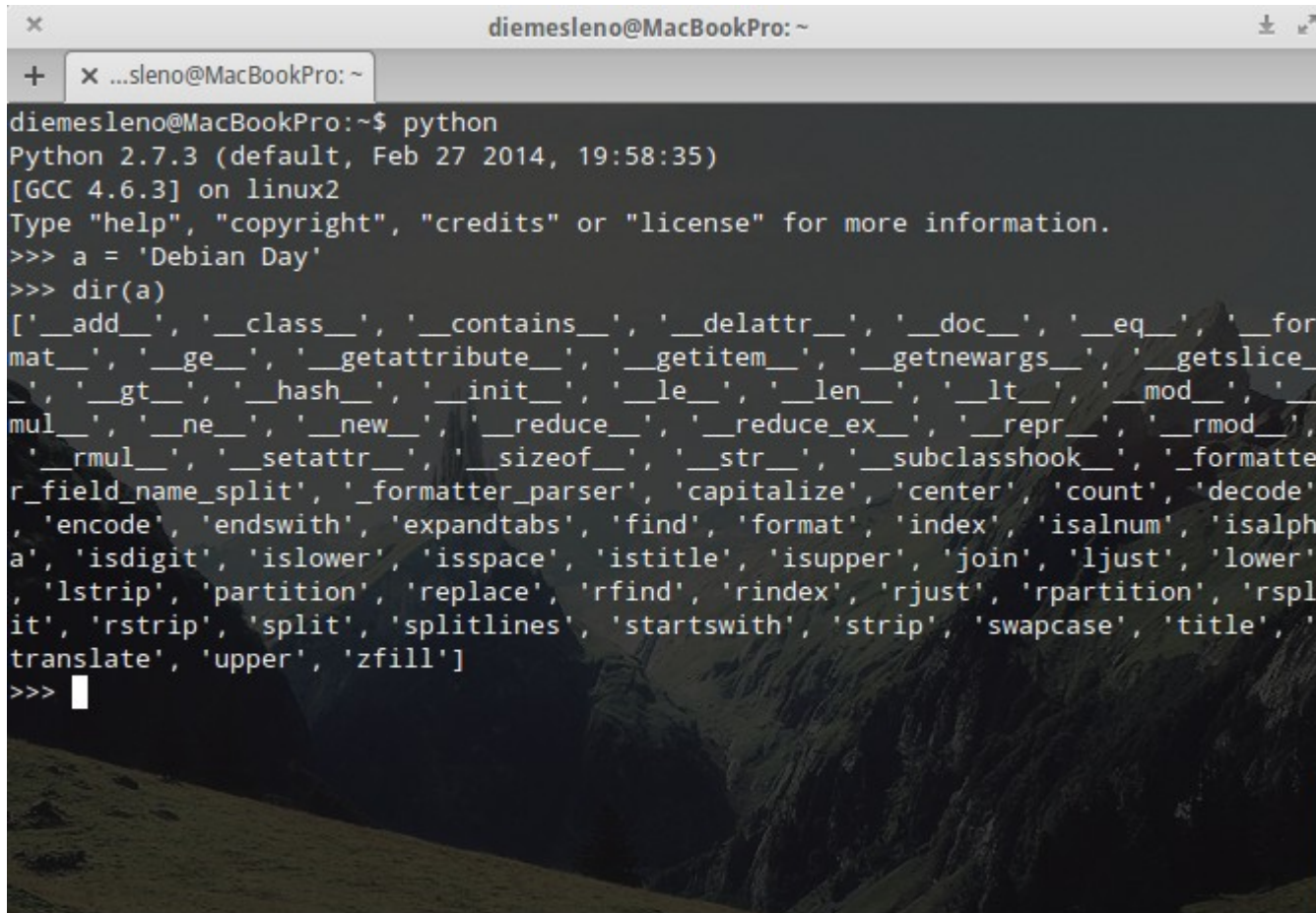
A screenshot of a terminal window titled 'diemeslento@MacBookPro: ~'. The terminal shows the execution of the Python interpreter. The user enters 'python', and the prompt changes to 'Python 2.7.3 (default, Feb 27 2014, 19:58:35)'. The user then enters a series of commands: 'a = 1', 'type(a)', 'a = \'Debian Day\'', 'type(a)', and finally a blank line. The output shows the type of 'a' as '<type \'int\'>' and then '<type \'str\'>'. The terminal background features a dark, mountainous landscape with a prominent, jagged peak in the center.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1
>>> type(a)
<type 'int'>
>>> a = 'Debian Day'
>>> type(a)
<type 'str'>
>>>
```

A variável 'a' agora é do tipo string

Python Básico

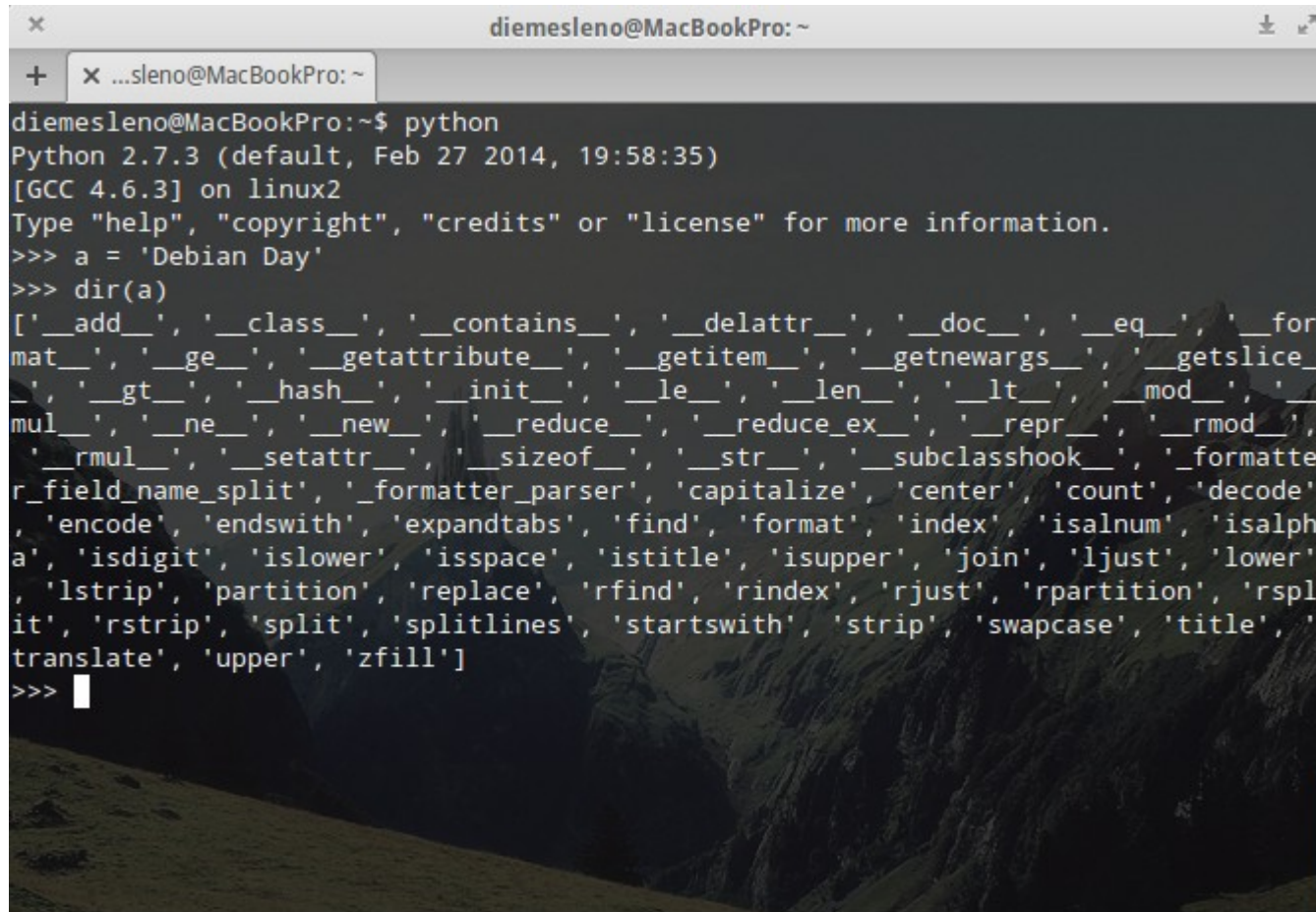
★ OBS: O método “dir” é seu amigo. Lembre-se dele.

A screenshot of a terminal window titled 'diemeslento@MacBookPro: ~'. The terminal shows a Python 2.7.3 prompt where the user has entered 'python'. The prompt then shows 'Python 2.7.3 (default, Feb 27 2014, 19:58:35)' and '[GCC 4.6.3] on linux2'. The user enters 'Type "help", "copyright", "credits" or "license" for more information.' followed by '>>> a = \'Debian Day\''. Then the user enters '>>> dir(a)' and the terminal displays a long list of attributes and methods for the string object 'Debian Day', including '__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill'. The prompt '>>>' is followed by a cursor.

O método 'dir' mostra os atributos e métodos suportados pelo objeto.

Python Básico

★ Mas como utilizar estes atributos / métodos?

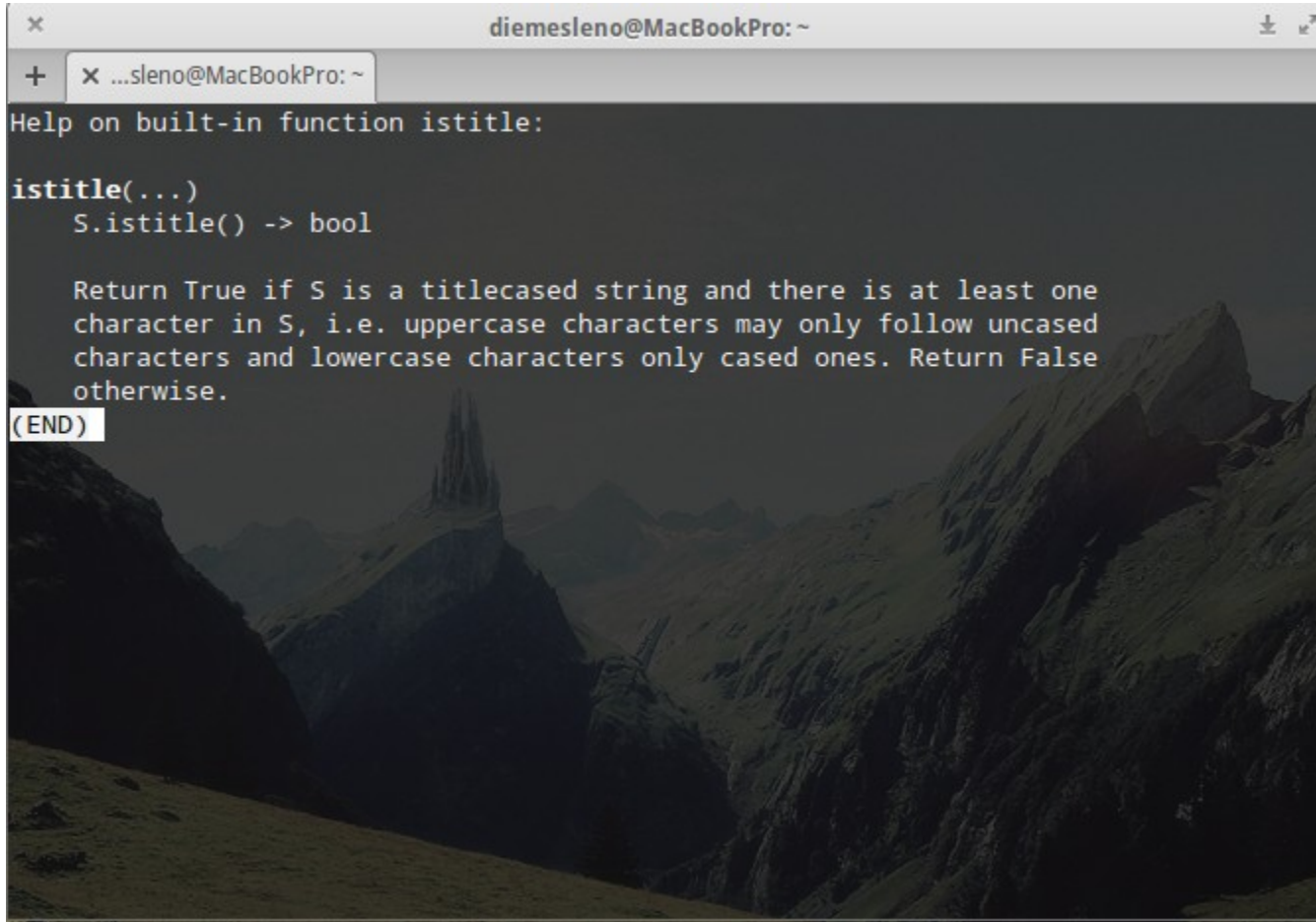


