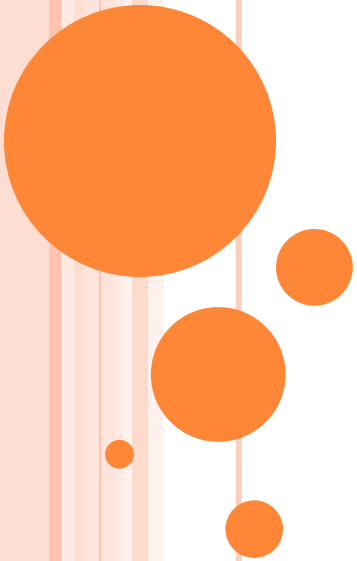


Python para PLN

Introdução ao Python

Rogério Figueredo de Sousa rogerfig@usp.br
Roney Lira de Sales Santos roneysantos@usp.br
Prof. Thiago A. S. Pardo

Introdução à Linguagem

- Linguagem de alto nível
  - Ao ler o comando, já se presume o que ele significa!
- Vários tipos de programação:
  - Modular: divisão do algoritmo em blocos
  - Orientada a objetos: classes e objetos referenciados
  - Funcional: aplicação de funções matemáticas
- Tipagem **forte** e dinâmica
- Grande coleção de bibliotecas
- Código aberto

Introdução à Linguagem

- Python é uma linguagem interpretada
- Não existe etapa de compilação de código
  - Assim como em C, C++, Java
- Basta executar o comando Python e pronto!

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> print(1+1)
2
>>> print("roney" + "lira")
roneylira
>>> x = 4+6
>>> x
10
>>>
```

Introdução à Linguagem

- Não é preciso terminar comandos com ;
- Não é preciso declarar o tipo de dados das variáveis

```
>>> x = 4+6
>>> x
10
>>> y = "roney lira"
>>> y
'roney lira'
>>> k = 1.8 - 1.5
>>> k
0.30000000000000004
>>> type(x)
<class 'int'>
>>> type(y)
<class 'str'>
>>> type(k)
<class 'float'>
>>>
```

Introdução à Linguagem

- Instalação e uso: site oficial do Python
  - https://www.python.org/downloads/

- Os códigos podem ser executados:
  - Por meio de linha de comando (prompt Windows, terminal Linux, IDLE Python...)
    - python programa.py
    - ./programa.py
  - Em programas como PyCharm ou Sublime

Tipos de Dados

```
>>> x
10
>>> y = "roney lira"
>>> y
'roney lira'
>>> k = 1.8 - 1.5
>>> k
0.30000000000000004
>>> type(x)
<class 'int'>
>>> type(y)
<class 'str'>
>>> type(k)
<class 'float'>
>>> x > 6
True
>>> x > 15
False
>>>
```

- Tipos de dados básicos
  - **`int, string, float, bool`**...
- O tipo de uma variável muda conforme o valor que lhe é atribuído.
  - Princípio da dinâmica
- **`type(var)`**

Tipagem Forte

```
>>> v = "1"
>>> q = 1
>>> v + q
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    v + q
TypeError: can only concatenate str (not "int") to str
>>>
```

- Ou seja, Python é uma linguagem que tem a característica de que não permite um mesmo dado ser tratado como se fosse de outro tipo.
- Mais sobre isso aqui.

Indentação

- Bom, Python não usa nenhum limitante para estrutura de bloco, comum em outras linguagens
  - **{**, **}**, **;**, **begin**, **end**, e por aí vai.

- A estrutura do bloco é definida pela indentação, ou seja, o alinhamento dos comandos é que define a estrutura!

