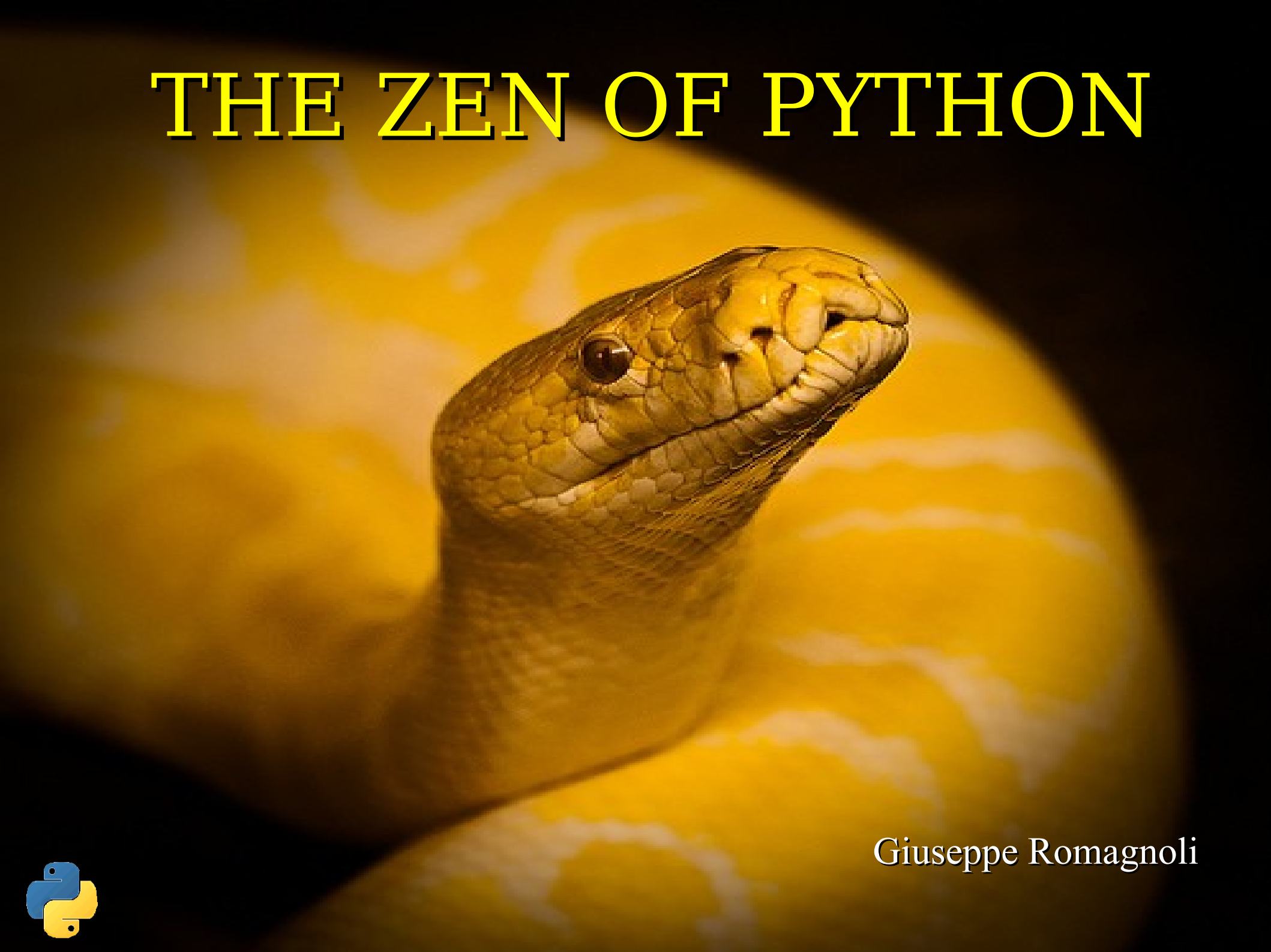


# THE ZEN OF PYTHON



Giuseppe Romagnoli





# Trabalho

## SERPRO

**Serviço Federal de Processamento de Dados**





# SERPRO



- **Maior Provedor do Governo de tecnologia de Informações e Comunicações**
- **Clientes: Ministério da Fazenda, Planejamento, Justiça, Transportes, Educação e Gabinete Civil e Presidência.**
- **Mais de 3 bilhões de Transações/ano**
- **Múltiplas plataformas e padrões abertos**
- **ASP + ISP + NSP**



# Conteúdo

- O que é o Python ?
- Quem usa Python ?
- Por que usar Python ?
- Como é o Python ?

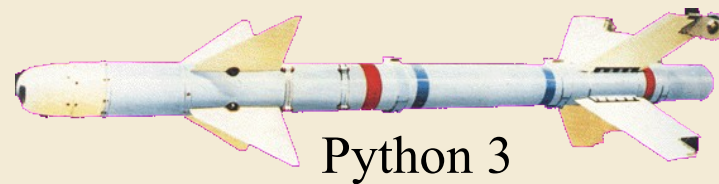
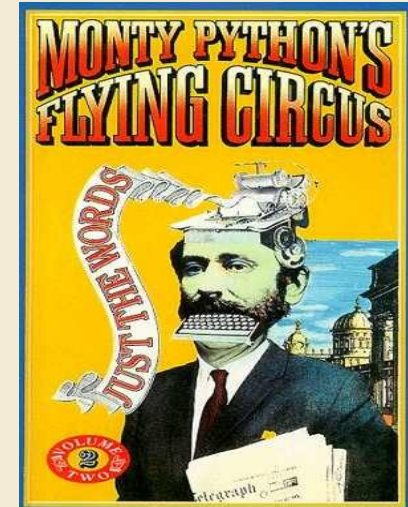




