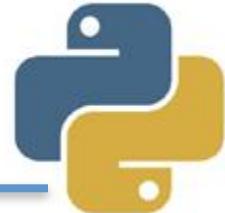


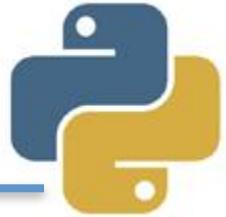
Capítulo 07



Manipulação de Arquivos e controle de Exceções



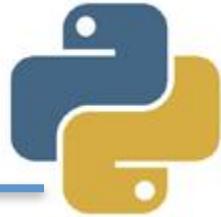
Márcio Palheta, M.Sc.
marcio.palheta@gmail.com



Apresentação

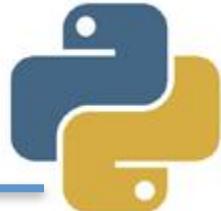
- Programador desde 2000
- Professor de programação desde 2009
- Mestre pelo ICOMP/UFAM – 2013
- Fundador da Buritech – 2014
- Doutorando pelo ICOMP/UFAM
- Pesquisador das áreas: Banco de Dados, Recuperação da Informação, Big Data, Mineração de dados e Aprendizado de Máquina





Agenda

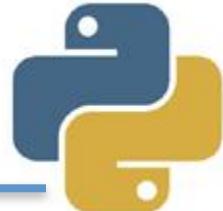
- Revisão da aula anterior
 - Manipulação de arquivos
 - Abertura para escrita/gravação
 - Tratamento de exceções
 - Try, Catch e Finally
 - Criação e lançamento de exceções
-



Revisão

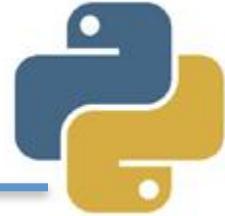
- O papel da classe object
 - Sobrescrita de métodos
 - Como implementar a Herança múltipla ?
 - Herança múltipla
 - Qual a ordem de execução de métodos na herança múltipla?
 - Teste de tipos de atributos
-

O que temos por aqui?



- Como trabalhar com arquivos?
 - Como faço leitura e gravação de arquivos?
 - Como posso tratar exceções
 - Lançamento de exceções
 - Criação de exceções personalizadas
-

Trabalhando com arquivos



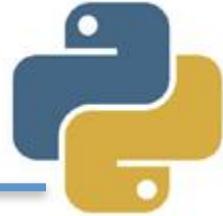
- Em ambientes **corporativos**, é comum a necessidade de **envio de dados** entre serviços
 - Uma estratégia comum, é a criação de **arquivos CSV**, para este fim
 - Vamos começar, **lendo um arquivo com dados de GIBIS**
-



Arquivo gibis.csv

	ID;NUMERO;TITULO;personagem;editora
1	11;24;"MEMORIA APAGADA";WOLVERINE;"MARVEL COMICS"
2	12;26;"ENTERRADO VIVO";WOLVERINE;"MARVEL COMICS"
3	13;3;"SAMURAIS EM GUERRA";WOLVERINE;"MARVEL COMICS"
4	14;39;"CODINOME WOLVERINE";WOLVERINE;"MARVEL COMICS"
5	15;44;"MORTE EM FAMÍLIA";WOLVERINE;"MARVEL COMICS"
6	1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
7	2;2;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
8	3;3;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
9	4;2002;"CORPORAÇÃO DO SUPERMAN";SUPERMAN;"DC COMICS"
10	5;0;"O ULTIMO FILHO";SUPERMAN;"DC COMICS"
11	6;0;"A MORTE DO ROBIN";BATMAN;"DC COMICS"
12	7;0;"A MASCARA DA MORTE";BATMAN;"DC COMICS"
13	8;1;"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN";BATMAN;"DC COMICS"
14	9;0;"A VITÓRIA DE BANE";BATMAN;"DC COMICS"
15	10;16;"MORCEGO VERSUS MORCEGO";BATMAN;"DC COMICS"
16	
17	

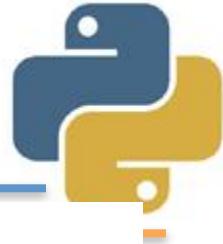
Exercício 01: leitura



- Crie o arquivo `teste_leitura.py`

```
1      # -*- coding: UTF-8 -*-
2      # teste_leitura.py
3
4      arquivo = open('gibis.csv', 'r')
5      for linha in arquivo:
6          print linha
7      arquivo.close()
8
```

Exercício 01: leitura



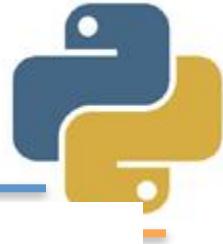
- Crie o arquivo teste leitura.

Abertura do
arquivo gibis.csv



```
1 #  
2 #  
3  
4 arquivo = open('gibis.csv', 'r')  
5 for linha in arquivo:  
6     print(linha)  
7 arquivo.close()  
8
```

Exercício 01: leitura



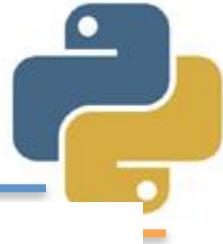
- Crie o arquivo teste_leitura.

Modo leitura



```
1 # - - - - -  
2 # - - - - -  
3  
4 arquivo = open('gibis.csv', 'r')  
5 for linha in arquivo:  
6     print(linha)  
7 arquivo.close()  
8
```

Exercício 01: leitura



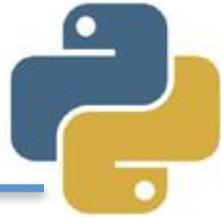
- Crie o arquivo teste_leitura.

Lista de linhas do arquivo



```
1 #  
2 #  
3  
4 arquivo = open('gibis.csv', 'r')  
5 for linha in arquivo:  
6     print(linha)  
7 arquivo.close()  
8
```

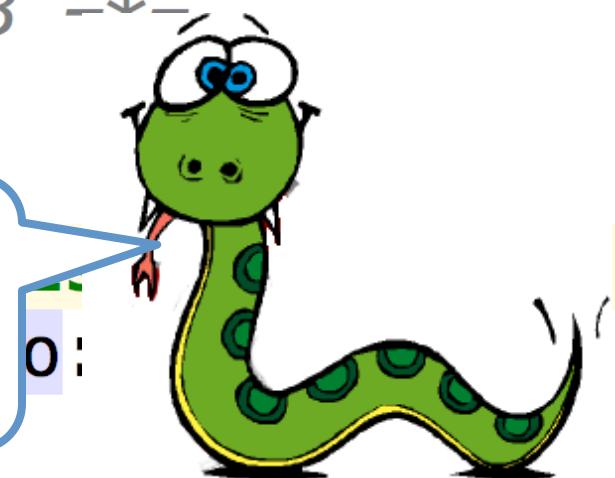
Exercício 01: leitura



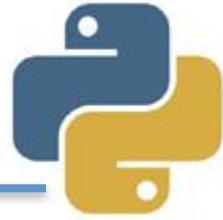
- Crie o arquivo teste_leitura.py

```
1 # -*- coding: UTF-8 -*-
2 # teste_leitura.py
3
4 ar
5 fo
6
7 arquivo.close()
8
```

Fechando arquivo



Resultado no console



```
Run teste_leitura
/Library/Frameworks/Python.framework/Versions/2.7/bin/p
ID;NUMERO;TITULO;personagem;editora

11;24;"MEMORIA APAGADA";WOLVERINE;"MARVEL COMICS"

12;26;"ENTERRADO VIVO";WOLVERINE;"MARVEL COMICS"

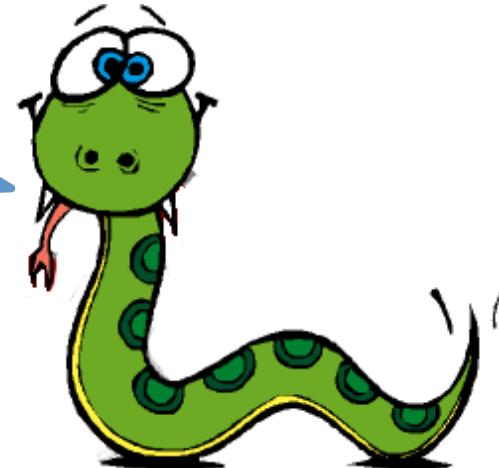
13;3;"SAMURAIS EM GUERRA";WOLVERINE;"MARVEL COMICS"

14;39;"CODINOME WOLVERINE";WOLVERINE;"MARVEL COMICS"

15;44;"MORTE EM FAMÍLIA";WOLVERINE;"MARVEL COMICS"

1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
```

R A 1^a linha contém
o cabeçalho das
colunas



Run teste_leitura

```
/Library/Frameworks/Python.framework/Versions/3.7/bin/python3 teste_leitura.py
```

ID;NUMERO;TITULO;personagem;editora

11;24;"MEMORIA APAGADA";WOLVERINE;"MARVEL COMICS"

12;26;"ENTERRADO VIVO";WOLVERINE;"MARVEL COMICS"

13;3;"SAMURAIS EM GUERRA";WOLVERINE;"MARVEL COMICS"

14;39;"CODINOME WOLVERINE";WOLVERINE;"MARVEL COMICS"

15;44;"MORTE EM FAMÍLIA";WOLVERINE;"MARVEL COMICS"

1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"

R

Está lendo o
carácter de
quebra de linha

\n

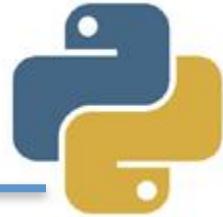


Run teste_leitura

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/test/test_leitura.py
```

11;24;"MEMORIA APAGADA";WOLVERINE;"MARVEL COMICS"
12;26;"ENTERRADO VIVO";WOLVERINE;"MARVEL COMICS"
13;3;"SAMURAIS EM GUERRA";WOLVERINE;"MARVEL COMICS"
14;39;"CODINOME WOLVERINE";WOLVERINE;"MARVEL COMICS"
15;44;"MORTE EM FAMÍLIA";WOLVERINE;"MARVEL COMICS"
1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"

Exercício 2: strip()



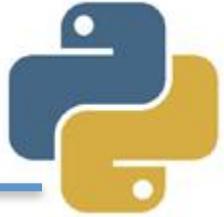
- Remova caracteres de quebra de linha

The screenshot shows a code editor with two tabs: 'gibis.csv' and 'teste_leitura.py'. The 'teste_leitura.py' tab is active, displaying the following Python script:

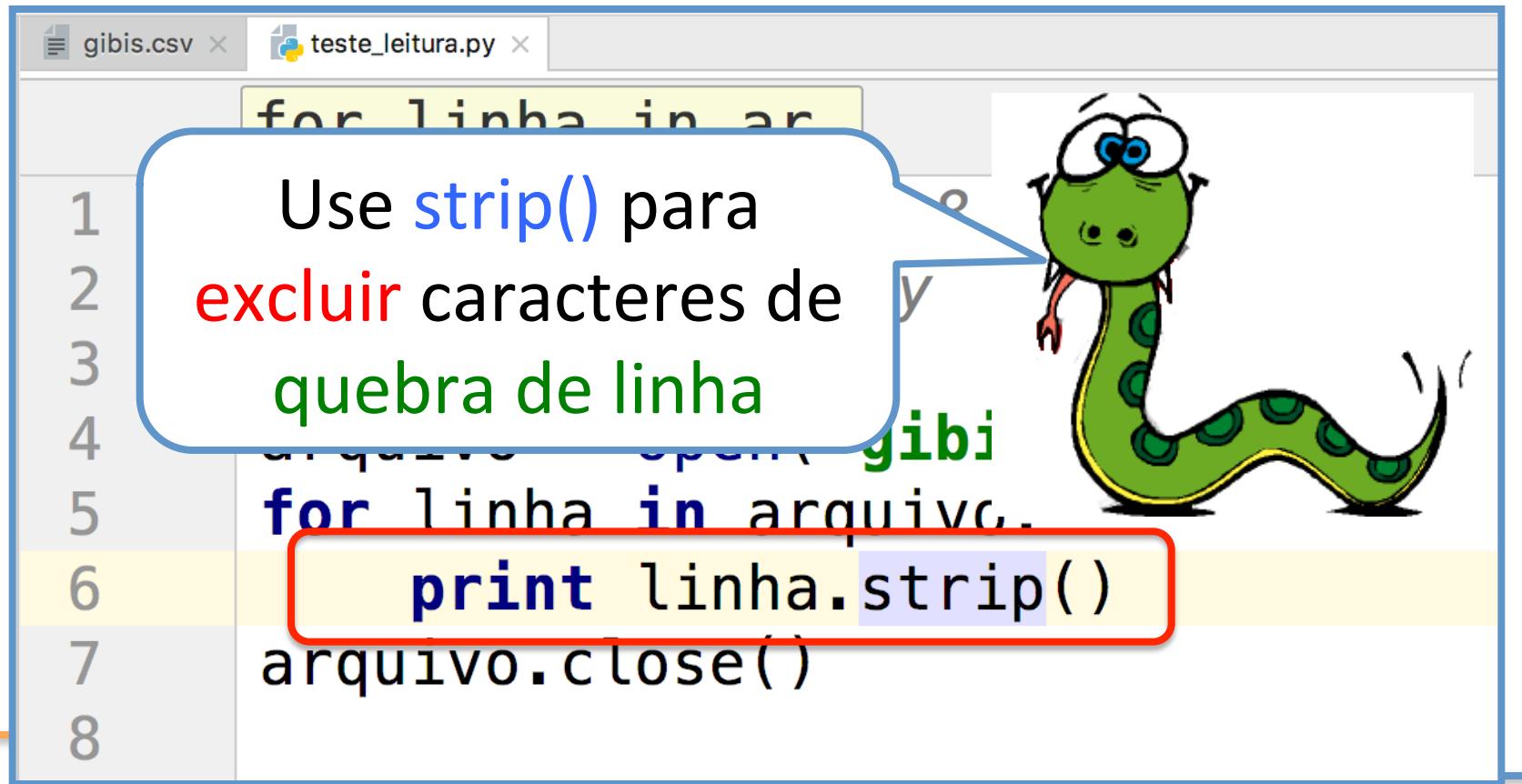
```
for linha in ar...
1 # -*- coding: UTF-8 -*-
2 # teste_leitura.py
3
4 arquivo = open('gibis.csv', 'r')
5 for linha in arquivo:
6     print linha.strip()
7 arquivo.close()
8
```

The code uses the `strip()` method on each line of the CSV file to remove leading and trailing whitespace before printing it.

Exercício 2: strip()



- Remova caracteres de quebra de linha



A cartoon illustration of a green snake with large eyes and a pink tongue, pointing its head towards the text in the speech bubble.

