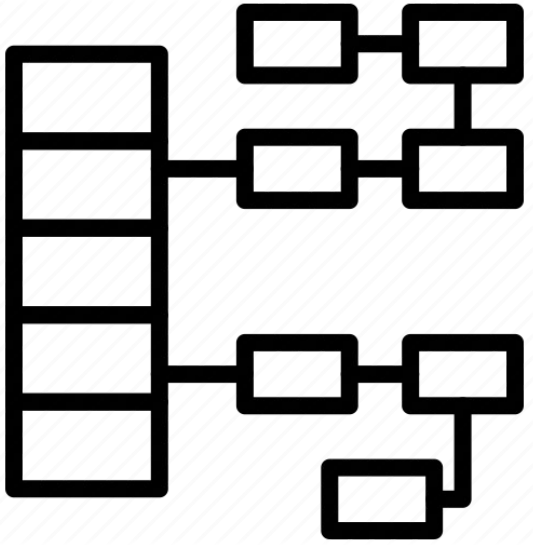
INE5404 Programação Orientada a Objetos II

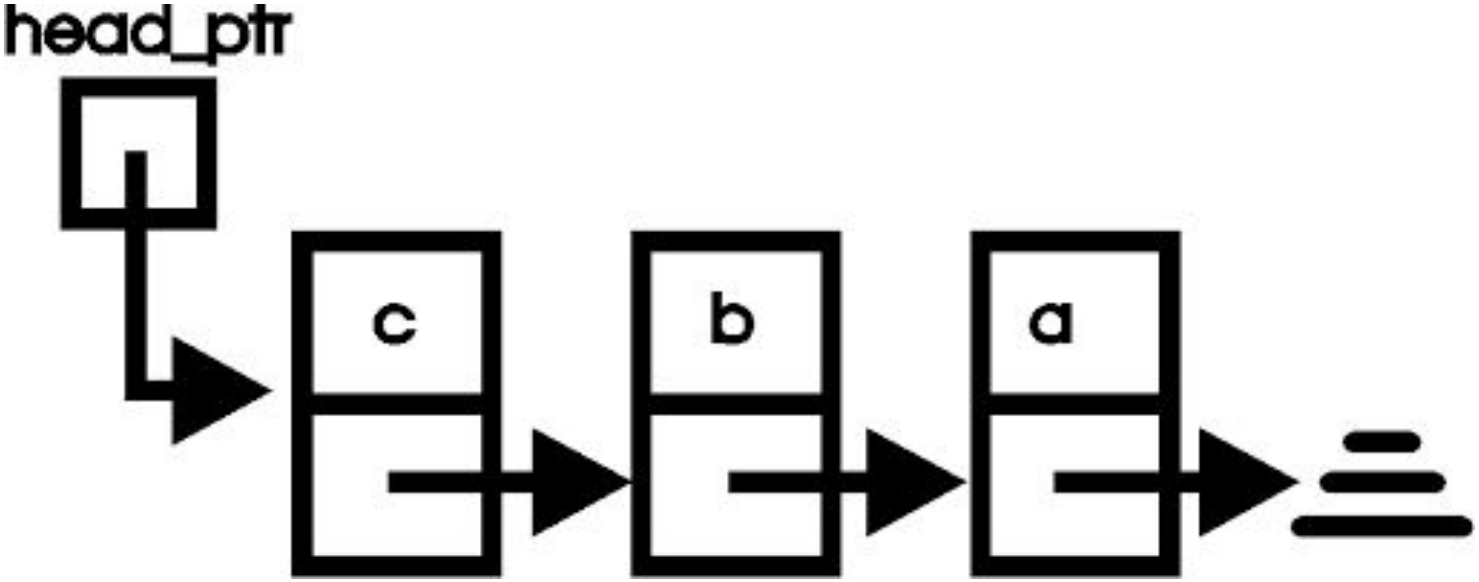
Listas, dicionários e similares

Prof. Jônata Tyska Carvalho
Prof. Mateus Grellert

Listas, Dicionários e Similares

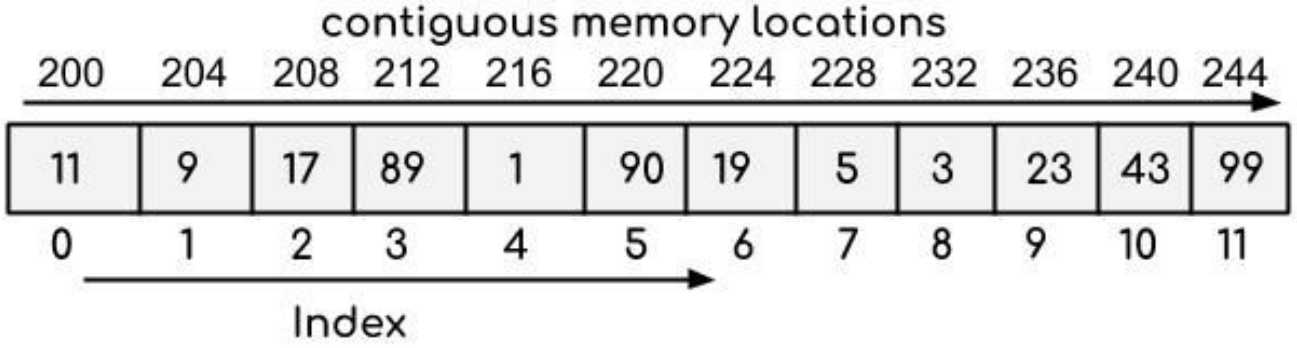


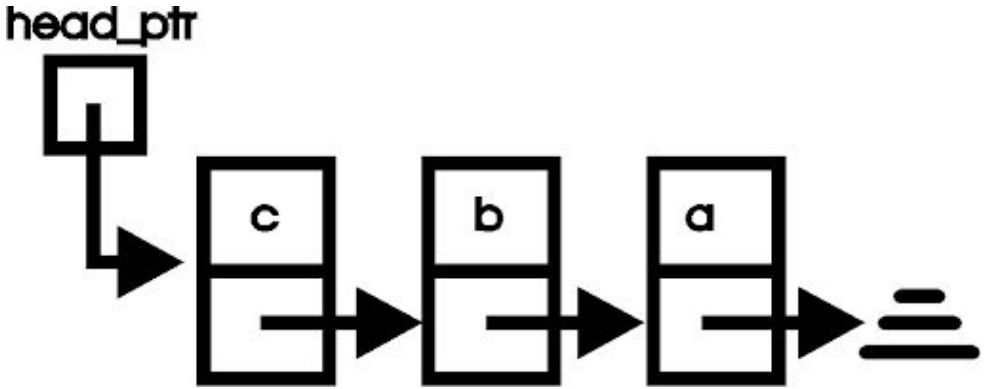
Listas em Computação



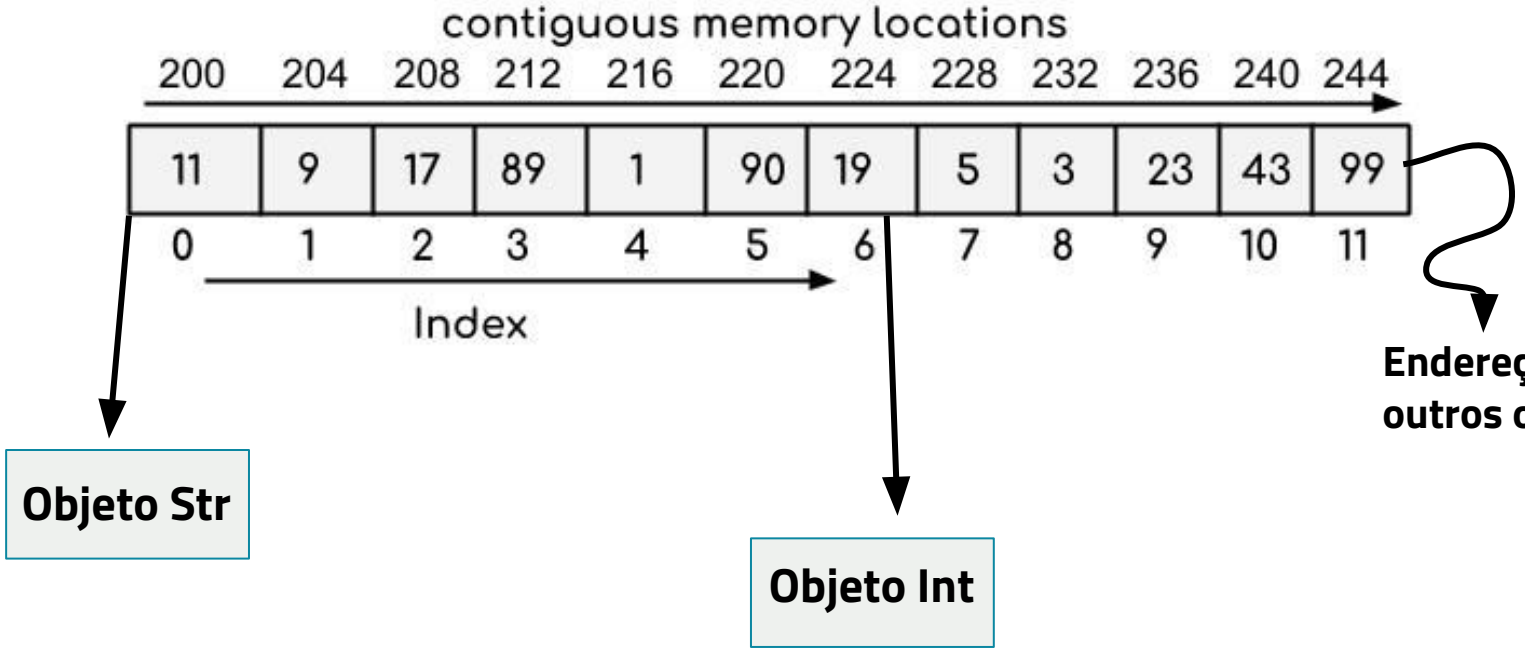
Listas em Python

Arrays x Listas





Listas em Python – internamente



Listas

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

Listas em Python

- Coleções heterogêneas de objetos;
    - Diferentes tipos de dados;
    - Incluindo outras listas;
- Podem ser ‘fatiadas’ (slice) como as strings;
- Diferente das strings, são mutáveis.

Listas - Exemplos - Criação

```