# Mas afinal o que significa Python ?



Colt Python .357 Magnum



Python 3





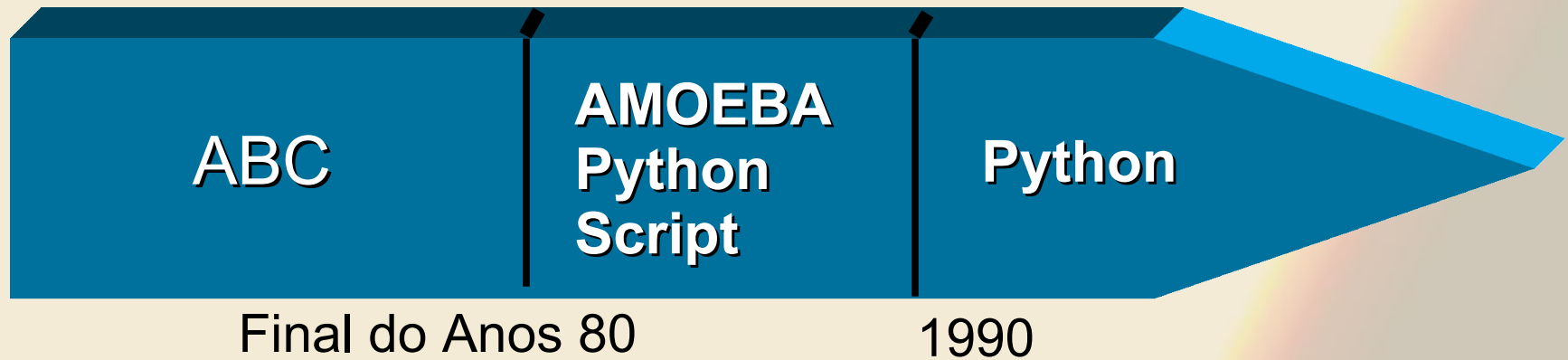
# Introdução

## Histórico

O **Python** foi criado em 1990 por **Guido van Rossum**, a partir de uma outra linguagem chamada **ABC**, que tinha como foco original usuários como físicos e engenheiros.



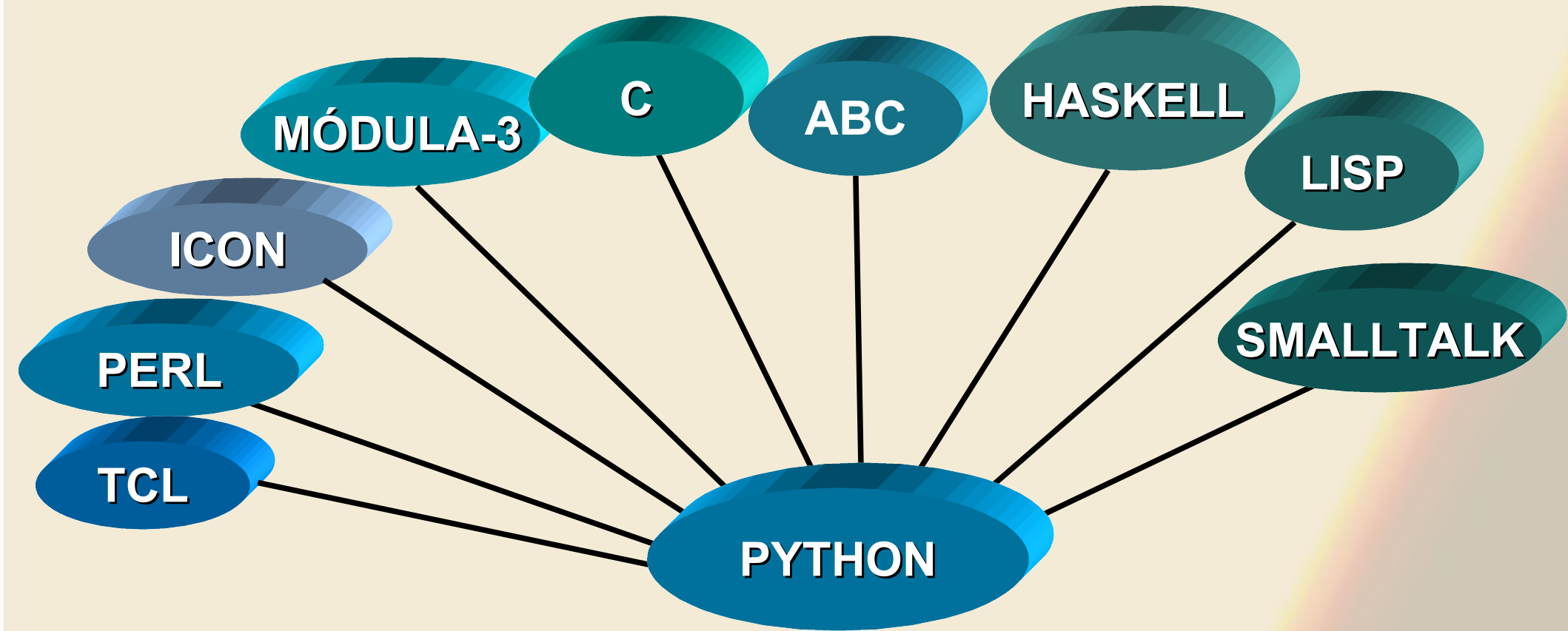
Segundo seu criador a linguagem foi concebida para ocupar o espaço que existia entre as linguagens C/C++ e o shell.