```
gibis.csv x teste_leitura.py x
for linha in ar
1     Use strip() para
2     excluir caracteres de
3     quebra de linha
4     gibis
5     for linha in arquivo.
6     print linha.strip()
7     arquivo.close()
8
```

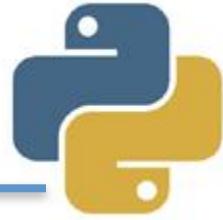
The code in the image is a Python script named `teste_leitura.py` that reads from a CSV file named `gibis.csv`. It uses the `strip()` method on each line of the file to remove whitespace characters (like newlines).

A callout bubble contains the following text:

Use `strip()` para
excluir caracteres de
quebra de linha

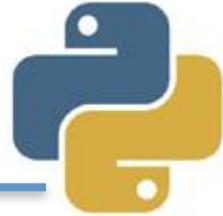
The line `print linha.strip()` is highlighted with a red rectangle.

Resultado no console



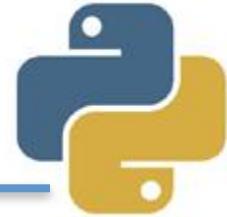
```
Run teste_leitura
▶ /Library/Frameworks/Python.framework/Versions/2.7/bin/
ID;NUMERO;TITULO;personagem;editora
11;24;"MEMORIA APAGADA";WOLVERINE;"MARVEL COMICS"
12;26;"ENTERRADO VIVO";WOLVERINE;"MARVEL COMICS"
13;3;"SAMURAIS EM GUERRA";WOLVERINE;"MARVEL COMICS"
14;39;"CODINOME WOLVERINE";WOLVERINE;"MARVEL COMICS"
15;44;"MORTE EM FAMÍLIA";WOLVERINE;"MARVEL COMICS"
1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
2;2;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
3;3;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
4;2002;"CORPORAÇÃO DO SUPERMAN";SUPERMAN;"DC COMICS"
5;0;"O ULTIMO FILHO";SUPERMAN;"DC COMICS"
6;0;"A MORTE DO ROBIN";BATMAN;"DC COMICS"
7;0;"A MASCARA DA MORTE";BATMAN;"DC COMICS"
```

Organizando os gibis



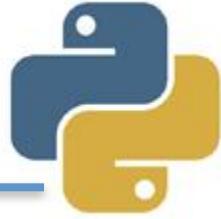
- Vamos criar uma **classe** para armazenar os dados dos **Gibis**
 - Na classe Gibi, crie um **método estático** para ler o arquivo **gibis.csv** e devolver uma **lista de objetos Gibi**
 - Atualize o arquivo **teste_leitura.py** para testar o método de carga de Gibis
-

Exercício 3: A classe Gibi

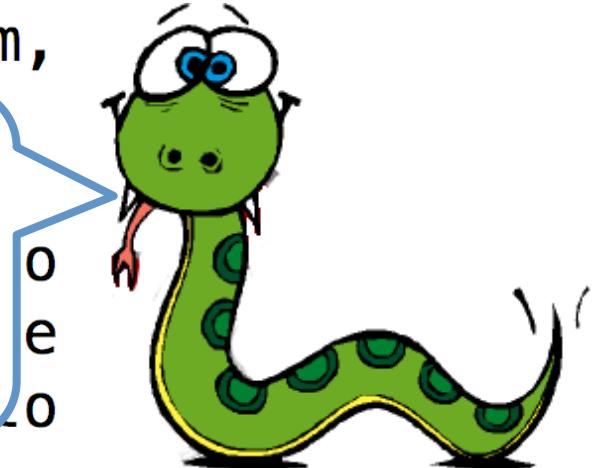


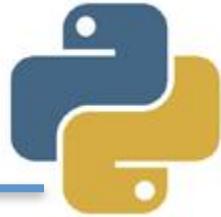
```
7  class Gibi(object):
8      """
9          Classe padrão para dados de GIBIs
10         """
11     def __init__(self, id, numero, titulo,
12                  personagem, editora):
13         self.id = id
14         self.numero = numero
15         self.titulo = titulo
16         self.personagem = personagem
17         self.editora = editora
18 
```

Exercício 3: A classe Gibi



```
7  class Gibi(object):
8      """
9          Classe padrão para dados de GIBIs
10         """
11     def __init__(self, id, numero, titulo,
12                  personagem,
13
14      Podemos documentar
15      nossas classes e métodos
16      com docstrings
17
18
```





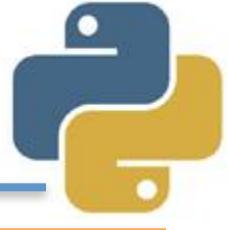
Exercício 3: A classe Gibi

```
7  class Gibi(object):
8      """
9          Classe
10         ...
11     def __i
12         personagem.
13         self.id = id
14         self.numero = numero
15         self.titulo = titulo
16         self.personagem = personagem
17         self.editora = editora
18     """
```

Definição de
atributos de
instância



Exercício 4: Método estático



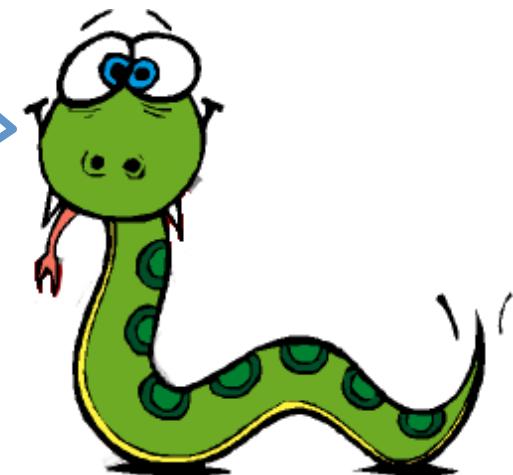
```
19     @staticmethod
20     def carregar_lista_gibi():
21         lista = []
22         arquivo = open("gibis.csv", "r")
23         for linha in arquivo:
24             linha = linha.strip().split(";")
25             if linha[0] != 'ID':
26                 gibi = Gibi(linha[0], linha[1],
27                             linha[2], linha[3],
28                             linha[4])
29                 lista.append(gibi)
30         arquivo.close()
31     return lista
```

Exercício 4: Método estático



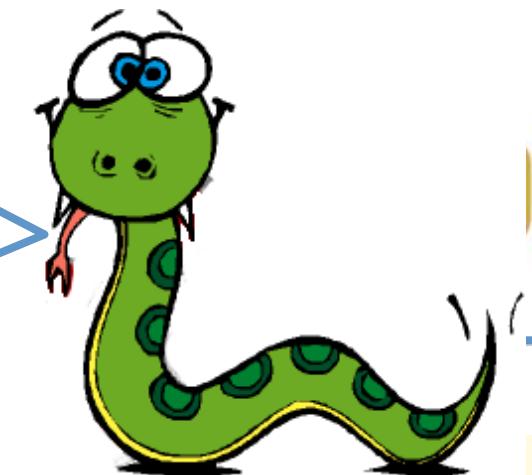
```
19 @staticmethod  
20     def carregar_lista_gibi():  
21         lista = []  
22         arquivo = open('gibis.txt', 'r')  
23         for linha in arquivo:  
24             gibis = linha.split(',')  
25             for gibi in gibis:  
26                 gibi = gibi.replace('\n', '')  
27                 lista.append(gibi)  
28             arquivo.close()  
29             return lista  
30  
31         lista.append(gibi)
```

Na classe **Gibi**,
crie o **método**
estático



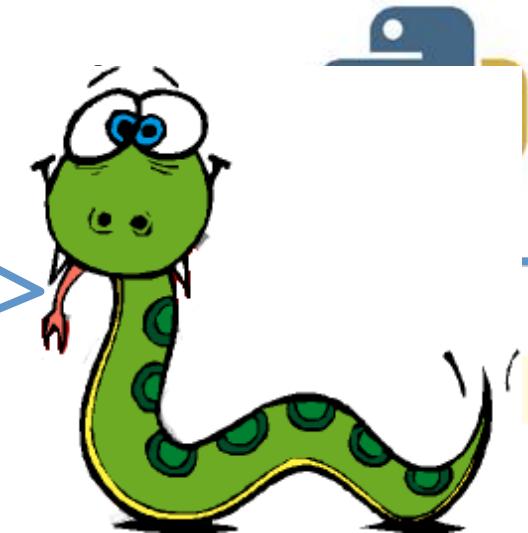
E

Vamos **ler o arquivo** e
guardar os dados em
uma **lista**



```
19
20
21 lista = []
22 arquivo = open("gibis.csv", "r")
23 for linha in arquivo:
24     linha = linha.strip().split(";")
25     if linha[0] != 'ID':
26         gibi = Gibi(linha[0], linha[1],
27                     linha[2], linha[3],
28                     linha[4])
29         lista.append(gibi)
30     arquivo.close()
31 return lista
```

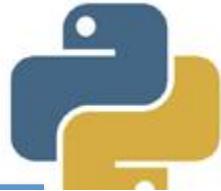
Exercício 4: Método especial



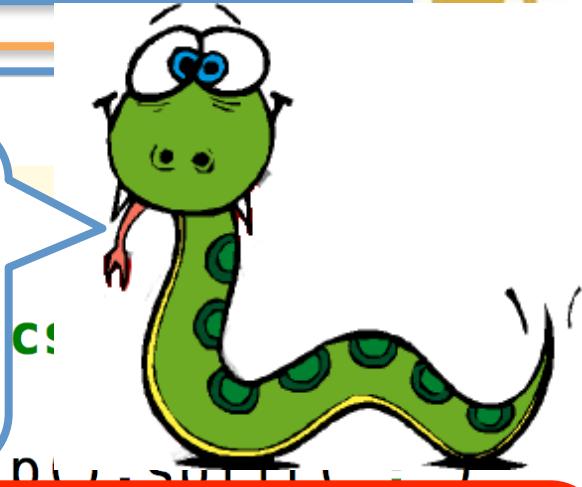
Cada **linha** do arquivo se torna uma **lista de atributos**

```
for linha in arquivo:  
    linha = linha.strip().split(';')  
    if linha[0] != 'ID':  
        gibi = Gibi(linha[0], linha[1],  
                    linha[2], linha[3],  
                    linha[4])  
        lista.append(gibi)  
arquivo.close()  
return lista
```

Exercício 4: Método estático



Vamos criar um Gibi com os **atributos extraídos** da **linha** do arquivo



```
if linha[0] != 'ID':
```

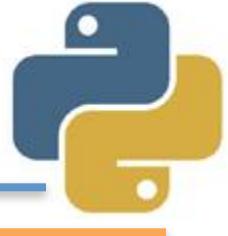
```
    gibi = Gibi(linha[0], linha[1],  
                linha[2], linha[3],  
                linha[4])
```

```
    lista.append(gibi)
```

```
arquivo.close()
```

```
return lista
```

Exercício 4: Método estático



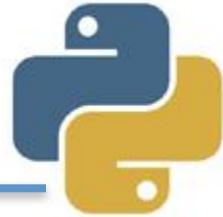
```
19     @staticmethod  
20     def carregar_lista_gibi():  
21         lista = []  
22         arquivo = open("gibis.cs")  
23         for linha in arquivo:  
24             gibis.append(linha)  
25             lista.append(gibi)  
26             print(gibi)  
27             print(lista)  
28             print(gibis)  
29             print(linha)  
30             print(arquivo)  
31             return lista
```

Cada gibi criado
ficará na lista de
gibis

lista.append(gibi)



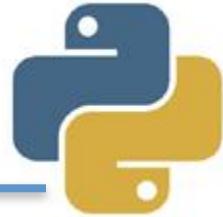
Exercício 5: Teste de carga



```
model.py x gibis.csv x teste_leitura.py x

1 #-*- coding: UTF-8 -*-
2 # teste_leitura.py
3 from model import Gibi
4
5 lista = Gibi.carregar_lista_gibi()
6
7 for gibi in lista:
8     print gibi
9
```

Exercício 5: Teste de carga



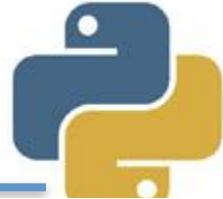
```
model.py x gibis.csv x teste_leitura.py x
```

```
1 #-*- coding: UTF-8 -*-
2 # teste_leitura.py
3 from model import Gibi
4
5 Vamos atualizar o
6 arquivo teste_leitura.py
7
8 print(gibis)
9
```

Vamos atualizar o arquivo **teste_leitura.py**

A cartoon illustration of a green snake with large eyes and a yellow belly, pointing towards the text in the speech bubble.

Exercício 5: Teste de carga



Chame o **método estático** da classe **Gibi** e **imprima** os dados de **cada objeto**

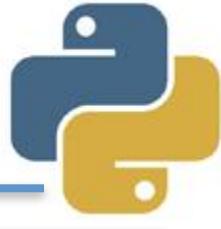


```
1  
2  
3  
4  
5 lista = Gibi.carregar_lista_gibi()  
6  
7 for gibi in lista:  
8     print gibi  
9
```



Resultado no console

```
Run teste_leitura
/Library/Frameworks/Python.framework/Vers
<model.Gibi object at 0x1005d0950>
<model.Gibi object at 0x1005d0990>
<model.Gibi object at 0x1005d0a10>
<model.Gibi object at 0x1005d0a90>
<model.Gibi object at 0x1005d0ad0>
<model.Gibi object at 0x1005d0b50>
<model.Gibi object at 0x1005d0bd0>
<model.Gibi object at 0x1005d0c50>
<model.Gibi object at 0x1005d0cd0>
<model.Gibi object at 0x1005d0d10>
```



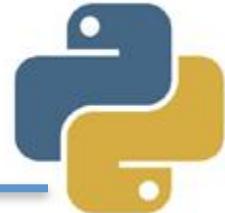
Resultado no console