```
diemeslento@MacBookPro: ~  
+ x ...sleno@MacBookPro: ~  
diemeslento@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> a = 'Debian Day'  
>>> dir(a)  
['_add_', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__for  
mat__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice_  
_', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__  
mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__',  
 '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatte  
r_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decode'  
, 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalph  
a', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower'  
, 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rspl  
it', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', '  
translate', 'upper', 'zfill']  
>>> 
```

O método 'help(objeto.metodo)' como utilizar o atributo / método.

Python Básico

★ Digite: `help(a.istitle)` (enter)

A screenshot of a terminal window titled 'diemeslano@MacBookPro: ~'. The window shows the output of the command 'help(a.istitle)'. The text displayed is: 'Help on built-in function istitle:', followed by 'istitle(...)' and 'S.istitle() -> bool'. Below this, a detailed description states: 'Return True if S is a titlecased string and there is at least one character in S, i.e. uppercase characters may only follow uncased characters and lowercase characters only cased ones. Return False otherwise.' The window ends with '(END)'. The background of the terminal window features a dark, mountainous landscape.

```
x
diemeslano@MacBookPro: ~
+ x ...slano@MacBookPro: ~
Help on built-in function istitle:

istitle(...)
  S.istitle() -> bool

  Return True if S is a titlecased string and there is at least one
  character in S, i.e. uppercase characters may only follow uncased
  characters and lowercase characters only cased ones. Return False
  otherwise.
(END)
```

O método '`help(objeto.metodo)`' como utilizar o atributo / método.

Python Básico



★ Gerar um baralho

- No console do Python digite:

```
naipes = 'copas ouros espadas paus'.split()
```

```
cartas = 'A 2 3 4 5 6 7 8 9 10 J Q K'.split()
```

```
baralho = [(c, n) for n in naipes for c in cartas]
```

```
baralho
```

```
len(baralho)
```

Python Básico

★ Gerar um baralho

```
diemesleno@MacBookPro: ~/Desktop/Debian Day/curso-python/programas
+ x ...rso-python/programas
diemesleno@MacBookPro:~/Desktop/Debian Day/curso-python/programas$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> naipes = 'copas ouros espadas paus'.split()
>>> cartas = 'A 2 3 4 5 6 7 8 9 10 J Q K'.split()
>>> baralho = [(c, n) for n in naipes for c in cartas]
>>> baralho
[('A', 'copas'), ('2', 'copas'), ('3', 'copas'), ('4', 'copas'), ('5', 'copas'), ('6', 'copas'), ('7', 'copas'), ('8', 'copas'), ('9', 'copas'), ('10', 'copas'), ('J', 'copas'), ('Q', 'copas'), ('K', 'copas'), ('A', 'ouros'), ('2', 'ouros'), ('3', 'ouros'), ('4', 'ouros'), ('5', 'ouros'), ('6', 'ouros'), ('7', 'ouros'), ('8', 'ouros'), ('9', 'ouros'), ('10', 'ouros'), ('J', 'ouros'), ('Q', 'ouros'), ('K', 'ouros'), ('A', 'espadas'), ('2', 'espadas'), ('3', 'espadas'), ('4', 'espadas'), ('5', 'espadas'), ('6', 'espadas'), ('7', 'espadas'), ('8', 'espadas'), ('9', 'espadas'), ('10', 'espadas'), ('J', 'espadas'), ('Q', 'espadas'), ('K', 'espadas'), ('A', 'paus'), ('2', 'paus'), ('3', 'paus'), ('4', 'paus'), ('5', 'paus'), ('6', 'paus'), ('7', 'paus'), ('8', 'paus'), ('9', 'paus'), ('10', 'paus'), ('J', 'paus'), ('Q', 'paus'), ('K', 'paus')]
>>> len(baralho)
52
>>> 
```


e Tipos

- strings, int, long, boolean, float, listas, tuplas e dicionários.

a = 'Debian Day' ou “Debian Day” (string)

```
c = 1 (int)
```

[illegible]

e = True ou False (boolean)

f = 3.4 (float)

`g = [1,2,3,4,5]` (list)

h = (1,2,3, 'Debian Day', 'Python') (tuple)

```
i = {'evento': 'Debian Day', 'curso': 'Python'} (dict)
```

Exercício 1 - Tempo 15 minutos



- ★ Verifique os métodos suportados pelos tipos de dados apresentados no slide anterior e tente utilizar alguns deles.
- ★ Utilize `dir()` para saber quais atributos / métodos são suportados.
- ★ Utilize `help(variavel.metodo)` para saber como utilizar.

Python Básico: Variáveis e Tipos

★ Strings

- Objeto iterável.
- Imutável
- Podemos utilizar aspas simples (') e dupas (")
- Acessível através de índice
- Podemos fazer Slice e Substring através de índice
- Membership
- ...

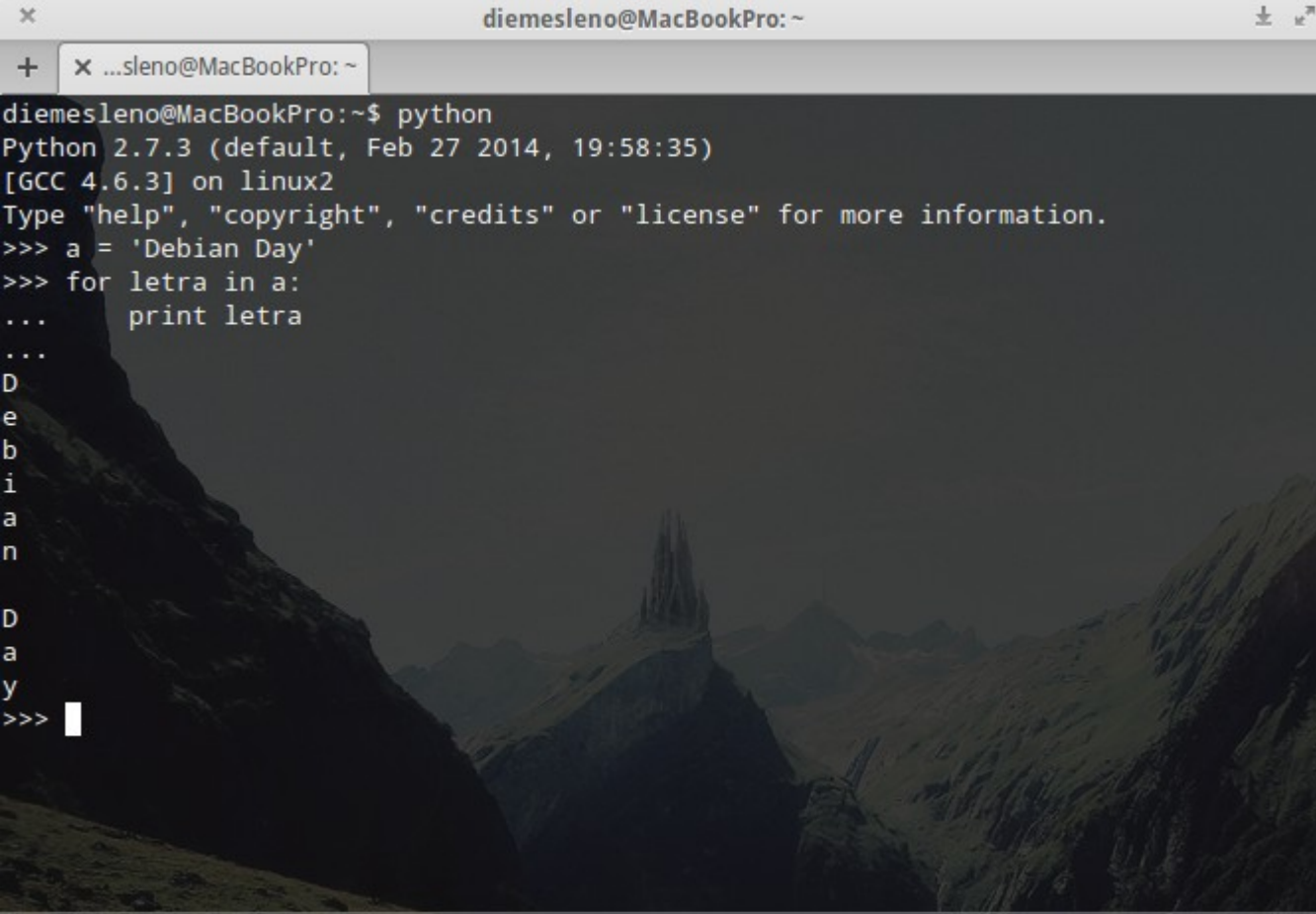
Python Básico: Variáveis e Tipos

★ Strings - Exemplo iteração