# DNA



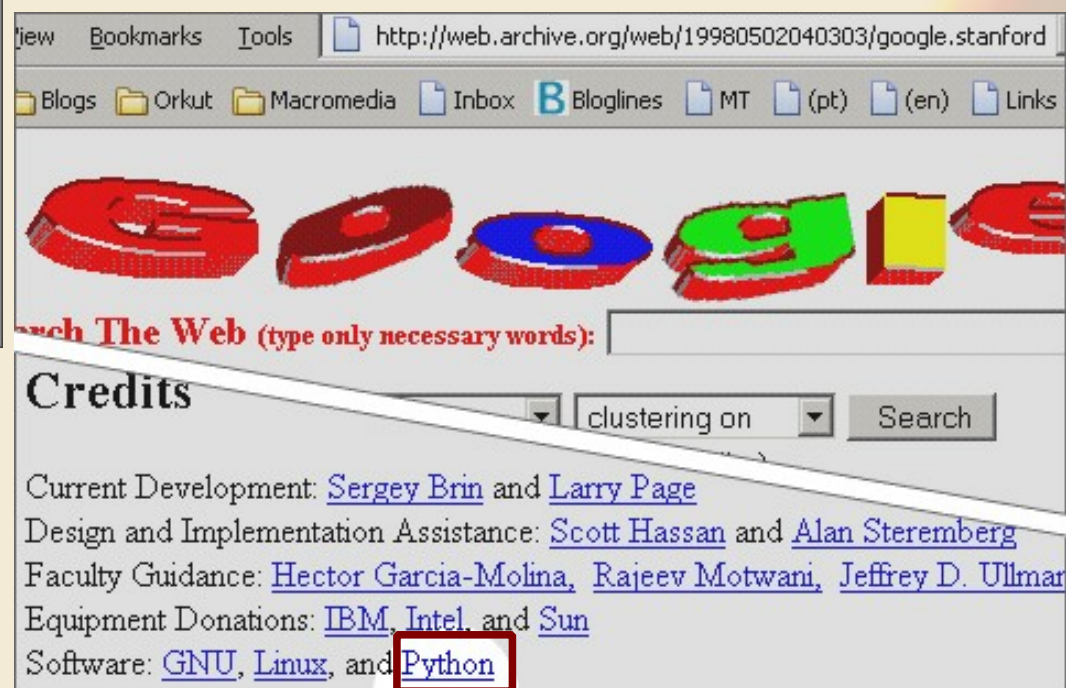
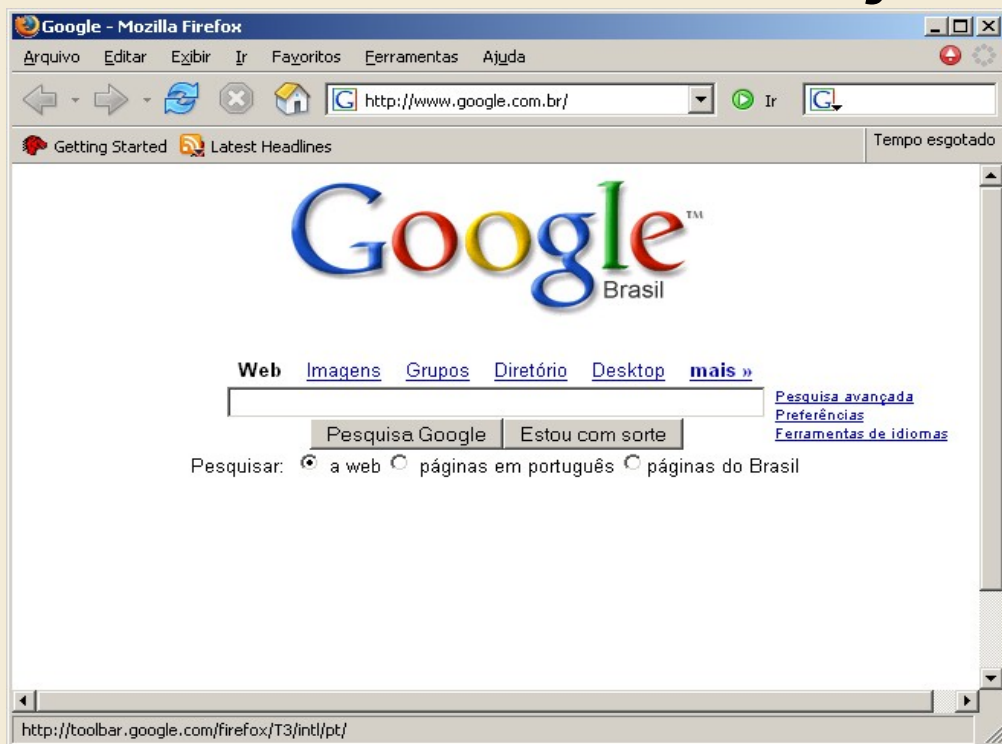
Wikipedia – linguagens que influenciaram o Python





# Você já usou Python ?

## Você já usou o Python ?







# Introdução

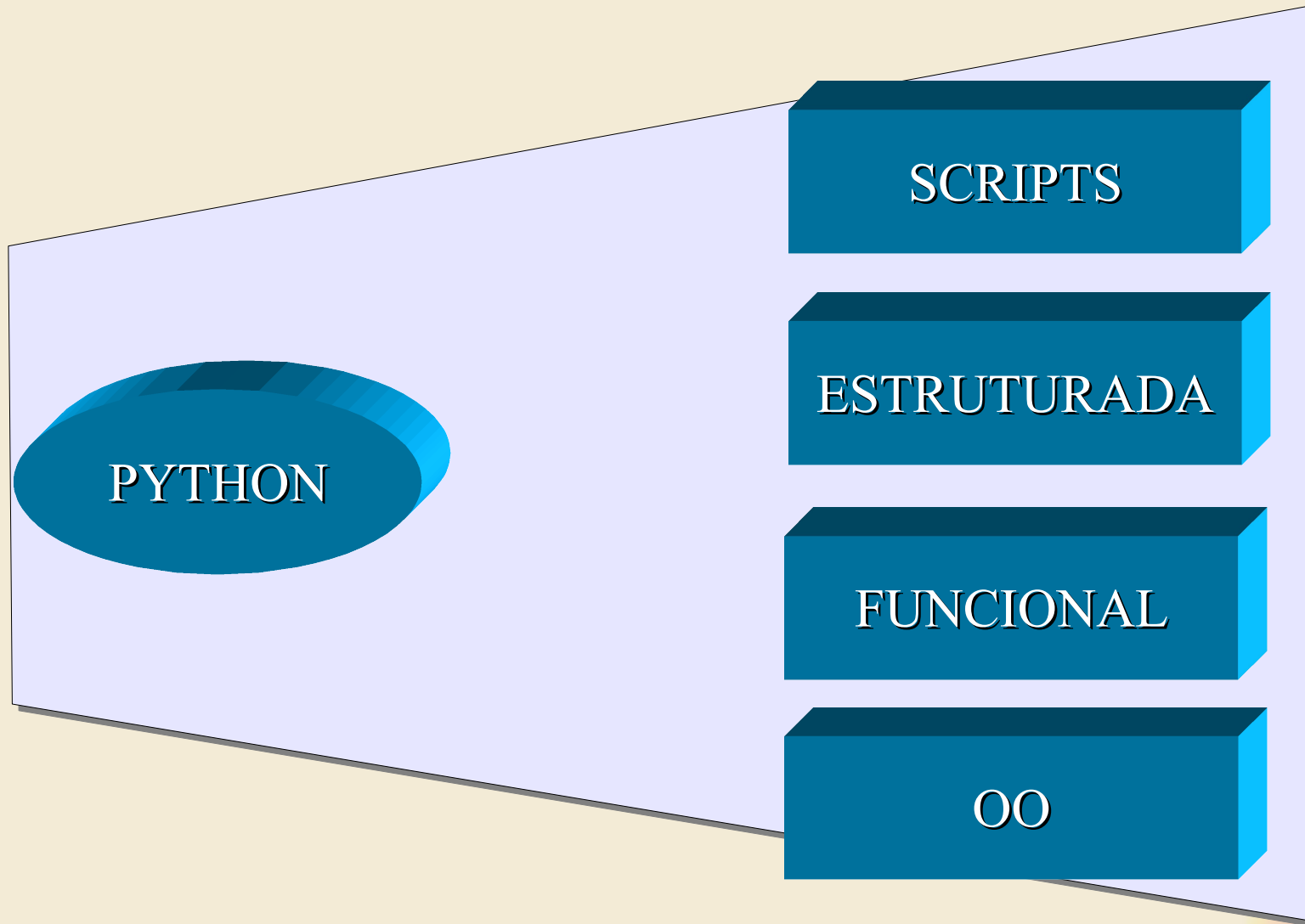
## Python:

- Fácil Aprendizado
- Fácil Manutenção
- Possui uma extensa biblioteca padrão
- Interpretada e interativa
- Possui uma poderosa estruturas de dados nativa
- Disponível com código aberto e livre
- Interface para outras Linguagens ( Lua, Java, C)
- Multi-plataforma
- É um canivete-suíço !!!





# Introdução



**Múltiplos paradigmas**



# Introdução

## Exemplo de um script

```
# Busca em uma página web o valor do dolar comercial
import urllib
import re

site = urllib.urlopen('http://economia.uol.com.br/cotacoes/').readlines()

for linha in site:
    if linha.find('paralelo (em R$)') > 0:
        valores = linha
        numeros = re.findall(r'[0-9]+(?:\.[0-9]+)?', linha)
        print 'Dolar Paralelo'
        print 'Compra  %s' %numeros[0]
        print 'Venda   %s' %numeros[1]
        print 'Variacao %s' %numeros[2]
```

Dolar Paralelo	
Compra	2,137
Venda	2,139
Variacao	1,97



# Introdução

## Estruturando o exemplo do script

```
# -*- coding: utf-8 -*-  
# modulo finanças.py  
# Busca em uma página web o valor do dolar comercial  
import urllib,  
import re  
  
def extrai_cotacao(moeda,mercado='R$'):  
    """ função para retornar a cotacao de uma moeda em seu mercado  
    moeda - nome da moeda – mercado – tipo (ex.paralelo) """  
    site = urllib.urlopen('http://economia.uol.com.br/cotacoes/').readlines()  
    numeros = []  
    for linha in site:  
        if linha.find(moeda) > 0 and linha.find(mercado) > 0:  
            numeros = re.findall(r'[0-9]+(?:\.[0-9]+)?', linha)  
    return numeros
```

Euro - compra (2,842) venda (2,844)

```
# modulo cotacao.py  
from finanças import extrai_cotacao  
  
cotacao= extrai_cotacao('Euro') # para o euro na página não tinha mercado  
print "Euro - compra (%s) venda (%s)" % (cotacao[0],cotacao[1])
```



# Introdução

## Programação Funcional

```
def fatorial(num):  
    """fatorial de forma recursiva"""  
    if num == 0:  
        return 1  
    return num*fatorial(num-1)  
  
print fatorial(5)
```

```
def fatorial(num):  
    """fatorial de forma funcional"""  
    return reduce(lambda x,y:y*x,[1]+range(1,num+1))  
  
print fatorial(5)
```





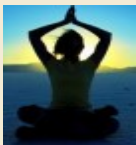
# Introdução

## Orientação à objetos

```
class Cesta:  
    def __init__(self, conteudo=None):  
        self.conteudo = conteudo or []  
    def inclui(self, elemento):  
        self.conteudo.append(elemento)  
    def abre_a_cesta(self):  
        resultado = ""  
        for elemento in self.conteudo:  
            resultado = resultado + " " + `elemento`  
        print "Contém: "+resultado
```

```
nova_cesta = cesta()  
nova_cesta.inclui('uva')  
nova_cesta.inclui('melao')  
nova_cesta.abre_a_cesta()
```

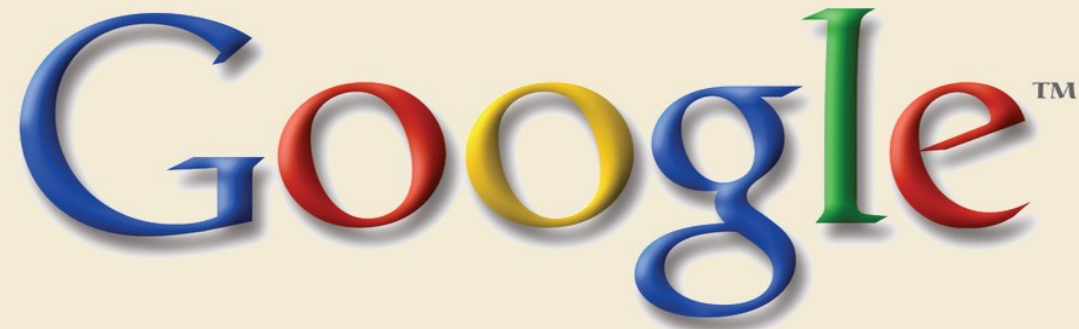
Contém: 'uva' 'melao'



Explicit is better than implicit (ZoP)



# Quem usa Python ?



"Python tem sido uma parte importante na Google desde o início e continua de acordo com o crescimento e a evolução do sistema. Hoje dúzias de engenheiros da Google usam Python, e estamos procurando por mais pessoas com habilidades na linguagem."