```
if num % 2 == 0:
    par = par + 1
    print("PAR")
else:
    impar = impar + 1
    print("IMPAR")
```

```
14  if num % 2 == 0:
15  print("PAR")
```

```
  File "C:\Users\roney\OneDrive\Documentos\10.
  PLN\aula8.py", line 15
    print("PAR")
        ^
IndentationError: expected an indented block
[Finished in 0.4s]
```

Para nossas práticas, usaremos...

- Listas e Tuplas
- Dicionários
- Arquivos
- Strings
  - Textos, cara do nosso PLN né! =)

- Claro, existem beeeem mais coisas relacionadas ao Python!
  - Estruturas de controle, funções, classes, programação funcional, uso de *frameworks* diversos...

- E também *frameworks* relacionados ao PLN, como **NLTK** e **spaCy**
  - Em breve nas telinhas do curso!

Listas e Tuplas

- Estruturas de dados nativas da linguagem
  - **`list`** e **`tuple`**
- Informações dentro das listas e tuplas podem ser de tipos diferentes
- Acesso sequencial: fatias (*slicing*) ou diretamente
- Métodos prontos para adicionar, remover, ordenar, procurar, contar, entre vários outros
- Listas: mutáveis e delimitadas por colchetes
- Tuplas: imutáveis e delimitadas por parênteses

Listas - Métodos

```
l = list(range(5))
print(l)
l.append(5)
print(l)
l.insert(0,6)
print(l)
l.reverse()
print(l)
l.sort()
print(l)
```

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

- **`extend(L)`**: inclusão de uma lista **`12`** (**`append`**)
- **`remove(x)`**: remove a primeira ocorrência de **`x`**
- **`index(x)`**: índice da primeira ocorrência de **`x`**
- **`count(x)`**: número de ocorrências de **`x`** na lista

Listas - Slice

- Funciona como uma sublista da lista

```
>>> l = list(range(10))
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> l[2:7]
[2, 3, 4, 5, 6]
```

- A notação **`[2:7]`** significa qual o intervalo da lista original você pretende retornar
  - Lembrando que os índices são contados a partir do zero. Assim, no nosso exemplo, queremos do 3º elemento até o 7º.

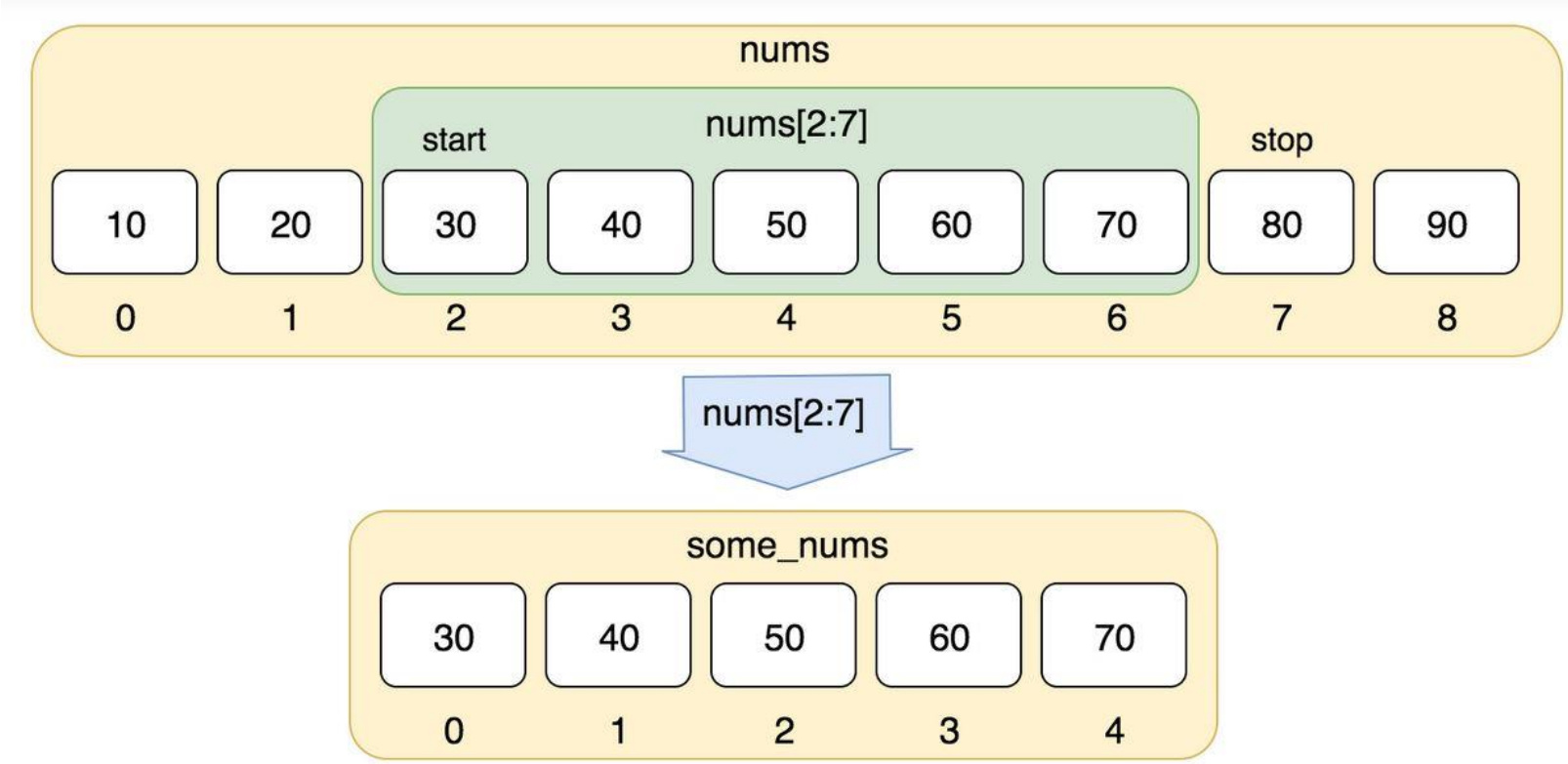
Listas - Slice

- Também é possível pular elementos, incluindo a quantidade em um outro índice.