```
Run teste_leitura
/Lib/Python3.7/site-packages/gibidev/api.py:10: DeprecationWarning: The
<model.Gibi object at 0x1005d0950>
<model.Gibi object at 0x1005d0990>
<model.Gibi object at 0x1005d0a10>
<model.Gibi object at 0x1005d0a20>
<model.Gibi object at 0x1005d0a30>
<model.Gibi object at 0x1005d0a40>
<model.Gibi object at 0x1005d0a50>
<model.Gibi object at 0x1005d0a60>
<model.Gibi object at 0x1005d0a70>
<model.Gibi object at 0x1005d0a80>
<model.Gibi object at 0x1005d0a90>
<model.Gibi object at 0x1005d0aa0>
<model.Gibi object at 0x1005d0ab0>
<model.Gibi object at 0x1005d0ac0>
<model.Gibi object at 0x1005d0ad0>
<model.Gibi object at 0x1005d0ae0>
<model.Gibi object at 0x1005d0af0>
<model.Gibi object at 0x1005d0b00>
<model.Gibi object at 0x1005d0b10>
<model.Gibi object at 0x1005d0b20>
<model.Gibi object at 0x1005d0b30>
<model.Gibi object at 0x1005d0b40>
<model.Gibi object at 0x1005d0b50>
<model.Gibi object at 0x1005d0b60>
<model.Gibi object at 0x1005d0b70>
<model.Gibi object at 0x1005d0b80>
<model.Gibi object at 0x1005d0b90>
<model.Gibi object at 0x1005d0ba0>
<model.Gibi object at 0x1005d0bb0>
<model.Gibi object at 0x1005d0bc0>
<model.Gibi object at 0x1005d0bd0>
<model.Gibi object at 0x1005d0be0>
<model.Gibi object at 0x1005d0bf0>
<model.Gibi object at 0x1005d0c00>
<model.Gibi object at 0x1005d0c10>
<model.Gibi object at 0x1005d0c20>
<model.Gibi object at 0x1005d0c30>
<model.Gibi object at 0x1005d0c40>
<model.Gibi object at 0x1005d0c50>
<model.Gibi object at 0x1005d0c60>
<model.Gibi object at 0x1005d0c70>
<model.Gibi object at 0x1005d0c80>
<model.Gibi object at 0x1005d0c90>
<model.Gibi object at 0x1005d0ca0>
<model.Gibi object at 0x1005d0cb0>
<model.Gibi object at 0x1005d0cc0>
<model.Gibi object at 0x1005d0cd0>
<model.Gibi object at 0x1005d0ce0>
<model.Gibi object at 0x1005d0cf0>
<model.Gibi object at 0x1005d0d00>
<model.Gibi object at 0x1005d0d10>
```

Os dados **não** estão muito **legíveis**. Podemos **melhorar** isso.

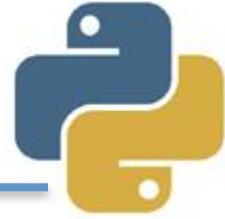


Imprimindo objetos



- Em **Python**, representamos objetos com **Strings**, **sobrescrevendo** os métodos:
 - **`__repr__`** devolve a **String** com a **representação formal** do objeto. Método acessado via **`repr(obj)`**. Usado pelo **`eval()`** para criação de **instâncias**
 - **`__str__`** representaçāo **informal** do objeto
-

Exercício 6: Representação



- Atualize a classe **Gibi**, com os métodos para representações **formal** e **informal**

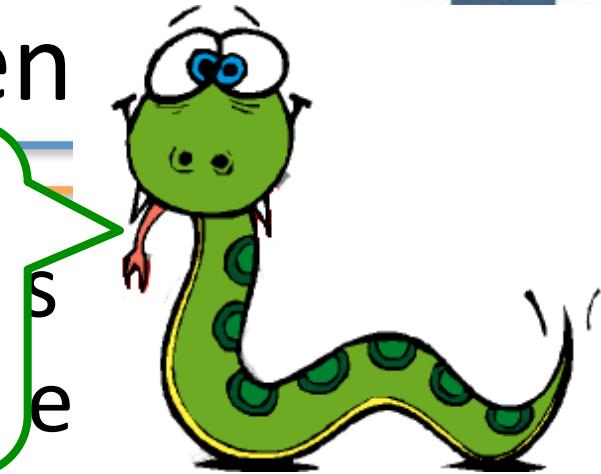
```
def __str__(self):
    return self.titulo + " - " + self.personagem \
           + " [" + self.editora + "]"

def __repr__(self):
    return self.__class__.__name__+ \
        '(id=%s, numero=%s, titulo=%s, ' \
        'personagem=%s, editora=%s)' \
        % (self.id, self.numero, self.titulo,
           self.personagem, self.editora)
```

Exercício 6: Represen

- Atua para

Sobrescreva o método `__str__`, que será chamado pelo `print()`



```
def __str__(self):  
    return self.titulo + " - " + self.personagem \  
        + " [" + self.editora + "]"
```

```
def __repr__(self):  
    return self.__class__.__name__+ \  
        '(id=%s, numero=%s, titulo=%s, ' \  
        'personagem=%s, editora=%s)' \  
        % (self.id, self.numero, self.titulo,  
            self.personagem, self.editora)
```

Exercício 6: Represen

- Atua para gerar a String para criação de Gibis

Sobrescreva o `__repr__`

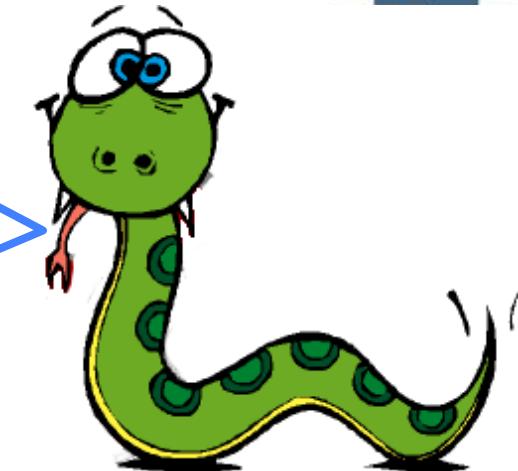


```
def __str__(self):  
    return self.titulo + " - " + self.personagem +  
           " [" + self.editora + "]"  
  
def __repr__(self):  
    return self.__class__.__name__ + '  
        '(id=%s, numero=%s, titulo=%s, ' + '\n  
        'personagem=%s, editora=%s)' + '\n  
        % (self.id, self.numero, self.titulo,  
             self.personagem, self.editora)
```

Exercício 6: Represen

- Atua para

A “\” indica que o comando continua na próxima linha



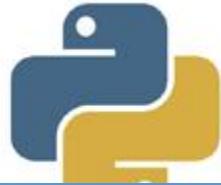
```
def __str__(self):  
    return self.titulo + " - " + self.personagem + "\\\n" + "[" + self.editora + "]"  
  
def __repr__(self):  
    return self.__class__.__name__ + '\n' + '(id=%s, numero=%s, titulo=%s,\n' + '\n' + 'personagem=%s, editora=%s)' + '\n' + '% (self.id, self.numero, self.titulo,\n' + '\n' + self.personagem, self.editora)
```

Execute o teste novamente



Run teste_leitura

```
/Library/Frameworks/Python.framework/Versions/2.7/bin/
"MEMORIA APAGADA" - "WOLVERINE" ["MARVEL COMICS"]
"ENTERRADO VIVO" - "WOLVERINE" ["MARVEL COMICS"]
"SAMURAIS EM GUERRA" - "WOLVERINE" ["MARVEL COMICS"]
"CODINOME WOLVERINE" - "WOLVERINE" ["MARVEL COMICS"]
" MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"O ULTIMO FILHO" - SUPERMAN ["DC COMICS"]
"A MORTE DO ROBIN" - BATMAN ["DC COMICS"]
"A MASCARA DA MORTE" - BATMAN ["DC COMICS"]
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN [
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]
" MORCEGO VERSUS MORCEGO" - BATMAN ["DC COMICS"]
```



Execute o teste novamente

Run teste_leitura

```
/Library/Frameworks/Python.framework/Versions/2.7/bin/
```

```
"MEMORIA APAGADA" - "WOLVERINE" ["MARVEL COMICS"]  
"ENTERRADO VIVO" - "WOLVERINE" ["MARVEL COMICS"]  
"SAMURAIS EM GUERRA" - "WOLVERINE" ["MARVEL COMICS"]  
"CODINOME WOLVERINE" - "WOLVERINE" ["MARVEL COMICS"]
```

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MA
```

```
"A MO
```

```
"A MO
```

```
"A MO
```

```
"CORF
```

```
"O UL
```

```
"A MO
```

```
"A MASCARA DA TURTLE
```

```
BATMAN [ "DC COMI
```

```
DC
```

```
DC
```

```
I
```

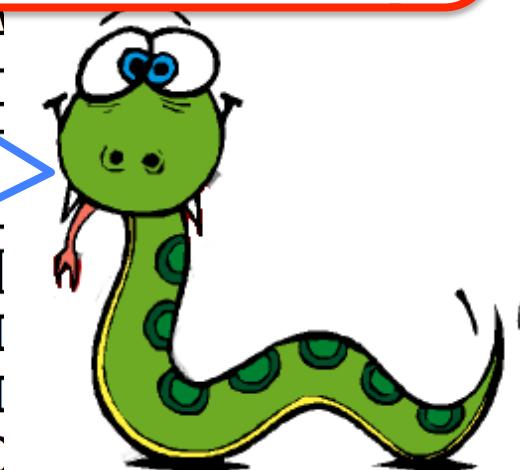
```
MI
```

```
MI
```

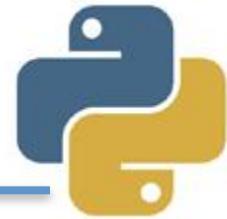
```
CC
```

```
CC
```

Quando executamos
print(gibi), o método
print invoca **gibi.__str__**

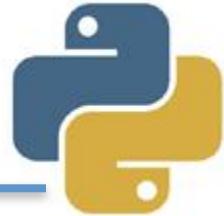


```
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN [  
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]  
"MORCEGO VERSUS MORCEGO" - BATMAN ["DC COMICS"]
```



Exercício 7: testando eval()

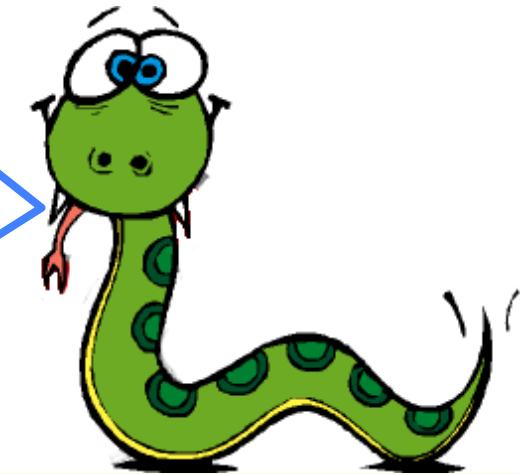
```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = Gibi.carregar_lista_gibi()
6  for gibi in lista:
7      print gibi
8
9  representacao = repr(lista[0])
10 print "\nRepresentação Formal:\n", representacao
11
12 print "\nTeste com eval()"
13 a = eval(representacao)
14 print "type(a):", type(a)
15 print "repr(a):", repr(a)
16
```

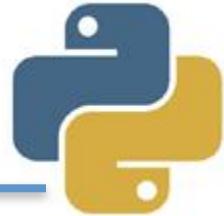


Exercício 7: testando eval()

```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = Gibi.carre
6  for gibi in lista:
7      print gibi
8
9  representacao = repr(lista[0])
10 print "\nRepresentação Formal:\n", representacao
11
12 print "\nTeste com eval()"
13 a = eval(representacao)
14 print "type(a):", type(a)
15 print "repr(a):", repr(a)
```

Pega o 1º
Gibi da lista

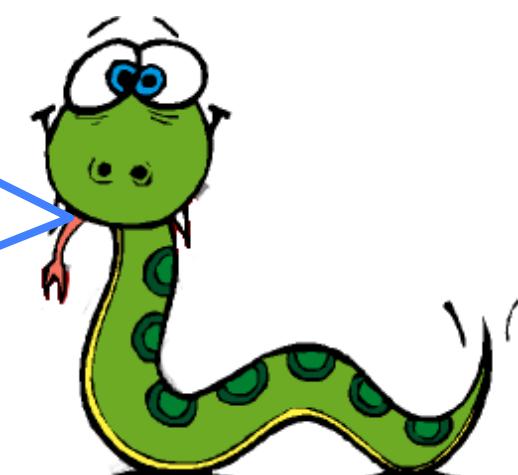


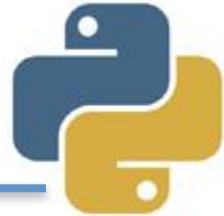


Exercício 7: testando eval()

```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = G
6  for gibi
7      print
8
9  representacao = repr(lista[0])
10 print "\nRepresentação Formal:\n", representacao
11
12 print "\nTeste com eval()"
13 a = eval(representacao)
14 print "type(a):", type(a)
15 print "repr(a):", repr(a)
```

Chama o método
gibi.__repr__()

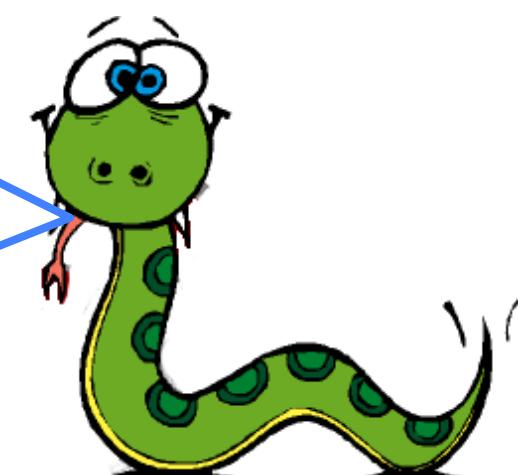


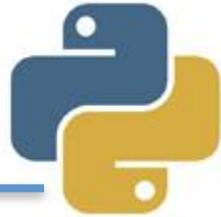


Exercício 7: testando eval()

```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = G
6  for gibi
7      print
8
9  representacao = repr(lista[0])
10 print "\nRepresentação Formal:\n", representacao
11
12 print "\nTeste com eval()"
13 a = eval(representacao)
14 print "type(a):", type(a)
15 print "repr(a):", repr(a)
16
```

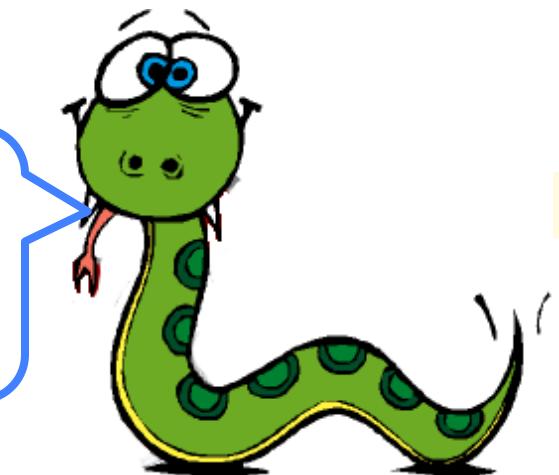
Imprime o resultado de `gibi.__repr__()`

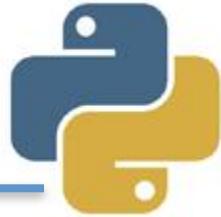




Exercício 7: testando eval()

```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = Gibi.carregar_lista_gibi()
6  for gibi in lista:
7      print gibi
8
9  Cria um objeto Gibi, a partir do
10 resultado de gibi.__repr__()
11
12 print type(gibi)
13 a = eval(representacao)
14 print "type(a):", type(a)
15 print "repr(a):", repr(a)
16
```

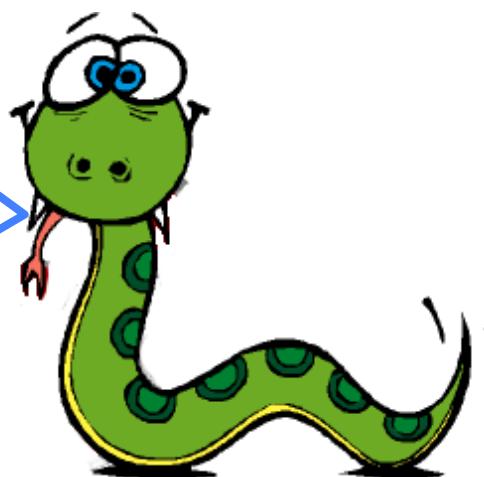




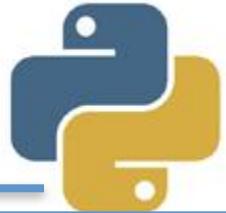
Exercício 7: testando eval()

```
1  #-*- coding: UTF-8 -*-
2  # teste_leitura.py
3  from model import Gibi
4
5  lista = Gibi.carregar_lista_gibi()
6  for gibi in lista:
7      print gibi
8
9
10
11
12
13
14  print "type(a):", type(a)
15  print "repr(a):", repr(a)
```

