

# Programação para Jornalistas

Fernando Masanori

<https://about.me/fmasanori>

# Motivação

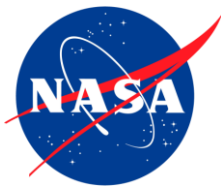
- Quanto o Brasil gastou na Copa do Mundo 2014
- Quantas escolas existem, em funcionamento, sem água, luz e esgoto
- Como acessar dados das eleições americanas 2016

# Quem sou eu

- Criador do [Python para Zumbis](#)
- 107 palestras e minicursos em 11 países



Worshop de programação para mulheres, Django Girls, na Namíbia



# Python



“Nós queremos que a nossa comunidade seja mais diversificada: quem quer que você seja, e qualquer que seja sua origem, você é bem vindo”

<http://www.python.org/psf/diversity/>

“Nós criamos esta declaração, pois acreditamos que uma comunidade Python diversificada é mais forte e mais vibrante. A comunidade diversificada, onde as pessoas se tratam com respeito, tem mais contribuidores e mais fontes de ideias.”

<http://www.python.org/psf/diversity/>

# Números

- 2014: 1/3 palestrantes PyCon mulheres
- 2015: 1/3 palestrantes PyCon mulheres
- 2016: 1/3 palestrantes PyCon mulheres
- 2015: 1/3 palestrantes DjangoCon mulheres
- 2016: 1/2 palestrantes DjangoCon Europe mulheres
- 2015: 7 mulheres, 4 homens (nova diretoria PSF)

# Modo interativo do Python

- Existem duas versões: Python 2 e Python 3
- Neste curso iremos usar Python 3 e o IDLE para editar os nossos programas



# Primeiro programa

```
>>> print ("Primeira mensagem!")  
Primeira mensagem!  
>>>
```

- Este programa possui apenas uma linha de código
- Observe que as aspas não aparecem na saída
- Precisamos marcar ou limitar o início e o fim de nossas mensagens com um símbolo, nesse caso, as aspas

# Primeira mensagem de erro

```
>>> Print ("Primeira mensagem!")  
Traceback (most recent call last):  
  File "<pyshell#39>", line 1, in <module>  
    Print ("Primeira mensagem!")  
NameError: name 'Print' is not defined
```

```
>>>
```

- Letras maiúsculas e minúsculas são diferentes
- Você reparou que Print não está na cor roxa?

# Primeira mensagem de erro

```
>>> print (Primeira mensagem)
SyntaxError: invalid syntax (<pyshell#7>, line 1)
>>>
```

- Se não utilizarmos aspas, o computador interpretará nossa mensagem como um comando da linguagem Python, gerando um erro de sintaxe
- Você reparou que a mensagem não está na cor verde?

# Primeira mensagem de erro

```
>>> print "Primeira mensagem!"  
SyntaxError: invalid syntax (<pyshell#7>, line 1)  
>>>
```

- Na versão do Python que usamos os parênteses não são opcionais no print

# Primeira mensagem de erro



```
>>> print ("Primeira mensagem!")
```

```
SyntaxError: unexpected indent (<pyshell#9>, line 1)
```

```
>>>
```

- Os espaços iniciais possuem um significado em Python que veremos mais adiante

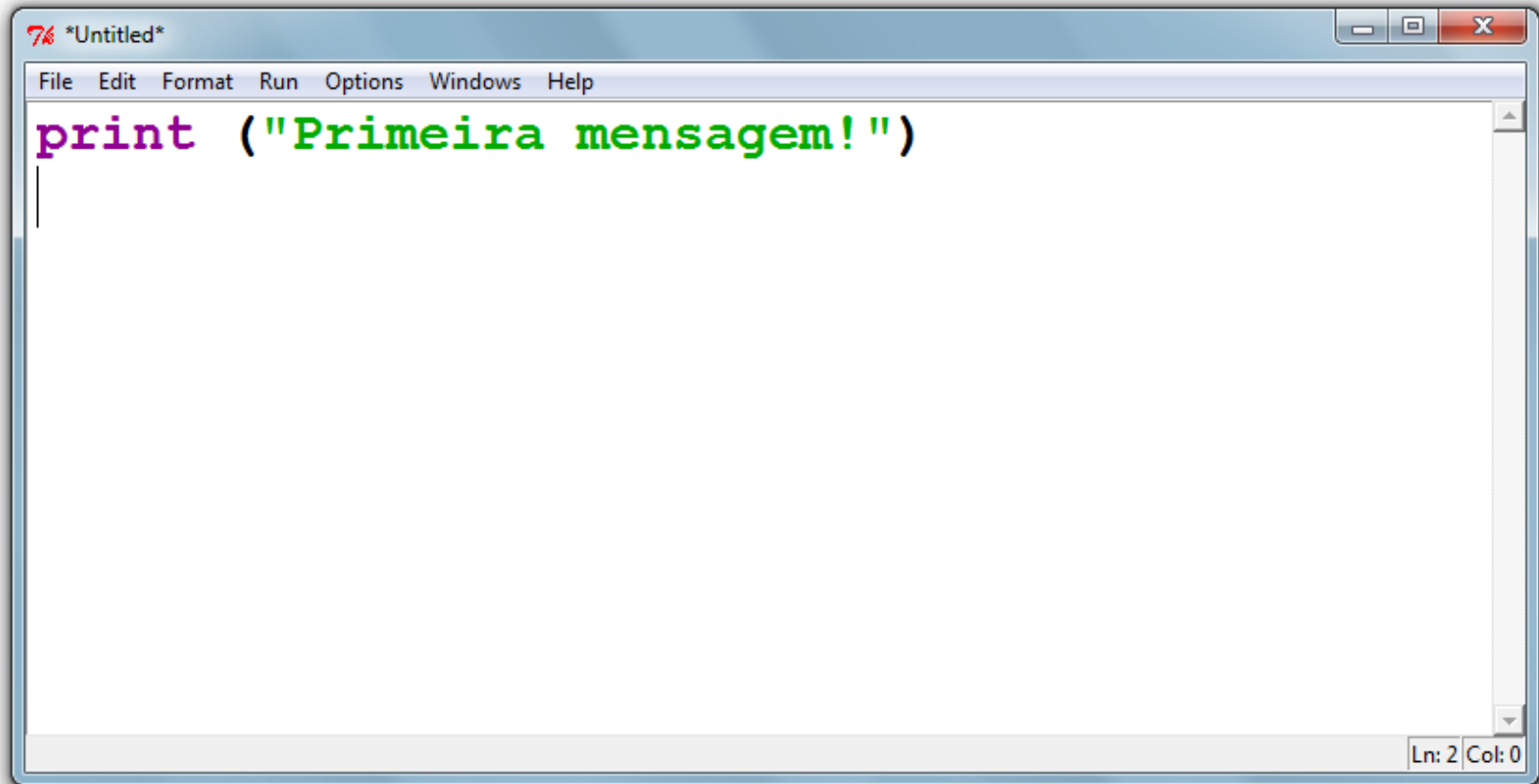
# Interpretador Python

- O interpretador é um programa que aceita comandos escritos em Python e os executa, linha a linha
- Sem o interpretador, nossos programas não podem ser executados, sendo considerados apenas um texto
- O interpretador verifica se escrevemos corretamente o programa, mostrando mensagens de erro caso haja algum problema

# Interpretador Python

- Existem dois modos do interpretador Python: modo interativo e modo de edição
- Usamos nos exemplos anteriores o modo interativo
- Uma vantagem do modo interativo é poder testar comandos e obter a resposta instantaneamente