```
a = 'Debian Day'
```

```
for letra in a:
```

```
    print letra
```

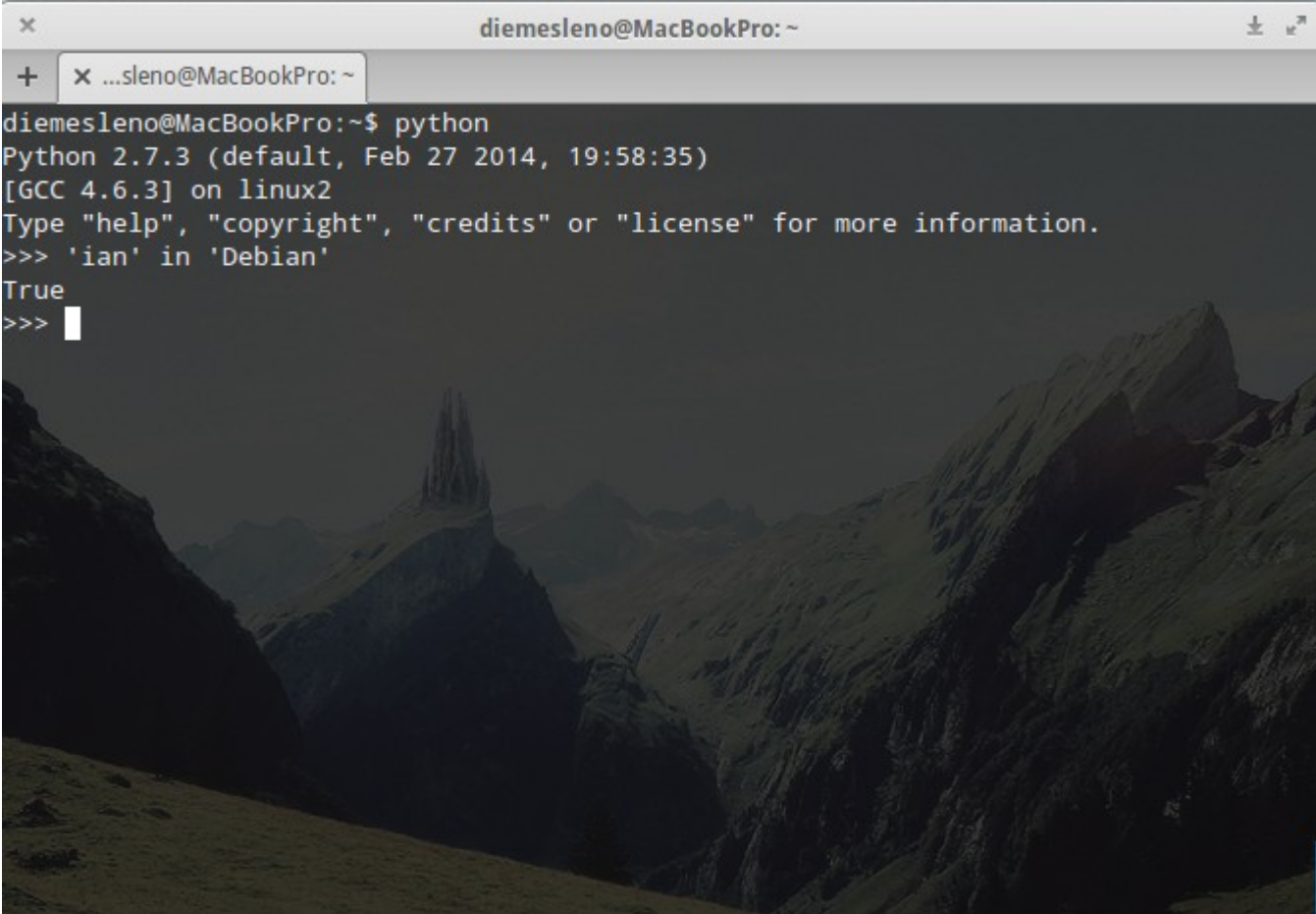
A screenshot of a terminal window titled 'diemeslino@MacBookPro: ~'. The terminal shows a Python 2.7.3 shell session. The user enters 'python', followed by 'a = \'Debian Day\'', and then a for loop 'for letra in a: print letra'. The output shows each letter of 'Debian Day' on a new line. The terminal background is a dark, mountainous landscape.

```
diemeslino@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'Debian Day'
>>> for letra in a:
...     print letra
...
D
e
b
i
a
n
D
a
y
>>>
```

Python Básico: Variáveis e Tipos

★ Strings - Exemplo membership

ian in 'debian'

A screenshot of a terminal window on a Mac. The window title is 'diemeslento@MacBookPro: ~'. The terminal shows the execution of 'python', which starts the Python 2.7.3 interpreter. The user enters the expression 'ian' in 'Debian', and the interpreter returns 'True'. The terminal background has a dark, mountainous landscape wallpaper.

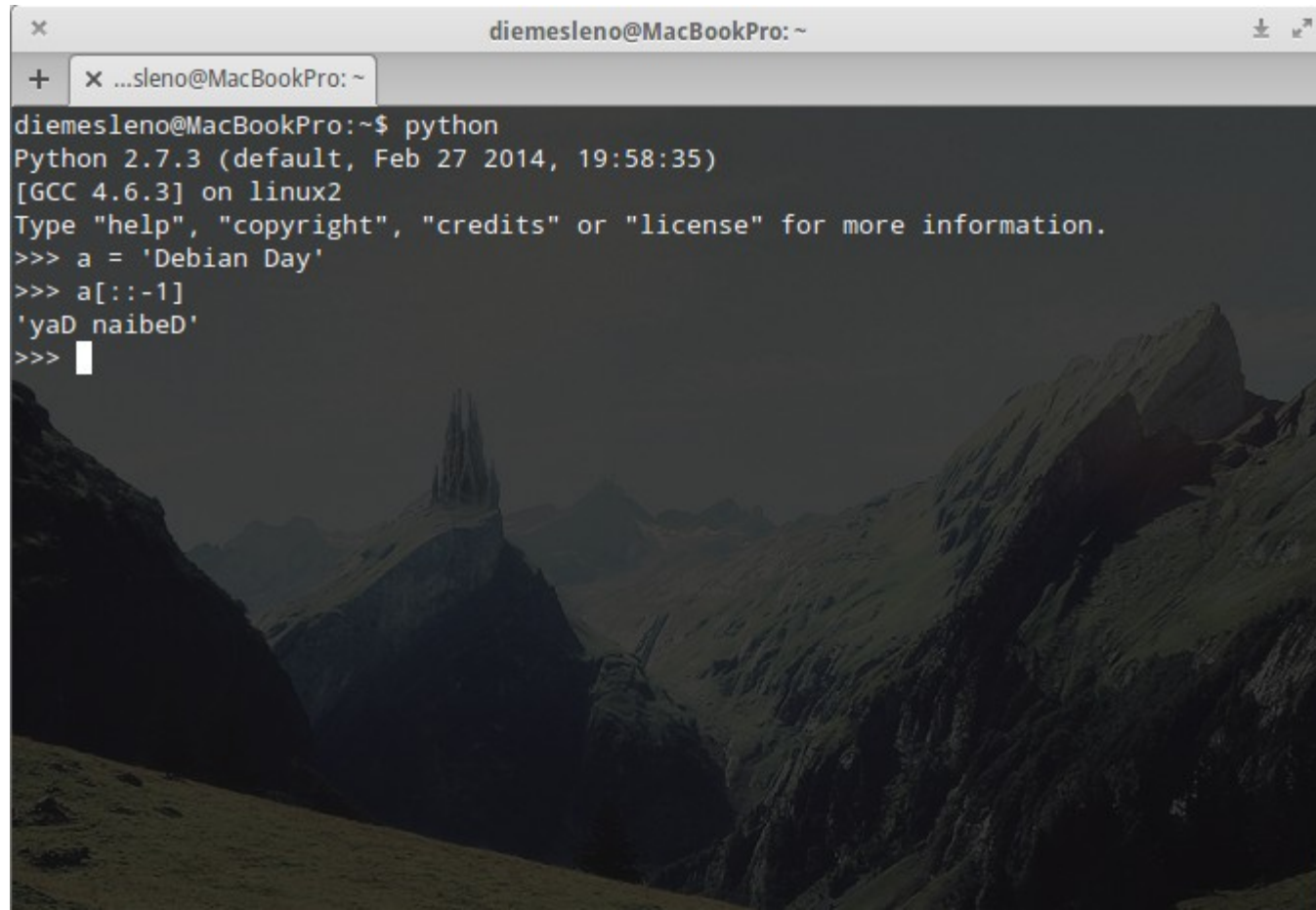
```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 'ian' in 'Debian'
True
>>>
```

Python Básico: Variáveis e Tipos

★ Strings - Exemplo reverse

a = 'Debian Day'

a[::-1]

A screenshot of a terminal window on a Mac. The window title is 'diemeslento@MacBookPro: ~'. The terminal shows a Python 2.7.3 shell session. The user enters 'python', then 'a = \'Debian Day\'', and finally 'a[::-1]'. The output is 'yaD naibeD'. The terminal background has a dark, mountainous landscape wallpaper.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'Debian Day'
>>> a[::-1]
'yaD naibeD'
>>>
```