Impressão do **tipo** e da
representação de “a”



cao



Resultado no console

Run teste_leitura

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"O ULTIMO FILHO" - SUPERMAN ["DC COMICS"]
"A MORTE DO ROBIN" - BATMAN ["DC COMICS"]
"A MASCARA DA MORTE" - BATMAN ["DC COMICS"]
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN ["DC COMICS"]
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]
"MOREGO VERSUS MOREGO" - BATMAN ["DC COMICS"]
```

Representação Formal:

```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE",
```

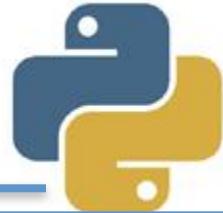
Teste com eval()

```
type(a): <class 'model.Gibi'>
```

```
repr(a): Gibi(id=11, numero=24, titulo=MEMORY APAGADA, personagem=WOLVE
```

```
Process finished with exit code 0
```

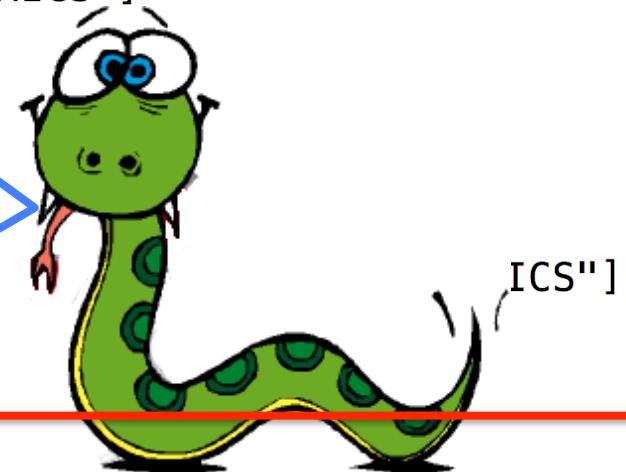
Resultado no console



Run teste_leitura

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]  
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]  
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]  
"A MORTE DO SUPERMAN" - SUPERMAN ["DC C  
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC  
"O ULTIMO REINHO" - SUPERMAN ["DC COMICS"]
```

Resultado do comando
`repr(lista[0])`



Representação Formal:

```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE",
```

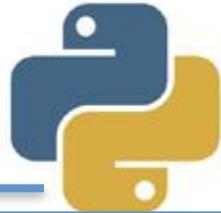
Teste com eval()

```
type(a): <class 'model.Gibi'>
```

```
repr(a): Gibi(id=11, numero=24, titulo=MEMORIA APAGADA, personagem=WOLVE
```

```
Process finished with exit code 0
```

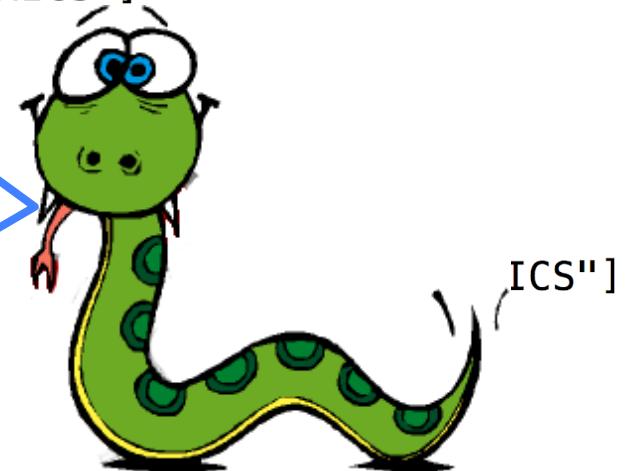
Resultado no console



Run teste_leitura

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC C
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC
"O ULTIMO REINHO" - SUPERMAN ["DC COMICS"]
```

Resultado dos comandos
`type(a)` e `repr(a)`



Representação Formal:

```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE",
```

Teste com `eval()`

```
type(a): <class 'model.Gibi'>
repr(a): Gibi(id=11, numero=24, titulo=MEMORIA APAGADA, personagem=WOLVE
```

Process finished with exit code 0

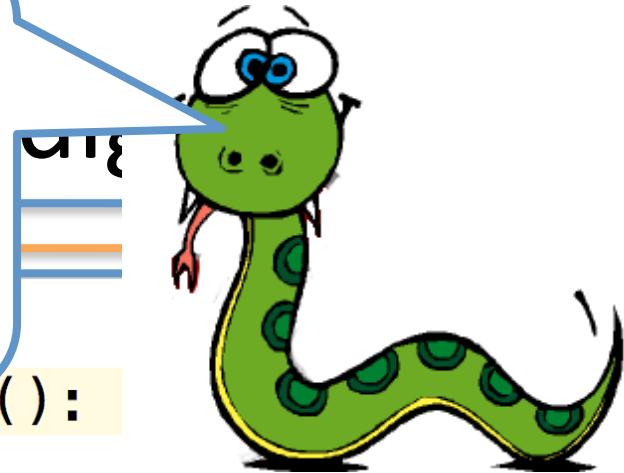
Revisando o código



```
19     @staticmethod
20     def carregar_lista_gibi():
21         lista = []
22         arquivo = open("gibis.csv", "r")
23         for linha in arquivo:
24             linha = linha.strip().split(";")
25             if linha[0] != 'ID':
26                 gibi = Gibi(linha[0], linha[1],
27                               linha[2], linha[3],
28                               linha[4])
29                 lista.append(gibi)
30         arquivo.close()
31     return lista
```

De volta ao método
`carregar_lista_gibi()`.

Que **melhorias** poderíamos
implementar?



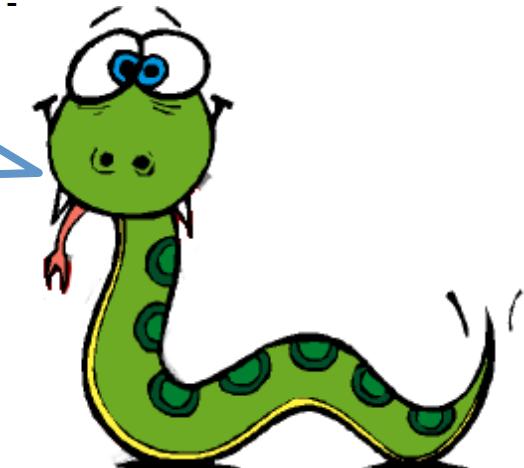
```
19
20     def carregar_lista_gibi():
21         lista = []
22         arquivo = open("gibis.csv", "r")
23         for linha in arquivo:
24             linha = linha.strip().split(";")
25             if linha[0] != 'ID':
26                 gibi = Gibi(linha[0], linha[1],
27                             linha[2], linha[3],
28                             linha[4])
29                 lista.append(gibi)
30         arquivo.close()
31     return lista
```



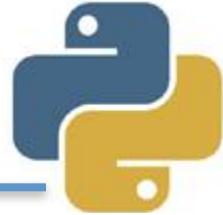
Revisando o código

```
19      @staticmethod  
20  def carregar_lista_gibi():  
21      lista = []  
22      arquivo = open("gibis.csv", "r")  
23      for linha in arquivo:  
24          linha = linha.strip().split(";")  
25          if linha[0] != 'ID':  
26              lin  
27              lin(gi  
28  
29  
30  
31      return lista
```

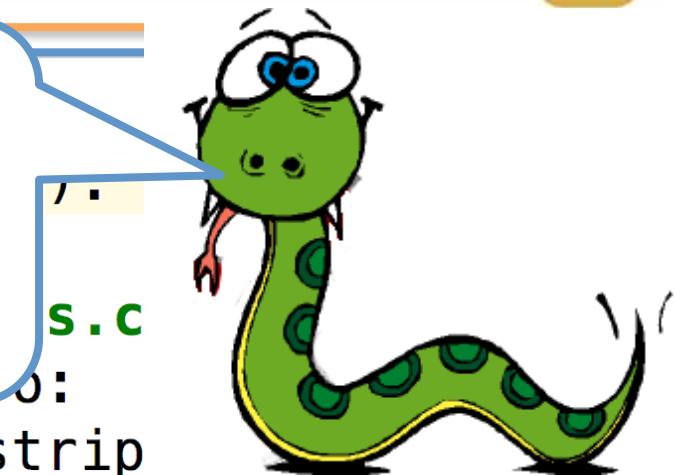
Poderíamos usar
documentação com
docstrings



Revisando o código

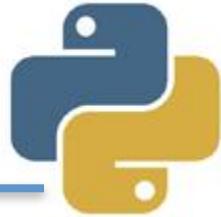


Poderíamos usar * para passar todos os atributos de uma vez só



```
    linha = linha.strip
    if linha[0] != 'ID':
        gibi = Gibi(linha[0], linha[1],
                    linha[2], linha[3],
                    linha[4])
        lista.append(gibi)
```

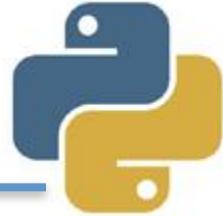
```
    arquivo.close()
    return lista
```



Exercício 8: docstrings e atributos

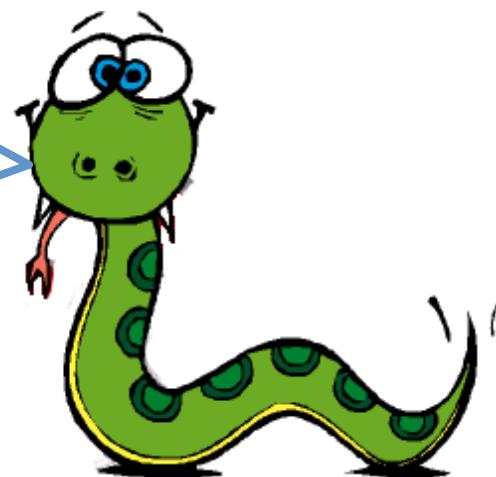
```
@staticmethod
def carregar_lista_gibi(nome_arquivo):
    """Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo a ser carregado.
    Returns:
        list: Lista contendo Gibis."""
    lista = []
    arquivo = open(nome_arquivo, "r")
    for linha in arquivo:
        linha = linha.strip().split(";")
        if linha[0] != 'ID':
            gibi = Gibi(*linha)
            lista.append(gibi)
    arquivo.close()
    return lista
```

Exercício 8: docstrings e atributos

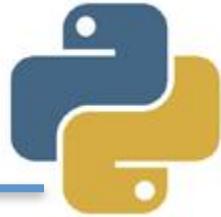


```
@staticmethod  
def carregar_lista_gibi(nome_arquivo):  
    """Função para carga de lista de GIBIS  
  
    Para que nosso método  
    fique mais dinâmico, vamos  
    pedir o nome do arquivo  
  
    for linha in arquivo:  
        linha = linha.strip().split(';')  
        if linha[0] != 'ID':  
            gibi = Gibi(*linha)  
            lista.append(gibi)  
    arquivo.close()  
    return lista
```

Para que nosso método
fique mais dinâmico, vamos
pedir o nome do arquivo



ado.



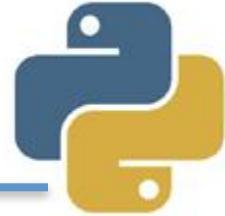
Exercício 8: docstrings e atributos

```
@staticmethod
def carregar_lista_gibi(nome_arquivo):
    """Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo a ser carregado.
    Returns:
        list: Lista contendo Gibis."""
    lista = []
    arquivo = open(nome_arquivo)
    for linha in arquivo:
        lista.append(linha)
    arquivo.close()
    return lista
```

Google Style Python
Docstrings



Exercício 8: docstrings e atributos



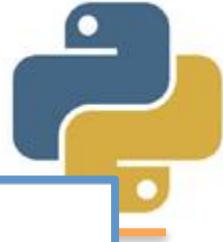
```
@staticmethod
def carregar_lista_gibi(nome_arquivo):
    """Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo a ser carregado.
    """
    lista = []
    with open(nome_arquivo, "r") as arquivo:
        for linha in arquivo:
            gibis = Gibi(*linha)
            lista.append(gibi)
    arquivo.close()
    return lista
```

Passando a **lista de atributos** da **linha** para o **construtor** de Gibi

gibi = Gibi(*linha)



Exercício 8: Atualização de testes



```
#-*- coding: UTF-8 -*-
# teste_leitura.py
from model import Gibi

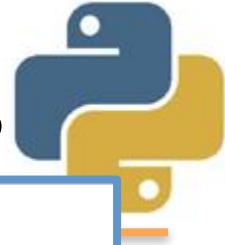
help(Gibi.carregar_lista_gibi)
```

Usando **help()** para
mostrar a documentação
do método

```
print "\nRepresentação Formal:\n"
print "\nTeste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)
```



Exercício 8: Atualização de testes

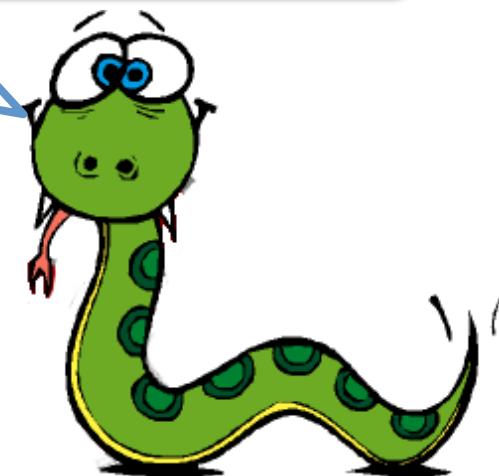


```
#-*- coding: UTF-8 -*-
# teste_leitura.py
from model import Gibi

help(Gibi.carregar_lista_gibi)
```

Agora, precisamos informar
o nome do arquivo

+("gibis.csv")



```
representacao = repr(lista[0])
print "\nRepresentação Formal:\n"
print "\nTeste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)
```

Resultado no console

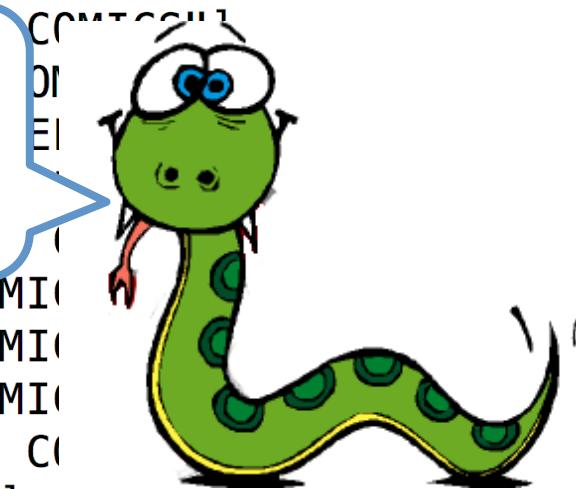


```
Run teste_leitura
> /Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
Help on function carregar_lista_gibi in module model:

carregar_lista_gibi(nome_arquivo)
    Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo a ser carregado.
    Returns:
        list: Lista contendo Gibis.