# Modo edição

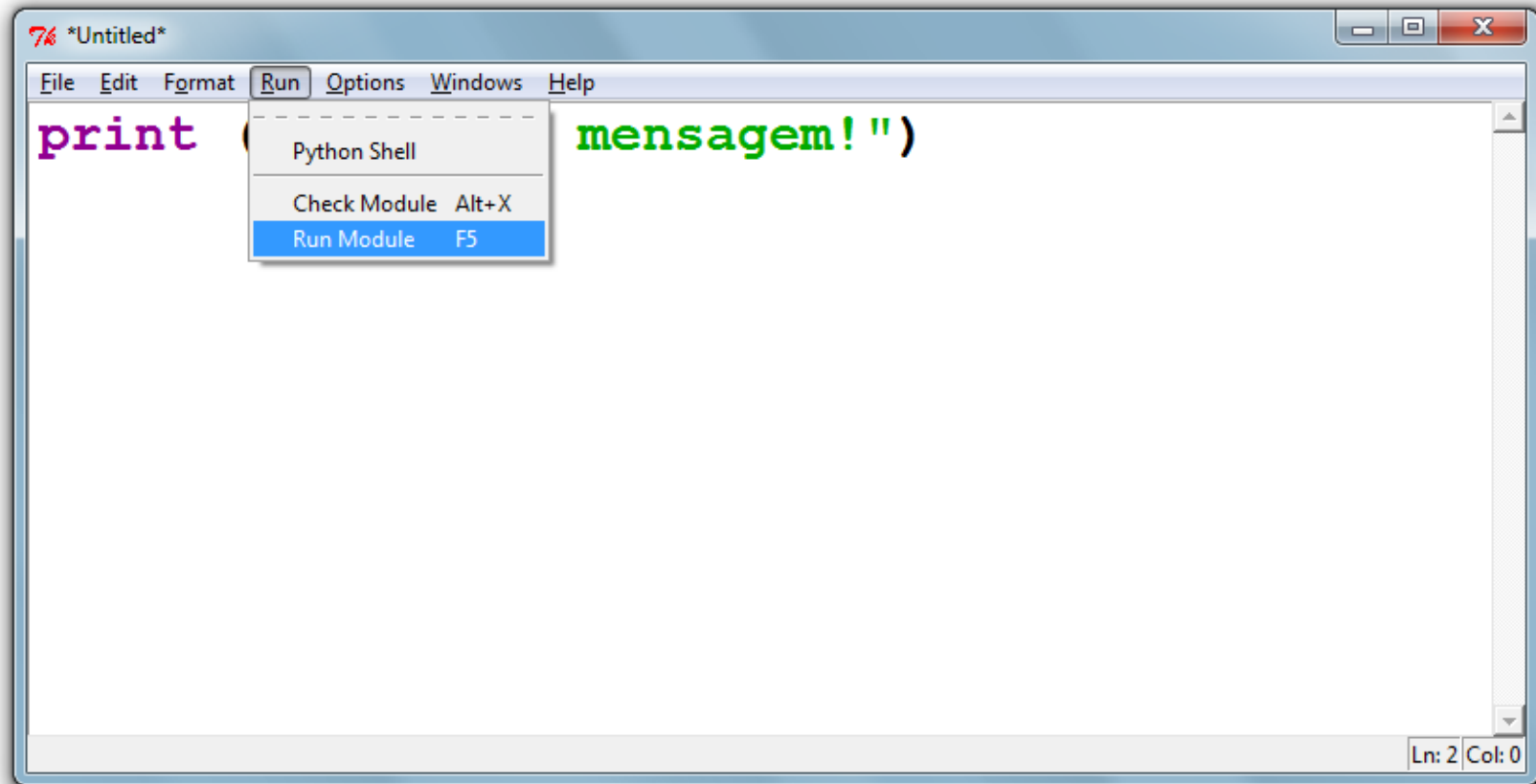


```
print ("Primeira mensagem!")
```

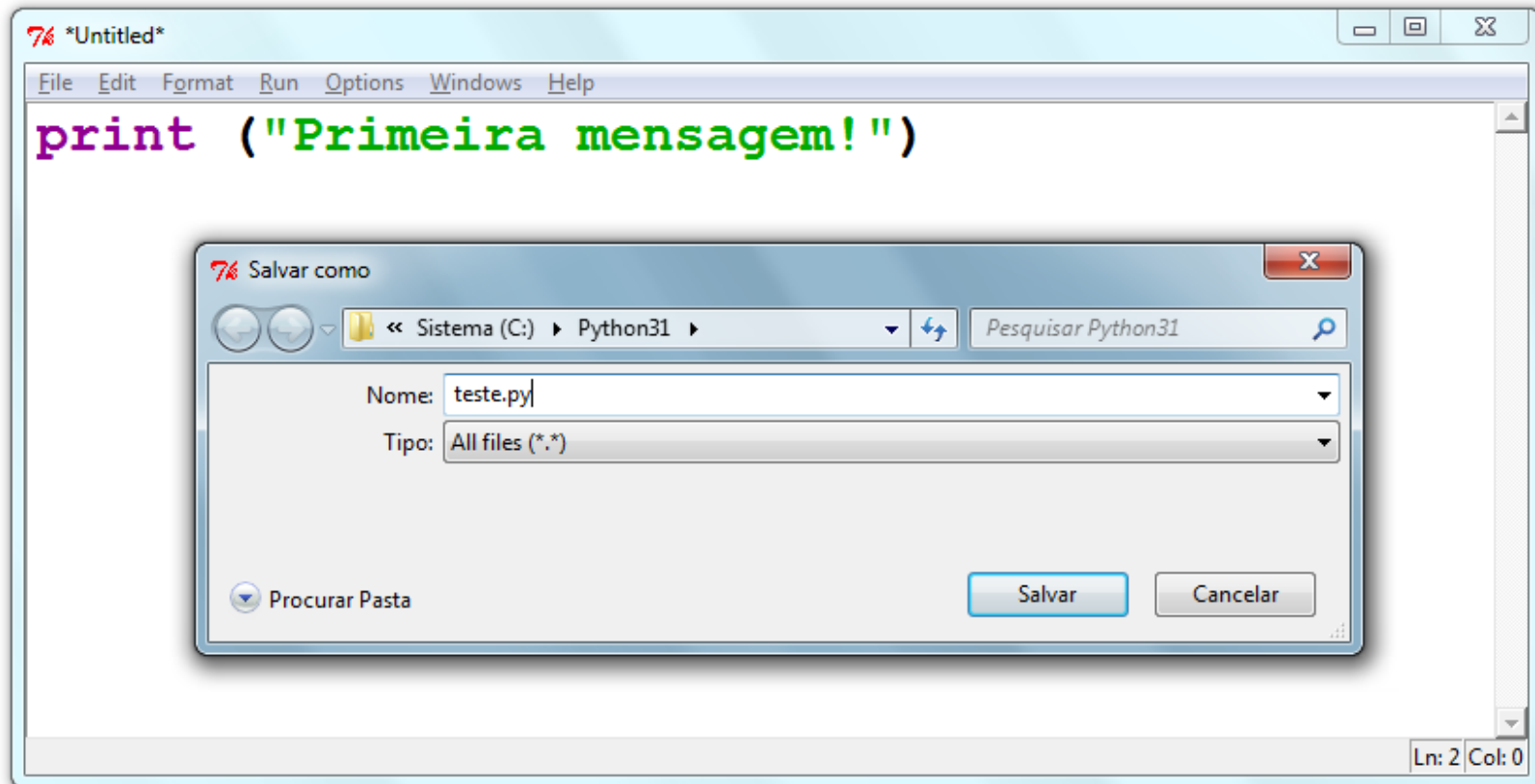
Observe que não aconteceu nada ao digitar enter no final da linha  
É necessário “rodar” o programa no modo edição (Run Module F5)



# Rodar o programa



# Salvar o programa



Sempre use a extensão “.py” para não perder as cores do seu programa

# Mesmo resultado!

The screenshot shows a Python Shell window titled "Python Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area contains several red prompt characters ">>>". A smaller window titled "teste.py - C:\Python31\teste.py" is overlaid on top, showing a single line of Python code: `print ("Primeira mensagem!")`. Below the code window, the text "==== RESTART ===" is visible. The output of the script is displayed as "Primeira mensagem!". At the bottom of the main shell window, a blue arrow points to the left, and the status bar shows "Ln: 30 Col: 0".

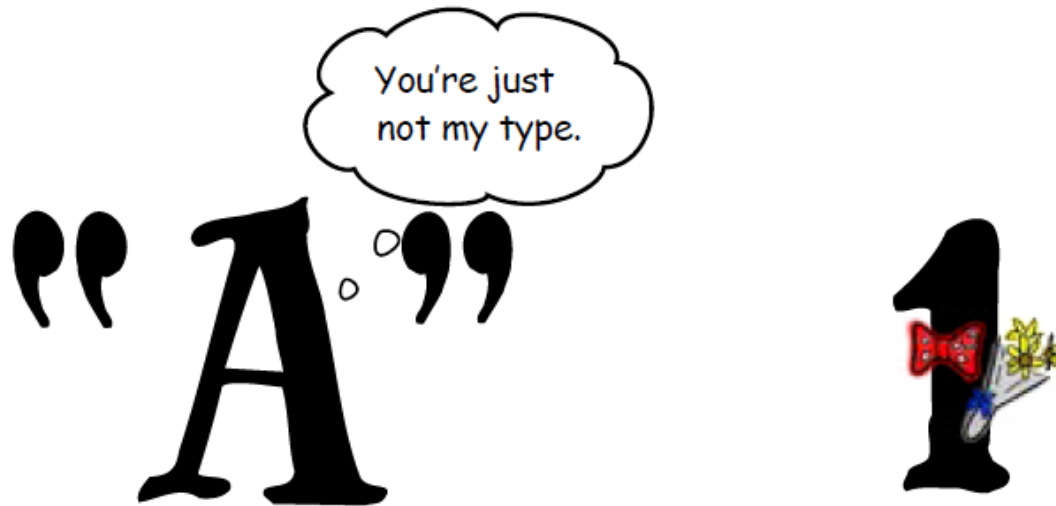
# Variáveis e atribuição

- Variáveis são utilizadas para armazenar valores e para dar nome a uma área da memória do computador
- O símbolo para atribuição é o igual (=)

```
>>> nome = "Fernando"  
>>> print (nome)  
Fernando  
>>> resposta = 42  
>>> print (resposta)  
42
```



# Tipos de variáveis



Strings são diferentes de números

# Tipos de variáveis

- O conteúdo de uma variável possui um tipo
- O tipo define a natureza dos dados que a variável armazena
- Os tipos mais comuns são inteiros, números em ponto flutuante e strings (texto)
- Além de poder armazenar números e letras, as variáveis em Python também armazenam valores como True e False

# Variáveis numéricas

- Inteiros não possuem casas decimais: 42, -7
- O tipo inteiro em Python é chamado int
- Números em ponto flutuante possuem casa decimal: 1.0, 3.1415, 1234.56
- Note que 1.0, mesmo tendo zero na parte decimal, é um número em ponto flutuante
- O tipo ponto flutuante em Python é chamado float

# Listas

- Edifício de apartamentos

```
edifício_térreo    = "Família Souza"  
edifício_1o_andar = "Família Brito"  
edifício_2o_andar = "Sr Jorge"  
edifício_3o_andar = "Família Tanaka"
```



# Edifício

- Podemos associar o térreo ao andar zero, o primeiro é o andar 1 e assim por diante

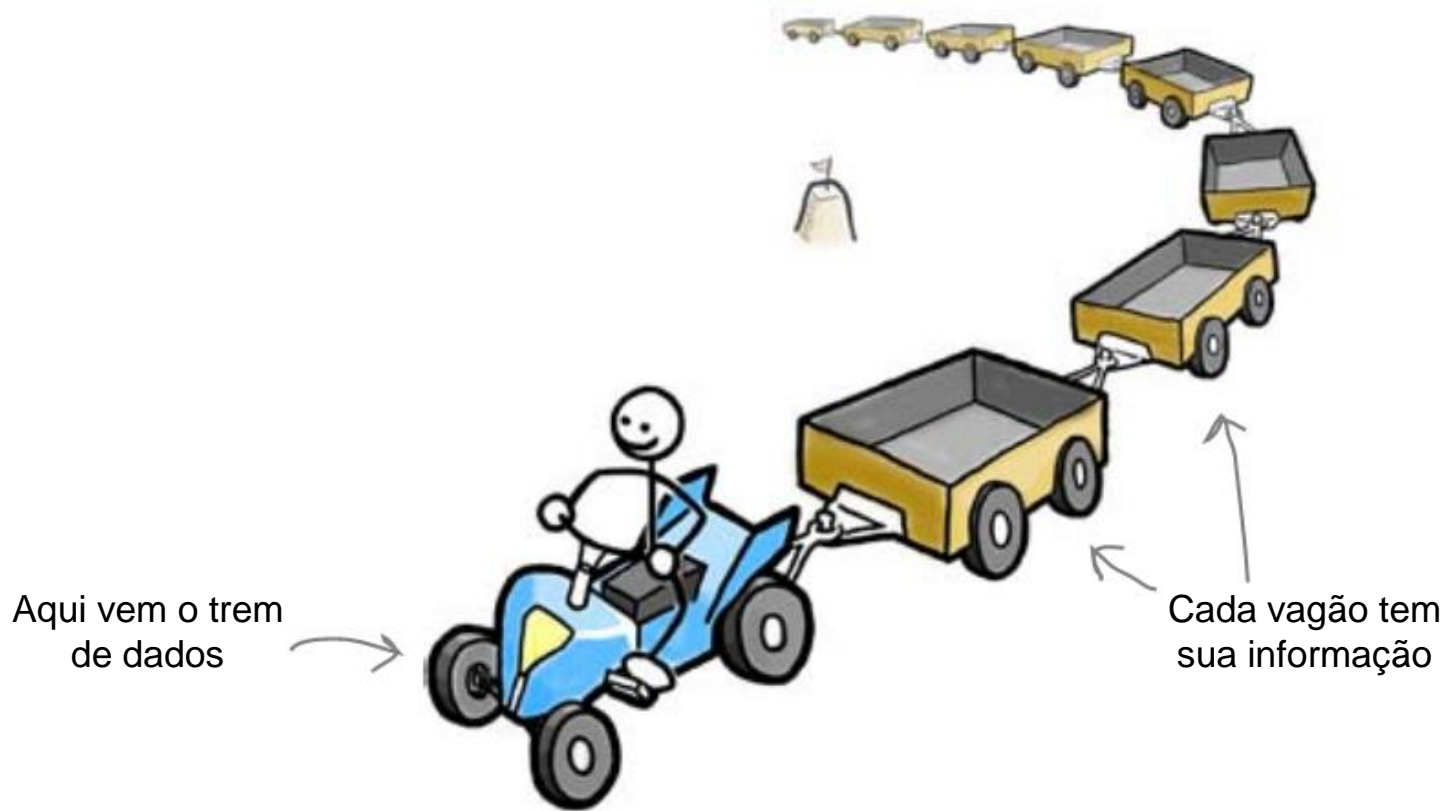
```
edifício = ["Família Souza",  
            "Família Brito",  
            "Sr Jorge",  
            "Família Tanaka"]  
  
print (edifício[0])  
print (edifício[1])  
print (edifício[2])  
print (edifício[3])
```

```
>>>
```