Python Básico: Variáveis e Tipos

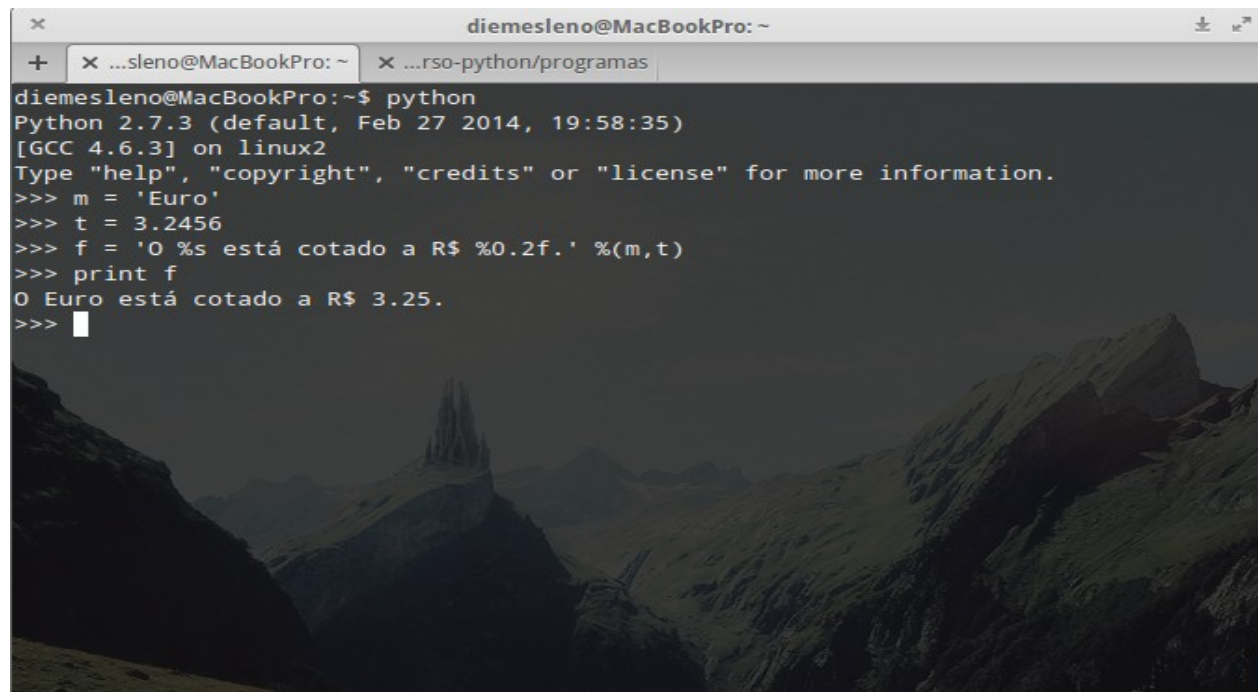
★ Strings - Exemplo com variáveis

`m = 'Euro'`

`t = 3.2456`

`f = 'O %s está cotado a R$ %0.2f.' %(m,t)`

`print f`

A screenshot of a terminal window titled 'diemeslento@MacBookPro: ~'. The window shows the execution of a Python script. The code entered is: `m = 'Euro'`, `t = 3.2456`, `f = 'O %s está cotado a R$ %0.2f.' %(m,t)`, and `print f`. The output of the script is: `O Euro está cotado a R$ 3.25.`. The terminal window has a dark background and a light-colored text. The window title bar shows the name of the user and the machine. The window content shows the prompt `diemeslento@MacBookPro:~$` followed by `python`. The Python version is `Python 2.7.3 (default, Feb 27 2014, 19:58:35)`. The compiler is `[GCC 4.6.3] on linux2`. The prompt is `>>>`. The code is entered line by line. The output is shown after the `print f` command. The terminal window is open on a desktop with a mountain landscape background.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> m = 'Euro'
>>> t = 3.2456
>>> f = 'O %s está cotado a R$ %0.2f.' %(m,t)
>>> print f
O Euro está cotado a R$ 3.25.
>>>
```


Python Básico: Variáveis e Tipos

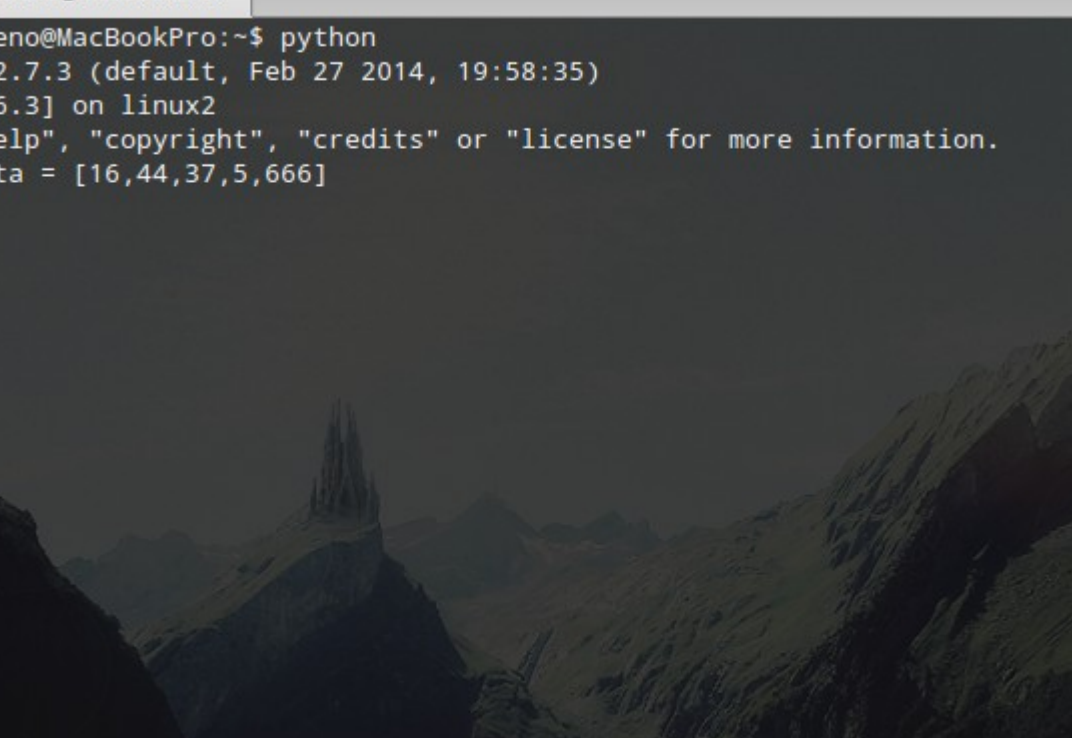
★ Listas

- Objeto iterável.
- Imutável
- Objetos dentro de colchetes []
- Acessível através de índices
- ...

e Tipos

★ Listas - Exemplo

```
lista = [16, 44, 37, 5, 666]
```



```
diemeslino@MacBookPro: ~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> lista = [16,44,37,5,666]
>>>
```

Python Básico: Variáveis e Tipos

★ Listas - Exemplo

Verifique os métodos disponíveis com o dir

Testar:

- append()
- insert()
- pop()
- remove()
- sort()
- reverse()
- count()

```
diemeslento@MacBookPro: ~  
+ x ...slento@MacBookPro: ~  
diemeslento@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> lista = [16,44,37,5,666]  
>>> dir(lista)  
['_add_', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__setslice__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']  
>>> █
```

Python Básico: Variáveis e Tipos



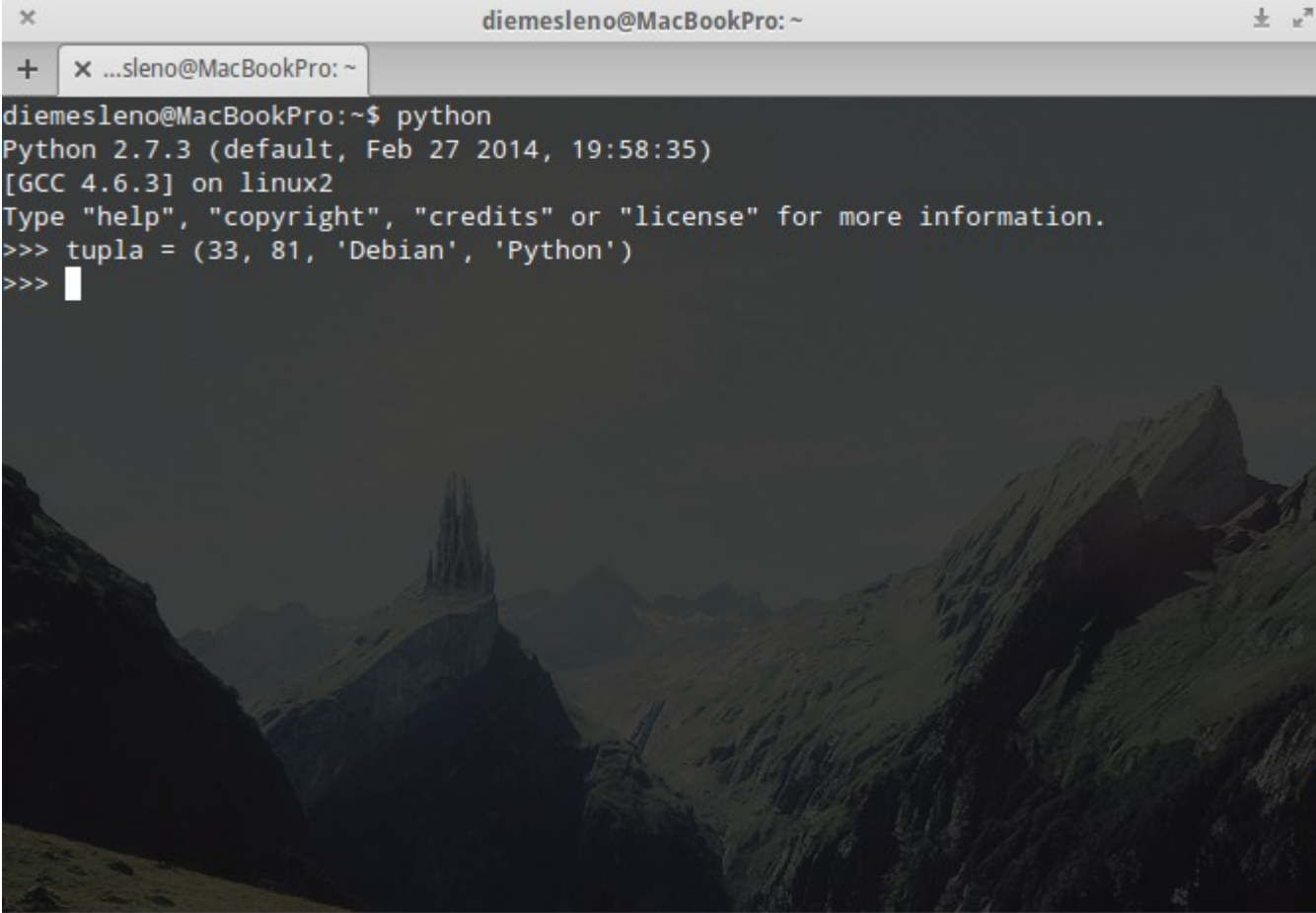
Tuplas

- Imutável
- Objetos dentro de parênteses ()
- Objetos acessíveis pelo índice
- Indicado para retorno em métodos com múltiplo valores
- ...

Python Básico: Variáveis e Tipos

★ Tuplas - Exemplo

tupla = (33, 81, 'Debian', 'Python')

A screenshot of a terminal window on a Mac. The window title is 'diemeslino@MacBookPro: ~'. The terminal shows the command 'python' being executed, which starts the Python 2.7.3 interpreter. The prompt is '>>>'. The user enters 'tupla = (33, 81, 'Debian', 'Python')' and presses enter, resulting in a new prompt '>>>'. The background of the terminal window shows a dark, mountainous landscape.

```
diemeslino@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> tupla = (33, 81, 'Debian', 'Python')
>>>
```

Python Básico: Variáveis e Tipos

★ Tuplas - Exemplo

Verifique os métodos disponíveis com o dir

Testar:

tupla[3]

```
diemeslino@MacBookPro: ~  
+ x ...sleno@MacBookPro: ~  
diemeslino@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> tupla = (33, 81, 'Debian', 'Python')  
>>> dir(tupla)  
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__for  
mat__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice_  
_', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '  
_mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul_  
_', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']  
>>> |
```

Python Básico: Variáveis e Tipos



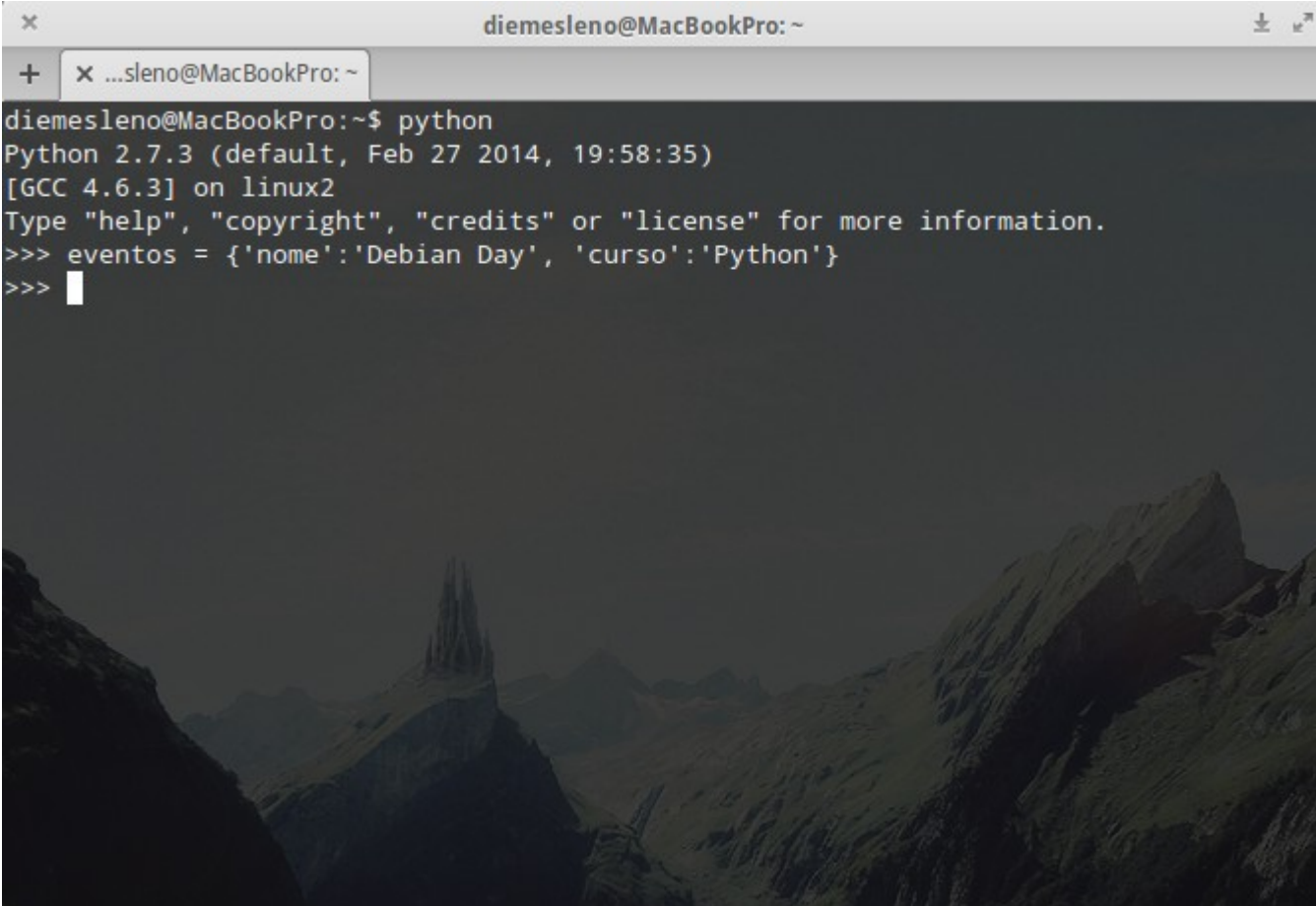
Dicionários

- Mapping
- Delimita os objetos com chaves {}
- Par CHAVE:VALOR
- ...

Python Básico: Variáveis e Tipos

★ Dicionários - Exemplo