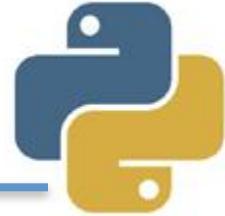
"""
"""
    Agora, temos acesso à
    documentação do método
"""

"A MORTE DO SUPERMAN" -> SUPERMAN [{"DC COMICS": "A MORTE DO SUPERMAN"}, {"DC COMICS": "CORPORACAO DO SUPERMAN"}, {"DC COMICS": "O ULTIMO ETIHO"}]
```



Agora, temos acesso à documentação do método

Gibis em formato americano



- Gibis em formato americano (17cm x 26cm) Podem conter capa especial
 - Precisamos criar uma classe, filha da classe Gibi, para representar essa linha
 - Vamos criar a GibiFormatoAmericano, no arquivo model.py
-

Exerc. 9: GibiFormatoAmericano



```
class GibiFormatoAmericano(Gibi):
    """
        Classe padrão para dados de GIBIs em Formato Americano
    """

    def __init__(self, id, numero, titulo,
                 personagem, editora, capa_especial=None):

        super(GibiFormatoAmericano, self).__init__(id,
                                                numero, titulo, personagem, editora)

        self.capa_especial = capa_especial

    def __repr__(self):
        return self.__class__.__name__ + \
            '(id=%s, numero=%s, titulo=%s, ' \
            'personagem=%s, editora=%s, ' \
            'capa_especial=%s) ' \
            % (self.id, self.numero, self.titulo,
               self.personagem, self.editora, self.capa_especial)
```

Exerc. 9: GibiFormatoAmericano



```
class GibiFormatoAmericano(Gibi):
    """
    Classe filha de Gibi
    """

    def __init__(self, id, numero, titulo, personagem, editora, capa_especial):
        super(GibiFormatoAmericano, self)
            numero, titulo, personagem, editora,
            self.capa_especial = capa_especial

    def __repr__(self):
        return self.__class__.__name__ + \
            '(id=%s, numero=%s, titulo=%s, ' \
            'personagem=%s, editora=%s, ' \
            'capa_especial=%s) ' \
            % (self.id, self.numero, self.titulo,
               self.personagem, self.editora, self.capa_especial)
```



Exerc. 9: GibiFormatoAmericano



```
class GibiFormatoAmericano(Gibi):
    """
        Classe padrão para dados de GIBIs em Formato Americano
    """

    def __init__(self, id, numero, titulo,
                 personagem, editora, capa_especial=None):

        super(GibiFormatoAmericano, self).__init__(id,
                                                numero, titulo, personagem, editora)

        self.capa_especial = capa_especial

    def __repr__(self):
        return "%s(%s, %s, %s, %s, %s, %s)" % (self.id, self.numero, self.titulo,
                                                self.personagem, self.editora, self.capa_especial)
```

Método **construtor**



Exerc. 9: GibiFormatoAmericano



```
class GibiFormatoAmericano(Gibi):
    """
        Classe padrão para dados de GIBIs em Formato Americano
    """

    def __init__(self, id, numero, titulo,
                 personagem, editora, capa_especial=None):
        super(GibiFormatoAmericano, self).__init__(id,
                                                numero, titulo, personagem, editora)
```

Chamada ao **construtor** da **classe Pai**

```
        self.titulo = titulo
        self.numero = numero
        self.id = id
        self.editora = editora
        self.personagem = personagem
        self.capa_especial = capa_especial
```



Exerc. 9: GibiFormatoAmericano



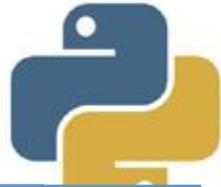
```
class GibiFormatoAmericano(Gibi):
    """
        Classe padrão para dados de GIBIs em Formato Americano
    """

    def __init__(self, id, numero, titulo,
                 personagem, editora, capa_especial):
        self.id = id
        self.numero = numero
        self.titulo = titulo
        self.personagem = personagem
        self.editora = editora
        self.capa_especial = capa_especial

    def __repr__(self):
        return self.__class__.__name__ + \
            '(id=%s, numero=%s, titulo=%s, ' \
            'personagem=%s, editora=%s, ' \
            'capa_especial=%s)' \
            % (self.id, self.numero, self.titulo,
               self.personagem, self.editora, self.capa_especial)
```

Atributo específico para
Formato Americano





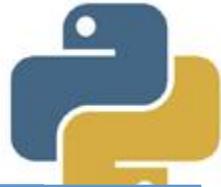
Exercício 10: Teste nova classe

```
#-*- coding: UTF-8 -*-
# teste_leitura.py
from model import GibiFormatoAmericano

lista = GibiFormatoAmericano.carregar_lista_gibi()
for gibi in lista:
    print gibi

representacao = repr(lista[0])
print "\nRepresentação Formal:\n", representacao

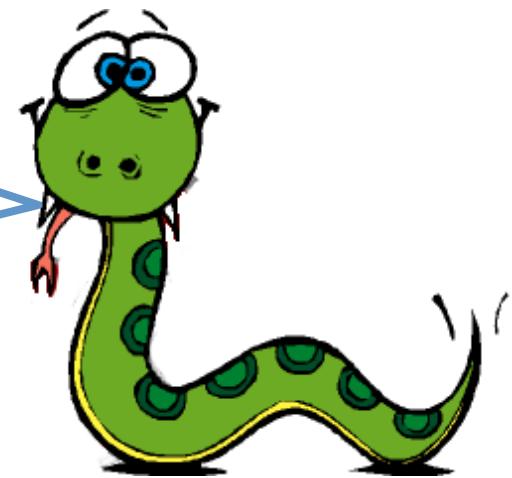
print "\nTeste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)
```



Exercício 10: Teste nova classe

```
#-*- coding: UTF-8 -*-
# teste_leitura.py
from model import GibiFormatoAmericano
lista = GibiFormatoAmericano.carregar_lista_gibi()
for gibi in lista:
    print "\nTeste com eval()"
    a = eval(representacao)
    print "type(a):", type(a)
    print "repr(a):", repr(a)
```

Altere a classe de **Gibi** para
GibiFormatoAmericano





Resultado no console

```
"A MORTE DO ROBIN" - BATMAN ["DC COMICS"]
"A MASCARA DA MORTE" - BATMAN ["DC COMICS"]
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN ["DC COMICS"]
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]
"MOREGO VERSUS MOREGO" - BATMAN ["DC COMICS"]
```

Repre
Gibi(

Teste
Trace
File

O Interpretador está
pedindo a classe Gibi.
Mas por que?

```
a = eval(representacao)
File "<string>", line 1, in <module>
```

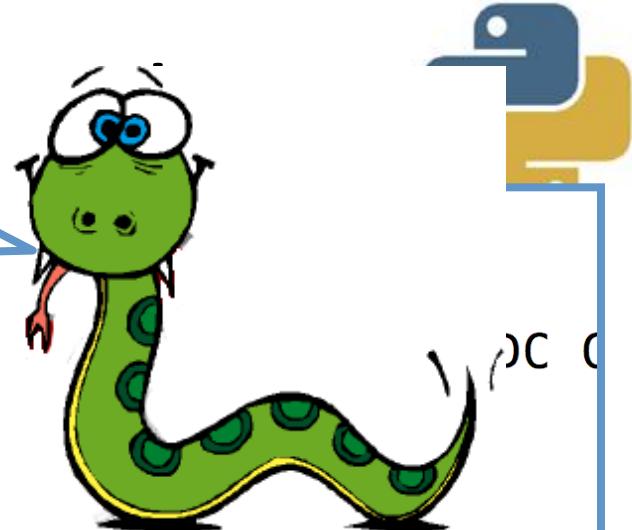
```
NameError: name 'Gibi' is not defined
```

```
Process finished with exit code 1
```



Resultado no console

```
"A MORCEGO VERSUS MORCEGO - BATMAN"  
"A M  
"A OR  
"A VI  
"MORCEGO VERSUS MORCEGO - BATMAN  
Veja que o __repr__  
está retornando Gibi
```



Representação Formal:

```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagens=[{"DC": "GIBI", "COMIC": "GIBI", "COR": "VERDE", "BATMAN": "GIBI", "VERSA": "GIBI"}])
```

Teste com eval()

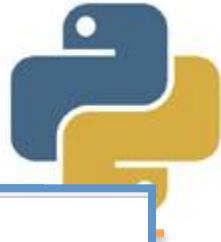
Traceback (most recent call last):

```
File "/Users/marciopalheta/python/codigofonte/financeiro/gibi.py", line 1, in <module>  
    a = eval(representacao)
```

```
File "<string>", line 1, in <module>
```

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

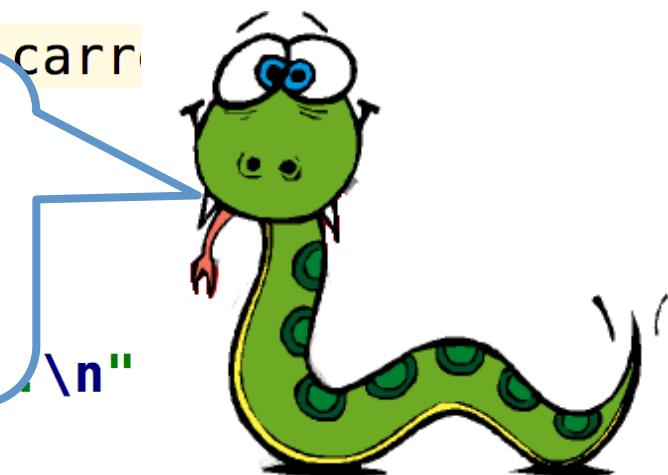
```
Process finished with exit code 1
```



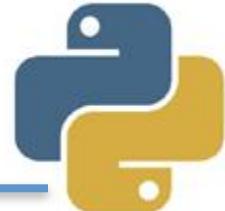
Exercício 10: Teste nova classe

```
#-*- coding: UTF-8 -*-
# teste_leitura.py
from model import GibiFormatoAmericano
from model import Gibi
lista = GibiFormatoAmericano.carr
for Gibi in lista:
    print Gibi
    print Gibi.representacao
    print "print \"%s\" % Gibi" % Gibi.representacao
    print "print \"\nTeste com eval()\""
    print "a = eval(representacao)"
    print "print \"type(a):\"," type(a)
    print "print \"repr(a):\"," repr(a)
```

Retorne o **import**,
mesmo **sem uso**
aparente



Execute novamente



Run teste_leitura

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"O ULTIMO FILHO" - SUPERMAN ["DC COMICS"]
"A MORTE DO ROBIN" - BATMAN ["DC COMICS"]
"A MASCARA DA MORTE" - BATMAN ["DC COMICS"]
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN ["DC COMICS"]
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]
"MOREGO VERSUS MOREGO" - BATMAN ["DC COMICS"]
```

Representação Formal:

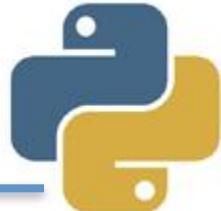
```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE",
```

```
Teste com eval()
```

```
type(a): <class 'model.Gibi'>
```

```
repr(a): Gibi(id=11, numero=24, titulo=MEMORY APAGADA, personagem=WOLVE
```

```
Process finished with exit code 0
```

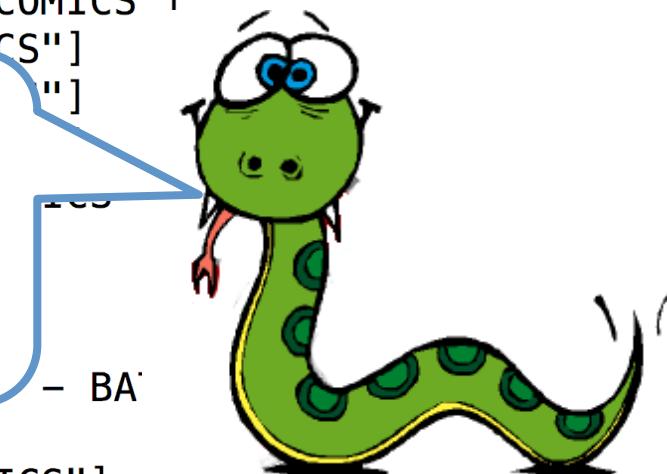


Execute novamente

Run teste_leitura

```
"MORTE EM FAMILIA" - "WOLVERINE" ["MARVEL COMICS"]  
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]  
"A MORTE DE CATWOMAN" - CATWOMAN ["DC COMICS"]  
"A MORTE DE SPIDER-MAN" - SPIDER-MAN ["MARVEL COMICS"]  
"CORPORACAO VENDEU A MULHER MARAVilha" - MULHER MARAVilha ["DC COMICS"]  
"O ULTIMO LAR" - LAR [ ]  
"A MORTE DE CATWOMAN" - CATWOMAN ["DC COMICS"]  
"A MASCARA DO CRIMSON" - CRIMSON ["DC COMICS"]  
"A ORIGEM DE SPIDER-MAN" - SPIDER-MAN ["MARVEL COMICS"]  
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]  
"MORCEGO VERSUS MORCEGO" - BATMAN ["DC COMICS"]
```

Voltou a funcionar,
mas continuamos
com Objetos Gibi



Representação Formal:

```
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE",
```

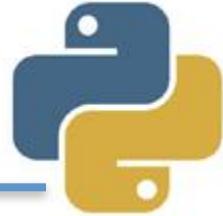
Teste com eval()

```
type(a): <class 'model.Gibi'>
```

```
repr(a): Gibi(id=11, numero=24, titulo=MEMORIA APAGADA, personagem=WOLVE
```

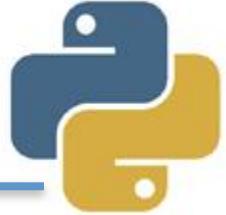
```
Process finished with exit code 0
```

Revisando o código



```
@staticmethod
def carregar_lista_gibi(nome_arquivo):
    """Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo a ser carregado.
    Returns:
        list: Lista contendo Gibis."""
    lista = []
    arquivo = open(nome_arquivo, "r")
    for linha in arquivo:
        linha = linha.strip().split(";")
        if linha[0] != 'ID':
            gibi = Gibi(*linha)
            lista.append(gibi)
    arquivo.close()
    return lista
```

Revisando o código

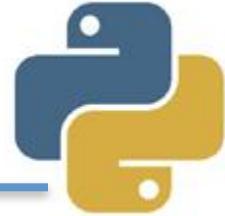


```
@staticmethod  
def carregar_lista_gibi(nome_arquivo):  
    """Função para carga de lista de GIBIS  
    Args:  
        nome_arquivo (str): Nome do arquivo  
    Returns:  
        lista (list): Lista de Gibis  
    """  
    lista = []  
    with open(nome_arquivo, 'r') as arquivo:  
        for linha in arquivo:  
            linha = linha.strip()  
            if linha[0] != '#':  
                gabi = Gabi(*linha)  
                lista.append(gabi)  
    arquivo.close()  
    return lista
```

