

Design de Software

Aula 8 – Python (Arquivos e Matrizes)

2016 – Engenharia

Fábio Ayres <fabioja@insper.edu.br>

Raul Ikeda <rauligs@insper.edu.br>

Objetivos de Aprendizado

- Acessar e modificar arquivos
- Fazer operações com matrizes

Arquivos

Arquivos são estruturas de dados que ficam armazenados em dispositivos secundários de memória. Em geral:

- Mais lentos;
- Maior capacidade;
- Persistentes.

Os principais dispositivos de armazenamento atualmente são:

- Discos Rígidos (HDs)
- Pen drives
- CDs e DVDs



Os arquivos armazenados nestes dispositivos possuem sempre um identificação (nome) e sua localização. Outros atributos como data e tamanho também são normalmente usados no índice.

Acessando Arquivos

- Arquivos são lidos como uma sequência de bytes
 - Estes bytes podem estar organizados de forma de códigos que só um programa específico entenda;
 - Estes bytes podem ser também somente caracteres (ASCII ou Unicode), usando os recursos nativos do Python.
Programas de edição de texto puro trabalham com estes arquivos.
- Os arquivos podem ser divididos assim em dois sub-grupos:
 - Texto;
 - Binários.

Tabela ASCII

000	(nul)	016	► (dle)	032	sp	048	ô	064	@	080	P	096	`	112	p
001	☉ (soh)	017	◄ (dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	⊗ (stx)	018	↑ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥ (etx)	019	!! (dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦ (eot)	020	ℙ (dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	♣ (enq)	021	§ (nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠ (ack)	022	— (syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	• (bel)	023	‡ (etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	■ (bs)	024	↑ (can)	040	(056	8	072	H	088	X	104	h	120	x
009	(tab)	025	↓ (em)	041)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂ (vt)	027	← (esc)	043	+	059	;	075	K	091	[107	k	123	{
012	♀ (np)	028	L (fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	↔ (gs)	045	-	061	=	077	M	093]	109	m	125	}
014	♫ (so)	030	▲ (rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	⊗ (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	△

ASCII – American Standard Code for Information Interchange

Tabela ASCII estendida

128 Ç	143 Å	158 ß	172 ¼	186	200 ℒ	214 ⌈	228 Σ	242 ≥
129 ü	144 É	159 f	173 ;	187 ⌋	201 ⌌	215 ⌉	229 σ	243 ≤
130 é	145 æ	160 á	174 «	188 ⌌	202 ⌍	216 ⌊	230 μ	244 ∫
131 â	146 Æ	161 í	175 »	189 ⌍	203 ⌎	217 ⌋	231 τ	245 ∫
132 ä	147 ô	162 ó	176 ▯	190 ⌎	204 ⌏	218 ⌌	232 Φ	246 ÷
133 à	148 ö	163 ú	177 ▯	191 ⌏	205 =	219 ■	233 Θ	247 ≈
134 å	149 ò	164 ñ	178 ▯	192 ⌐	206 ⌐	220 ■	234 Ω	248 °
135 ç	150 û	165 Ñ	179	193 ⊥	207 ⊥	221 ■	235 δ	249 ·
136 ê	151 ù	166 ª	180	194 ⊥	208 ⊥	222 ■	236 ∞	250 ·
137 ë	152 ŷ	167 °	181 =	195 ⊥	209 ⊥	223 ■	237 φ	251 √
138 è	153 Ö	168 ¿	182	196 −	210 ⊥	224 α	238 ε	252 ∞
139 ï	154 Ü	169 ¬	183 π	197 ⊥	211 ⊥	225 β	239 ∩	253 ²
140 î	155 ç	170 ¬	184 ¶	198 ¶	212 ⊥	226 Γ	240 ≡	254 ■
141 ì	156 £	171 ½	185 ¶	199 ¶	213 F	227 π	241 ±	255
142 Ä	157 ¥							

