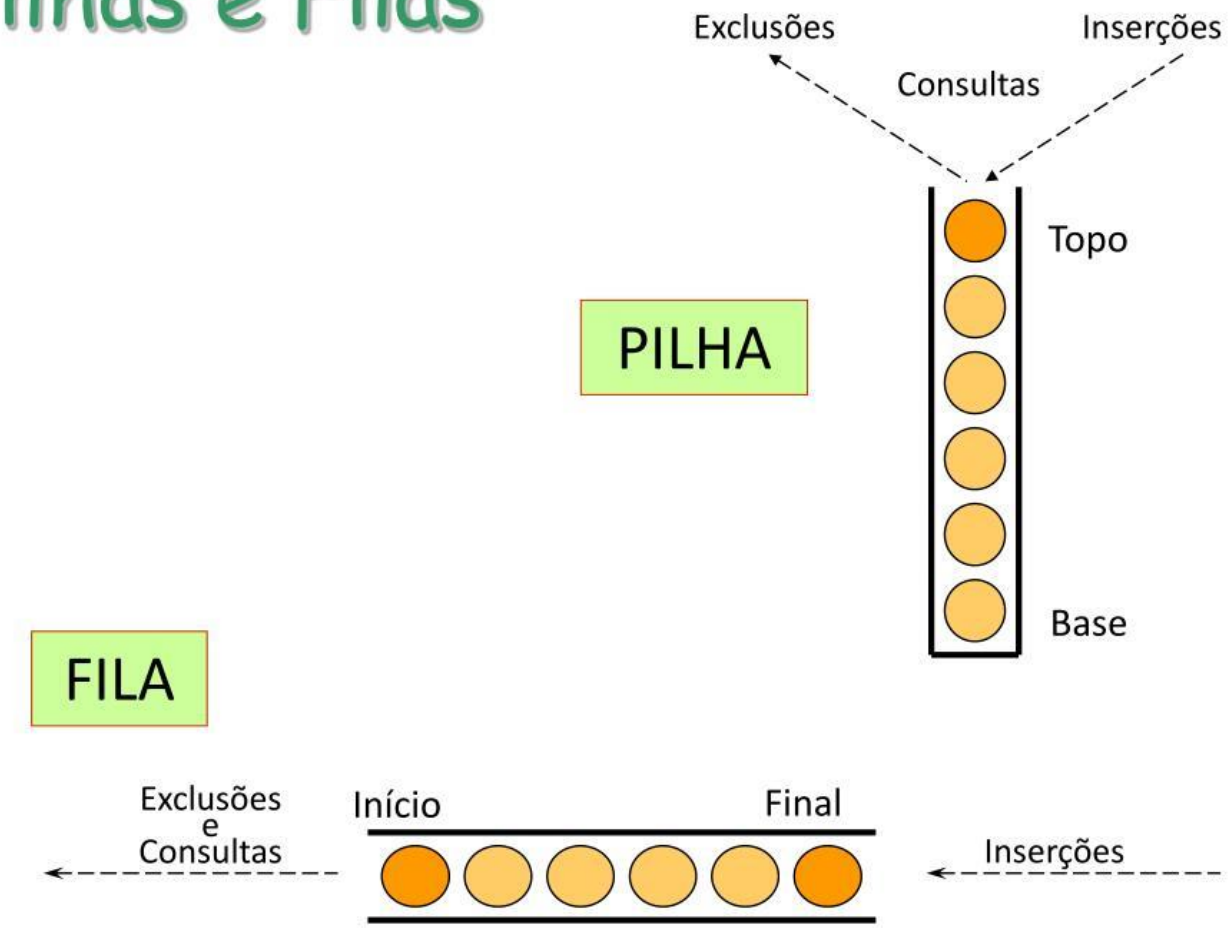
Listas em Python

Disciplina: Programação para Engenharia

Professores: Cristiane Pavei Martinello Fernandes
Douglas de Medeiros Deolindo
Giovani Martins Cascaes
Joel Barbosa Panchyniak
Marcelo Marcos Amoroso
Marcos Antônio Jeremias Coelho
Ramon de Souza Coan

Estruturas de dados

Pilhas e Filas



Variáveis de estrutura de dados

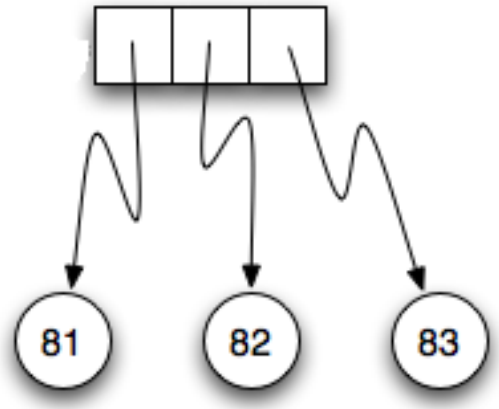
Em Python é possível compor estruturas de dados primitivas, e armazená-las em outras estruturas, chamadas de estruturas compostas.