Nosso método estático só cria objetos Gibi

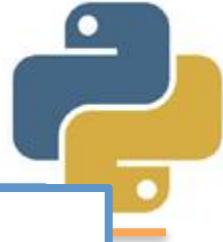


@staticmethod e @classmethod



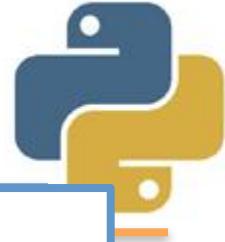
- `@staticmethod` são métodos que pertencem à classe, mas **não sabem** nada a **respeito do objeto** que o executa
 - `@classmethod` métodos que também pertencem à classe, mas recebem uma **referência para a classe** que o executa, como **1º parâmetro**
-

Exercício 11: @classmethod



```
@classmethod
def carregar_lista_gibi(cls, nome_arquivo):
    """Função para carga de lista de GIBIS
    Args:
        nome_arquivo (str): Nome do arquivo
    Returns:
        list: Lista contendo Gibis.
    """
    lista = []
    arquivo = open(nome_arquivo, "r")
    for linha in arquivo:
        linha = linha.strip().split(";")
        if linha[0] != 'ID':
            gibi = cls(*linha)
            lista.append(gibi)
    arquivo.close()
    return lista
```

Exercício 11: @classmethod



@classmethod

```
def carregar_lista_gibi(cl
```

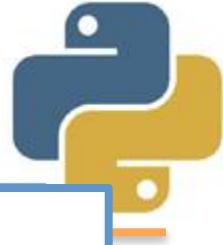
Use o decorator
@classmethod

```
    lista = []
    arquivo = open(nome_arquivo, "r")
    for linha in arquivo:
        linha = linha.strip().split(";")
        if linha[0] != 'ID':
            gibi = cls(*linha)
            lista.append(gibi)
    arquivo.close()
    return lista
```



quivo

Exercício 11: @classmethod

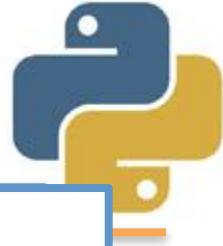


```
@classmethod  
def carregar_lista_gibi(cls, nome_arquivo):  
    """Função para carga de lista de GIBIS  
    Args:  
        nome_arquivo (str) - nome do arquivo que deve ser lido  
    Returns:  
        lista (list) - lista de gibis carregados  
    """  
  
    lista = []  
    with open(nome_arquivo, "r") as arquivo:  
        for linha in arquivo:  
            linha = linha.strip().split(";")  
            if linha[0] != 'ID':  
                gibi = cls(*linha)  
                lista.append(gibi)  
    arquivo.close()  
    return lista
```

Referência para a
classe do objeto



Exercício 11: @classmethod



```
@classmethod
```

```
def carregar_lista_gibi(cls, nome_arquivo):
```

"""\nFunção para carga de lista de GIBIS

Args:

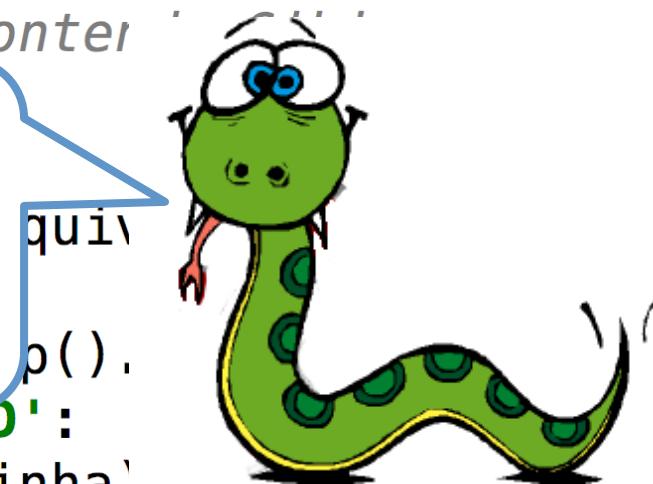
nome_arquivo (str): Nome do arquivo

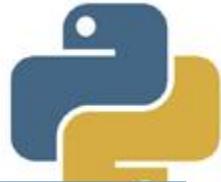
Returns:

list: Lista contei

Troque Gibi pela
referência à classe

```
    if linha[0] == 'ID':  
        gibi = cls(*linha)  
        lista.append(gibi)  
    arquivo.close()  
return lista
```

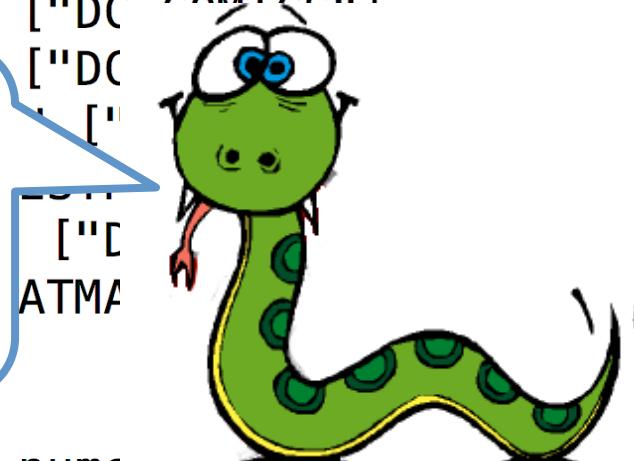




Execute novamente

```
Run teste_leitura
A MORTE DO SUPERMAN - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"O ULTIMO FILHO" - SUPERMAN ["DC COMICS"]
"A"
"A
"A
"A
"Mo
"Re
Representação formal:
GibiFormatoAmericano(id=11, numero=24, título=MORIA
Teste com eval()
type(a): <class 'model.GibiFormatoAmericano'>
repr(a): GibiFormatoAmericano(id=11, numero=24, título=MORIA
Process finished with exit code 0
```

Veja que agora os objetos são da classe correta





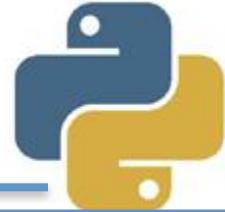
Execute novamente

```
Run teste_leitura
A MORTE DO SUPERMAN - SUPERMAN ["DC COMICS"]
"A MORTE DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"CORPORACAO DO SUPERMAN" - SUPERMAN ["DC COMICS"]
"O ULTIMO FILHO" - SUPERMAN ["DC COMICS"]
"A MORTE DO ROBIN" - BATMAN ["DC COMICS"]
"A MASCARA DA MORTE" - BATMAN ["DC COMICS"]
"A ORIGEM DO HOMEM QUE VAI DESTRUIR BATMAN" - BATMAN []
"A VITORIA DE BANE" - BATMAN ["DC COMICS"]
"MOREGO VERSUS MOREGO" - BATMAN ["DC COMICS"]

Representação Formal:
GibiFormatoAmericano(id=11, numero=24, titulo="MEMORIA")

Teste com eval()
type(a): <class 'model.GibiFormatoAmericano'>
repr(a): GibiFormatoAmericano(id=11, numero=24, titulo=)
Process finished with exit code 0
```

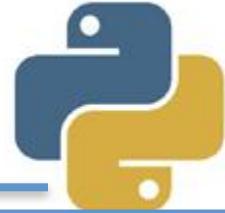
Exercício 12: Super e Sub classes



```
lista = Gibi.carregar_lista_gibi("gibis.csv")
representacao = repr(lista[0])
print "\nRepresentação Formal GIBI:\n", representacao
print "Teste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)

lista1 = GibiFormatoAmericano.carregar_lista_gibi("gibis.csv")
representacao = repr(lista1[0])
print "\nRepresentação Formal GibiFormatoAmericano:\n", \
      representacao
print "Teste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)
```

Exercício 12: Super e Sub classes



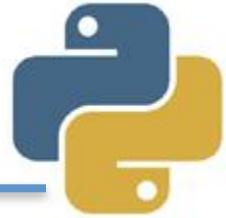
```
lista = Gibi.carregar_lista_gibi("gibis.csv")
representacao = repr(lista[0])
print "\nRepresentação Formal GIBI:\n", representacao
print "Teste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)

lista1 = GibiFormatoAmericano.carregar_lista_gibi("gibis.csv")
representacao = repr(lista1[0])
print "\nRepresentação Formal GIBI AMERICANO:\n", representacao
print "Teste com eval()"
a = eval(representacao)
print "type(a):", type(a)
print "repr(a):", repr(a)
```

Teste o método, a partir de cada uma das classes



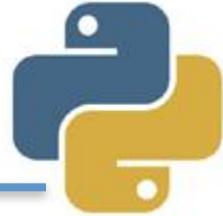
Resultado no console



- Agora, o método `carregar_lista_gibi()` sabe a diferença entre os tipos de Gibi

```
Run teste_leitura
  Run
  Stop
  Refresh
  Help
  Exit
  Teste com eval()
  type(a): <class 'model.Gibi'>
  repr(a): Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE"
  Representação Formal GIBI:
  Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE"
  Teste com eval()
  type(a): <class 'model.GibiFormatoAmericano'>
  repr(a): GibiFormatoAmericano(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLV
  Representação Formal GibiFormatoAmericano:
  GibiFormatoAmericano(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE"
  Teste com eval()
  type(a): <class 'model.GibiFormatoAmericano'>
  repr(a): GibiFormatoAmericano(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLV
Process finished with exit code 0
```

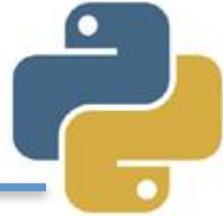
Resultado no console



- Agora, o método `carregar_lista_gibi()` sabe a **diferença** entre os **tipos** de Gibi

```
Run teste_leitura
Representação Formal GIBI:
Gibi(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE"
Teste com eval()
type(a): <class 'model.Gibi'>
repr(a): Gibi(id=11, numero=24, titulo=MEMORIA APAGADA, personagem=WOLV
Representação Formal GibiFormatoAmericano:
GibiFormatoAmericano(id=11, numero=24, titulo="MEMORIA APAGADA", personagem="WOLVERINE"
Teste com eval()
type(a): <class 'model.GibiFormatoAmericano'>
repr(a): GibiFormatoAmericano(id=11, numero=24, titulo=MEMORIA APAGADA, personagem=WOLV
Process finished with exit code 0
```

Testes mais detalhados



- Quando informamos um **arquivo válido**,
tudo funciona perfeitamente

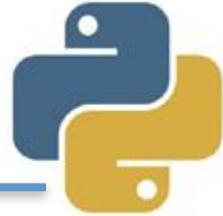
```
4     from model import Gibi
5
6     lista = Gibi.carregar_lista_gibi("gibis.csv")
7     print 'Total', len(lista)
```

Run teste_excecao

/Library/Frameworks/Python.framework/Versions/2
Total 15

Process finished with exit code 0

Testes mais detalhados

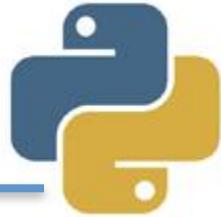


- Quando informamos um **arquivo válido**,
tudo funciona perfeitamente

```
4     from model import Gibi
5
6     lista = Gibi.carregar_lista_gibi("gibis.csv")
7     print 'Total', len(lista)
```

The code block shows a Python script named 'teste_excecao'. The file path is '/Library/Frameworks/Python.framework/Versions/2'. The output window displays the result of the script's execution: 'Total 15'. The line 'print 'Total'', len(lista)' is highlighted with a red box, and the resulting output 'Total 15' is also highlighted with a red box.

```
Run teste_excecao
/Library/Frameworks/Python.framework/Versions/2
Total 15
Process finished with exit code 0
```



Testes mais detalhados

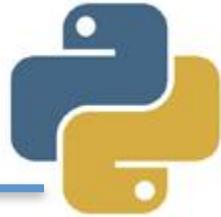
- Mas e quando o arquivo **não existe?**

```
4 from model import Gibi
5
6 lista = Gibi.carregar_lista_gibi("gib.csv")
7 print 'Total', len(lista)
```

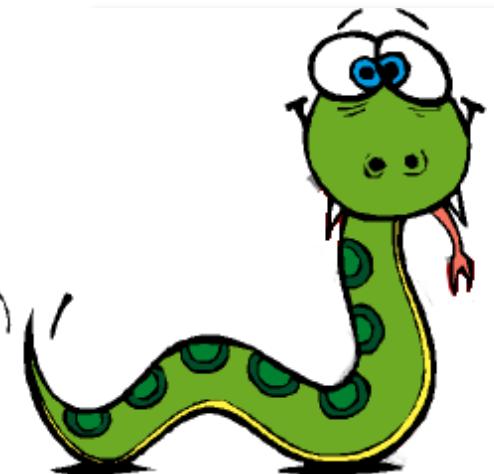
Run teste_excecao

```
/Library/Frameworks/Python.framework/Versions/2.7/bin/python
Traceback (most recent call last):
  File "/Users/marciopalheta/python/codigofonte/financeiro/teste_excecao.py", line 6, in <module>
    lista = Gibi.carregar_lista_gibi("gib.csv")
  File "/Users/marciopalheta/python/codigofonte/financeiro/Gibi.py", line 14, in carregar_lista_gibi
    arquivo = open(nome_arquivo, "r")
IOError: [Errno 2] No such file or directory: 'gib.csv'

Process finished with exit code 1
```



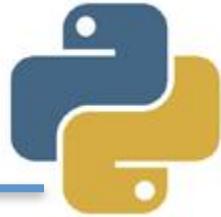
Tratamento de exceções



Ocorre um **erro** em tempo de execução

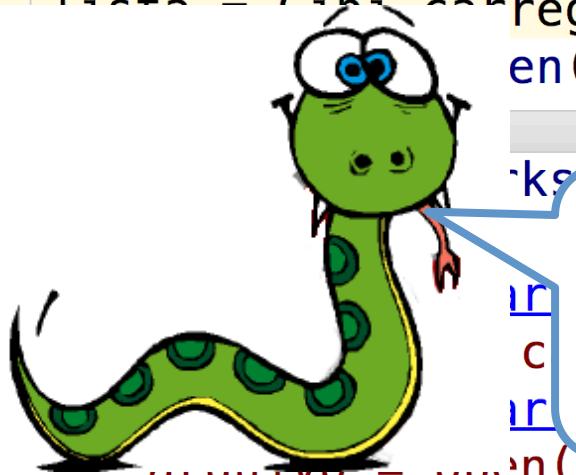
```
4
5
6    regar_lista_gibi("gib.csv")
7    en(lista)
/Libra
Traceback (most recent call last):
  File "/Users/marciopalheta/python/codigofonte/financeiro
      lista = Gibi.carregar_lista_gibi("gib.csv")
  File "/Users/marciopalheta/python/codigofonte/financeiro
      arquivo = open(nome_arquivo, "r")
IOError: [Errno 2] No such file or directory: 'gib.csv'
Process finished with exit code 1
```

The slide illustrates a Python exception handling scenario. A cartoon green snake with large eyes is positioned next to a code editor window. The code editor shows a script with a line highlighted in red: `regar_lista_gibi("gib.csv")`. A callout bubble from the snake points to this line with the text: "Ocorre um **erro** em tempo de execução". Below the code editor, a red box encloses the entire traceback output, which details an `IOError` for a non-existent file named `'gib.csv'`. The Python logo is located in the top right corner.