Disse Peter Norvig, diretor de busca de qualidade na Google, Inc.





# Quem usa Python ?



"Python realiza um papel chave no ciclo de produção. Sem ele um projeto do tamanho de Star Wars: Episode II seria complicado de ser realizado. Desde a renderização das pessoas em processamento batch até a composição, Python junta todas as coisas juntas," disse o Tommy Burnette, Diretor Técnico da Industrial Light & Magic.





# Quem usa Python ?



“Nós escolhemos Python porque **provê a máxima produtividade**, código **claro e fácil de manter**, **forte** and **extensível** (e crescente !) **bibliotecas**, e excelente capacidade de **integração** com outras aplicações em qualquer plataforma.

**Python atingiu ou excedeu todas as necessidades que nós tínhamos,**" disse Steve Waterbury, Software Group Leader, NASA STEP Testbed.



# Quem usa Python ?



“ O Serpro criou uma fábrica virtual para desenvolver **portais** para o governo federal. As ferramentas **ZOPE** e **PLONE** feitas em **PYTHON**, foram selecionadas para o projeto, representaram uma economia inicial de **R\$ 10 milhões** para a Empresa em custo de aquisição de softwares proprietários além de um aumento visível de **produtividade**.”

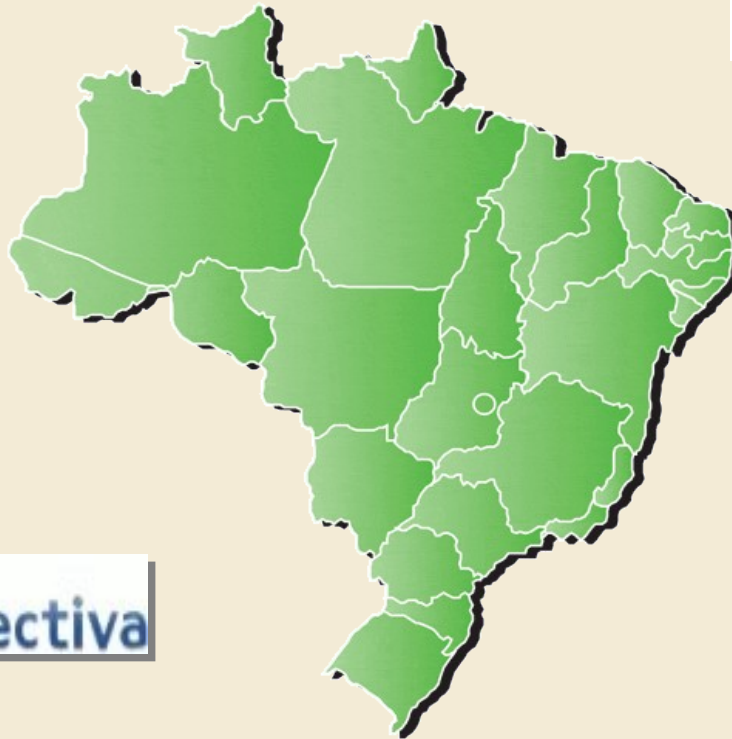
**Sérgio Borba Cangiano – Diretor do SERPRO**







# Quem usa Python ?

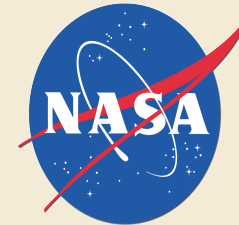




# Quem usa Python ?

## Utilizado pelas seguintes Empresas :

- Nasa, United Space Alliance
- Google, Yahoo , YouTube
- Nokia, Nortel
- RedHat, Gentoo
- Apple
- Disney
- Philips
- Canonical
- Los Alamos National Laboratory
- ILM - Industrial Ligth & Magic
- SERPRO, PETROBRAS, GLOBO
- Interlegis

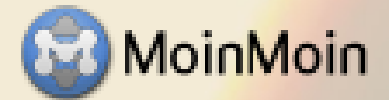




# Aplicações com Python

## Encontramos o Python no :

- Blender, Maya
- OpenOffice
- Zope/Plone
- BitTorrent
- MoinMoin
- Mailman
- Chandler
- Gimp
- Gnumeric
- Anaconda, Portage, Yum
- Trac

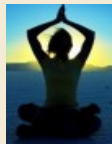




# Por que usar o Python ?

Código é mais vezes lido do que escrito !

Clareza na linguagem é fundamental para o aprendizado e para a manutenção do código.



Readability counts ! (ZoP)

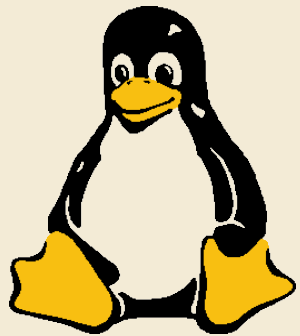




# Por que usar o Python ?

## MULTIPLATAFORMA

- Alta portabilidade
- Presente em 99,99% das distribuições LINUX
- Alta integração com Windows COM
- Iron Python
- PDA's, Celulares e tablets



**UNIX**







# Por que usar o Python ?

## Implementações :

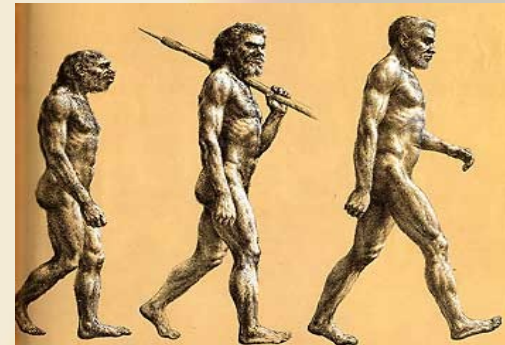
- CPython – Escrita em C
- Jython (máquina virtual Java)
- IronPython .NET (Microsoft)
- PyPy – Python escrito em Python
- PyS60 – Nokia