```
>>> l = list(range(10))
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> l[2:7]
[2, 3, 4, 5, 6]
>>> l[2:7:2]
[2, 4, 6]
```

- A notação **`[2:7:2]`** significa, então, que queremos uma sublista do 3º ao 7º elemento, pulando de 2 em 2 elementos.

Listas - Slice



Disponível em https://railsware.com/blog/python-for-machine-learning-indexing-and-slicing-for-lists-tuples-strings-and-other-sequential-types/

Listas - Slice

- Conseguimos também retornar os **n** primeiros elementos e os **n** últimos elementos por meio do *slicing*.

- Os **n** primeiros elementos: **[:n]**
  - Ou seja, deixa-se vazio o primeiro index da lista

```
>>> l[:5]
[0, 1, 2, 3, 4]
```

- Os **n** últimos elementos: **[-n:]**
  - Deixa-se vazio o segundo index da lista

```
>>> l[-5:]
[5, 6, 7, 8, 9]
```

Listas - Slice

- Além disso, várias outras combinações podem ser feitas...

- Retornar todos os elementos menos os **`n`** últimos:
  - **`l[:-n]`**

- Pular de **`n`** em **`n`** elementos na lista:
  - **`l[::n]`**

- O **`reverse()`** pode ser representado por **`l[::-1]`**

- E vááárias outras possibilidades!
  - Dá uma olhada aqui, ó!

Tuplas

- Tuplas seguem o mesmo conceito das listas, porém com uma diferença importante: são **imutáveis**.

- Existe uma outra diferença: tuplas são para elementos **heterogêneos**, ou seja, tipos diferentes dentro da tupla.
  - Porém, como o Python é uma linguagem dinâmica, essa característica também aparece nas listas.
  - Cabe ao programador decidir tal característica.

TUPLAS

- Por exemplo, se você tentar modificar um elemento da tupla, você não consegue. Exemplo:

```
>>> t = (1, 'roney', 2, 'lira')
>>> t
(1, 'roney', 2, 'lira')
>>> t[0] = 5
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    t[0] = 5
TypeError: 'tuple' object does not support item assignment
>>>
```

Tuplas

- Então você pode perguntar: **Mas então, qual a diferença entre usar uma tupla em vez de uma lista e vice-versa?**

- Por serem imutáveis, tuplas representam informações que **não devem ser modificadas**,
  - Exemplos: vetores dos embeddings e classes morfossintáticas retornadas por um *tagger*.

- Arranjo das tuplas é similar ao das listas, só não se usam os métodos para modifica-las.
  - Apenas o **`count()`** e o **`index()`** estão disponíveis.