Parte da tabela UTF-8

Unicode code point	character	UTF-8 (hex.)	name
U+00A0		c2 a0	NO-BREAK SPACE
U+00A1	¡	c2 a1	INVERTED EXCLAMATION MARK
U+00A2	¢	c2 a2	CENT SIGN
U+00A3	£	c2 a3	POUND SIGN
U+00A4	¤	c2 a4	CURRENCY SIGN
U+00A5	¥	c2 a5	YEN SIGN
U+00A6	¦	c2 a6	BROKEN BAR
U+00A7	§	c2 a7	SECTION SIGN
U+00A8	¨	c2 a8	DIAERESIS
U+00A9	©	c2 a9	COPYRIGHT SIGN
U+00AA	ª	c2 aa	FEMININE ORDINAL INDICATOR
U+00AB	«	c2 ab	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
U+00AC	¬	c2 ac	NOT SIGN
U+00AD		c2 ad	SOFT HYPHEN
U+00AE	®	c2 ae	REGISTERED SIGN
U+00AF	ˉ	c2 af	MACRON
U+00B0	°	c2 b0	DEGREE SIGN
U+00B1	±	c2 b1	PLUS-MINUS SIGN
U+00B2	²	c2 b2	SUPERSCRIP TWO
U+00B3	³	c2 b3	SUPERSCRIP THREE

Unicode code point	character	UTF-8 (hex.)	name
U+00B4	´	c2 b4	ACUTE ACCENT
U+00B5	µ	c2 b5	MICRO SIGN
U+00B6	¶	c2 b6	PILCROW SIGN
U+00B7	·	c2 b7	MIDDLE DOT
U+00B8	¸	c2 b8	CEDILLA
U+00B9	¹	c2 b9	SUPERSCRIP ONE
U+00BA	º	c2 ba	MASCULINE ORDINAL INDICATOR
U+00BB	»	c2 bb	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
U+00BC	¼	c2 bc	VULGAR FRACTION ONE QUARTER
U+00BD	½	c2 bd	VULGAR FRACTION ONE HALF
U+00BE	¾	c2 be	VULGAR FRACTION THREE QUARTERS
U+00BF	¿	c2 bf	INVERTED QUESTION MARK
U+00C0	À	c3 80	LATIN CAPITAL LETTER A WITH GRAVE
U+00C1	Á	c3 81	LATIN CAPITAL LETTER A WITH ACUTE
U+00C2	Â	c3 82	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
U+00C3	Ã	c3 83	LATIN CAPITAL LETTER A WITH TILDE

Unicode & UTF-8

- Um caractere Unicode é uma sequência de pontos de código, que são numerados de 0 a 0x10FFFF (1.114.111 decimal) identificando as letras.
- UTF-8 é a codificação mais comum, e padrão no Python 3, porém no cabeçalho do programa Python se pode definir qualquer codificação.

```
1 # -*- coding: utf-8 -*-  
2 print("Três formas diferentes de imprimir o A com til:")  
3 print("Ã")  
4 print("\N{LATIN CAPITAL LETTER A WITH TILDE}")  
5 print("\u00C3")
```


Spider UTF

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script named 'Untitled33.py' with the following code:

```
1 palavra="abacaxi"
2 conta_a=0
3 for l in palavra:
4     if l=="a":
5         conta_a += 1
6 print(conta_a)
```

The status bar at the bottom indicates 'Permissions: RW', 'End-of-lines: LF', and 'Encoding: ASCII'. A blue circle highlights the 'Encoding: ASCII' text.

On the right side, the 'Variable explorer' panel is empty. Below it, the 'Internal console' panel shows a traceback error:

```
- use IPython.nbformat for read/write/validate public API
- use IPython.nbformat.vX directly to composing notebooks
of a particular version

"""
Traceback (most recent call last):
  File "/Users/lpsoares/anaconda/lib/python3.4/site-packa
ges/spyderlib/widgets/editor.py", line 1372, in analyze s
cript
    finfo.run_code_analysis(run_pyflakes, run_pep8)
  File "/Users/lpsoares/anaconda/lib/python3.4/site-packa
ges/spyderlib/widgets/editor.py", line 312, in run_code a
nalysis
    source_code = self.get_source_code().encode(enc)
UnicodeError: 'ascii' codec can't encode character
'\xe1' in position 97: ordinal not in range(128)
```

The status bar at the bottom right shows 'Line: 6', 'Column: 15', 'Memory: 73 %', and 'CPU: 8 %'.

Abrindo e fechando arquivos

- Um arquivo pode ser aberto com o comando:
`open(nome_do_arquivo, modo_de_acesso)`

```
1 arquivo = open("texto.txt", "r")
```

- Os principais modos de acesso são:

r - somente leitura em modo texto

rb - somente leitura em modo binário

w - leitura e escrita, apagando arquivo

wb - leitura e escrita binária apagando arquivo

a - leitura e escrita para atualizar arquivo a partir do final

- Para fechar o arquivo use o método `close()`

```
2 arquivo.close()
```

Lendo e Escrevendo em Arquivos

- As duas formas mais simples de ler e escrever em arquivos são com os métodos **read([count])** e **write()**.

```
1 arquivo = open("texto.txt", "r")
2 leitura = arquivo.read(6)
3 print("texto = {}".format(leitura))
4 arquivo.close()
```

Obs: count = número de posições a ler.