```
eventos = {'nome':'Debian Day', 'curso':'Python'}
```

A screenshot of a terminal window on a Mac. The window title is 'diemeslento@MacBookPro: ~'. The terminal shows the execution of 'python', which starts the Python 2.7.3 interpreter. The user enters a dictionary assignment: 'eventos = {'nome':'Debian Day', 'curso':'Python'}'. The prompt '>>>' is visible at the end of the line.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> eventos = {'nome':'Debian Day', 'curso':'Python'}
>>>
```


Python Básico: Variáveis e Tipos

★ Dicionários - Exemplo

Verifique os métodos disponíveis com o dir

Testar:

- keys()
- values()
- viewkeys()
- viewitems()
- viewvalues()
- __sizeof__()
- clear()

```
diemeslino@MacBookPro: ~  
+ x ...sleno@MacBookPro: ~  
diemeslino@MacBookPro:~$ python  
Python 2.7.3 (default, Feb 27 2014, 19:58:35)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> eventos = {'nome':'Debian Day', 'curso':'Python'}  
>>> dir(eventos)  
['__class__', '__cmp__', '__contains__', '__delattr__', '__delitem__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'has_key', 'items', 'iteritems', 'iterkeys', 'itervalues', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values', 'viewitems', 'viewkeys', 'viewvalues']  
>>> 
```

Python Básico



Range

- Função geradora de iteradores
- `range(4)`
- `range(4,9)`
- `range(0, 10, 2)`
- `a = range(4)`
- `a.insert(5, 666)`
- `a.sort()`
- `a`

Python Básico



raw_input()

- Função para receber dados via teclado
- Dados recebidos são tratados como string
- nome = raw_input()

Python Básico



input()

- Função para receber dados via teclado
- Dados recebidos são tratados como int
- idade = input()

Python Básico



Palavras Reservadas

- `and`
- `assert`
- `break`
- `class`
- `continue`
- `def`
- `del`
- `elif`
- `else`
- `except`
- `exec`
- `finally`
- `for`
- `from`
- `global`
- `if`
- `import`
- `in`
- `is`
- `lambda`
- `not`
- `or`
- `pass`
- `print`
- `raise`
- `return`
- `try`
- `while`
- `yield`

Python Básico

Blocos

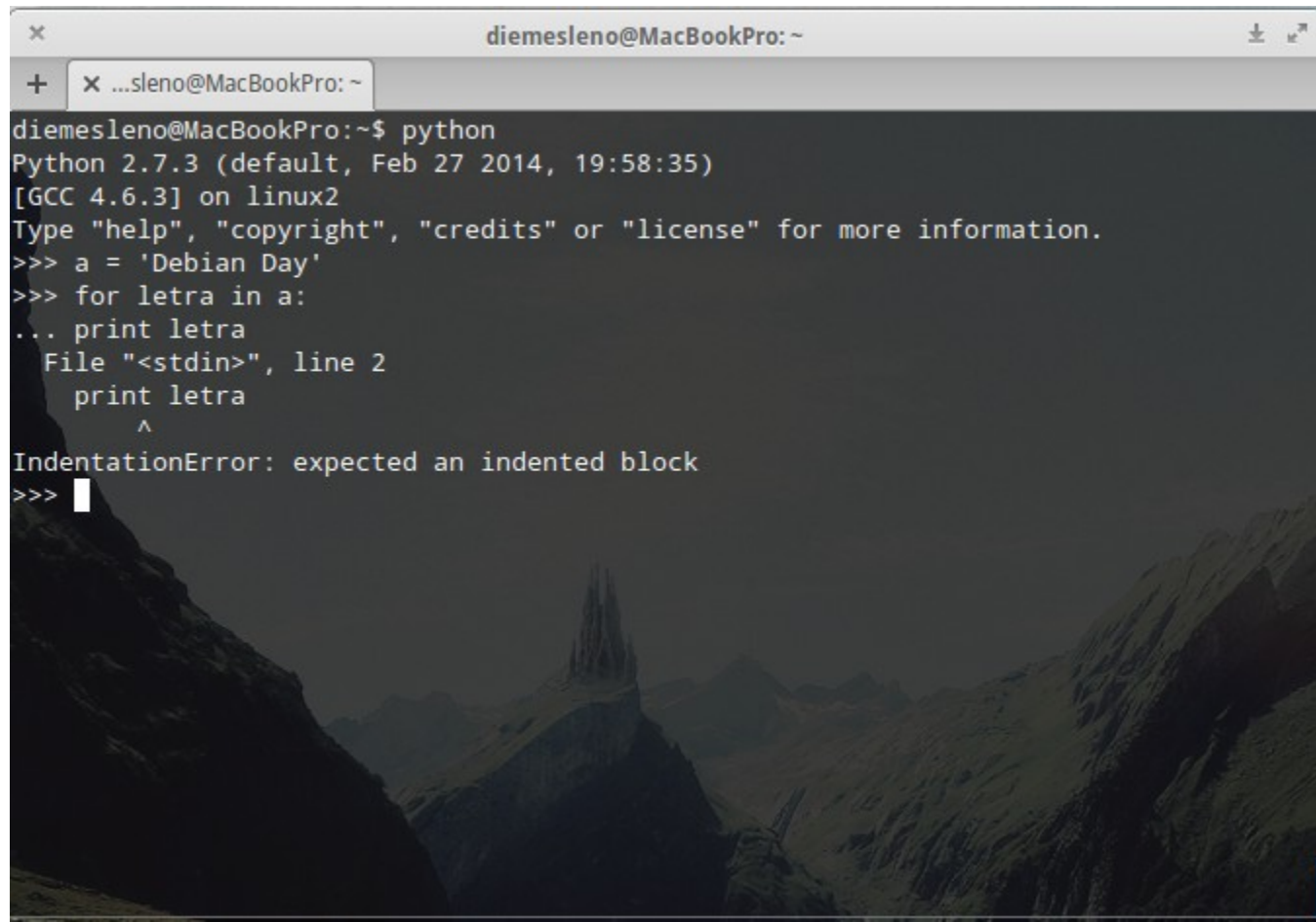
- Identação obrigatória

- Tab's ou

- 4 espaços

- * Recomendado

- 4 espaços

A terminal window titled 'diemeslento@MacBookPro: ~' with a single tab '...sleno@MacBookPro: ~'. The terminal shows the execution of 'python', displaying 'Python 2.7.3 (default, Feb 27 2014, 19:58:35) [GCC 4.6.3] on linux2'. It then shows a code snippet: '>>> a = 'Debian Day'' followed by '>>> for letra in a:' and '... print letra'. The next line is 'File "<stdin>", line 2' followed by 'print letra' with an arrow pointing to the indentation. The terminal then displays 'IndentationError: expected an indented block' and '>>>' with a cursor.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'Debian Day'
>>> for letra in a:
... print letra
File "<stdin>", line 2
  print letra
    ^
IndentationError: expected an indented block
>>> 
```

Python Básico

Operadores

==

!=

&

><

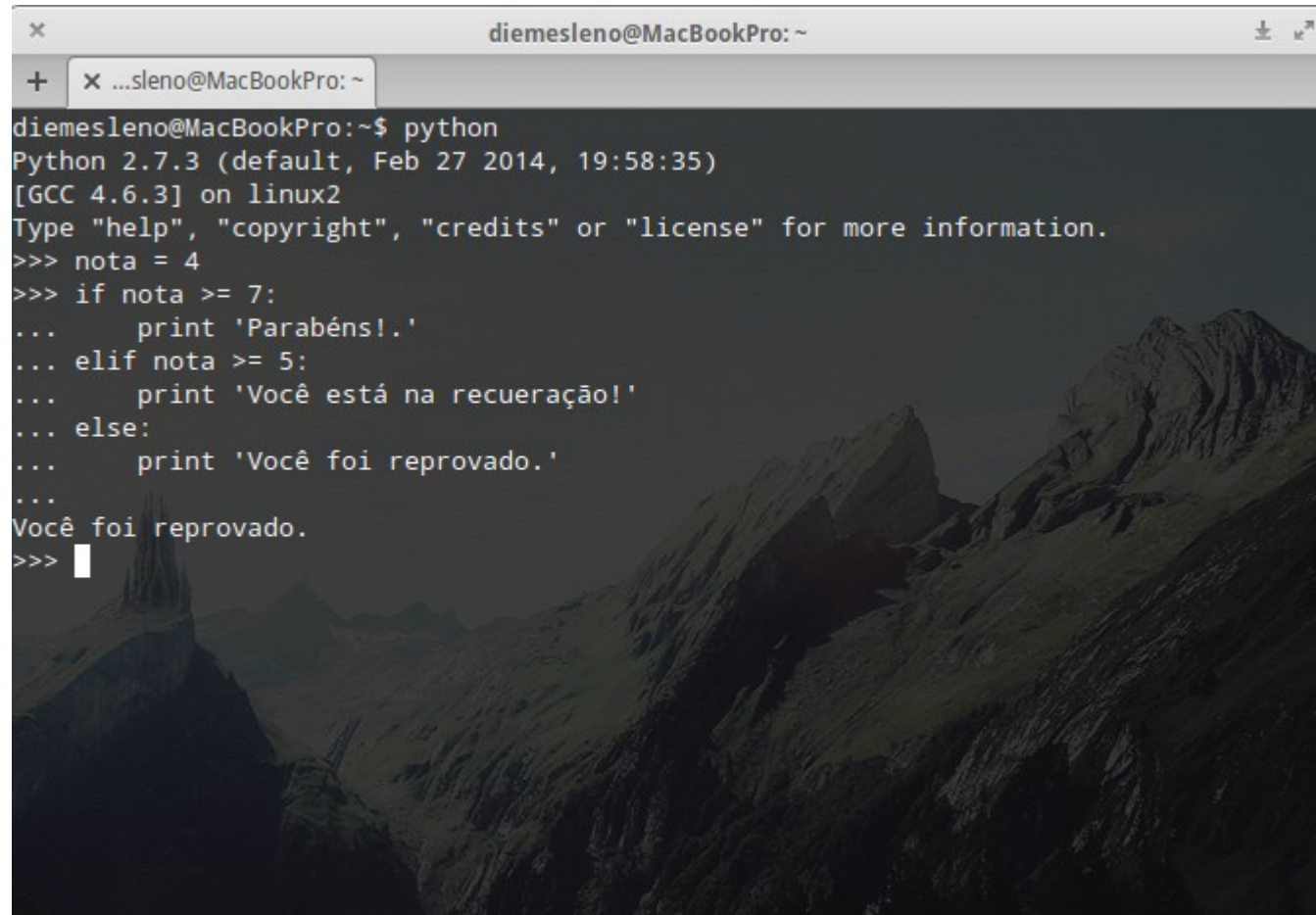
is

in

if

elif

else

A screenshot of a terminal window on a Mac. The window title is "diemeslino@MacBookPro: ~". The terminal shows a Python 2.7.3 shell session. The user enters "python", and the shell displays version and platform information. Then, the user enters a series of commands: "nota = 4", "if nota >= 7:", " print 'Parabéns!.'", "elif nota >= 5:", " print 'Você está na recuperação!'", "else:", " print 'Você foi reprovado.'". The output shows "Você foi reprovado." and the prompt returns to the user.