# Por que Python ?


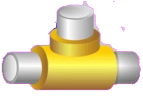







## MATURIDADE

- 1990 - Nascimento
- 1994 - Python ?? Que isso ?
- 1997 – Mas ninguém usa isso.
- 1999 – Onde podemos encontrar programadores?
- 2004 – Infoworld – 6ª linguagem
- 2005 - Python na Nokia
- 2007 – A linguagem que mais cresceu (TIOBE)
- 2008 – Google App Engine
- 2009 – Melhor Linguagem Open Source  
(Linux New Media Award – CEBIT 2009)



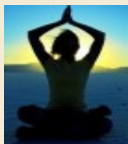
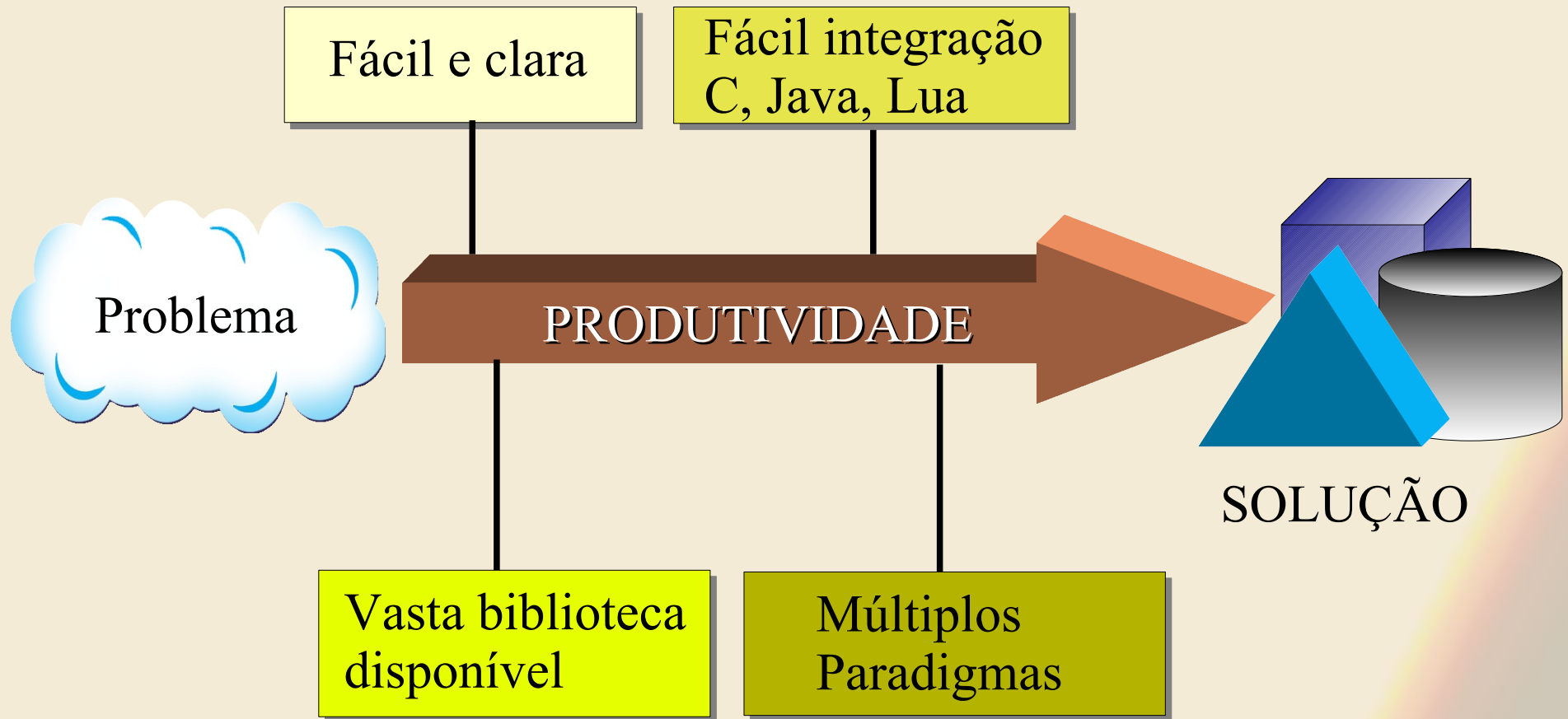


# Por que Python ?

	Bancos de Dados : ODBC, MySQL, Postgres, Oracle, SQLServer, sqlite, gadfly, ZODB/Durus, PyDO, Metakit
	Rede : Twisted, asyncore, httplib, SimpleHTTPServer, urllib, ftplib, poplib, smtpplib, telnetlib
	GUI : Tkinter, wxPython, PyGTK, PyQt, PyKDE, Pythonwin
	Ciência : NumPY, SciPy, BioPython, AstroPy
	Processamento. de Imagens : PIL, PythonMagick, Gimp-python
	XML : PyXML, 4Suite, ElementTree, RDFLib, Cwm
	Web : ZOPE, CherryPy, Webware, Quixote, PSP, mod_python, Nevow, Django, TurboGears, Pylons, Web2Py
	IDE : Emacs, vi, idle, PyDev (Eclipse), SPE, Pythonwin, Komodo, BlackAdder, WingIDE, PyScripter, NetBeans, Boa Constructor
	Jogos : Pygame, Pykra, Panda3D, Blender3D,