Escrevendo em Arquivos

```
1 arquivo = open("texto_novo.txt", "w")  
2 arquivo.write("Hello world!\n")  
3 arquivo.close()
```

Lendo e Escrevendo Linhas

- As duas formas mais simples de ler e escrever linhas em arquivos são com os métodos **readlines()** e **writelines()**.

```
1 arquivo = open("texto.txt", "r")  
2 lista = arquivo.readlines()  
3 arquivo.close()  
4 print(lista)
```



['banana\n', 'abacate\n', 'laranja\n']

Escrevendo Linhas

```
1 dados = ["banana\n", "abacate\n", "laranja\n"]  
2 arquivo = open("texto.txt", "w")  
3 arquivo.writelines(dados)  
4 arquivo.close()
```

Equivale à:

```
1 dados = ["banana\n", "abacate\n", "laranja\n"]  
2 arquivo = open("texto.txt", "w")  
3 arquivo.writelines(dados)  
4 arquivo.close()
```

Métodos convenientes de Strings

- `str.strip()` - remove espaços em branco do início e fim de uma string. Por exemplo:

```
>>> print(" palavra ").strip()  
'palavra'
```

- `str.split()` - separa uma string pelo seu delimitador em uma lista de strings. Exemplo:

```
>>> '1 2 3 '.split()  
['1', '2', '3']  
>>> '1,2,3'.split(",")  
['1', '2', '3']
```

Operações com Arquivos

- O módulo **os** possui uma série de recursos para operações com arquivos:
- Renomear arquivos `os.rename(nome_atual, novo_nome)`
- Apagando arquivos `os.remove(nome_do_arquivos)`
- Criando diretório `os.mkdir(nome_da_pasta)`
- Mudando de diretório `os.chdir(nome_da_pasta)`

Arquivos com UTF

- Por padrão não é possível gravar caracteres especiais em arquivos.

```
1 # -*- coding: utf-8 -*-  
2 arquivo = open("texto.txt", "r+")  
3 arquivo.write("áéíóú")
```

Traceback (most recent call last):

File "/Users/lpsoares/acentos.py", line 3, in <module>
arquivo.write("áéíóú")

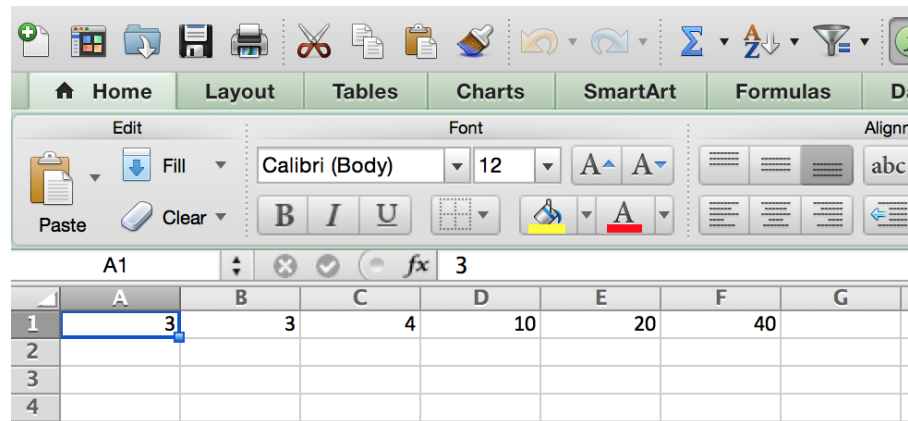
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-4: ordinal not in range(128)

- Para isso se deve especificar o *encoding*

```
1 # -*- coding: utf-8 -*-  
2 arquivo = open("texto.txt", "r+", encoding='utf-8')  
3 arquivo.write("áéíóú")
```

Exercício desafio

No Excel gere uma planilha com uma série de números. Salve em um formato de texto como CSV (Comma-Separated Values). Em um programa em Python, carregue esse arquivo e some todos os valores. Salve de volta no arquivo CSV o valor da somatória e carregue novamente no Excel.



Obs: feche o arquivo no Excel senão ele bloqueia o arquivo.