```
diemeslino@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> nota = 4
>>> if nota >= 7:
...     print 'Parabéns!.'
... elif nota >= 5:
...     print 'Você está na recuperação!'
... else:
...     print 'Você foi reprovado.'
...
Você foi reprovado.
>>>
```

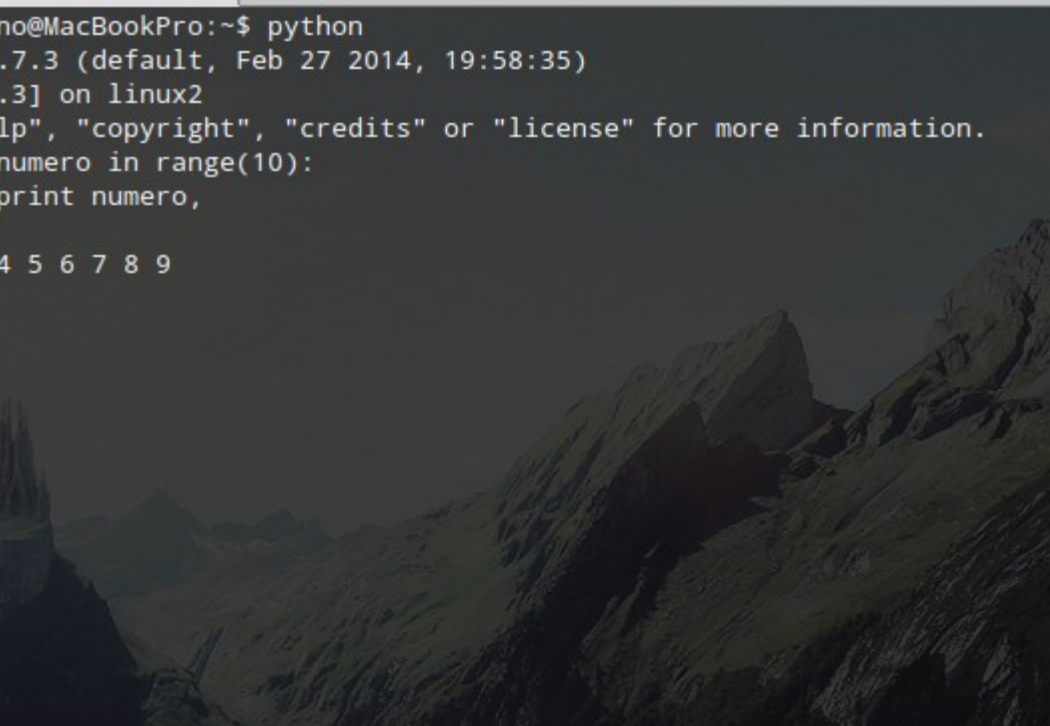
Python Básico

Instruções de Repetição - while

```
diemeslino@MacBookPro: ~  
+ x ...sleno@MacBookPro: ~  
>>> numero = 15  
>>> while numero > 10:  
...     numero = input("Digite um numero: ")  
...  
Digite um numero: 30  
Digite um numero: 50  
Digite um numero: 33  
Digite um numero: 89  
Digite um numero: 60  
Digite um numero: 2  
>>> 
```



```
for var in objeto_iteravel:
```

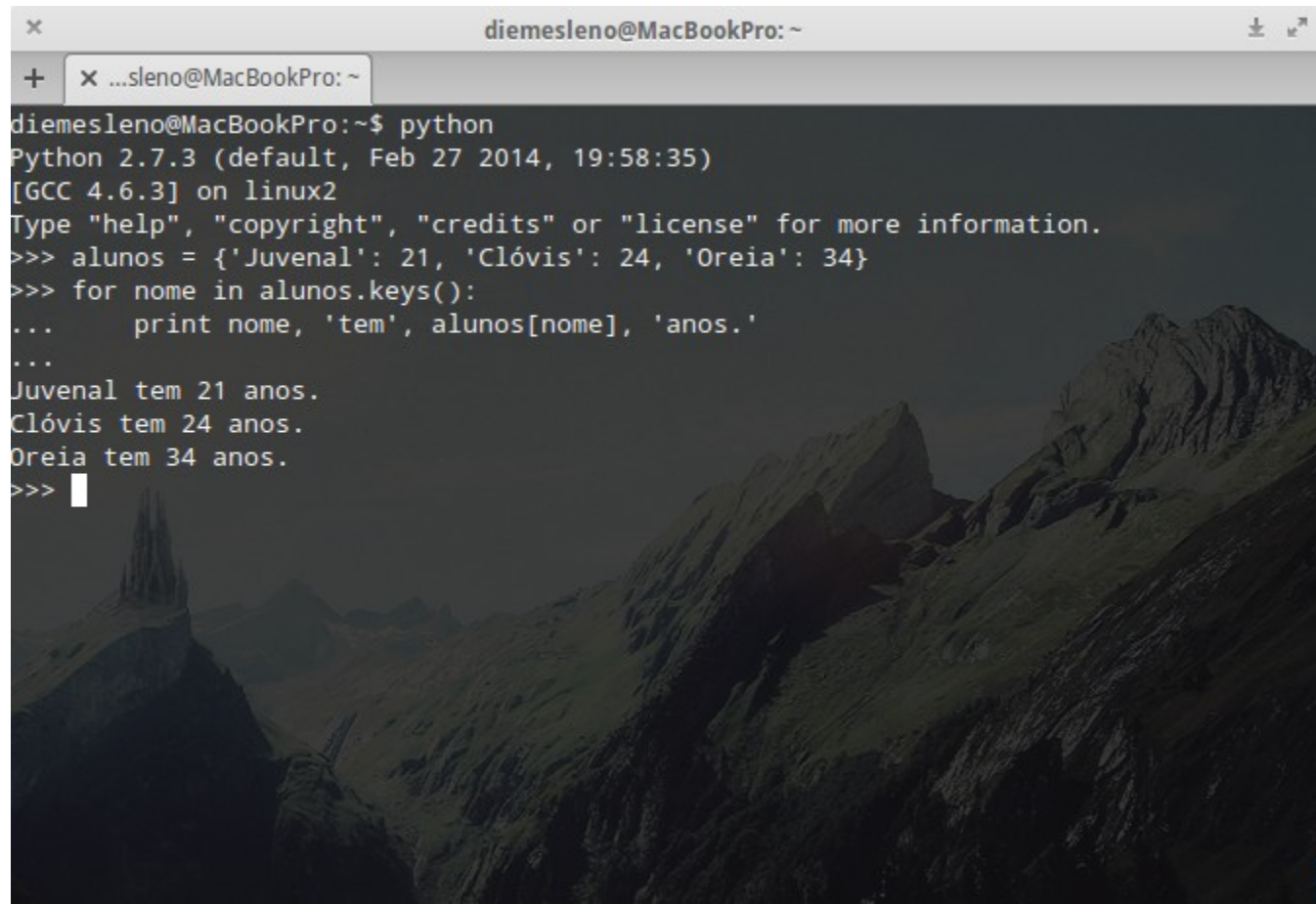
A screenshot of a terminal window on a Mac. The window title is "diemesleno@MacBookPro: ~". The terminal shows a Python prompt where the user has run "python". The output shows "Python 2.7.3 (default, Feb 27 2014, 19:58:35) [GCC 4.6.3] on linux2". The user then enters a loop: ">>> for numero in range(10):", followed by an indented "print numero," on the next line. The output of the loop is the numbers "0 1 2 3 4 5 6 7 8 9" on a single line. The prompt ">>>" is followed by a cursor. The background of the terminal is a dark, grayscale image of a mountain range. The window has a standard Mac OS X title bar with a close button, a zoom button, and a full-screen button. The terminal window is titled "diemesleno@MacBookPro: ~". The terminal text is as follows:

```
diemesleno@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> for numero in range(10):
...     print numero,
...
0 1 2 3 4 5 6 7 8 9
>>>
```

Python Básico

Instruções de Repetição - for (Exemplo 2)

for **var** in **objeto_iteravel**:

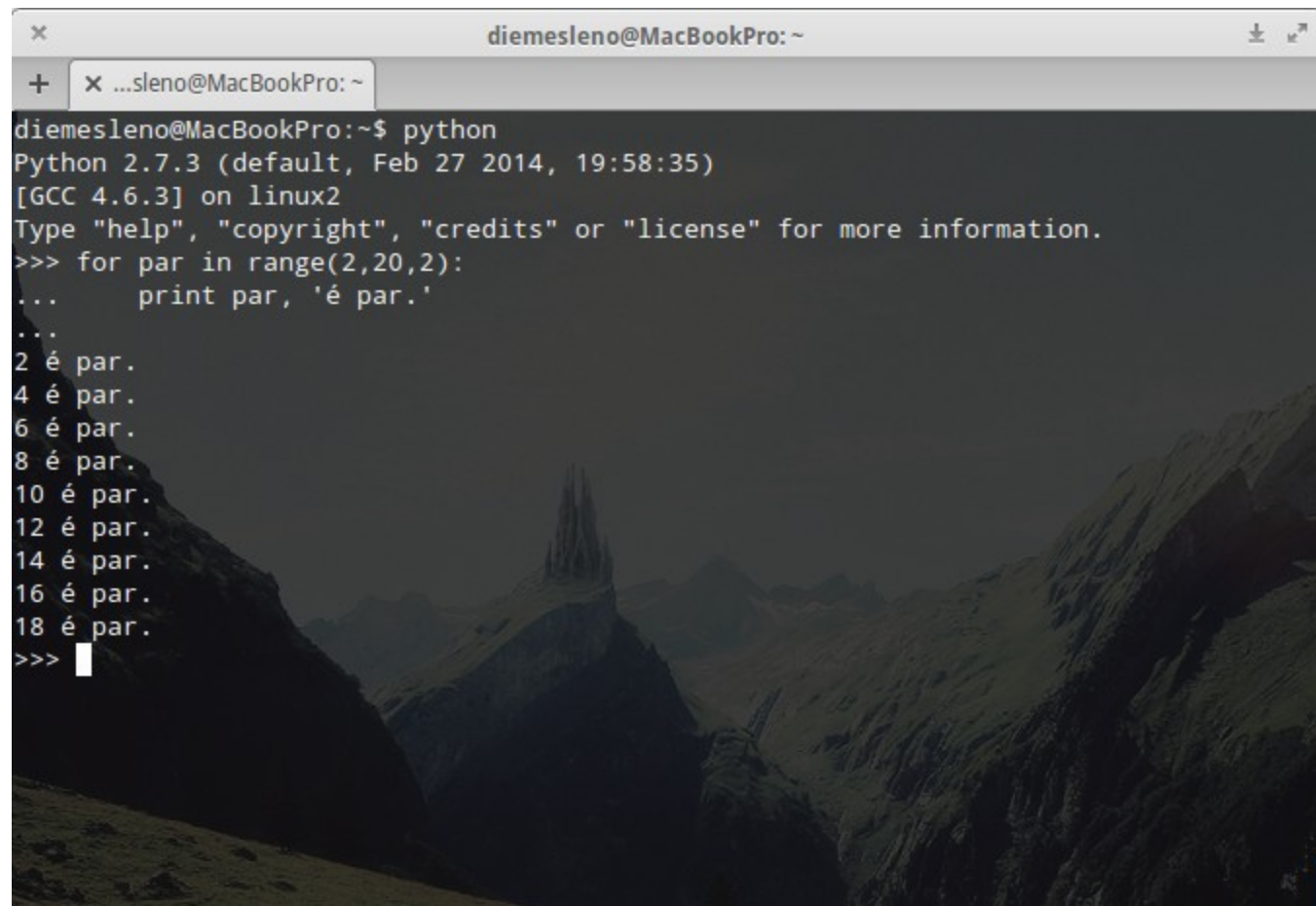
A screenshot of a terminal window titled 'diemeslento@MacBookPro: ~'. The terminal shows a Python 2.7.3 shell session. A dictionary 'alunos' is defined with three entries: 'Juvenal' (21), 'Clóvis' (24), and 'Oreia' (34). A 'for' loop iterates over the keys of the dictionary, printing each name followed by 'tem' and the corresponding age. The output shows: 'Juvenal tem 21 anos.', 'Clóvis tem 24 anos.', and 'Oreia tem 34 anos.'. The cursor is at the prompt '>>>' on the line following the last print statement. The background of the terminal window features a dark, mountainous landscape.