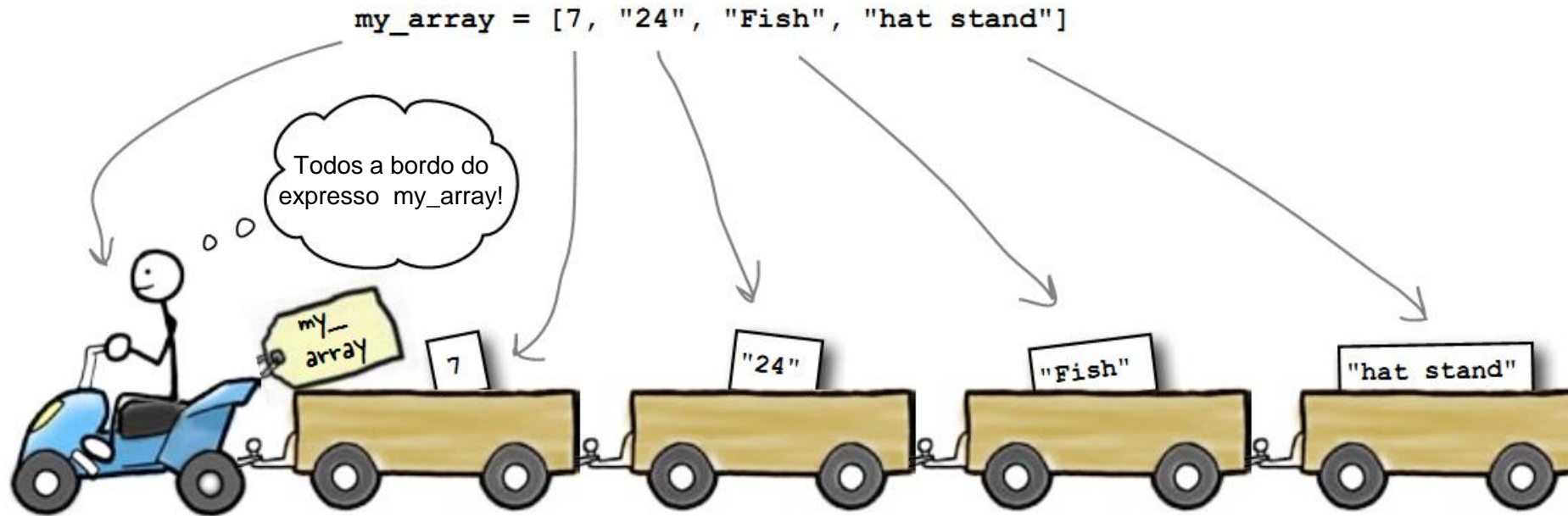
```
Família Souza  
Família Brito  
Sr Jorge  
Família Tanaka
```



# Trem de dados



# Trem de dados



O trem de dados `my_array` é uma única variável

# Operadores relacionais

Operador	Operação	Símbolo matemático
==	igual	=
>	maior que	>
<	menor que	<
!=	diferente	≠
>=	maior ou igual	≥
<=	menor ou igual	≤

Observe que o operador de igualdade são dois iguais (==)

# Exemplo

- Podemos usar operadores relacionais para inicializar variáveis do tipo lógico

```
>>> nota = 8
>>> média = 6
>>> aprovado = nota > média
>>> print (aprovado)
True
```

# Operadores lógicos: and, or, not

- A condição para empréstimo de compra de uma moto é salário maior que R\$ 1.000,00 e idade acima de 18 anos. Verificar se o José pode pegar o empréstimo

```
>>> salário = 500.0
>>> idade = 20
>>> salário > 1000 and idade > 18
False
```

# Entrada de Dados

- Até agora nossos programas trabalharam com valores conhecidos
- Vamos começar a pegar os valores durante a execução dos programas e usar mais o modo de edição

```
nome = input ('nome: ')\nprint ('Olá', nome)\n>>>\nnome: Fernando\nOlá Fernando
```



# Conversão de Dados

- A função input retorna apenas strings
- Usamos int() para converter um valor para inteiro e float() para ponto flutuante

```
idade = int(input('idade: '))  
pi = float(input('Pi: '))  
print ('idade:', idade)  
print ('Pi:', pi)
```

```
>>>
```

```
idade: 42
```

```
Pi: 3.1415
```

```
idade: 42
```

```
Pi: 3.1415
```





# Condições

```
print ("Bem vindo ao meu programa!")  
print ("Volte sempre!")
```

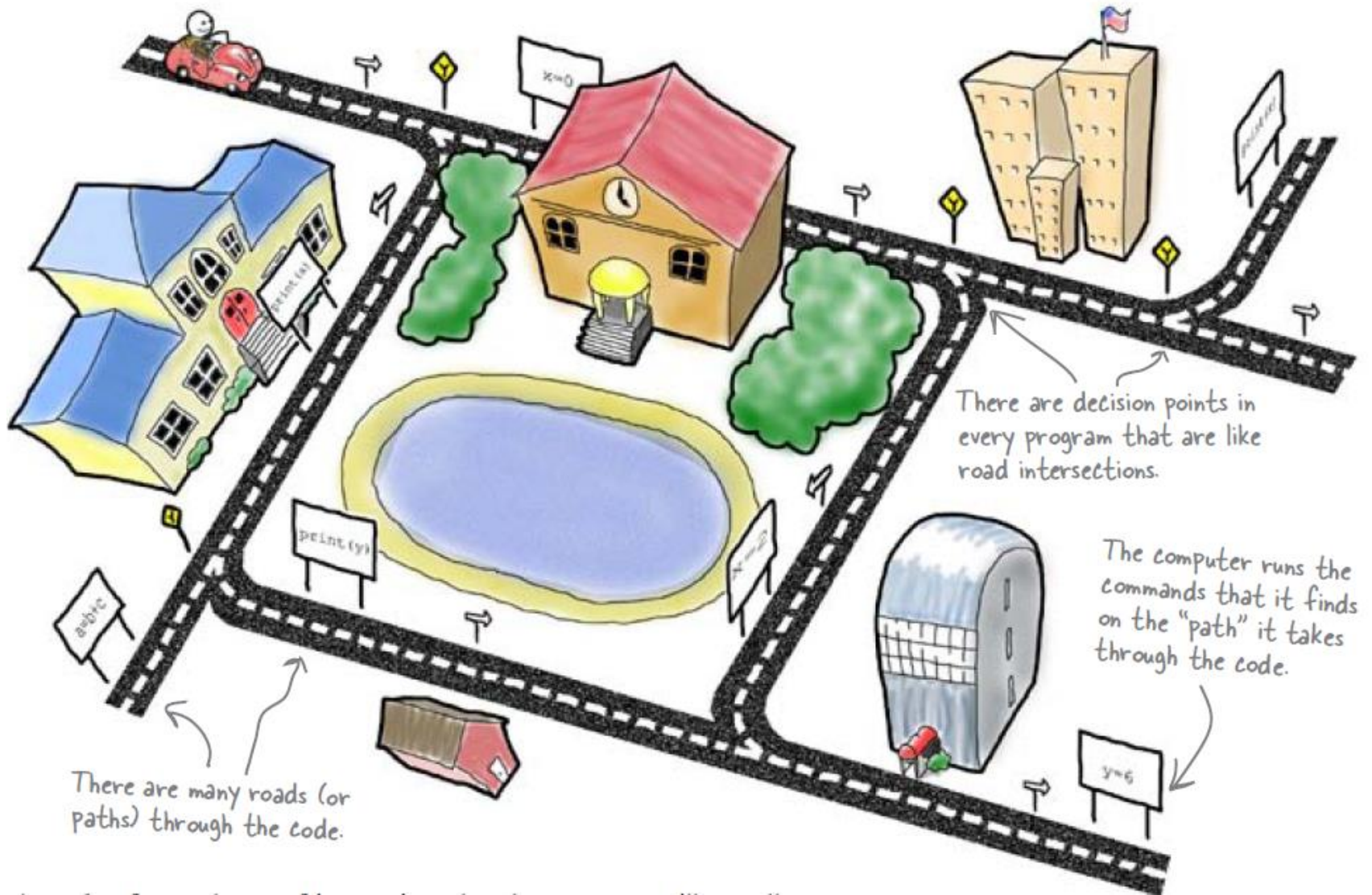


Seus programas nem sempre serão seqüências simples de comandos

# Condições

- “Executar ou não executar? Eis a questão...”
- Em geral não executo todas as linhas do programa
- Passar pelas linhas de um programa é como andar de carro numa cidade
- Existem pontos onde decidimos qual caminho escolher

# Condições



# if

- Ler dois valores inteiros e imprimir o maior deles

```
a = int(input("Primeiro valor: "))
b = int(input("Segundo valor: "))
if a > b:
    print ("O primeiro número é o maior!")
if b > a:
    print ("O segundo número é o maior!")
```