Arquivos Binários

Tente carregar o arquivo anterior no formato xls ou xlsx. O que aconteceu Python?

Para ver o conteúdo do arquivo, use:

hexdump -C (Linux e Mac)

ou senão online em <http://www.onlinehexeditor.com/>

```
00006a50  00 00 00 00 00 00 00 04 5f 00 00 78 6c 2f 73 74 |....._...xl/st|
00006a60  79 6c 65 73 2e 78 6d 6c 50 4b 01 02 2d 00 14 00 |yles.xmlPK..-...|
00006a70  06 00 08 00 00 00 21 00 12 f4 65 86 16 02 00 00 |.....!...e.....|
00006a80  f6 03 00 00 18 00 00 00 00 00 00 00 00 00 00 00 |.....|
00006a90  00 00 1d 61 00 00 78 6c 2f 77 6f 72 6b 73 68 65 |...a...xl/workshe|
00006aa0  65 74 73 2f 73 68 65 65 74 31 2e 78 6d 6c 50 4b |ets/sheet1.xmlPK|
00006ab0  01 02 2d 00 14 00 06 00 08 00 00 00 21 00 e4 d2 |..-.....!...|
00006ac0  40 49 3f 01 00 00 6d 02 00 00 11 00 00 00 00 00 |@I?...m.....|
00006ad0  00 00 00 00 00 00 00 00 69 63 00 00 64 6f 63 50 |.....ic..docP|
00006ae0  72 6f 70 73 2f 63 6f 72 65 2e 78 6d 6c 50 4b 01 |rops/core.xmlPK.|
00006af0  02 2d 00 14 00 06 00 08 00 00 00 21 00 47 67 c8 |.-.....!.Gg.|
00006b00  1a 93 01 00 00 21 03 00 00 10 00 00 00 00 00 00 |.....!.....|
00006b10  00 00 00 00 00 00 00 df 65 00 00 64 6f 63 50 72 |.....e..docPr|
00006b20  6f 70 73 2f 61 70 70 2e 78 6d 6c 50 4b 05 06 00 |ops/app.xmlPK...|
00006b30  00 00 00 0a 00 0a 00 83 02 00 00 a8 68 00 00 00 |.....h...|
```

Matrizes com arrays 2d do numpy

- Numpy é recomendado para cálculo com matrizes

```
from numpy import zeros, array, ones
```

```
m = zeros([4,4])
```

```
a = ones([2,3])
```

```
print(m)
```

```
print(a)
```



```
[[ 0.  0.  0.  0.]  
 [ 0.  0.  0.  0.]  
 [ 0.  0.  0.  0.]  
 [ 0.  0.  0.  0.]
```

```
[[ 1.  1.  1.]  
 [ 1.  1.  1.]
```

Facilidades do numpy

```
from numpy import *
```

```
a = array([[1,2,1], [2, -1, 1], [3, 1, -1]])
```

```
a[1][1] = 15
```

```
b = zeros((3,3))
```

```
uns = ones((3,3), dtype=int32)
```

```
ident = identity(3)
```

$2*a$


$$\begin{bmatrix} \begin{bmatrix} 2 & 4 & 2 \end{bmatrix} \\ \begin{bmatrix} 4 & -2 & 2 \end{bmatrix} \\ \begin{bmatrix} 6 & 2 & -2 \end{bmatrix} \end{bmatrix}$$

$a + uns$


$$\begin{bmatrix} \begin{bmatrix} 1 & 4 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \\ \begin{bmatrix} 4 & 4 & 4 \end{bmatrix} \end{bmatrix}$$

$a**2$


$$\begin{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \\ \begin{bmatrix} 4 & 4 & 4 \end{bmatrix} \\ \begin{bmatrix} 9 & 1 & 1 \end{bmatrix} \end{bmatrix}$$

$a*uns$


$$\begin{bmatrix} \begin{bmatrix} 4 & 4 & 4 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \end{bmatrix}$$

$a.dot(uns)$


$$\begin{bmatrix} \begin{bmatrix} 1. & 2. & 1. \end{bmatrix} \\ \begin{bmatrix} 2. & 15. & 1. \end{bmatrix} \\ \begin{bmatrix} 3. & 1. & -1. \end{bmatrix} \end{bmatrix}$$

$a.dot(ident)$


$$\begin{bmatrix} \begin{bmatrix} 1. & 2. & 1. \end{bmatrix} \\ \begin{bmatrix} 2. & 15. & 1. \end{bmatrix} \\ \begin{bmatrix} 3. & 1. & -1. \end{bmatrix} \end{bmatrix}$$