Testes mais detalhados

- Mas e quando o arquivo **não existe?**



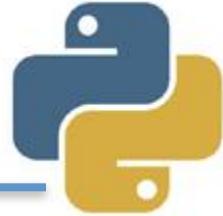
from model import Gibi
lista = Gibi.conregar_lista_gibi("gib.csv")
en(lista)

FileNotFoundError: [Errno 2] No such file or directory: 'gib.csv'

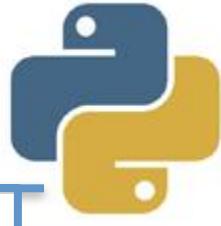
Process finished with exit code 1

Informando que o arquivo
não foi encontrado

Tratamento de exceções

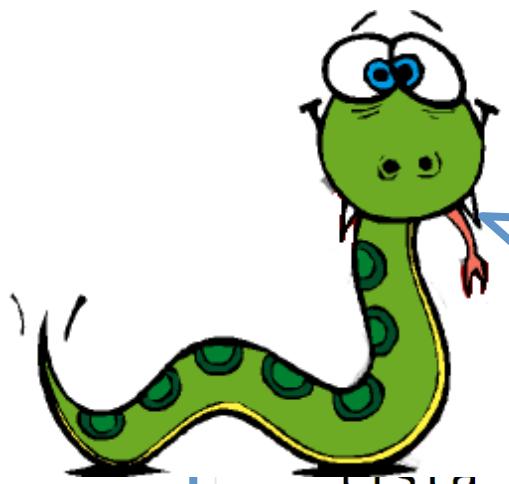


- Nossa aplicação precisa lidar com o problema de erro na abertura do arquivo (que pode ter várias causas)
 - Podemos usar um bloco try-except para capturar erros durante a execução
 - try: trecho de código protegido
 - except: executado quando ocorre erro
-

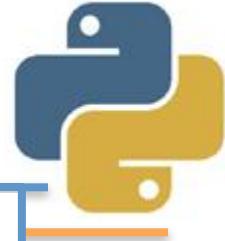


Exercício 13: tratamento

```
@classmethod
def carregar_lista_gibi(cls, nome_arquivo):
    """
    ...
    """
    lista = []
    try:
        print 'Carregando arquivo...'
        arquivo = open(nome_arquivo, "r")
        print 'Arquivo carregado'
        for linha in arquivo:
            linha = linha.strip().split(';')
            if linha[0] != 'ID':
                gibi = cls(*linha)
                lista.append(gibi)
        print 'Fechando arquivo...'
        arquivo.close()
        print 'Arquivo fechado'
    except IOError as error:
        print "A casa caiu - %s" % error
    return lista
```



Ício 12. Tratamento

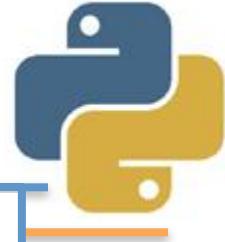


Bloco **protegido**
pela clausula **try**

```
try:  
    print 'Carregando arquivo...'  
    arquivo = open(nome_arquivo, "r")  
    print 'Arquivo carregado'  
    for linha in arquivo:  
        linha = linha.strip().split(';')  
        if linha[0] != 'ID':  
            gibi = cls(*linha)  
            lista.append(gibi)  
    print 'Fechando arquivo...'  
    arquivo.close()  
    print 'Arquivo fechado'  
except IOError as error:  
    print "A casa caiu - %s" % error  
return lista
```

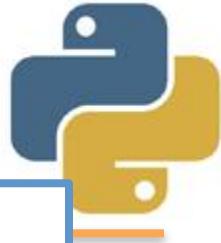


Ício 12. Tratamento



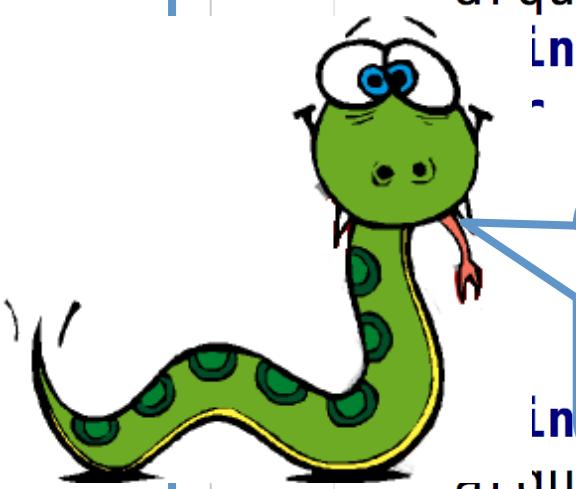
Inclua mensagens
de log de execução

```
d
r...
lista = []
try:
    print 'Carregando arquivo...'
    arquivo = open(nome_arquivo, "r")
    print 'Arquivo carregado'
    for linha in arquivo:
        linha = linha.strip().split(';')
        if linha[0] != 'ID':
            gibi = cls(*linha)
            lista.append(gibi)
    print 'Fechando arquivo...'
    arquivo.close()
    print 'Arquivo fechado'
except IOError as error:
    print "A casa caiu - %s" % error
return lista
```



Exercício 13: tratamento

```
@classmethod  
def carregar_lista_gibi(cls, nome_arquivo):  
    """..."""  
    lista = []  
    try:  
        print 'Carregando arquivo...'  
        arquivo = open(nome_arquivo, "r")  
        print 'Arquivo carregado...'  
        for linha in arquivo:  
            linha = linha.strip().split(',')  
    except IOError as error:  
        print "A casa caiu - %s" % error  
    finally:  
        arquivo.close()  
    return lista
```



Clausula chamada quando
ocorre um IOError

Exercício 14: Arquivo válido



```
1 #-*- coding: UTF-8 -*-
2 # teste_excecao.py
3
4 from model import Gibi
5
6 lista = Gibi.carregar_lista_gibi("gibis.csv")
7 print 'Total', len(lista)
```

Run teste_excecao

```
/Library/Frameworks/Python.framework/Versions/2.
Carregando arquivo...
Arquivo carregado...
Fechando arquivo
Total 15
```

Process finished with exit code 0

Exercício 14: Arquivo válido



```
1 #-*- coding: UTF-8 -*-
2 # teste_excecao.py
```

```
import Gibi
```

Crie e **execute** o arquivo
teste_excecao.py

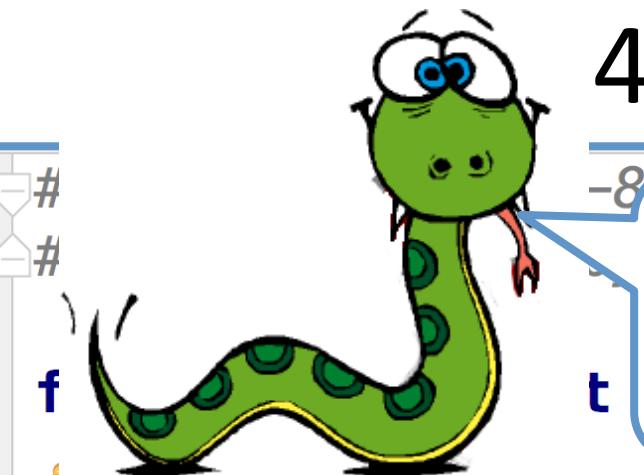
```
Carregando arquivo...
Arquivo carregado...
Fechando arquivo
Total 15
```

```
Process finished with exit code 0
```





4: Arquivo válido



Informe um **arquivo** que **existe** na **raiz** do projeto

```
1 #  
2 #  
3 )  
4 f  
5 t  
6 lista = Gibi.carregar_lista_gibi("gibis.csv")  
7 print 'Total', len(lista)
```

Run teste_excecao

```
/Library/Frameworks/Python.framework/Versions/2.  
Carregando arquivo...  
Arquivo carregado...  
Fechando arquivo  
Total 15
```

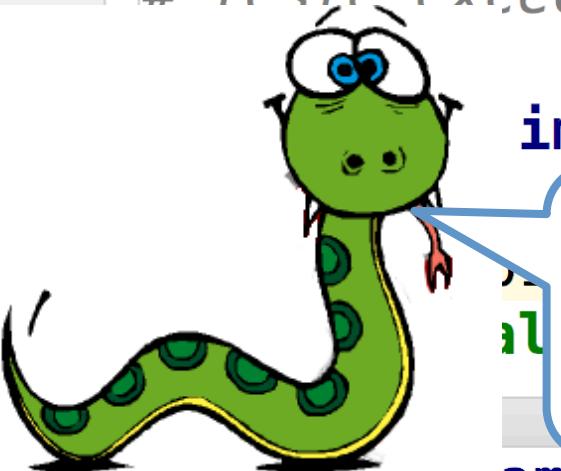
Process finished with exit code 0

Exercício 14: Arquivo válido



```
1 #-*- coding: UTF-8 -*-
2 # teste excecao.py
```

```
import Gibi
```

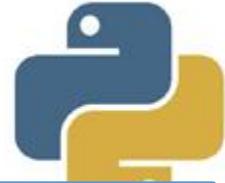


Veja que **todas** as mensagens
de log foram **impressas**

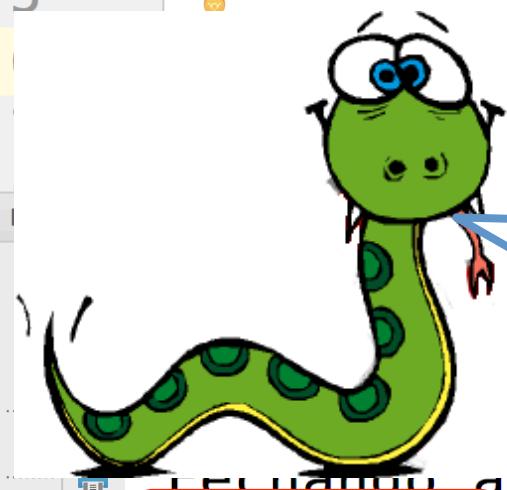
```
/Library/Frameworks/Python.framework/Versions/2.
Carregando arquivo...
Arquivo carregado...
Fechando arquivo
Total 15
```

Process finished with exit code 0

Exercício 14: Arquivo válido



```
1 #-*- coding: UTF-8 -*-
2 # teste_excecao.py
3
4 from model import Gibi
5
6
7     gibis.csv")
8     al', len(lista))
```



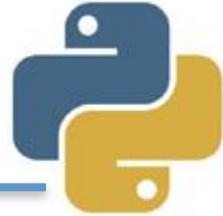
E recebemos a mensagem de que foram carregados **15 gibis**

Total 15

Process finished with exit code 0

2.

Exercício 15: Teste exceção



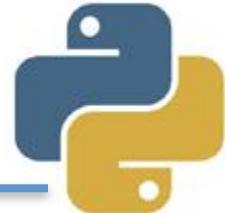
- Agora tente com um arquivo inexistente

```
1  #--*-- coding: UTF-8 -*-  
2  # teste_excecao.py  
3  
4  from model import Gibi  
5  
6  lista = Gibi.carregar_lista_gibi("gib.csv")  
7  print 'Total', len(lista)
```

Run teste_excecao

```
/Library/Frameworks/Python.framework/Versions/2.7/bin/python2  
Carregando arquivo...  
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'  
Total 0  
  
Process finished with exit code 0
```

Exercício 15: Teste exceção



e com um arquivo inexistente

O Arquivo “gib.csv” não existe
na pasta raiz do projeto

```
1
2
3
4
5
6
7
```

```
lista = Gibi.carregar_lista_gibi("gib.csv")
print 'Total', len(lista)
```

Run teste_excecao

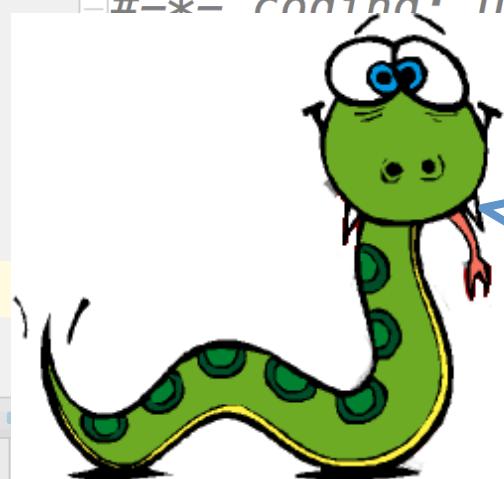
```
/Library/Frameworks/Python.framework/Versions/2.7/bin/python2
Carregando arquivo...
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'
Total 0
```

```
Process finished with exit code 0
```

Exercício 15: Teste exceção



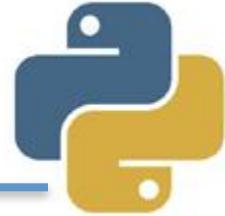
- Agora tente com um **arquivo inexistente**



```
1  #--*- coding: UTF-8 -*-  
2  -py  
3  
4  rt  
5  rr  
6  le  
7  rks, v, eon, framework, vcs, tools, /, python2  
8  
9  Carregando arquivo...  
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'  
Total 0  
Process finished with exit code 0
```

Foram impressas as mensagens de abertura do arquivo, erro e total

Exercício 15: Teste exceção



- Agora tente com um **arquivo inexistente**

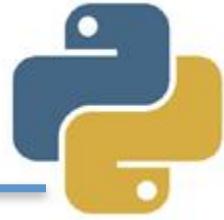


```
1 #--*- coding: UTF-8 -*-  
2  
3  
4  
5  
6  
7  
8  
9  
0  
Run  
Green triangle  
Grey square  
Grey square with vertical line  
Grey square with horizontal line  
Grey square with diagonal line  
Blue square  
Purple square  
Red square  
Yellow square  
Question mark  
File icon  
Save icon  
Run icon  
Stop icon  
Exit icon  
Help icon
```

Conseguimos tratar a exceção, mas não fechamos o arquivo, quando há erro

```
Carregando arquivo...  
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'  
Total 0
```

Process finished with exit code 0



Exercício 16: finally

```
@classmethod
def carregar_lista_gibi(cls, nome_arquivo):
    """
    ...
    """
    lista = []
    arquivo = None
    try:
        print 'Carregando arquivo...'
        arquivo = open(nome_arquivo, "r")
        print 'Arquivo carregado'
        for linha in arquivo:
            linha = linha.strip().split(';')
            if linha[0] != 'ID':
                gibi = cls(*linha)
                lista.append(gibi)
    except IOError as error:
        print "A casa caiu - %s" % error
    finally:
        print 'Fechando arquivo...'
        if arquivo:
            arquivo.close()
        print 'Arquivo fechado'
    return lista
```