# if

- Verificar se um carro é novo ou velho
- Se o carro tiver pelo menos três anos é novo

```
idade = int(input("Digite a idade de seu carro: "))  
if idade <= 3:  
    print("Seu carro é novo")  
if idade > 3:  
    print("Seu carro é velho")
```

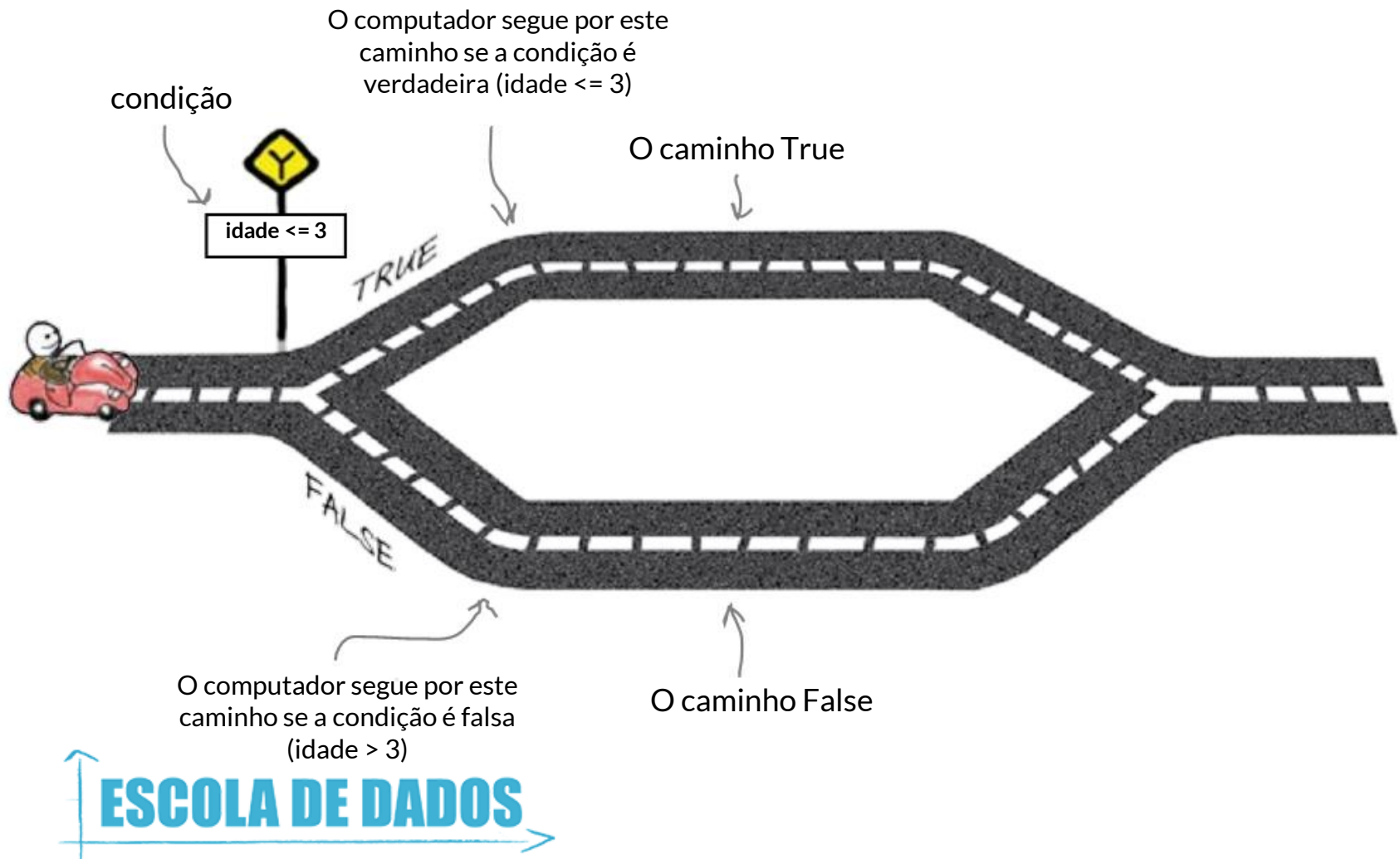
# else

- O que fazer quando a condição do if é falsa?
- Os dois códigos abaixo fazem a mesma coisa:

```
idade = int(input("Digite a idade de seu carro: "))  
if idade <= 3:  
    print("Seu carro é novo")  
if idade > 3:  
    print("Seu carro é velho")
```

```
idade = int(input("Digite a idade de seu carro: "))  
if idade <= 3:  
    print("Seu carro é novo")  
else:  
    print("Seu carro é velho")
```

# if / else







↑  
**ESCOLA DE DADOS** →



# Imprimindo de 1 a 3

- Forma simples

```
print (1)  
print (2)  
print (3)
```

- Usando uma variável

```
x = 1  
print (x)  
x = 2  
print (x)  
x = 3  
print (x)
```



# Imprimindo de 1 a 3

- Incrementando a variável

```
x = 1
print (x)
x = x + 1
print (x)
x = x + 1
print (x)
```

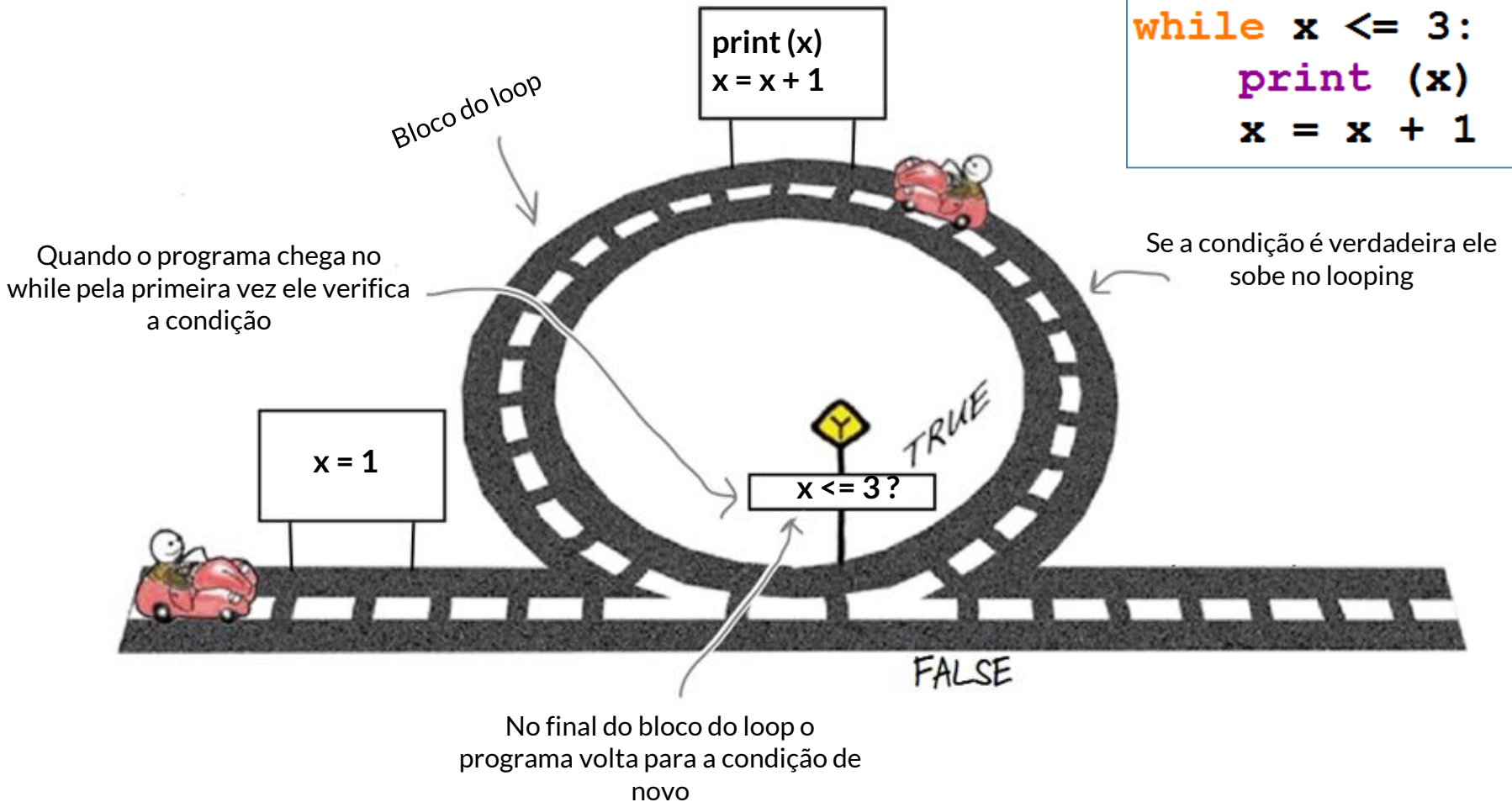
- Usando while

```
x = 1
while x <= 3:
    print (x)
    x = x + 1
```



# Imprimindo de 1 a 3

```
x = 1
while x <= 3:
    print (x)
    x = x + 1
```



# Revisar tudo fazendo um jogo

- Vamos fazer um jogo de adivinhar um número inteiro, que o computador irá escolher, entre 1 e 100
- Vamos revisar tudo o que aprendemos passo a passo