#lista vazia
progs = []

#lista com strings
progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

#lista com tipos diferentes
progs = [1, 'Genesis', 3.14]
```

Listas - Exemplos - Percorrendo

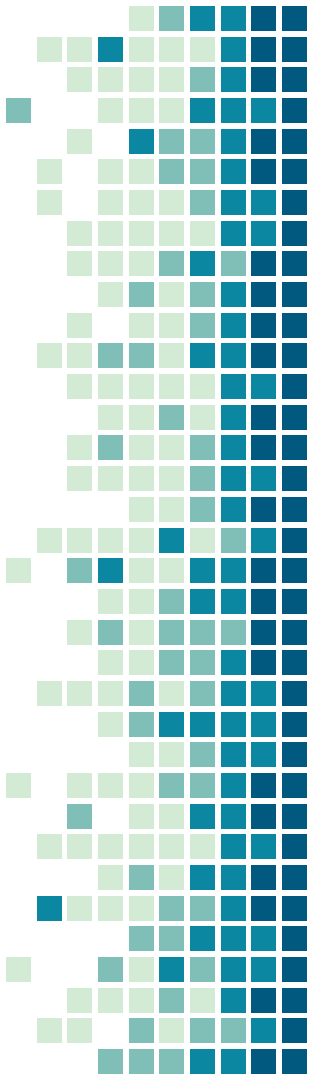
```
progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

#apenas valores
for prog in progs:
    print(prog)

#indice e valores
for i, prog in enumerate(progs):
    print(i+1, '=>',prog)
```

Listas - Exemplos - Percorrendo

```
progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

#indice e valores - elegant
for i, prog in enumerate(progs):
    print(i+1, '=>',prog)

#indice e valores - ugly
for i in range(len(progs)):
    print(i+1,'=>',progs[i])
```

Listas - Exemplos:
Incluindo , substituindo e removendo

```
#incluindo
progs.append('Camel')

#substituindo o ultimo elemento
progs[-1] = 'King Crimsom'

#substituindo o ultimo elemento
progs.remove('Pink Floyd')
```



Listas – Exemplos: Ordenando e invertendo

```
#ordena a lista
progs.sort()

#inverte a lista
progs.reverse()

#slicing - imprime os tres primeiros itens
print(progs[1:3])
```

Listas – Exemplos: Ordenando e invertendo

```
#ordena a lista
progs.sort()

#inverte a lista
progs.reverse()
```

Realizadas na própria lista, isto é, sem gerar uma nova lista

```
#slicing - imprime os tres primeiros itens
print(progs[1:3])
```

Listas - Exemplos: Copiando Listas

```
lista1 = [1,5,2]
lista2 = lista1
lista2[0] = 3
print(lista1)
print(lista2)
print(id(lista1))
print(id(lista2))
```

Explique o que acontece e por quê?

