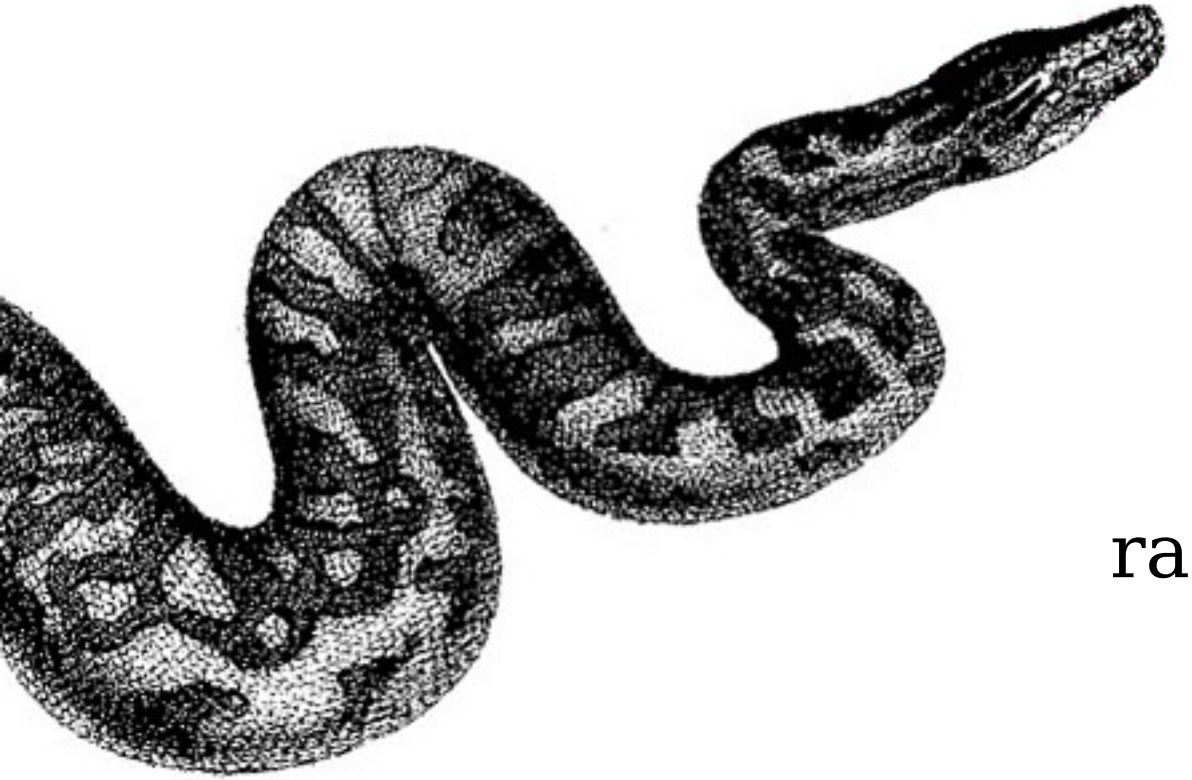
Python 2:
strings e unicode



Luciano Ramalho
ramalho@python.pro.br

PythonPro

Buscando substrings

sub in s

s contém **sub?**

s.startswith(sub), s.endswith(sub)

s começa ou termina com sub?

s.find(sub), s.index(sub)

posição de sub em s (se **sub** não existe em **s**, **find** retorna -1, **index** sinaliza **ValueError**)

rfind e **rindex** começam pela direita

s.replace(sub1, sub2)

substitui as ocorrências de sub1 por sub2

Strings

str: cada caractere é um byte; acentuação depende do encoding

strings podem ser delimitadas por:

- aspas simples ou duplas: `'x'`, `"x"`
- três aspas simples ou duplas: `'''x'''`, `"""x"""`

```
>>> fruta = 'maçã'
>>> fruta
'ma\xc3\xa7\xc3\xa3'
>>> print fruta
maçã
>>> print repr(fruta)
'ma\xc3\xa7\xc3\xa3'
>>> print str(fruta)
maçã
>>> len(fruta)
6
```

PythonPro

Strings unicode

- Padrão universal, compatível com todos os idiomas existentes (português, chinês, grego, híndi, árabe, suaíli etc.)
- Cada caractere é representado por dois bytes
- Utilize o prefixo **u** para denotar uma constante unicode: **u'maçã'**

```
>>> fruta = u'maçã'
>>> fruta
u'ma\xe7\xe3'
>>> print fruta
maçã
>>> len(fruta)
4
```

PythonPro

Conversao entre str e unicode

De **str** para **unicode**:

```
u = s.decode('iso-8859-15')
```

De **unicode** para **str**:

```
s2 = u.encode('utf-8')
```

O argumento de ambos métodos é uma string especifcando a codificação a ser usada

Codificações comuns no Brasil

iso-8859-1: padrão ISO Latin-1

iso-8859-15: idem, com símbolo € (Euro)

cp1252: MS Windows codepage 1252

- ISO Latin-1 aumentado com caracteres usados em editoração eletrônica (‘’ “” •)

utf-8: Unicode codificado em 8 bits

- compatível com ASCII até o código 127
- utiliza 2 bytes para caracteres não-ASCII
- este é o padrão recomendado pelo W3C, para onde todas os sistemas estão migrando

Codificação em scripts

As constantes str ou unicode são interpretadas segundo a codificação declarada num comentário especial no início do arquivo .py:

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

Codificação em scripts (2)

Exemplo:

```
# -*- coding: iso-8859-15 -*-

euro_iso = '€'
print '%x' % ord(euro_iso)
euro_unicode = u'€'
print '%x' % ord(euro_unicode)
```

Resultado:

```
a4
20ac
```

PythonPro

Como gerar strings com variáveis embutidas

Operador de interpolação: **`f % tupla`**

```
>>> m = 'Euro'
>>> t = 2.7383
>>> f = 'O %s está cotado a R$ %0.2f.'
>>> print f % (m,t)
O Euro está cotado a R$ 2.74.
```

Tipos de conversão mais comuns:

`%s, %f, %d`: string, float, inteiro decimal

Aprendendo a aprender:

Google: Python String Formatting Operations

Algumas funções com strings

chr(n): retorna uma string com um caractere de 8-bits cujo código é **n**

unichr(n): retorna uma string com um caractere Unicode cujo código é **n**

ord(c): retorna o código numérico do caractere **c** (pode ser Unicode)

repr(x): conversão de objeto para sua representação explícita em **Python**

len(s): número de caracteres da string

Alguns métodos de strings

s.strip()

retira os brancos (espaços, tabs e newlines) da frente e de trás de **s** (+ parâmetros)

rstrip e lstrip retiram à direita e à esquerda

s.upper(), s.lower(), s.capitalize()

converte todas maiúsculas, todas minúsculas, primeira maiúscula por palavra

s.isdigit(), s.isalnum(), s.islower()...

testa se a string contém somente dígitos, ou somente dígitos e letras ou só minúsculas