# Adivinhando números

## Funções

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

# Adivinhando números

Strings

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

# Adivinhando números

Variáveis

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

# Adivinhando números

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

Diretivas



# Adivinhando números

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

Identação

# Adivinhando números

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

= atribuição

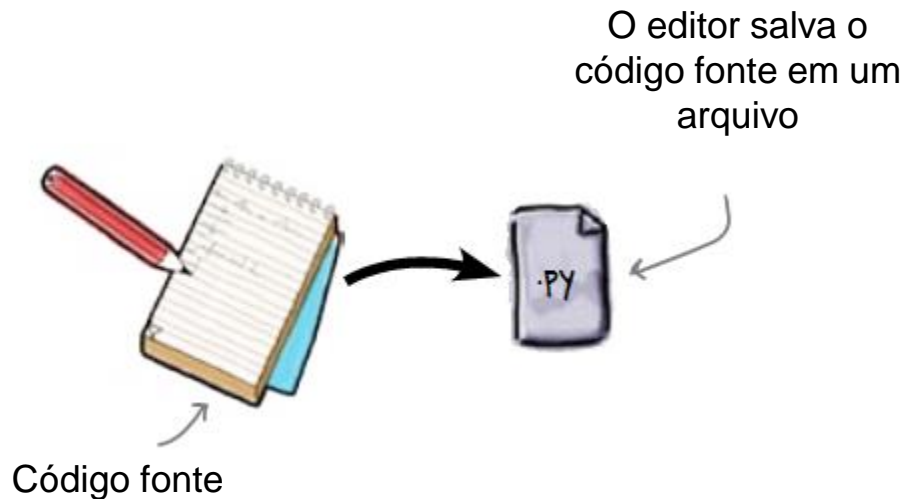
# Adivinhando números

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    print ('Você perdeu!')
print ('Fim do jogo!')
```

== comparação

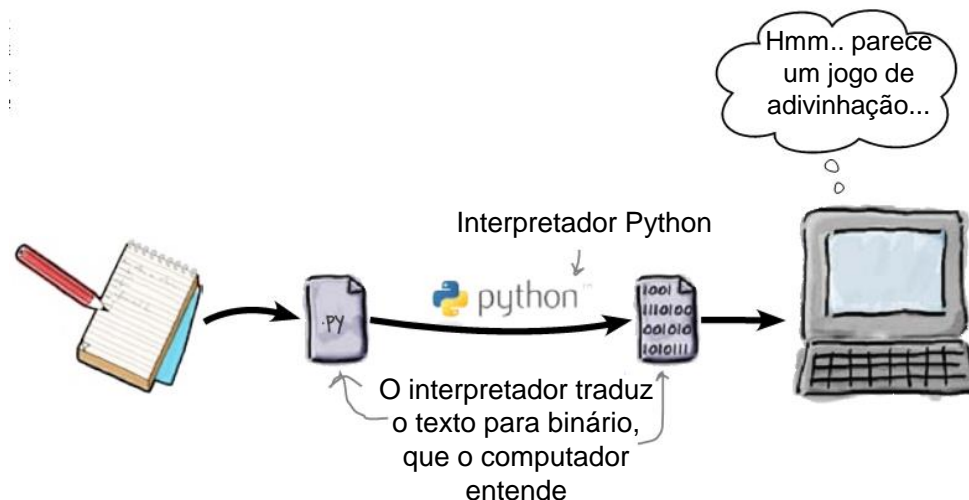
# Então, como você executa seu código?

- Há duas coisas para executar o programa de adivinhação: um *editor* e um *interpretador*
- O editor salva o código escrito em um arquivo no disco



# Então, como você executa seu código?

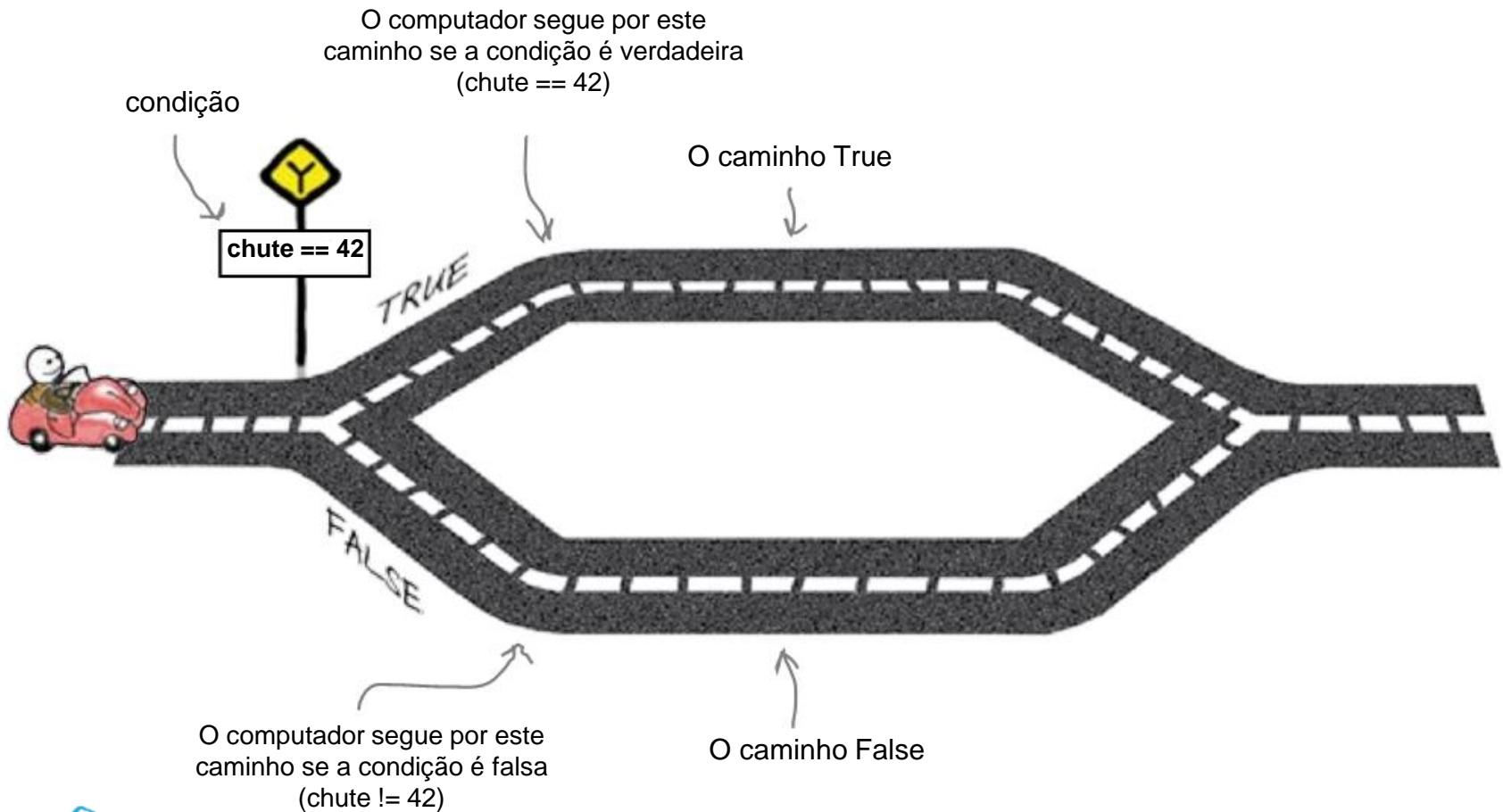
- Computadores não podem processar textos pois somente entendem binário (zeros e uns)
- O interpretador transforma o código fonte em um arquivo binário para o computador



# Então, como você executa seu código?

- O interpretador Python atua em dois modos: interativo e edição
- O modo interativo é ótimo para testar comandos e obter respostas instantâneas
- Porém o modo edição é o mais utilizado para desenvolver os programas
  - Nomes dos arquivos geralmente terminam com “.py”
  - Caso utilize outra extensão perderá as cores...

# Você escolhe seu caminho

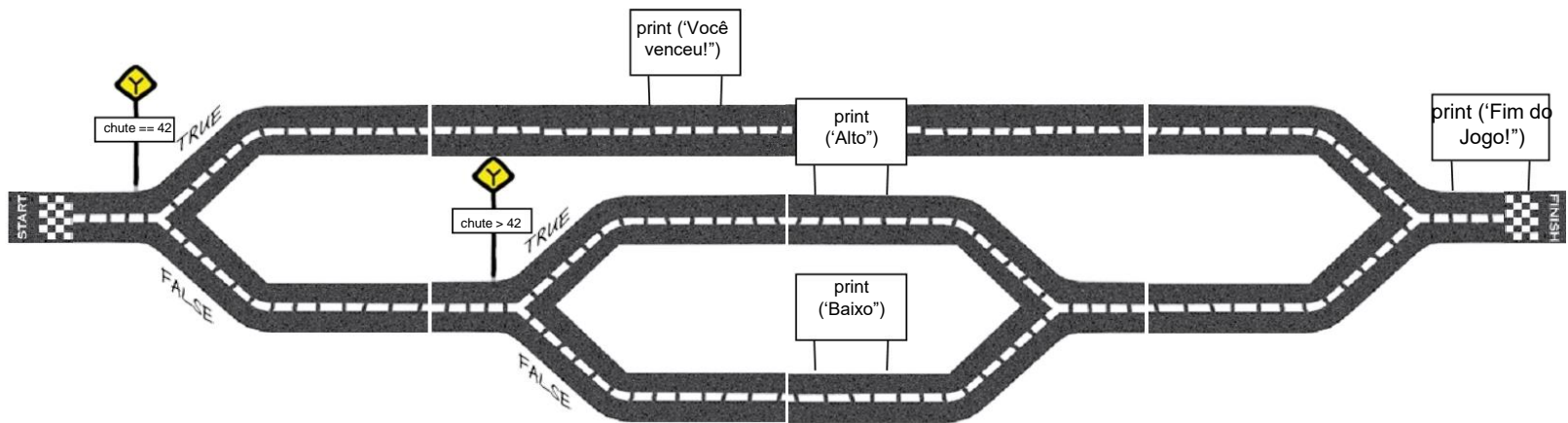


# Dando dicas

- O programa somente diz se acertou ou não
- Para ajudar você dirá “Alto” ou “Baixo” caso a pessoa erre
- Como ficaria a estrada?



# Dando dicas



# Dando dicas