Listas - Exemplos: Copiando Listas

```
#copia a apenas os valores
lista1 = [1,5,2]
lista2 = lista1[:]
    ou list(lista1)
    ou lista1.copy()
    ou copy.deepcopy(lista1) - listas de listas (import copy)
lista2[0] = 3
print(lista1)
print(lista2)
print(id(lista1))
print(id(lista2))
```

Listas – o método pop

- Facilita a implementação de Pilhas e Filas

```
#retorna e remove o primeiro elemento
progs.pop(0)

#retorna e remove o último elemento
progs.pop() #equivalente a pop(-1)
```

Listas – o método pop

- Facilita a implementação de Pilhas e Filas

```
fila = []
fila.append('B');fila.append('C');fila.append(1)
print(fila)

while fila:
    print('Saiu', fila.pop(0), ' da fila, faltam', len(fila))
```

Listas – o método pop

- Facilita a implementação de Pilhas e Filas

```
pilha = []
pilha.append('B');pilha.append('C');pilha.append(1)
print(pilha)

while pilha:
    print('Saiu', pilha.pop(), ' da pilha, faltam', len(pilha))
```

Tuplas

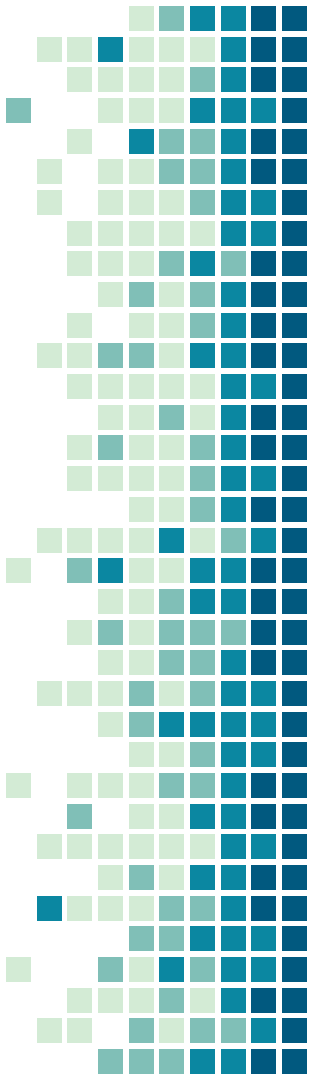
- Listas imutáveis;
- Não se pode acrescentar, apagar ou atribuir;
- Representada por parênteses (opcionais);
- Podem ser convertidas em listas e vice-versa;
- Pode conter elementos mutáveis, mas não podem ser ‘re-atribuídos’.

Tuplas – exemplos – criação

```
#criando uma tupla
tupla = (2,5,9,12)

#parentesis opcionais
tupla = 1,7,12

#funciona?
tupla[2]=5
```

Tuplas – exemplos – acesso

```
tupla = (2,5,9,12)

#acesso igual a Listas
segundo_elemento = tupla[1]

#particularidade de tuplas com um elemento
tupla = (2,)
```



Tuplas - exemplos - conversão

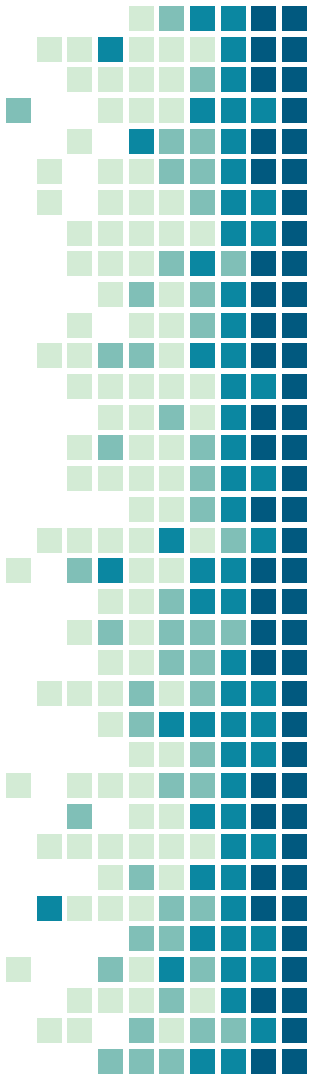
```
#convertendo de lista para tupla
tupla = tuple([2,5,8])

#convertendo de tupla para lista
lista = list(tupla)
```

Tuplas – exemplos – elementos mutáveis

```
#criando uma tupla com elemento mutável
tupla = (9,[2,5],12)

#funciona?
tupla[1].append(18)

#funciona?
tupla[1]=[2,5,18]
```

Listas e Tuplas – 'desempacotamento'

```
#desempacotando
lista = [2,4,6]
a, b, c = lista
print(a,b,c)

#funciona?
tupla = (2,4,6)
a, b, c = tupla
a=8
print(a,b,c)
```



Listas e Tuplas – 'desempacotamento' 2

```
#desempacotando - varios elementos
lista = [2,4,6,8,10]
a, b, *resto = lista
print(a,b,resto)

#funciona?
lista = [2,4,6,8,10]
a, *resto, b = lista
print(a,b,resto)
```

Sets and Frozensets

- sequencia unívoca (sem repetições) não ordenada;
  - set: mutável;
  - Frozenset: imutável;

- implementam operações de conjuntos: união, intersecção e diferença.