```
diemeslento@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> alunos = {'Juvenal': 21, 'Clóvis': 24, 'Oreia': 34}
>>> for nome in alunos.keys():
...     print nome, 'tem', alunos[nome], 'anos.'
...
Juvenal tem 21 anos.
Clóvis tem 24 anos.
Oreia tem 34 anos.
>>> 
```

Python Básico

Instruções de Repetição - for (Exemplo 3)

for **var** in **objeto_iteravel**:

A screenshot of a terminal window on a Mac. The window title is 'diemeslino@MacBookPro: ~'. The terminal shows a Python 2.7.3 shell session. The user enters a for loop that iterates over the range(2, 20, 2) and prints each value followed by 'é par.'. The output shows even numbers from 2 to 18. The terminal background has a dark, mountainous landscape.

```
diemeslino@MacBookPro:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> for par in range(2,20,2):
...     print par, 'é par.'
...
2 é par.
4 é par.
6 é par.
8 é par.
10 é par.
12 é par.
14 é par.
16 é par.
18 é par.
>>>
```

Aprofundando na linguagem



Criando programas python

- Abra um editor de textos / IDE*

*** Recomendo fortemente o Sublime Text**

Aprofundando na linguagem



Digite o código abaixo e salve* como programa1.py

```
#!/usr/bin/env python
#_*_ coding: utf-8 *_*

from datetime import datetime
from time import sleep

while True: #rodar eternamente
    hora = datetime.now()
    print hora.strftime('%H:%M:%S')
    sleep(1) #aguardar 1 segundo
```

* Recomendo criar um diretório para salvar todos os programas.

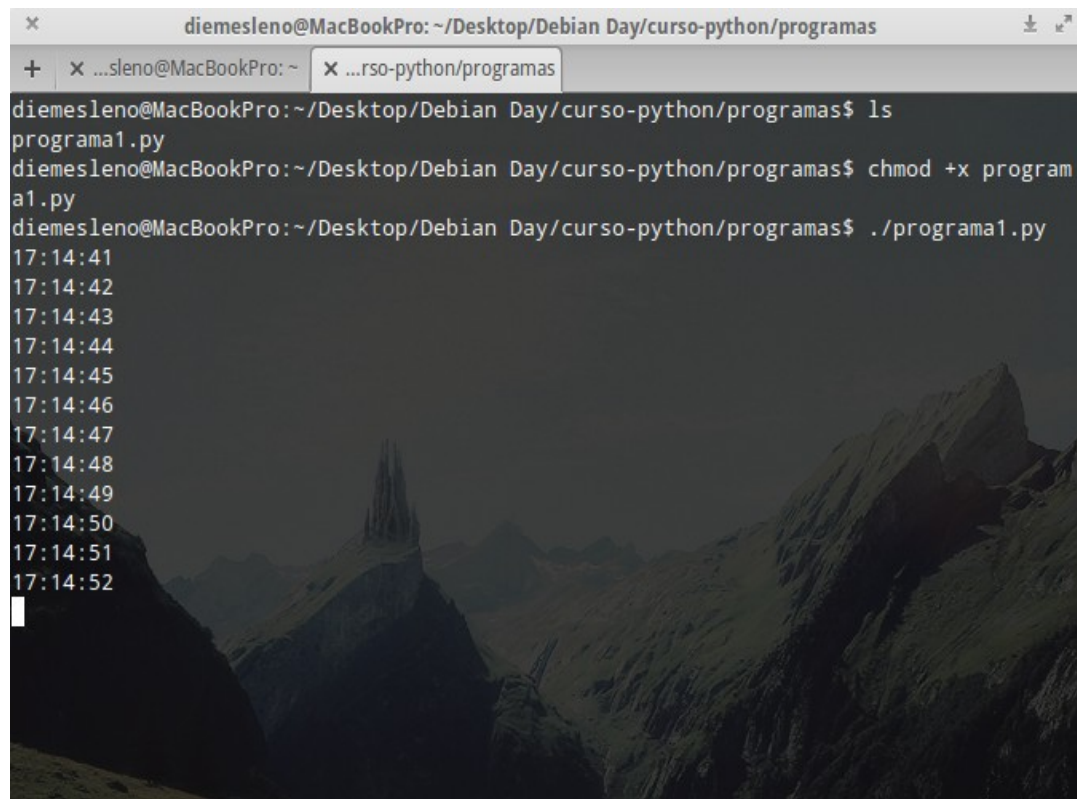
Aprofundando na linguagem

Utilizando o terminal, vá até o diretório onde está salvo o arquivo programa1.py e torne-o executável.

chmod +x programa1.py

Rode o programa.

./programa1.py

A screenshot of a terminal window on a Mac. The window title is 'diemeslino@MacBookPro: ~/Desktop/Debian Day/curso-python/programas'. The terminal shows the following commands and output:

```
diemeslino@MacBookPro:~/Desktop/Debian Day/curso-python/programas$ ls
programa1.py
diemeslino@MacBookPro:~/Desktop/Debian Day/curso-python/programas$ chmod +x programa1.py
diemeslino@MacBookPro:~/Desktop/Debian Day/curso-python/programas$ ./programa1.py
17:14:41
17:14:42
17:14:43
17:14:44
17:14:45
17:14:46
17:14:47
17:14:48
17:14:49
17:14:50
17:14:51
17:14:52
```

The background of the terminal window shows a dark, mountainous landscape.

Aprofundando na linguagem



Comentários em Python

comentários de 1 linha

*''' Comentários de
mais de 1 linha '''*

*""" Comentários de
mais de 1 linha """*

Exercício - 2



Crie um programa em Python, onde seja solicitado ao usuário o nome, a idade e a profissão. Após o usuário entrar com os dados, seja impresso na tela as informações deste usuário.

Exemplo de saída:

O Diemeslino tem 33 anos e é programador de sistemas.

Exercício - 3



Crie um programa em Python, onde seja solicitado ao aluno o nome, a nota do primeiro bimestre e a nota do segundo bimestre. O programa deve efetuar o cálculo da nota final e informar se o aluno está aprovado ou reprovado. Para ser aprovado o aluno tem que ter nota final maior ou igual a 7. Nota menor a 4 o aluno está reprovado. Nota maior ou igual a 4 e menor que 7 o aluno está de recuperação.

Exercício - 4



Crie um programa em Python, onde o usuário informe um item de cesta básica e seu preço. O item deve ser colocado em uma lista e os preços devem ser somados. A cada item adicionado deve-se imprimir em tela o valor atualizado da cesta básica.

Exercício – 5 [Desafio 1]



Crie um programa em Python, onde ao ser executado, imprima em tela o alfabeto de z até a.

Dica: importe o módulo string

Orientação a Objetos com Python



```
print("Hello, world!")
```

Orientação a Objetos com Python

Classe: Molde ou modelo do objeto do mundo real mapeado.

Atributo: Características do objeto.

Método: Ação que o objeto é capaz de realizar.

Objeto: Produto gerado a partir do molde. (Classe)

Exemplo:

Classe: Aluno

Atributo: 1.72, 23, masculino

Método: Estudar, Correr

Objeto: Juvenal

Orientação a Objetos com Python

Definindo Classes

```
class Veiculo(object):  
    pass
```

Palavra *class* tipicamente utilizada para definir classes! (:

Palavra reservada *pass*, significa: passe este bloco.

- Toda classe herda de 'object' ou de outra classe.
- Classes com iniciais maiúsculas, arquivo* com minúsculas.
- * O arquivo não precisa ter o mesmo nome. Podemos ter várias classes em um arquivo.

Orientação a Objetos com Python

Atributos de classe e de instância

```
class Veiculo(object):  
    marca = None  
    ano = None
```

← Atributos de classe

```
gol = Veiculo()  
gol.marca = "Volkswagen"  
gol.ano = 2008
```

```
class Veiculo(object):  
    pass
```

```
fusca = Veiculo()  
fusca.marca = "Volkswagen"  
fusca.ano = 1972
```

Atributos de instância →

Orientação a Objetos com Python



Atributos de classe e de instância?

- Um atributo de classe é um atributo definido na classe;
- Um atributo de instância é um atributo definido na instância da classe (objeto);
- Todo atributo de classe é refletido nas instâncias;
- Atributos de instâncias são únicos e pertencem apenas à uma instância específica;

Orientação a Objetos com Python

Atributos de classe e de instância?

```
class Veiculo(object):  
    pass
```

```
fusca = Veiculo()  
fusca.marca = "Volkswagen"  
fusca.ano = 1972
```

`marca` e `ano` são atributos de instância, se criarmos outro objeto da classe `Veiculo` estes atributos não existirão na nova instância.

Orientação a Objetos com Python

Instanciando objetos

- Já vimos isso de forma “não oficial”;
- Agora veremos oficialmente a sintaxe para instanciar objetos de uma classe!

Não tem *new*!

```
gol = Veiculo()  
gol.marca = "Volkswagen"  
gol.ano = 2008  
gol.andar()
```

Orientação a Objetos com Python

O que é um método construtor?

- É um método especial que constrói o objeto, ou seja, é o método que retorna uma instância de determinada classe;
- Em Python, o construtor chama-se `__new__`;
- Porém geralmente não se implementa o método construtor em Python!
- O que é implementado é o método `__init__`. Este método é chamado imediatamente após o construtor.

Orientação a Objetos com Python

O que é um método construtor?

```
class Veiculo(object):  
    marca = None  
    ano = None  
  
    def __init__(self, marca, ano):  
        self.marca = marca  
        self.ano = ano  
  
    def andar(self):  
        print("O carro esta andando!")
```

Orientação a Objetos com Python

Peculiaridade: Argumentos flexíveis

- Argumentos flexíveis, com valor *default* ou opcionais são argumentos que podem ou não serem passados para um método;
- Como Python não possui sobrecarga de métodos e funções, o uso de argumentos flexíveis é extremamente importante para trabalhar de forma similar;
- Definir um argumento flexível é fácil!

Orientação a Objetos com Python

Peculiaridade: Argumentos flexíveis

```
class Veiculo(object):  
    marca = None  
    ano = None  
  
    def __init__(self, marca = None, ano = None):  
        self.marca = marca  
        self.ano = ano  
  
    def andar(self):  
        print("O carro esta andando!")
```

Orientação a Objetos com Python

Exercício 6

- Utilizando seu editor de textos ou IDE favorito, crie uma classe chamada Tamagoshi com os atributos nome, idade, saude e fome.
- Abra o terminal, inicie o console python e instancie um objeto da classe Tamagoshi, colocando valores em seus atributos. Utilize o comando “dir” para conhecer a classe e o objeto.

OBS:

- Inicie o console Python dentro do diretório “programas”
- Importe a classe com o comando: *from tamagoshi import Tamagoshi* onde tamagoshi é o nome do arquivo que está sua classe.

Orientação a Objetos com Python



Exercício 7

- Defina um método inicializador na classe Tamagoshi. Este método receberá valores de fome, saúde e idade e armazenará estes valores nos atributos internos correspondentes;
- Utilize passagem flexível de parâmetros.

Orientação a Objetos com Python

Definindo métodos

- Um método é um atributo com comportamento;
- Para não fundir conceitos, vamos apenas trabalhar com métodos da forma tradicional, ou quase isso;
- Para definir métodos e funções em Python utilizamos a palavra reservada *def*;

Orientação a Objetos com Python

Definindo um método

```
class Veiculo(object):  
    marca = None  
    ano = None  
  
    def andar(self):  
        print("O carro esta andando!")
```

```
gol = Veiculo()  
gol.marca = "Volkswagen"  
gol.ano = 2008  
gol.andar()
```

Método “andar”

Orientação a Objetos com Python



Self?!?!

- Quando definimos nosso método, determinamos um parâmetro, o **self**;
- Quando utilizamos o método, porém, não passamos nenhum parâmetro!
- Quem é esse tal de self?

Orientação a Objetos com Python

Self?!?!