```
print ('Bem vindo!')
g = input ('Chute um número: ')
chute = int(g)
if chute == 42:
    print ('Você venceu!')
else:
    if chute > 42:
        print ('Alto')
    else:
        print ('Baixo')
print ('Fim do jogo!')
```

Utilize BACKSPACE  
e TAB para ir e voltar

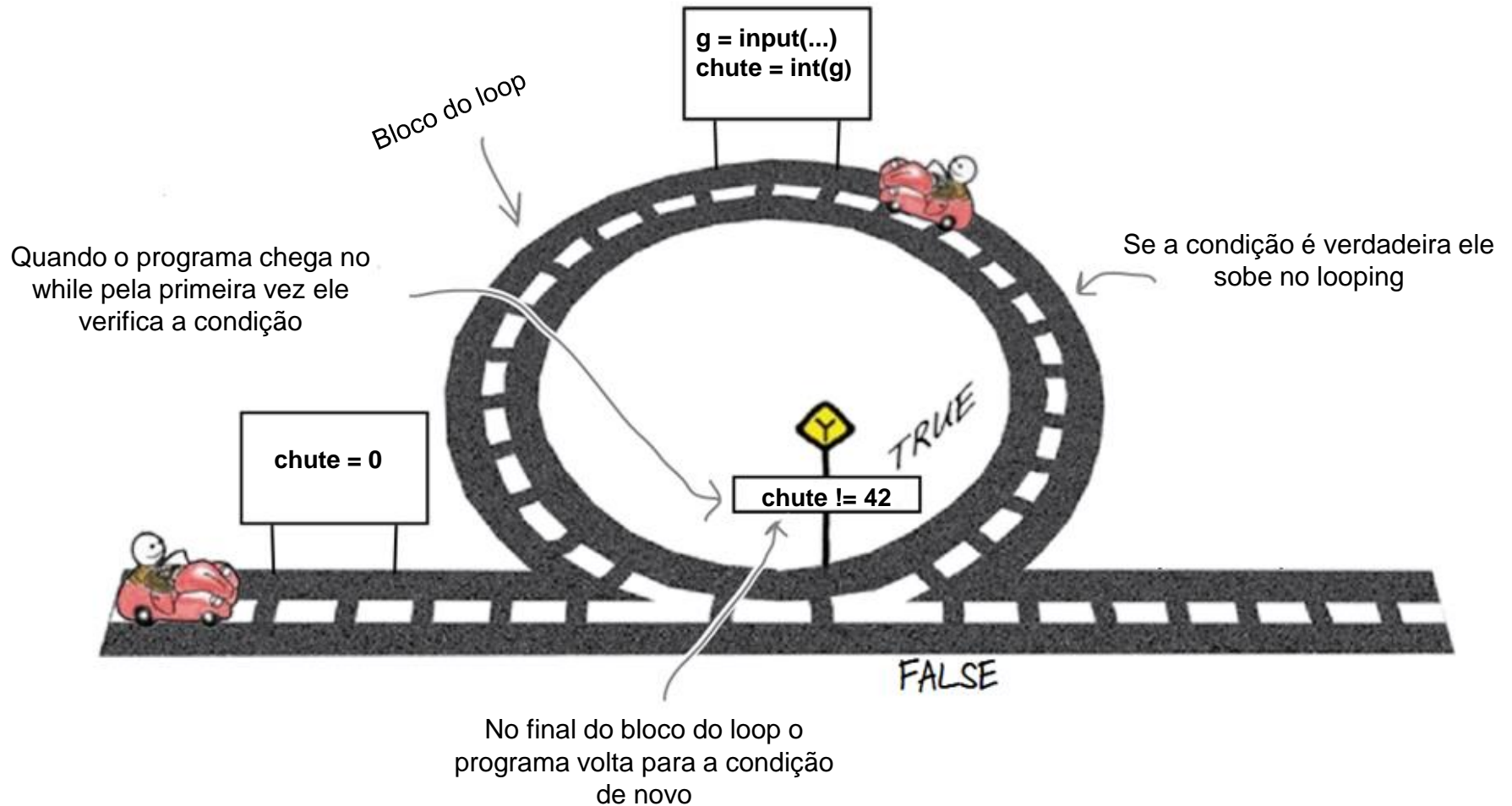
# Os usuários ainda não gostam



# Repetições

```
print ('Bem vindo!')  
→ chute = 0  
→ while chute != 42:  
    g = input ('Chute um número: ')  
    chute = int(g)  
    if chute == 42:  
        print ('Você venceu!')  
    else:  
        if chute > 42:  
            print ('Alto')  
        else:  
            print ('Baixo')  
print ('Fim do jogo!')
```

# Repetições



# Os usuários ainda não gostam



# Sortear o número a ser adivinhado



```
from random import randint
print ('Bem vindo!')
sorteado = randint (1, 100)
chute = 0
while chute != sorteado:
    g = input ('Chute um número: ')
    chute = int(g)
    if chute == sorteado:
        print ('Você venceu!')
    else:
        if chute > sorteado:
            print ('Alto')
        else:
            print ('Baixo')
print ('Fim do jogo!')
```

# Agora sim!

Show de bola! Não importa  
o quanto eu jogue, sempre  
terei um número novo!





# Strings e Acesso à Internet

- Vamos procurar onde estão as informações num texto == scraping
- Inicialmente iremos raspar o valor do café onde o preço está numa posição fixa
- Como conseguir raspar esse valor se a posição ficar mudando de posição?
- Iremos comprar o café apenas se o preço estiver abaixo de um valor, para isso deixaremos o programa repetir o acesso até o valor baixar

# Código Starbuzz

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices.html')
texto = pagina.read().decode('utf8')
print (texto)
```

```
>>>
```

```
<html><head><title>Welcome to the Beans'R'Us Pricing Page</title>
<link rel="stylesheet" type="text/css" href="beansrus.css" />
</head><body>
<h2>Welcome to the Beans'R'Us Pricing Page</h2>
<p>Current price of coffee beans = <strong>$5.31</strong></p>
<p>Price valid for 15 minutes from Tue Mar 15 13:16:01 2011.</p>
</body></html>
```



# O CEO quer apenas o preço



Você acha que pode  
obter apenas o preço?

# O preço está embutido no HTML

- Este é um texto HTML “bruto”, que é o formato das páginas Web
- O preço está embutido no HTML

>>>

```
<html><head><title>Welcome to the Beans'R'Us Pricing Page</title>
<link rel="stylesheet" type="text/css" href="beansrus.css" />
</head><body>
<h2>Welcome to the Beans'R'Us Pricing Page</h2>
<p>Current price of coffee beans = <strong>$5.31</strong></p>
<p>Price valid for 15 minutes from Tue Mar 15 13:16:01 2011.</p>
</body></html>
```



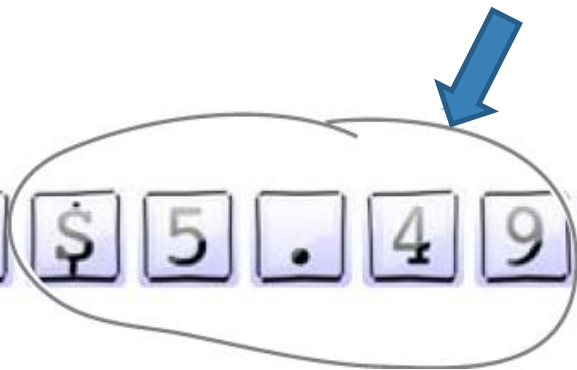
# Strings

- Strings são seqüências de caracteres

< h t m l > < h e a d > < t i

- Como obter apenas o preço?

< s t r o n g > \$ 5 . 4 9 < /



# Strings

Deslocamento

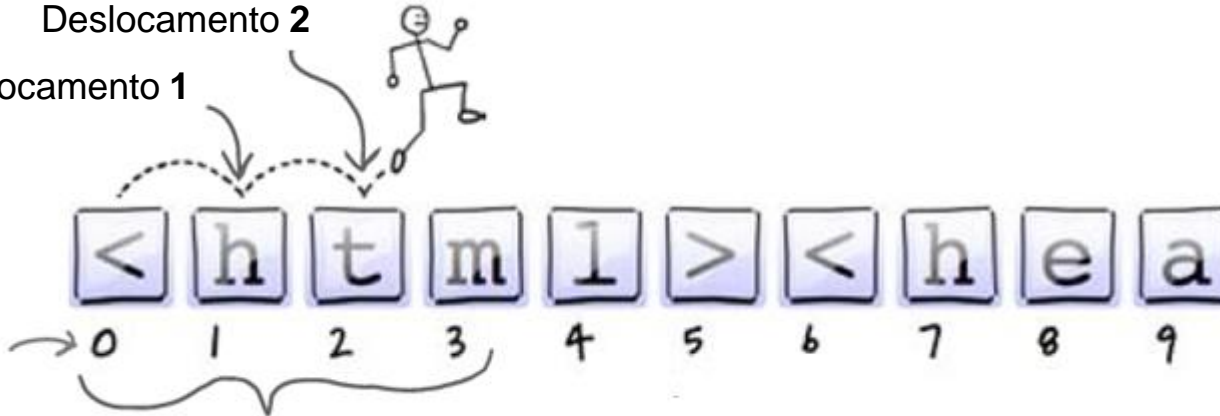
**zero**

Início da  
string

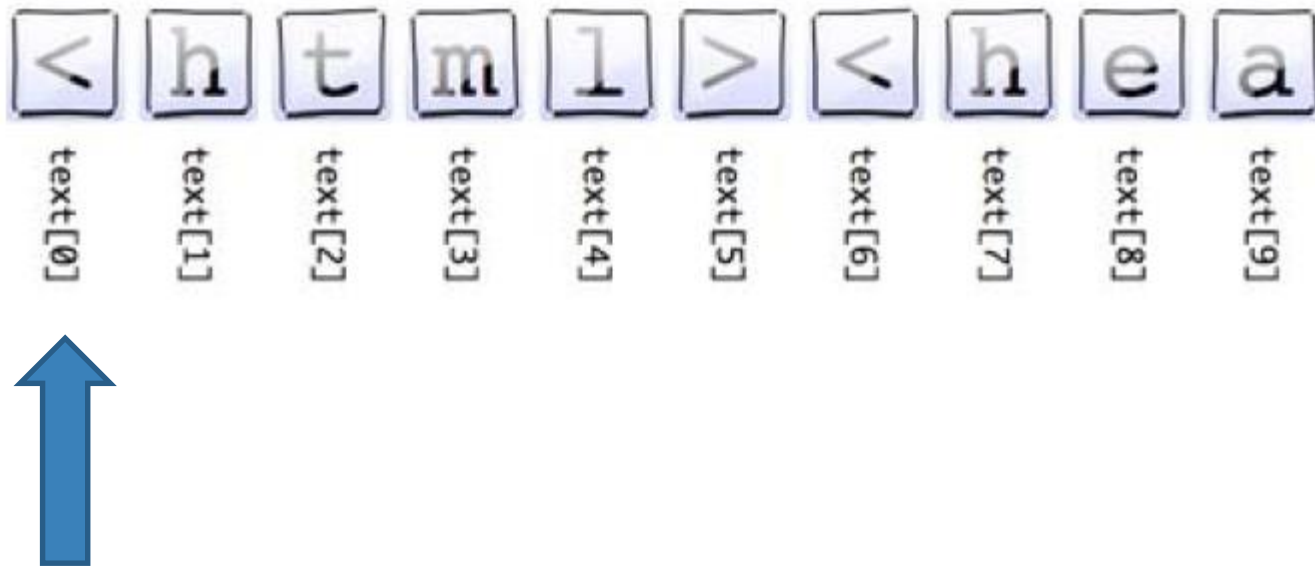


Deslocamento 2

Deslocamento 1



# Strings



O primeiro é zero!!

# Fatiamento

```
>>> time = "Palmeiras"
```

```
>>> time[2:5]
```

```
'lme'
```

```
>>> time[0:3]
```

```
'Pal'
```

```
>>> time[4:6]
```

```
'ei'
```

```
>>>
```

Não inclui o segundo número!