Facilidades interessantes

- Resolver o sistema linear com numpy:

$$x + 2y + z = 8$$

$$2x - y + z = 3$$

$$3x + y - z = 2$$

```
from numpy import *
```

```
# Os coeficientes das equações entram
```

```
# um por linha
```

```
a = array([[1,2,1], [2, -1, 1], [3, 1, -1]])
```

```
b = array([8,3,2])
```

```
x = linalg.solve(a,b)
```

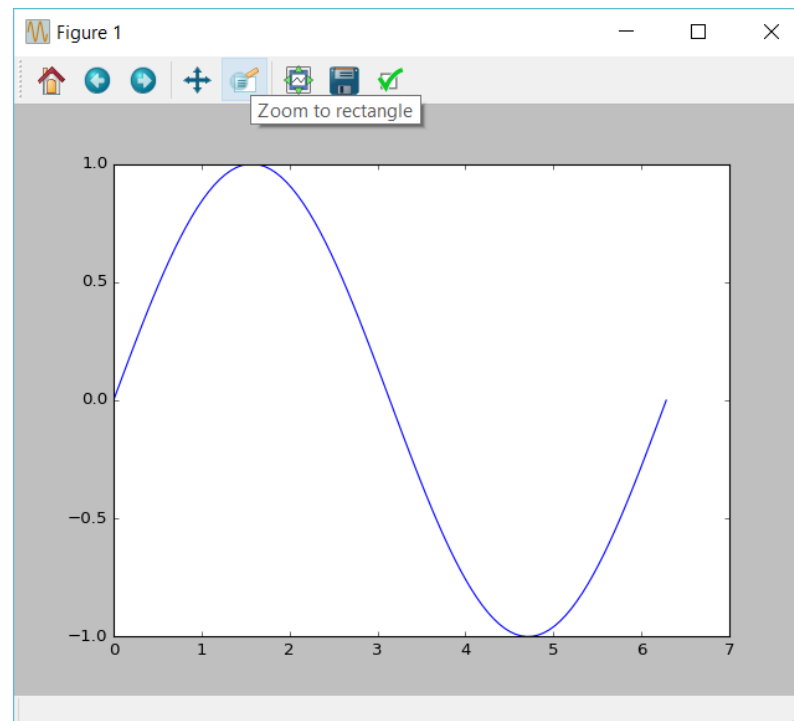
```
print(x)
```



```
[ 1.  2.  3.]
```

Cálculos com numpy

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.linspace(0, 2*np.pi, 100)
5 y = np.sin(x)
6
7 plt.plot(x, y)
8 plt.show()
```



Exercícios - matrizes

1. Escreva uma função que receba e multiplique duas matrizes 2×2 sem usar a função `dot`. Devolva o resultado
2. Resolva um antigo quebra-cabeça chinês clássico: Contamos 35 cabeças e 94 pernas entre as galinhas e coelhos em uma fazenda. Quantos coelhos e quantas galinhas que nós temos?

Exercícios - matrizes

3. Crie uma matriz chamada a. Preencha a com um padrão igual ao de um tabuleiro de xadrez, em que 0 = preto e 1 = branco.
4. Crie uma matriz de zeros de tamanho 6 x 6. Preencha a matriz com o triângulo de pascal de tamanho 6 usando as posições à esquerda da diagonal principal e a imprima:

```
[ [ 1  0  0  0  0  0 ]  
  [ 1  1  0  0  0  0 ]  
  [ 1  2  1  0  0  0 ]  
  [ 1  3  3  1  0  0 ]  
  [ 1  4  6  4  1  0 ]  
  [ 1  5 10 10  5  1 ] ]
```

Para a próxima aula

- Itens 6.17, 6.18 e 6.19 – Dicionários com listas

Capítulo ? do livro

Introdução à Programação com Python,
de Nilo Ney Menezes.

(disponível na biblioteca)



Insper

www.insper.edu.br

Guido van Rossum is the creator of Python, one of the major programming languages on and off the web. The Python community refers to him as the BDFL (Benevolent Dictator For Life), a title straight from a Monty Python skit. He moved from the Netherlands to the USA in 1995, where he met his wife. Until July 2003 they lived in the northern Virginia suburbs of Washington, DC with their son Orlijn, who was born in 2001. They then moved to Silicon Valley where Guido now works for Google (spending 50% of his time on Python!).