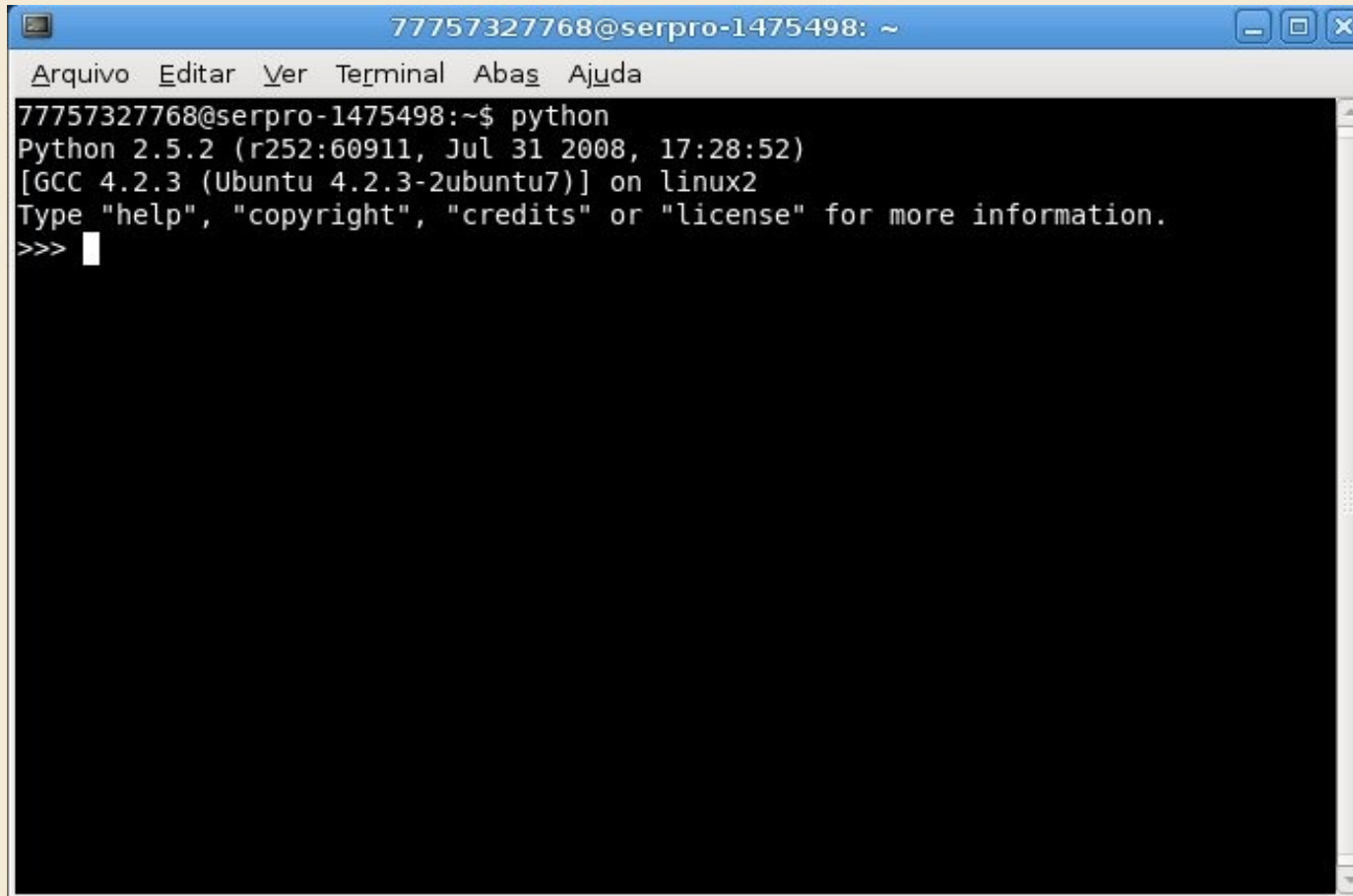
# Porque Python



“Simple is better than complex” (ZoP)