"Palmeiras"

0 1 2 3 4 5 6 7 8

lo



# Fatiamento



# Fatiamento

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices.html')
texto = pagina.read().decode('utf8')
preço = texto[234:238]
print (preço)

>>>
5.31
```

# O CEO está feliz!

Exatamente o que preciso!  
Você não sabe quanto  
tempo e dinheiro irá  
economizar para mim...



# Não existem perguntas idiotas

- Posso colocar qualquer página web neste código?
  - Sim. Fique à vontade, mas não esqueça o decode
  - Por exemplo, o site abaixo usa iso8859
  - [www.ime.usp.br/~pf/dicios/br](http://www.ime.usp.br/~pf/dicios/br)
- O que urllib.request faz?
  - Permite conversar com a internet
- Posso acessar uma página diretamente no navegador?
  - Sim. Digite no modo interativo “import antigravity”

# Descontos para clientes fiéis

Vocês poderiam ver o preço  
no programa de fidelidade?  
Acho que é simples mudar...



Clientes normais

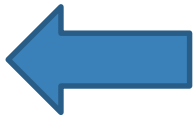
↪ `http://www.beans-r-us.biz/prices.html`

Clientes fiéis ↪ `http://www.beans-r-us.biz/prices-loyalty.html`

# Programa de fidelidade

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices-loyalty.html')
texto = pagina.read().decode('utf8')
preço = texto[234:238]
print (preço)
```

```
bean
>>>
```



Não deu certo! Apareceu “bean” no lugar do preço. Por quê será?

# O preço se moveu

## Welcome to the Beans'R'Us Pricing Page

Current price of coffee beans = **\$6.94**

Price valid for 15 minutes from Tue Mar 15 13:22:01 2011.

## Welcome to the Beans'R'Us Pricing Page

Special Offer!!! Current price of coffee beans = **\$4.39**

Limited time only - get 'em while they're roasting!

Price valid for 15 minutes from Tue Mar 15 13:22:01 2011.

# Método find

- Métodos find para strings

```
>>> "Palmeiras".find("P")
0
>>> "Palmeiras".find("lmei")
2
>>> "Palmeiras".find("Pa")
0
>>>
```

Para saber os métodos que posso dar ctrl + espaço após ponto



# Método find

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices-loyalty.html')
texto = pagina.read().decode('utf8')
onde = texto.find('>$')
inicio = onde + 2
fim = inicio + 4
preço = texto[inicio:fim]
print (preço)
```

```
>>>
4.90
```



# Só quando for menos que 4.74



# Só quando for menos que 4.74

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices-loyalty.html')
texto = pagina.read().decode('utf8')
onde = texto.find('>$')
inicio = onde + 2
fim = inicio + 4
preço = texto[inicio:fim]
if preço < 4.74:
    print (preço)
>>>
Traceback (most recent call last):
  File "C:/Python31/caneca05.py", line 9, in <module>
    if preço < 4.74:
TypeError: unorderable types: str() < float()
```

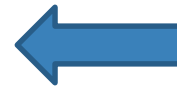


# Strings são diferentes de números



# Convertendo para float

```
import urllib.request
pagina = urllib.request.urlopen(
    'http://beans.itcarlow.ie/prices-loyalty.html')
texto = pagina.read().decode('utf8')
onde = texto.find('>$')
inicio = onde + 2
fim = inicio + 4
preço = float(texto[inicio:fim])
if preço < 4.74:
    print ('Comprar! Preço: %5.2f' %preço)
```



```
>>>
```

```
Comprar! Preço: 4.58
```

# Ele pode ficar testando o preço?



# Ele pode ficar tentando?

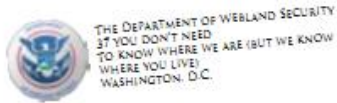
```
import urllib.request
preço = 99.99
while preço >= 4.74:
    pagina = urllib.request.urlopen(
        'http://beans.itcarlow.ie/prices-loyalty.html')
    texto = pagina.read().decode('utf8')
    onde = texto.find('>$')
    inicio = onde + 2
    fim = inicio + 4
    preço = float(texto[inicio:fim])
print ('Comprar! Preço: %5.2f' %preço)
>>>
Comprar! Preço: 4.47
```

# O CEO está muito feliz!





# Aconteceu algum problema



IF YOU DON'T NEED  
TO KNOW WHERE WE ARE (BUT WE KNOW  
WHERE YOU LIVE)  
WASHINGTON, D.C.

From: The Department of Webland Security  
Secret Service - Corporate Enforcement Unit

To Whom It May Concern:

A recent investigation into an apparent Distributed Denial of Service (DDoS) attack on the [www.beans-r-us.biz](http://www.beans-r-us.biz) domain showed that much of the traffic originated from machines located in various Starbuzz outlets from around the world. The number of web transactions (which reached a peak of several hundred thousand requests worldwide) resulted in a crash of the Beans'R'Us servers, resulting in a significant loss of business.

In accordance with the powers invested in this office by the United States Attorney General, we are alerting the developer of the very dim view we take of this kind of thing. In short:

**We're watching you, Bud. Consider yourself on notice.**

Yours faithfully,

A stylized, handwritten signature in black ink.

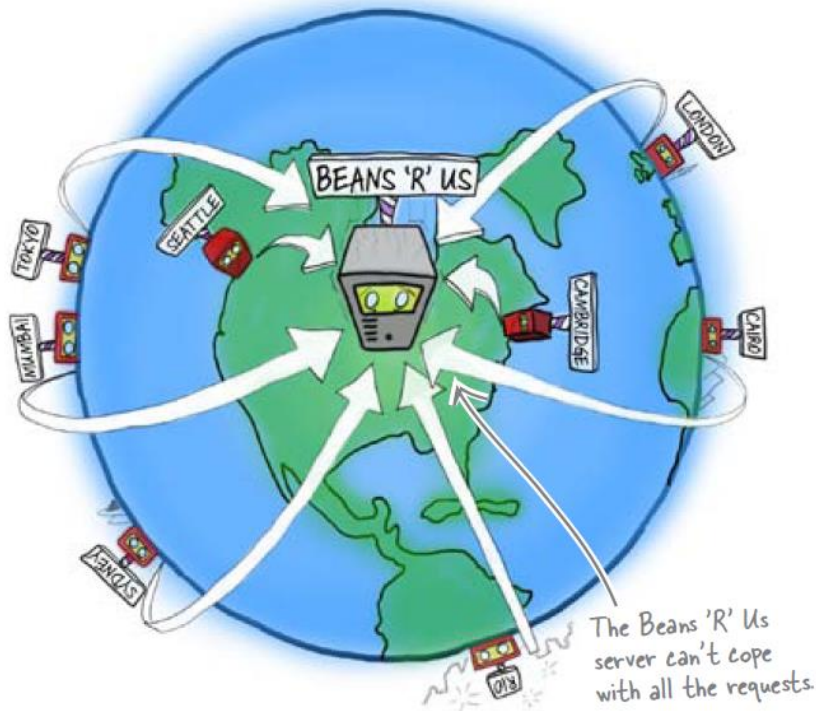
Head of Internet Affairs

O servidor da  
empresa de grãos  
travou!

E fomos acusados  
de ser hacker's!

# Acusação de DDoS

- DDoS – Distributed Denial of Service



Se o valor está acima de 4.74 o programa NÃO espera e já acessa o site de novo!

Seu programa gera milhares de solicitações por hora em todas as filiais

# 10 minutos entre cada acesso

```
import urllib.request
import time
preço = 99.99
while preço >= 4.74:
    pagina = urllib.request.urlopen(
        'http://beans.itcarlow.ie/prices-loyalty.html')
    texto = pagina.read().decode('utf8')
    onde = texto.find('>$')
    inicio = onde + 2
    fim = inicio + 4
    preço = float(texto[inicio:fim])
    if preço >= 4.74:
        time.sleep(600)
print ('Comprar! Preço: %5.2f' %preço)
>>>
Comprar! Preço: 4.44
```



# Documentação Dados Públicos Copa 2014

Figura 3 - Configurando um projeto Java.

O próximo passo é a inclusão das bibliotecas necessárias para a compilação e execução do cliente Java. Segue abaixo a lista de bibliotecas utilizadas e os links de onde obtê-las.

hk2-api-1.6.32 - <http://repo1.maven.org/maven2/org/glassfish/hk2/hk2-api/1.6.32/hk2-api-1.6.32.jar>

mail-1.4 - <http://repo1.maven.org/maven2/javax/mail/mail/1.4/mail-1.4.jar>

auto-depends-1.6.32 - <http://repo1.maven.org/maven2/org/glassfish/hk2/auto-depends/1.6.32/auto-depends-1.6.32.jar>

guava-11.0.1 - <http://repo1.maven.org/maven2/com/google/guava/guava/11.0.1/guava-11.0.1.jar>

javax.inject-1.6.32 - <http://repo1.maven.org/maven2/org/glassfish/hk2/external/javax.inject/1.6.32/javax.inject-1.6.32.jar>

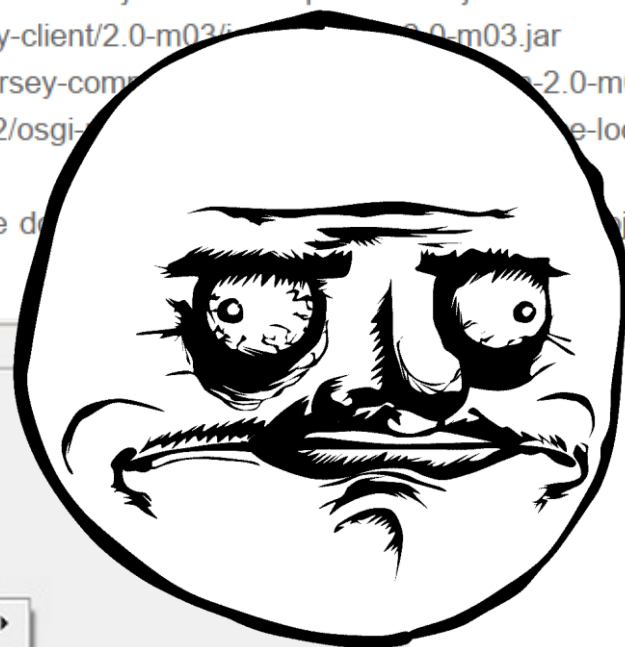
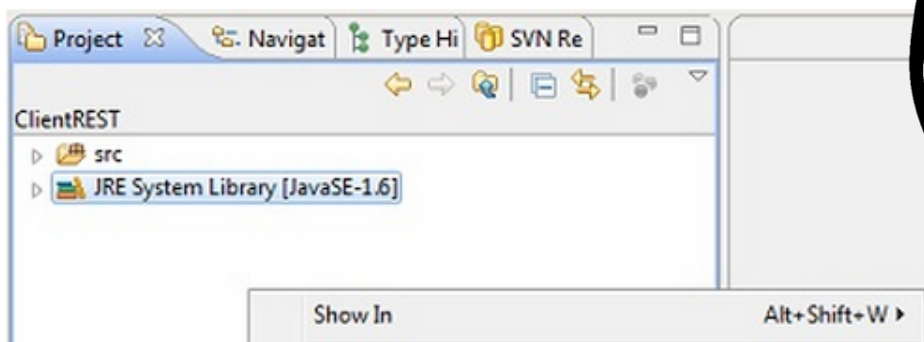
javax.ws.rs-api-2.0-m04 - <http://repo1.maven.org/maven2/javax/ws/rs/javax.ws.rs-api/2.0-m04/javax.ws.rs-api-2.0-m04.jar>

jersey-client-2.0-m03 - <http://repo1.maven.org/maven2/org/glassfish/jersey/core/jersey-client/2.0-m03/jersey-client-2.0-m03.jar>

jersey-common-2.0-m03 - <http://repo1.maven.org/maven2/org/glassfish/jersey/core/jersey-common/2.0-m03/jersey-common-2.0-m03.jar>

osgi-resource-locator-1.0.1 - <http://repo1.maven.org/maven2/org/glassfish/hk2/osgi-resource-locator/1.0.1/osgi-resource-locator-1.0.1.jar>

Para incluir as bibliotecas no projeto, clique com o botão direito do mouse no nome do projeto e selecione Build Path > Configure Build Path....