- Listas
- Tuplas
- Dicionários

Listas em Python

Lista são cadeias de valores, isto é, armazenam mais de um valor. São estruturas ordenadas, mutáveis e heterogêneas.

- Ordenadas: cada valor tem seu índice, na ordem que estão armazenados.
- Mutáveis: é possível alterar os valores.
- Heterogêneas : valores podem ser de tipos diferentes.



O que é uma lista?

Uma lista é um tipo de dados específico utilizado em Python que permite que vários tipos de informações sejam gravados nela.

Estas informações podem ser inteiras, pontos flutuantes ou strings, isto vai depender do que será utilizado.

Para criar uma lista, basta apenas utilizar uma variável, inserir os valores entre os “[]”, separando-os por vírgula, como no exemplo a seguir:

```
lista_exemplo = [1, 'Pedro', 3.54]
print(lista_exemplo)
[1, 'Pedro', 3.54]
```

Inicialização de uma lista

Inicialização de uma lista com valores

Nome_Lista = [valor1, valor2, ..., valorN]

```
>>> x = [1,2,3,4,5]
>>> x
[1, 2, 3, 4, 5]
```

Inicialização de uma lista vazia

```
>>> x = list()
>>> x = []
```

Acessando elementos de uma lista

O acesso aos elementos da lista é feito por meio de um índice, cuja numeração começa em zero.

```
>>> x = [1,'Maria',True,120.2]
>>> x[0]
1
>>> x[2]
True
>>> x[1]
'Maria'
>>> x[3]
120.2
```

Acessando elementos de uma lista

Um item pode ser selecionado individualmente pelo uso dos colchetes, Lista[índice]

```
>>> x = [1,['Maria','Da','Silva'],32,120.2]
>>> x[0]
1
>>> x[1]
['Maria', 'Da', 'Silva']
>>> x[1][0]
'Maria'
>>> x[1][0] + x[1][1] + x[1][2]
'MariaDaSilva'
```

Acessando elementos de uma lista

Acessar com índices negativos, a partir do -1, percorrem a lista de trás para frente.

```
>>> x = [1,['Maria','Da','Silva'],32,120.2]
>>> x[-1]
120.2
>>> x[-3]
['Maria', 'Da', 'Silva']
>>> x[-3][1]
'Da'
```

Exemplo 1

Elabore um script em linguagem Python que contenha uma lista com o nome de 5 dos seus melhores amigos. Mostrar na tela a seguir, cada nome em uma linha separada por meio de estrutura de repetição.

Agora, ao invés de apenas exibir o nome de cada um de seus amigos, adicione um mensagem para eles. A mensagem deve ser a mesma, no entanto será personalizado com o nome de cada um deles.

Modificando elementos de uma lista

Como a lista é mutável, os valores nela armazenados podem ser alterados e novos valores podem ser adicionados.

```
>>> x = [1,['Maria','Da','Silva'],32,120.2]
>>> x[2] = ['Pedro','Santos']
>>> x
[1, ['Maria', 'Da', 'Silva'], ['Pedro', 'Santos'], 120.2]
>>> x[1]= 2
>>> x
[1, 2, ['Pedro', 'Santos'], 120.2]
>>> x[3]
120.2
```

Operações em listas

append() - Adiciona elementos a lista.

```
>>> nomes = ['Yuri','Joao','Maria']
>>> nomes.append('Pedro')
>>> nomes
['Yuri', 'Joao', 'Maria', 'Pedro']
```

‘**+**’ – Concatena Listas

```
>>> nomes = ['Yuri','Joao','Maria']
>>> nomes = nomes + ['Pedro']
>>> nomes
['Yuri', 'Joao', 'Maria', 'Pedro']

>>> nomes = ['Yuri','Joao','Maria']
>>> nomes = nomes + ['Pedro','Debora']
>>> nomes
['Yuri', 'Joao', 'Maria', 'Pedro', 'Debora']
```

Exemplo 2

Desenvolva um script em linguagem Python com uma lista vazia. Em seguida, leia e insira 5 valores inteiros na lista. Ao final mostre na tela, os valores contidos na lista. Ordem de entrada e inversa.

