01 flertando com python

O prompt interativo esse ilustre desconhecido

Chamando

```
$ python
```

Temos

```
$ python
Python 2.6.5 (r265:79063)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits"
or "license" for more information.
>>>
```

Seu melhor amigo

```
>>> a = 5
>>> print a
5
>>> b = 4
>>> print a + b
9
```

Hello world

```
>>> from datetime import datetime
>>> from time import sleep
>>> while True: #rodar para sempre
...     hora = datetime.now()
...     print hora.strftime("%H:%M:%S")
...     sleep(1) #aguardar um segundo
...
```

Por partes

True e **False** com a primeira letra em Maiuscula

```
>>> while True: #rodar para sempre
                ^ marca o inicio do bloco
...     hora = datetime.now()
...     print hora.strftime("%H:%M:%S")
...     sleep(1) #aguardar um segundo
    ^^^^ a identacao tem que ser constante
```

padrão são 4 espaços, mas voce tem liberdade de escolher desde que seja consistente

Como chamar programas em python

Crie um arquivo teste.py

```
print "funciona"
```

Na linha de comando

```
$ python teste.py
funciona
```

Importando módulos

Módulos são arquivos. Eles são importados do sys.path

```
>>> import sys
>>> print sys.path
['', '/usr/lib/python2.6',
....
'/usr/lib/python2.6/dist-packages',
'/usr/local/lib/python2.6/dist-packages']
```

Criando módulos

cria o diretório

```
mkdir queijo
gedit queijo/quente.py
```

cria o módulo

```
print "importou queijo quente"
def foo(val):
    print val
```

importa dentro do python

```
>>> from queijo import quente
ImportError: No module named queijo
```

Módulos são definidos pelo init

Para o python entender o diretorio como um modulo ele precisa de um arquivo chamado __init__.py dentro dele

```
touch queijo/__init__.py
```

Salva o aquivo vazio

importa dentro do python

```
>>> from queijo import quente
importou queijo quente

>>> quente.foo("ola")
ola
```

ou

```
>>> from queijo.quente import foo

>>> foo("ola")
ola
```

Blocos  

```
if / elif / else

for / else

while / else

try / except / finally

class / def
```

Funções 1

```
def foo():
    pass
```

```
def foo(larari):
    print larari
```

Funções dentro de funções

```
def foo():
    print "antes de declarar bar"
    def bar():
        print "dentro do bar"
    print "depois de declarar bar"
    bar()

foo()
```

saida

```
antes de declarar bar
depois de declarar bar
dentro do bar
```

e bar()?

```
bar()

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

Parametros

```
def foo(nome, sobrenome, comp="bom", hora="dia"):
    print comp, hora, ":", nome, sobrenome

foo("adriano","petrich")
bom dia : adriano petrich

foo("adriano","petrich", "pessimo")
pessimo dia : adriano petrich

foo("adriano","petrich", hora="noite", comp="boa")
boa noite : adriano petrich
```

Mas Cuidado

Primeiro parâmetros sem nome e depois os nomeados

```
foo("adriano", hora="noite", "petrich")
SyntaxError: non-keyword arg after keyword arg
```

Sequências

Principais classes

- string e unicode
- listas e tuplas
- coisas que se comportam como sequência

Implica

```
len(s)    # length
min(s)    # ou max(s)
s[i]      # iesimo item de s (base 0)
s[i:j]    # slice do iesimo a jesimo item
s[i:j:k] # mesma coisa com passo k
s + t     # concaternação
s * i     # ou i * s;  i copias de s
x in s    # se s tem o elemento x
x not in s # ou nao tem
```

Strings

- Imutaveis
- suportam operações de sequência
- demarcadas com ' ou "  
- multilinhas com ' ' ' ou " " "

Strings como strings

```
a = '  abcd \n '
a.upper() # '  ABCD \n '
a.strip() # 'abcd'
a.islower() # True
# isalnum/isalpha/isdigit
# islower/isspace/istitle/isupper
a.startswith("  ") # True
# endswith
a.find("abc") # 2
a.split("b") # ['  a', 'cd \n ']
```

Strings como sequências

```
a = "ola python"
len(a)
min(a)
a[1]
a[-1]
a[-2]
a[:3]
a[4:]
a[:]
a[::-1]
```

Magia com Slices

```
len(a)  # 10
min(a)  # ' '
a[1]    # 'l'
a[-1]   # 'n'
a[-2]   # 'o'
a[:3]   # 'ola'
a[4:]   # 'python'
a[:]    # 'ola python'
a[::-1] # 'nohtyp alo'
```

Por que não?

```
a = "olá python"
```

Unicode

Padrão universal. Suporta todas as liguas do planeta (mesmo que algumas de forma muito simplificada)

Normalmente: usa um byte para ASCII baixo e dois para os outros caracteres

Em python é representada com o prefixo **u** antes de uma string

Strings vs

```
>>> a = "olá python"
>>> a
'ol\xc3\xa1 python'
>>> len(a)
11
>>> print a
olá python
```

Unicode

```
>>> b = u"olá python"
>>> b
u'ol\xe1 python'
>>> len(b)
10
>>> print b
olá python
```

Conversão entre Str e Unicode

```
>>> "olá".decode("iso-8859-15")
u'ol\xc3\xa1'
```

```
>>> u"olá".encode("utf-8")
'ol\xc3\xa1'
```

Codificações usadas no Brasil

ISO-8859-1 ou latin-1

ISO-8859-15 com o euro (€)

cp1252 MSWindows codepage 1252 (‘’ “” •)

utf-8 recomendado

Em scripts

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

Listas

Listas são sequencias mutaveis

```
a = ['ola', 2, 'queijo quente', 3.14]
a[0] # 'ola'
a[:2] # ['ola', 2]

for i in a:
    print i
```

for .. in ..

for .. in .. else

```
a = [1,2,3,4,5,6]
for i in a:
    print i * i

a = []
for i in a:
    print i
else:
    print "lista vazia"
```

for .. in .. else

```
1
4
9
16
25
36

lista vazia
```

file open

```
with open('arquivo.txt') as arquivo:
    for linha in arquivo:
        print linha.strip()
```