# Soma simples em Python

```
import urllib.request
url = 'http://www.portaltransparencia.gov.br/copa2014/api/rest/empreendimento'
resp = urllib.request.urlopen(url).read().decode('utf-8')
total = 0
j = 0
while True:
    abre = '<valorTotalPrevisto>'
    fecha = '</valorTotalPrevisto>'
    j = resp.find(abre, j)
    if j == -1:
        break
    k = resp.find(fecha, j)
    print (resp[j+len(abre):k])
    total = total + float (resp[j+len(abre):k])
    j = k + len(fecha)
print ('R$ %.2f' %total)

R$ 27346140056.43
>>>
```

<https://gist.github.com/fmasanori/a428ef783211b0ffdee1>



```
for letra in 'aeiou':  
    print (letra)
```

```
>>>
```

```
a
```

```
e
```

```
i
```

```
o
```

```
u
```

```
texto = 'aeiou'
```

```
k = 0
```

```
while k < len(texto):
```

```
    letra = texto[k]
```

```
    print (letra)
```

```
    k = k + 1
```

```
>>>
```

```
a
```

```
e
```

```
i
```

```
o
```

```
u
```

Códigos equivalentes: for durante o dia vira while à noite

```
for x in ['cpbr6', 42, 3.14]:  
    print (x)
```

```
>>>  
cpbr6  
42  
3.14
```

```
lista = ['cpbr6', 42, 3.14]  
k = 0
```

```
while k < len(lista):  
    x = lista[k]  
    print (x)  
    k = k + 1
```

```
>>>  
cpbr6  
42  
3.14
```

Códigos equivalentes



```
for i in range(5):  
    print (i)
```

```
>>>
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
lista = list(range(5))
```

```
k = 0
```

```
while k < len(lista):
```

```
    i = lista[k]
```

```
    print (i)
```

```
    k = k + 1
```

```
>>>
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

Códigos equivalentes



# Dicionários

- O dicionário em si consiste em relacionar uma chave a um valor específico
- Diferentemente das listas, onde o índice é um número, dicionários utilizam suas chaves como índice
- Para adicionar novos elementos basta fazer a atribuição
  - Se a chave já existe: o valor associado é alterado
  - Se a chave não existe: a nova chave é adicionada

```
>>> d = {}
>>> d['a'] = 'alpha'
>>> d['o'] = 'omega'
>>> d['g'] = 'gama'
>>> d
{'a': 'alpha', 'g': 'gama', 'o': 'omega'}
>>> d['a']
'alpha'
>>> d['x']
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    d['x']
KeyError: 'x'
```

```
import urllib.request
import json
resp = urllib.request.urlopen('http://worldcup.sfg.io/matches').read()
for jogo in json.loads(resp.decode('utf-8')):
    if jogo['status'] == 'completed':
        print (jogo['home_team']['country'], jogo['home_team']['goals'], 'x',
                jogo['away_team']['country'], jogo['away_team']['goals'])
.
.
.
Germany 2 x Algeria 1
Argentina 1 x Switzerland 0
Belgium 2 x USA 1
Brazil 2 x Colombia 1
France 0 x Germany 1
Netherlands 0 x Costa Rica 0
Argentina 1 x Belgium 0
Brazil 1 x Germany 7
Netherlands 0 x Argentina 0
Brazil 0 x Netherlands 3
Germany 1 x Argentina 0
>>>
```

<https://gist.github.com/fmasanori/1288160dad16cc473a53>



```
import urllib.request
import json

url = 'http://api.icndb.com/jokes/random?limitTo=[nerdy] '

resp = urllib.request.urlopen(url).read()
data = json.loads(resp.decode('utf-8'))

print (data['value']['joke'])
```

<https://gist.github.com/4745061>



```
import urllib.request
import json
url = 'http://educacao.dadosabertosbr.com/api/escolas/buscaavancada?situacaoFuncionamento=1'
situação = '&energiaInexistente=on&aguaInexistente=on&esgotoInexistente=on'
resp = urllib.request.urlopen(url+situação).read()
resp = json.loads(resp.decode('utf-8'))
print ('Número de Escolas em funcionamento sem energia, água e esgoto:', resp[0])
for x in resp[1]:
    print (x['nome'], x['cod'])
    print (x['cidade'], x['estado'], x['regiao'])
    print ()
```

Número de Escolas em funcionamento sem energia, água e esgoto: 968  
BAIRRO DOS MENDES E R M EF 41036026  
CANDIDO DE ABREU PR Sul

CEF CAJAZEIRAS 21237808  
SANTA HELENA MA Nordeste

E M E F AGUA AZUL 15008258  
MONTE ALEGRE PA Norte

E M E F AGUA BOA 15534340  
PLACAS PA Norte

E M E F AGUA PRETA 15590356  
MOJU PA Norte

<https://gist.github.com/fmasanori/fc0e98c8c486be80d5a5>



# Documentação API Educação

The screenshot shows a web browser displaying the API documentation for 'Educação Dados Abertos Brasil'. The page has a blue header with the title 'EDUCAÇÃO INTELIGENTE' and a subtitle 'Visualizando a Educação do Brasil'. Below the header, there are statistics for different levels of education: 272.049 Escolas, 558 Federais, 37.028 Estaduais, 175.326 Municipais, and 59.137 Privadas. The main content area is titled 'API para Desenvolvedores' and includes a sidebar with navigation links: Início, Escolas, Estatísticas, Alertas, API, Contato, and Sobre. The API section is selected. The main text describes the API's purpose and provides a list of endpoints under the heading 'Entradas Principais'.

**EDUCAÇÃO INTELIGENTE**  
Visualizando a Educação do Brasil

272.049 Escolas, 558 Federais, 37.028 Estaduais, 175.326 Municipais, 59.137 Privadas

**API para Desenvolvedores**

Esta página é voltada para Desenvolvedores, Geeks, Nerds, e pessoas que sabem programar computadores e desejam utilizar nossos dados para o desenvolvimento de sistemas baseados neles.

**Sobre a API**

Os dados do INEP não são publicados segundo os conceitos internacionais de dados abertos, sendo disponibilizados em páginas HTML sob diversas formas, desde grandes arquivos compactados contendo os dados, até "sistemas" feitos no excel. Isto leva a muitos problemas e dificuldades na utilização desses dados pela população, inclusive no desenvolvimento de aplicações como este projeto.

Com o objetivo de melhorar a disponibilização desses dados, de forma a incentivar o uso dos mesmos para geração de novas iniciativas por terceiros, criamos esta API (Application Programming Interface) para republicação dos dados. Ela ainda está na primeira versão, mas já completa. Objetivamos sempre atualizá-la e melhorá-la.

A API contém principalmente os dados das escolas e das estatísticas. O acesso é via chamadas HTTP (REST) e o retorno é sempre no formato JSON. Abaixo seguem os detalhes da API.

Entrada	Descrição
/api/cidades/{estado}	Método auxiliar que retorna o código e o nome de todas as cidades do referido estado. Necessário para poder realizar buscas pelo código da cidade
/api/escola/{id}	Detalha os dados da escola
/api/escolas	Busca simples por escolas
/api/escolas/buscaavancada	Busca avançada por escolas
/api/estatisticas	Retorna as estatísticas das escolas
/api/estatisticas/{indice}/{codescola}	Retorna as estatísticas de um índice específico para todos os estados, regiões e do país, com destaque para a cidade/estado/região da escola

<http://educacao.dadosabertosbr.com/api>



# Dados das eleições americanas 2016

```
from urllib.request import Request, urlopen
import json
```

```
url = "https://api.propublica.org/campaign-finance/v1/2016/president/totals.json"
```

```
q = Request(url)
q.add_header('X-API-Key', 'n8QLPm4lNS2Mfak1bam5X7HlevBAOSoF9epQkX0m')
data = urlopen(q).read()
data = json.loads(data.decode('utf-8'))
```

```
for candidate in data['results']:
    print (candidate['name'])
    print (candidate['total_contributions'])
    print (candidate['cash_on_hand'])
```

```
>>>
```

Hillary Clinton

110188447.67

37977647.61

Bernie Sanders

73483756.37

28304765.54

<https://gist.github.com/fmasanori/b7b56cf9fb0a4a4e4a17>



# <http://developer.nytimes.com/>

```
import urllib.request
import json

api_key = '7ba14b76dcea0086a03f8eca5a3a801f:1:74295566'
category = 'editorial'
for i in range(5):
    url = "http://api.nytimes.com/svc/search/v2/articlesearch.json?" \
          "q={0}&api-key={1}&page={2}".format(category, api_key, i)
    h = urllib.request.urlopen(url)
    result = json.loads(h.read().decode('utf-8'))
    print (result['response']['docs'][0]['headline']['main'])
    print (result['response']['docs'][0]['abstract'])
    print ()
```

<https://gist.github.com/fmasanori/d153bea5690164f75302>





Hackeando Dados Públicos com Python

Harness the power of Twitter through APIs

Coding for Journalists

Katharine Jarmul: Introduction to Web (and data!) Scraping with Python - PyCon 2014

Web scraping: Reliably and efficiently pull data from pages that don't expect it



# Questions?

[gist.github.com/fmasanori](https://gist.github.com/fmasanori)

<http://about.me/fmasanori>

[fmasanori@gmail.com](mailto:fmasanori@gmail.com)