# Porque Python

A screenshot of a terminal window titled '77757327768@serpro-1475498: ~'. The window has a menu bar with 'Arquivo', 'Editar', 'Ver', 'Terminal', 'Abas', and 'Ajuda'. The terminal output shows the command 'python' being executed, followed by the Python 2.5.2 startup banner: 'Python 2.5.2 (r252:60911, Jul 31 2008, 17:28:52) [GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7)] on linux2'. It then prompts the user to 'Type "help", "copyright", "credits" or "license" for more information.' and ends with the interactive prompt '>>>' followed by a cursor.

```
77757327768@serpro-1475498: ~
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
77757327768@serpro-1475498:~$ python
Python 2.5.2 (r252:60911, Jul 31 2008, 17:28:52)
[GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```







# Testemunhos



## Bruce Eckel

Autor de Best Sellers:

“Thinking in C++”

“Thinking in Java”

**Frases Inspiradoras:**

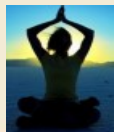
- ” Python: it fits your brain”
- “ Life is Better Without Braces”
- “ Life is short ! You need Python !
- “ Python: Batteries Included”

## Palestra: Why I love Python

“Python foi feito pra você “

“Python me ajuda a focar nos meus conceitos em vez de ficar brigando com a linguagem.”

“Eu não preciso digitar muito. Mas o que eu digito é o certo “



“Special cases aren't special enough to break the rules.” (ZoP)





# Testemunhos



## Eric Raymond

**Autor:** A Catedral e o Bazar

mantém o Jargon File (Hacker's Dictionary)  
Contribuidor do GNU Emacs, Linux, Fetchmail

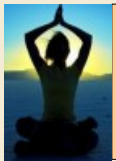
Frases famosas:

"Havendo olhos suficientes, todos os erros são óbvios"

### O que Eric pensa do Python:

“Entre todas as linguagens que aprendi, Python é a que menos interfere entre mim e o problema. É a mais efetiva para traduzir pensamentos em ações.”

“Python amplifica seu cérebro.”

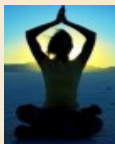


**“If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.” (ZoP)**





# Python foi feito para você !



**“Simple is better than complex” (ZoP)**



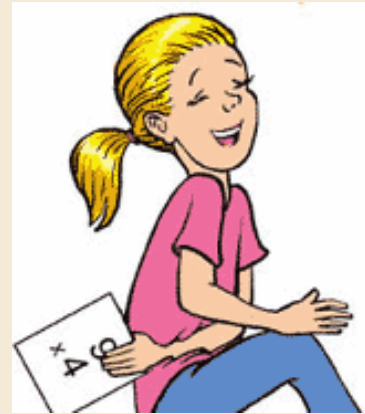
# In love with Python



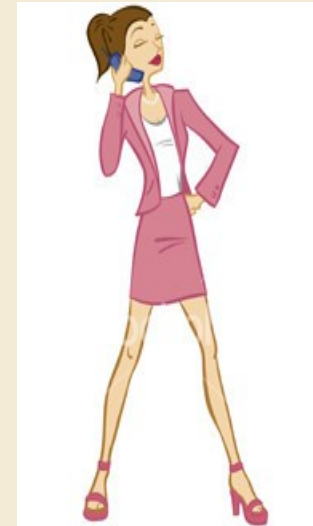
Fortran Girl



Cobol Girl



Prolog Girl



Java Girl



Python Girl



**“Beautiful is better than ugly.” (ZoP)**



# Por onde começar

# Python para desenvolvedores - Luiz Eduardo Borges



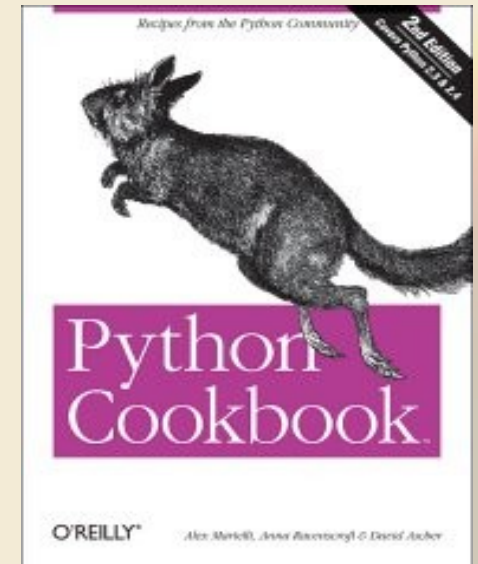
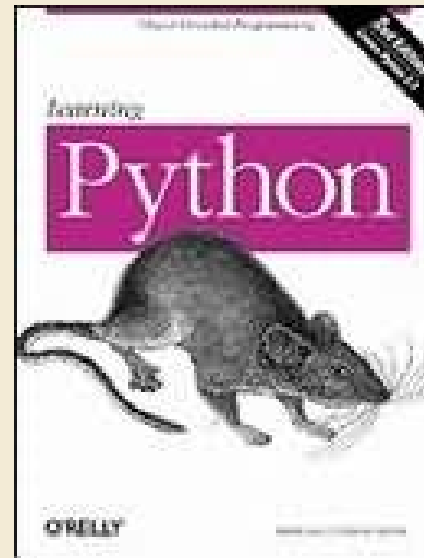
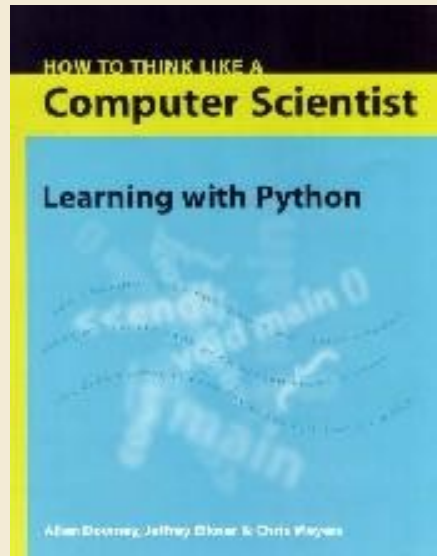
Livre para Download  
Licença Creative Commons  
<http://ark4n.wordpress.com/>



# Por onde começar

[www.python.org.br](http://www.python.org.br)

Documentação  
How to's  
Cookbook



Dive into Python  
Mark Pilgrim  
Livre pra download

How to think like a  
computer scientist  
Allen Downey, Jeff Elkner  
and Chris Meyers  
Livre pra download

Aprendendo Python  
David Ascher e Mark Lutz  
Python Cookbook  
Alex Martelli, Anna  
Ravenscroft e David Ascher







# Links Importantes

Site Python - [python.org](http://python.org)

Site PythonBrasil - [www.python.org.br](http://www.python.org.br)

Site PythonRio - [www.pythonrio.org](http://www.pythonrio.org)

APyB - [associacao.pythonbrasil.org/](http://associacao.pythonbrasil.org/)

Lista PythonRio - [br.groups.yahoo.com/group/pythonrio/](http://br.groups.yahoo.com/group/pythonrio/)

Lista PythonBrasil - [br.groups.yahoo.com/group/python-brasil/](http://br.groups.yahoo.com/group/python-brasil/)

PyconBrasil 2008 - [pyconbrasil.com.br/](http://pyconbrasil.com.br/)

PyConBrasil 2009 - [associacao.pythonbrasil.org/associacao/imprensa/noticias/caxias-do-sul-2009](http://associacao.pythonbrasil.org/associacao/imprensa/noticias/caxias-do-sul-2009)





# Zen of Python

## O Zen do Python, por Tim Peters

Bonito é melhor que feio.

Explícito é melhor que implícito.

Simples é melhor que complexo.

Complexo é melhor que complicado.

Linear é melhor do que aninhado.

Esperso é melhor que denso.

Legibilidade conta.

Casos especiais não são especiais o bastante para quebrar as regras.

Ainda que praticidade vença a pureza.

Erros nunca devem passar silenciosamente.

A menos que sejam explicitamente silenciados.

Diante da ambiguidade, recuse a tentação de adivinhar.

Deveria haver um — e preferencialmente só um — modo óbvio para fazer algo.

Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.

Agora é melhor que nunca.

Embora nunca frequentemente seja melhor que \*já\*.

Se a implementação é difícil de explicar, é uma má ideia

Se a implementação é fácil de explicar, pode ser uma boa ideia

Namespaces são uma grande idéia — vamos ter mais dessas!



# Contatos

```
Import Apresentacao
```

```
try:
```

```
    if Apresentacao.Boa:
```

```
        print "OBRIGADO !!!"
```

```
    else:
```

```
        print "Heeerr, Obrigado mesmo assim"
```

```
except PalestraRuim:
```

```
    print "SAI CORRENDO !!!!"
```

```
palestrante = "Giuseppe Romagnoli"
```

```
email = "giuseppe.romagnoli@gmail.com"
```



O importante é saber que os caminhos existem