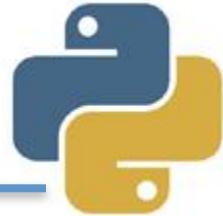
Exercício 16: finally

```
@classmethod  
def carregar_lista_gibi(cls, nome_arquivo):  
    """..."""  
    lista = []  
    arquivo = None  
  
    print 'Carregando arquivo...'  
    arquivo = open(nome_arquivo)  
  
    try:  
        for linha in arquivo:  
            lista.append(linha.strip())  
    except IOError as error:  
        print "A casa caiu - %s" % error  
    finally:  
        print 'Fechando arquivo...'  
        if arquivo:  
            arquivo.close()  
        print 'Arquivo fechado'  
  
    return lista
```



A cláusula opcional **finally** SEMPRE é **executada**, haja ou não erro

Execute novamente

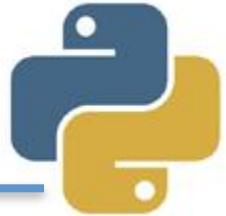


```
model.py x teste_excecao.py x

1 # -*- coding: UTF-8 -*-
2 # teste_excecao.py
3
4 from model import Gibi
5
6 lista = Gibi.carregar_lista_gibi("gib.csv")
7 print 'Total', len(lista)

Run teste_excecao
▶ /Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
Carregando arquivo...
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'
Fechando arquivo...
Arquivo fechado
Total 0

Process finished with exit code 0
```



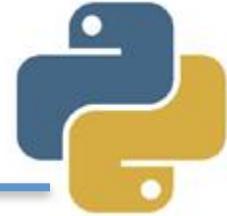
Execute novamente



Agora, mesmo com a exceção, todos os logs foram impressos no console

```
model.py x teste_excecao.py x
1
2
3
4
5
6
7
8 --*-
py
re
en
Run teste_excecao
/I library/Frameworks/Python.framework/Versions/2.7/bin/python2.7
Carregando arquivo...
A casa caiu - [Errno 2] No such file or directory: 'gib.csv'
Fechando arquivo...
Arquivo fechado
Total 0
Process finished with exit code 0
```

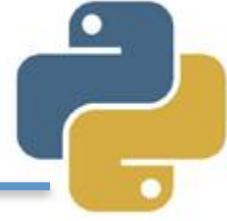
Exercício 17: Múltiplas exceções



- Ao final da linha do gibi com **ID == 11**,
inclusa ;"**CAPA DOURADA**"

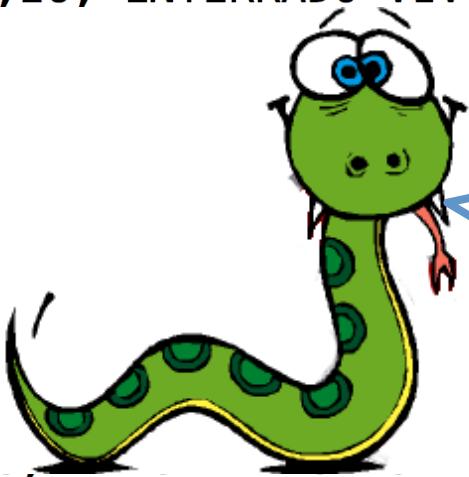
```
ID;NUMERO;TITULO;personagem;editora
11;24;"MEMORIA APAGADA";"WOLVERINE";"MARVEL COMICS";"CAPA DOURADA"
12;26;"ENTERRADO VIVO";"WOLVERINE";"MARVEL COMICS"
13;3;"SAMURAIS EM GUERRA";"WOLVERINE";"MARVEL COMICS"
14;39;"CODINOME WOLVERINE";"WOLVERINE";"MARVEL COMICS"
15;44;"MORTE EM FAMILIA";"WOLVERINE";"MARVEL COMICS"
1;1;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
2;2;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
3;3;"A MORTE DO SUPERMAN";SUPERMAN;"DC COMICS"
4;2002;"CORPORACAO DO SUPERMAN";SUPERMAN;"DC COMICS"
5;0;"O ULTIMO FILHO";SUPERMAN;"DC COMICS"
6;0;"A MORTE DO ROBIN";BATMAN;"DC COMICS"
```

Exercício 17: Múltiplas exceções



- Ao final da linha do gibi com **ID == 11**, inclua ;"**CAPA DOURADA**"

```
TD;NUMERO;TTTIII 0;personagem;editora
11;24;"MEMORIA APAGADA";"WOLVERINE";"MARVEL COMICS";"CAPA DOURADA"
12;26;"ENTERRADO VIVO";"WOLVERINE";"MARVEL COMICS"
13;"LARVA";"WOLVERINE";"MARVEL COMICS"
14;"RRA";"WOLVERINE";"MARVEL COMICS"
15;"RINE";"WOLVERINE";"MARVEL COMICS"
16;"A"
17;"M"
18;"IA"
19;"S"
20;"SU"
21;"';BATMAN, DC COMICS
```



Com isso, o **1º Gibi** passa a ter **mais atributos**, do que o esperado pelo **construtor**

Exercício 18: Múltiplas exceções

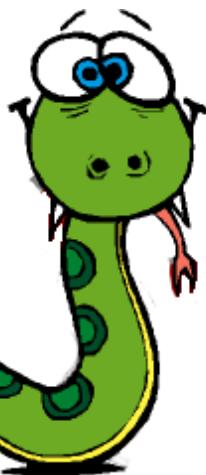
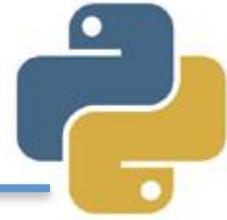


- Execute novamente `teste_excecao.py`

```
4 from model import Gibi
5
6 lista = Gibi.carregar_lista_gibi("gibis.csv")
7 print 'Total', len(lista)

Run teste_excecao
/Library/Frameworks/Python.framework/Versions/2.7/bin/python
Traceback (most recent call last):
  File "/Users/marciopalheta/python/codigofonte/financeiro/
    lista = Gibi.carregar_lista_gibi("gibis.csv")
  File "/Users/marciopalheta/python/codigofonte/financeiro/
    gibi = cls(*linha)
TypeError: __init__() takes exactly 6 arguments (7 given)
Carregando arquivo...
```

Exercício 18: Múltiplas exceções



vamente teste_excecao.py

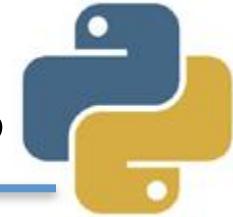
Novo erro lançado

```
arregar_lista_gibi("gibis.csv")
len(lista)
```

Run teste_excecao

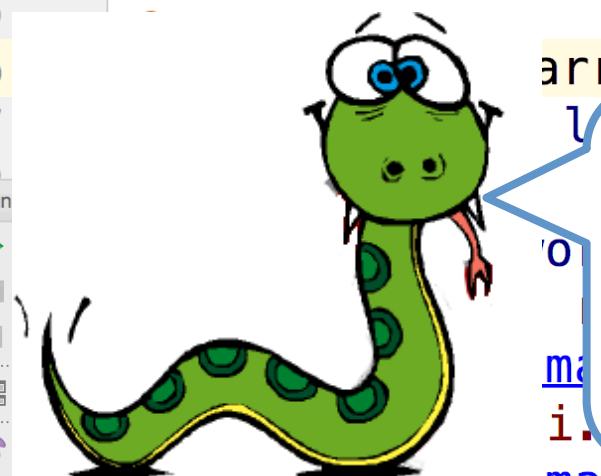
```
/Library/Frameworks/Python.framework/Versions/2.7/bin/python
Traceback (most recent call last):
  File "/Users/marciopalheta/python/codigofonte/financeiro/
    lista = Gibi.carregar_lista_gibi("gibis.csv")
  File "/Users/marciopalheta/python/codigofonte/financeiro/
    gibi = cls(*linha)
TypeError: __init__() takes exactly 6 arguments (7 given)
Carregando arquivo...
```

Exercício 18: Múltiplas exceções



- Execute novamente `teste_excecao.py`

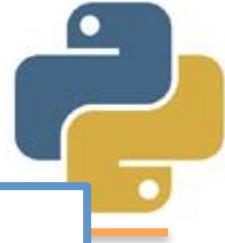
```
from model import Gibi  
arregar lista gibi("gibis.csv")
```



Foi lançado um **TypeError**, indicando que **passamos 7 argumentos**, ao **invés de 6**

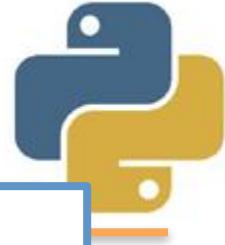
```
gibi = cls(*linha)
TypeError: __init__() takes exactly 6 arguments (7 given)
Carregando arquivo...
```

Exercício 19: Nova exceção



```
try:  
    print 'Carregando arquivo...'  
    arquivo = open(nome_arquivo, "r")  
    print 'Arquivo carregado'  
    for linha in arquivo:  
        linha = linha.strip().split(';')  
        if linha[0] != 'ID':  
            gibi = cls(*linha)  
            lista.append(gibi)  
except IOError as error:  
    print "Falha de abertura - %s" % error  
except TypeError as error:  
    print "Falha no nro de atributos %s" % error  
finally:  
    print 'Fechando arquivo...'  
    if arquivo:  
        arquivo.close()  
    print 'Arquivo fechado'
```

Exercício 19: Nova exceção



```
try:
```



```
    'rregando arquivo...'
```

```
    o  
    =  
    h  
    ib
```

Atualize a **lista de erros** tratados
pelo método **de classe**
carregar_lista_gibi()

```
except IOError as error:  
    print "Falha de abertura - %s" % error  
except TypeError as error:  
    print "Falha no nro de atributos %s" % error  
finally:  
    print 'Fechando arquivo...'  
    if arquivo:  
        arquivo.close()  
    print 'Arquivo fechado'
```

Execute novamente



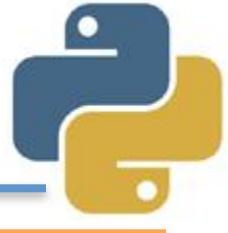
```
model.py x teste_excecao.py x gibis.csv x

4 from model import Gibi
5
6 lista = Gibi.carregar_lista_gibi("gibis.csv")
7 print 'Total', len(lista)
8

Run teste_excecao
▶ /Library/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/marcelo/PycharmProjects/Excecoes/teste_excecao.py
Carregando arquivo...
Arquivo carregado
Falha no nro de atributos __init__() takes exactly 6 arguments (7 given)
Fechando arquivo...
Arquivo fechado
Total 0

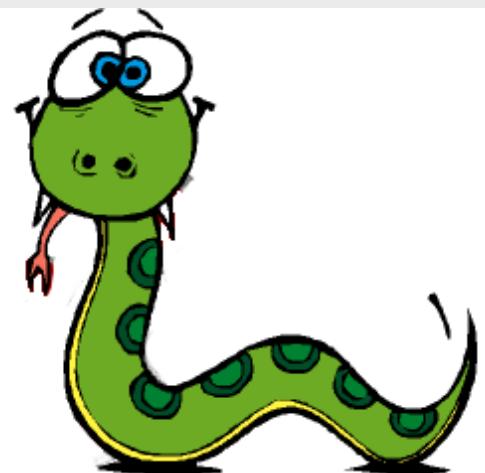
Process finished with exit code 0
```

Execute novamente



Conseguimos
abrir o arquivo

gibi()



/Library/Frameworks/Python.framework/Versions/2.7 /Users/marcelo/

Carregando arquivo...

Arquivo carregado

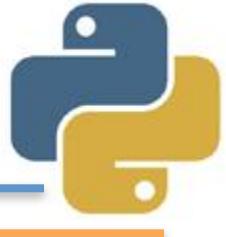
Falta no nro de atributos __init__() takes exactly 6 arguments (7 given)

Fechando arquivo...

Arquivo fechado

Total 0

Process finished with exit code 0



Execute novamente

Tratamos o erro
de quantidade
de atributos

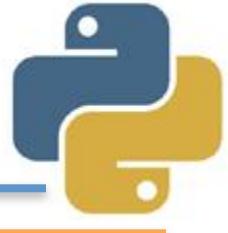
gibi()



```
/Library/Frameworks/Python.framework  
Carregando arquivo...  
Arquivo carregado  
Falha no nro de atributos __init__() takes exactly 6 arguments (7 given)  
Fechando arquivo...  
Arquivo fechado  
Total 0
```

Process finished with exit code 0

Execute novamente



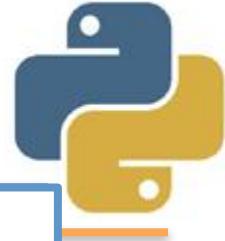
E fechamos o
arquivo

```
Falha no pro de atributos __init__  
Fechando arquivo...  
Arquivo fechado  
Total 0
```

```
Process finished with exit code 0
```

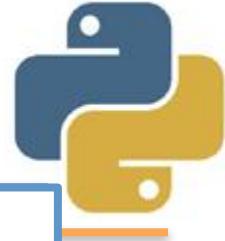


Tratamento em um único bloco



```
try:  
    print 'Carregando arquivo...'  
    arquivo = open(nome_arquivo, "r")  
    print 'Arquivo carregado'  
    for linha in arquivo:  
        linha = linha.strip().split(';')  
        if linha[0] != 'ID':  
            gibi = cls(*linha)  
            lista.append(gibi)  
except (IOError, TypeError) as error:  
    print "Falha no arquivo - %s" % error  
finally:  
    print 'Fechando arquivo...'  
    if arquivo:  
        arquivo.close()  
    print 'Arquivo fechado'
```

Tratamento em um único bloco



```
try:  
    print 'Lendo o arquivo'  
    arquivo = open('meu_arquivo.txt')  
    lista = []  
    for linha in arquivo:  
        gibis.append(linha)  
    arquivo.close()  
    print 'Arquivo lido com sucesso'  
  
except (IOError, TypeError) as error:  
    print "Falha no arquivo - %s" % error  
  
finally:  
    print 'Fechando arquivo...'  
    if arquivo:  
        arquivo.close()  
    print 'Arquivo fechado'
```

Podemos tratar os **dois erros** no mesmo bloco

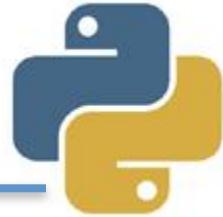




FIM

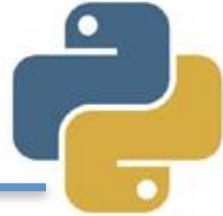


Contatos



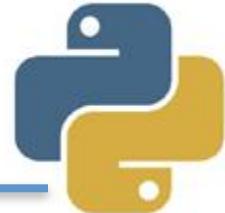
- Márcio Palheta
 - marcio.palheta@gmail.com
 - @marciopalheta
 - <https://sites.google.com/site/marciopalheta/>
-

Bibliografia



- LIVRO: Apress - Beginning Python From Novice to Professional
 - LIVRO: O'Reilly - Learning Python
 - <http://www.python.org>
 - <http://www.python.org.br>
 - Mais exercícios:
 - <http://wiki.python.org.br/ListaDeExercicios>
 - Documentação do python:
 - <https://docs.python.org/2/>
-

Capítulo 07



Manipulação de Arquivos e controle de Exceções



Márcio Palheta, M.Sc.
marcio.palheta@gmail.com