```
public class Veiculo {  
    public String marca = null;  
    public Integer ano = null;  
  
    public void setAno(Integer ano){  
        this.ano = ano;  
    }  
}
```

Quem é esse tal de **this**?

Orientação a Objetos com Python

Exercício 8

- Defina os seguintes métodos na classe Tamagoshi: `alterarNome`, `alterarFome`, `alterarSaude`, `retornarNome`, `retornarFome` e `retornarSaude`. Estes métodos deverão acessar os atributos anteriormente declarados;
- Defina o método `retornarHumor`, o humor não é um atributo, mas sim a soma entre a fome e a saúde;

Orientação a Objetos com Python



Herança de classes

- A herança permite criar um relacionamento entre classes utilizando subclasses. Uma *subclasse* herda atributos e métodos de sua *superclasse*.
- Utilizar herança pode poupar trabalho se métodos puderem ser escritos uma vez em uma superclasse em vez de muitas vezes em subclasses separadas.
- Para definirmos herança de classes em Python, utilizamos parênteses;
- Já utilizamos herança!

Orientação a Objetos com Python

Herança de classes

Prática! Implemente a classe Pessoa

```
class Pessoa(object):  
    nome = None  
    idade = None  
  
    def __init__(self, nome, idade):  
        self.nome = nome  
        self.idade = idade  
  
    def envelhecer(self):  
        self.idade += 1
```

Orientação a Objetos com Python

Herança de classes

Prática! Implemente a classe Atleta

```
class Atleta(Pessoa):  
    peso = None  
    aposentado = None  
  
    def __init__(self, nome, idade, peso):  
        super(Atleta, self).__init__(nome, idade)  
        self.peso = peso  
        self.aposentado = False  
  
    def aquecer(self):  
        print("Atleta aquecido!")  
  
    def aposentar(self):  
        self.aposentado = True
```

Orientação a Objetos com Python

Herança de classes

Prática! Implemente as classes Corredor, Nadador e Ciclista

```
class Corredor(Atleta):  
    def correr(self):  
        print("Corredor correndo!")
```

```
class Nadador(Atleta):  
    def nadar(self):  
        print("Nadador nadando!")
```

```
class Ciclista(Atleta):  
    def pedalar(self):  
        print("Ciclista pedalando!")
```

Orientação a Objetos com Python

Herança múltipla

Prática! Implemente a classe TriAtleta

- Há uma peculiaridade no diagrama de classes: a classe **TriAtleta** herda de três outras classes;
- Muitas pessoas podem pensar: mas isso é impossível!
- Em Python, assim como em C++, isso é possível sim! Logo, a classe TriAtleta fica bem simples:

```
class TriAtleta(Ciclista, Nadador, Corredor):  
    pass
```


Orientação a Objetos com Python



Exercício 9

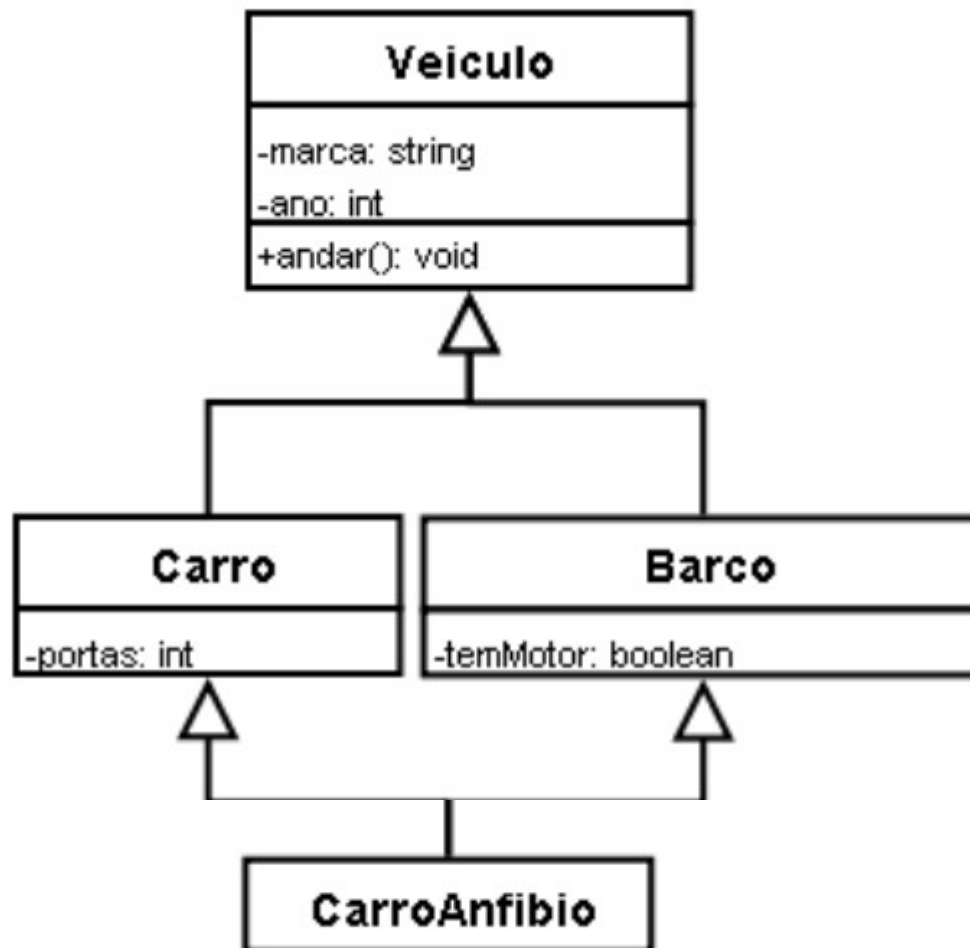
Instancie 1 objeto de cada uma das classes e execute seus métodos:

- Pessoa
- Atleta
- Corredor
- Nadador
- Ciclista
- TriAtleta

Orientação a Objetos com Python

Exercício 10

- Implemente o diagrama de classes abaixo:



Polimorfismo

- Mesmo que Python não suporte sobrecarga de métodos em uma mesma classe, é possível reimplementar métodos em uma hierarquia de classes;
- Vamos reimplementar em todas as subclasses de Atleta o método aquecer.

Orientação a Objetos com Python

Polimorfismo

```
class Corredor(Atleta):  
    def correr(self):  
        print("Corredor correndo!")  
  
    def aquecer(self):  
        print("Corredor aquecido!")
```

Prática!

```
class Nadador(Atleta):  
    def nadar(self):  
        print("Nadador nadando!")  
  
    def aquecer(self):  
        print("Nadador aquecido!")
```

```
class Ciclista(Atleta):  
    def pedalar(self):  
        print("Ciclista pedalando!")  
  
    def aquecer(self):  
        print("Ciclista aquecido!")
```

Orientação a Objetos com Python

Encapsulamento

- Em Python, todos os atributos e métodos são públicos!
- Porém existe uma forma de dificultar o acesso aos atributos e métodos, indicando que acessar aquele atributo diretamente não é a operação aconselhada;
- Basta adicionar dois `_` (underline) à frente dos atributos e métodos;
- Podemos então definir métodos de acesso em nossas classes;

Orientação a Objetos com Python

Encapsulamento

```
Classe Pessoa
def retornarNome(self):
    return self.__nome

def alterarNome(self, nome):
    self.__nome = nome

def retornarIdade(self):
    return self.__idade

def alterarIdade(self, idade):
    self.__idade = idade
```


Encapsulamento

OBS: Lembre-se que temos atributos de instância e atributos de classe
Quando trabalhamos com OO em Python.

Orientação a Objetos com Python

Exercício 11 – Parte 1

Escreva um programa de bancos que possua:

- Uma classe **Banco**:
 - com os atributos:
 - private total
 - public taxa_reserva
 - private reserva_xigida
 - com os métodos:
 - private calcular_reserva
 - public pode_fazer_emprestimo(valor) → boolean
 - consultar_total
 - adicionar_total
 - diminuir_total

Informações:

total = O total de dinheiro que o banco tem.

taxa_reserva = Taxa exigida pelo Banco Central para o banco manter reserva.

reserva_exigida = Cálculo entre o total e a taxa_reserva para saber quanto o banco tem que manter. O Banco nunca pode ficar com menos que esse valor.

Orientação a Objetos com Python

Exercício 11 – Parte 2

- Uma classe Conta:
 - com os atributos:
 - private saldo
 - private id_conta
 - private senha
 - private banco
 - com os métodos:
 - public depositar(senha, valor)
 - public sacar(senha, valor)
 - public solicitar_emprestimo(valor) → boolean
 - public consultar_saldo → float

Regras:

- Para efetuar um depósito deverá checar a senha, o valor será acrescido no saldo do cliente e também no total do Banco.
- Para efetuar um saque, deverá ser checada a senha, o valor tem que ser menor ou igual o saldo e deve-se debitar no total do Banco.
- Caso o cliente possa efetuar empréstimo o valor deverá ser debitado do total do banco e acrescido no saldo do cliente.

Como seguir daqui para frente?



```
print("Hello, world!")
```

Como seguir daqui para frente?



Quer aprender mais Python?

Python:

- *<http://www.python.org>*

Python Brasil:

- *<http://www.python.org.br>*

Como seguir daqui para frente?

Quer utilizar Python para Desktop?

PythonTk:

- *<https://wiki.python.org/moin/TkInter>*

PythonQT:

- *<https://qt-project.org/search/tag/python>*

Kivy:

- *<http://kivy.org/>*

Recomendo olhar bem de perto o Kivy!

Como seguir daqui para frente?

Quer utilizar Python para Web?

Django:

- <https://www.djangoproject.com/>
- <http://www.djangobrasil.org/>



Web2py:

- <http://www.web2py.com/>
- <http://bit.ly/1AbHxSi>



Flask:

- <http://flask.pocoo.org/>



Como seguir daqui para frente?

Quer utilizar Python em Games?

Pygame:

- <http://www.pygame.org/>



Cocos2D:

- <http://cocos2d.org/>

Pyglet:

- <http://www.pyglet.org/>

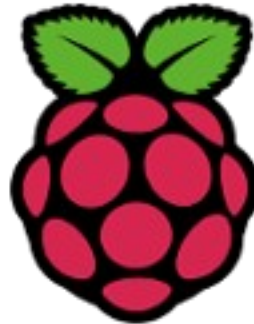


Como seguir daqui para frente?

Quer utilizar Python para mobile?

Kivy:

- <http://kivy.org/>



Como seguir daqui para frente?



Seu melhor amigo e pior inimigo

Google:

- *<http://www.google.com.br>*

Conclusões



Python apesar de não ter todo o marketing que Java tem, é muito utilizada no mercado e consagrada dentro das melhores universidades.

Para quem quer aprender uma linguagem para poder atuar em diferentes frentes de trabalho, Python com certeza é uma boa opção.

Referências



- ★ Mini-curso de Python de Francisco A. S. Souza
- ★ Introdução a Linguagem de Programação Python de Flávio Ribeiro.
- ★ Python para Desenvolvedores 2ed. De Luiz Eduardo Borges
- ★ Python a primeira mordida de Marco André Lopes Mendes
- ★ www.python.org/doc



Perguntas?

Diemesleno Souza Carvalho
diemesleno@gmail.com
www.diemesleno.com.br