Operações em listas

insert() - Insere na posição especificada um elemento

```
>>> x = ['a','c','d']
>>> x.insert (1, 'b')
>>> x
['a', 'b', 'c', 'd']
```

pop() - Remove e retorna o elemento da posição especificada

```
>>> x = ['a','b','c','d']
>>> x.pop(3)
'd'
>>> x
['a', 'b', 'c']
```

Operações em listas

remove() - Remove o elemento especificado

```
>>> x = ['a','b','c','d']
>>> x.remove('c')
>>> x
['a', 'b', 'd']
```

Operações em listas

‘*’ - Multiplica a concatenação da lista

```
>>> nomes = ['Yuri','Joao','Maria'] * 2
>>> nomes
['Yuri', 'Joao', 'Maria', 'Yuri', 'Joao', 'Maria']
```

clear() – Esvazia a lista

```
>>> x = [1,'Maria',True,120.2]
>>> x.clear()
>>> x
[]
```

Operações em listas

count() - Retorna quantas ocorrências há do argumento passado

```
>>> x = ['Pedro','Maria','Joao','Maria']
>>> x.count('Maria')
2
```

index() – Retorna o índice da primeira ocorrência do argumento passado

```
>>> x = ['Pedro','Maria','Joao','Maria']
>>> x.index('Maria')
1
```

Operações em listas

reverse() - Inverte a ordem dos elementos

```
>>> x = [100,50,20,0]
>>> x.reverse()
>>> x
[0, 20, 50, 100]
```

sort(reverse = True ou False**)** – Ordena a lista em ordem crescente ou decrescente.

```
>>> x = [4,19,5,2,99,102]
>>> x.sort(reverse=True)
>>> x
[102, 99, 19, 5, 4, 2]
```

```
>>> x = [4,19,5,2,99,102]
>>> x.sort(reverse=False)
>>> x
[2, 4, 5, 19, 99, 102]
```

Operações em listas

len() - retorna o tamanho da lista.

```
>>> x = [4,19,5,2,99,102]
>>> len(x)
6
```

min(), **max() e sum()** - retorna o menor/maior/somatório da lista.

```
>>> x = [4,19,5,2,99,102]
>>> min(x)
2
>>> max(x)
102
```

Exemplo 3

Em um script Python, crie uma lista vazia, que seja preenchida com notas de 3 alunos. Após a leitura das notas, mostrar a nota mais alta, a nota mais baixa, a média geral de notas, além de ordenar a lista de forma crescente.

Fatiamento de listas

É possível acessar partes da lista por meio de fatiamentos. Para isso, há três parâmetros, separados por dois pontos:

lista[inicio: fim: passo]

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> x[0:2:1]
[1, 2]
>>> x[0:4:2]
[1, 3]
>>> x[4::-1]
[5, 4, 3, 2, 1]
```

O passo negativo, que indica que o fatiamento percorre a lista no sentido inverso

Fatiamento de listas

É possível também omitir os parâmetros. Por padrão, o início vale 0, o fim vale o índice do último elemento + 1 (tamanho da lista) e o passo vale 1.

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> x[:2]
[1, 2]
>>> x[5:]
[6, 7, 8, 9, 0]
>>> x[::2]
[1, 3, 5, 7, 9]
```

Criação de listas com **range()**

A função range() define um intervalo de valores inteiros. Associada a **list()**, cria uma lista com os valores do intervalo. A função range() pode ter de 1 a 3 parâmetros:

range(n) - gera um intervalo de 0 a n-1

range(i , n) - gera um intervalo de i a n-1

range(i , n, p) - gera um intervalo de i a n-1 com intervalo p entre os números

Criação de listas com **range()**

A função range() define um intervalo de valores inteiros. Associada a **list()**, cria uma lista com os valores do intervalo.

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

>>> lista = list(range(2,10,2))
>>> lista
[2, 4, 6, 8]
```

Praticando com o Python

1) Elabore um script em linguagem Python que leia de 10 números reais, inserindo-os numa lista e ao final, mostre-os na ordem inversa.

2) Desenvolva um script em linguagem Python que peça as quatro notas de 10 alunos, calcule e armazene num vetor a média de cada aluno, imprima o número de alunos com média maior ou igual a 7.

3) Crie um script em linguagem Python que leia dois vetores com 5 elementos cada. Gere um terceiro vetor de 10 elementos, cujos valores deverão ser compostos pelos elementos intercalados dos dois outros vetores. Exibir na tela todos os vetores em linhas separadas.


Obrigado!