Sets and Frozensets - operações

```
#cria conjuntos
s1 = set(range(3))
s2 = set(range(10,7,-1))
s3 = set(range(2,10,2))

#exibe
print('s1:',s1,'\ns2:',s2,'\ns3:',s3)
```

Sets and Frozensets - operações

```
#uniao
s1s2 = s1.union(s2)
print(s1s2)

#diferenca
print(s1s2.difference(s3))

#interseccao
print(s1s2.intersection(s3))
```

Sets and Frozensets - operações

```
#subset
if s1.issuperset([1,2]):
    print('s1 inclui 1 e 2')

#inexistencia de elementos em comum
if s1.isdisjoint(s2):
    print('s1 e s2 nao tem elementos em comum')
```

Sets and Frozensets

O que acontece quando uma lista é convertida para set ou frozenset?

Sets and Frozensets

O que acontece quando uma lista é convertida para set ou frozenset?

Repetições são descartadas!

Sets and Frozensets - operações

```
#uniao
s1s2 = s1.union(s2)
print(s1s2)

#diferenca
print(s1s2.difference(s3))

#interseccao
print(s1s2.intersection(s3))
```

EXERCÍCIO!

Criar um tipo de dado (classe) MinhaLista que possui o comportamento de uma lista *builtin* mas é implementada usando tuplas. O novo tipo de dado deve permitir atribuição e acesso aos elementos da lista, e conter os seguintes métodos:

- **append()**
- **remove(element)**
- **sort()**
- **reverse()**
- **pop(index=-1)**

dica: usar magic methods

Dicionários

- Listas de associações
  - chave única ⇒ valor (qualquer tipo de dado)
  - mutáveis como as listas
  - chave precisa ser imutável: normalmente string
  - Até o Python 3.7: não existe garantia de ordenação das chaves, ou preservação da ordem

Dicionários - Exemplos

```
#criacao
dicionario = {"nome": "Shirley Manson",
              "banda":"Garbage",
              "Ano":1993}
#funciona?
dicionario = {"nome": "Shirley Manson",
              "banda":"Garbage",
             (1,5,4):1993}
```

EXERCÍCIO!

Criar 10 frozensets com 30 números aleatórios cada, e construir um dicionário que contenha a soma de cada um deles.

Dicionários - Exemplos

```
#adicionando elementos
dicionario["album"] = "Version 2.0"


#apagando elementos
del dicionario["album"]


#obtendo itens, chaves e valores:
print('itens:', dicionario.items())
print('chaves:', dicionario.keys())
print('valores:', dicionario.values())
```

Dicionários - Exemplos

```
progs = {"Yes":['Close to the Edge','Fragile'],
         "Genesis":['Foxtrot','The Nursery Crime'],
         "ELP":['Brain Salad Surgery']}
progs["King Crimsom"] = ['Red','Discipline']
a = progs.items()
for prog, albuns in a:
   print(prog,"=>",albuns)
print(len(progs),"bandas")
if 'ELP' in progs:
   del progs["ELP"]
print("Agora",len(progs),"bandas")
for prog, albuns in a:
   print(prog,"=>",albuns)
```

Bibliografia

BORGES, L. E. Python: para desenvolvedores. São Paulo: Novatec, 2014

EXERCÍCIO!

Crie uma classe MatrizEsparsa que pode ser construida das duas formas abaixo, e contenha métodos para mostrar a matriz no formato String e também no formato de dicionário.

1) **receber uma dupla (tupla com 2 elementos) indicando quantas linhas e colunas tem a matriz, e um dicionário com duplas como chave (linha, coluna) e um valor numérico.**
2) **Receber uma string no seguinte formato:**

```
matriz = '''0 8 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 2 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
0 0 0 7 0 0 0 0 0
0 0 0 0 0 0 4 0 0'''
```

Dica: use os métodos splitlines() e split da classe String, e o método get() da classe Dict

Programação Orientada a Objetos
INE5404

Fim

jonata.tyska@ufsc.br
Sala INE 404