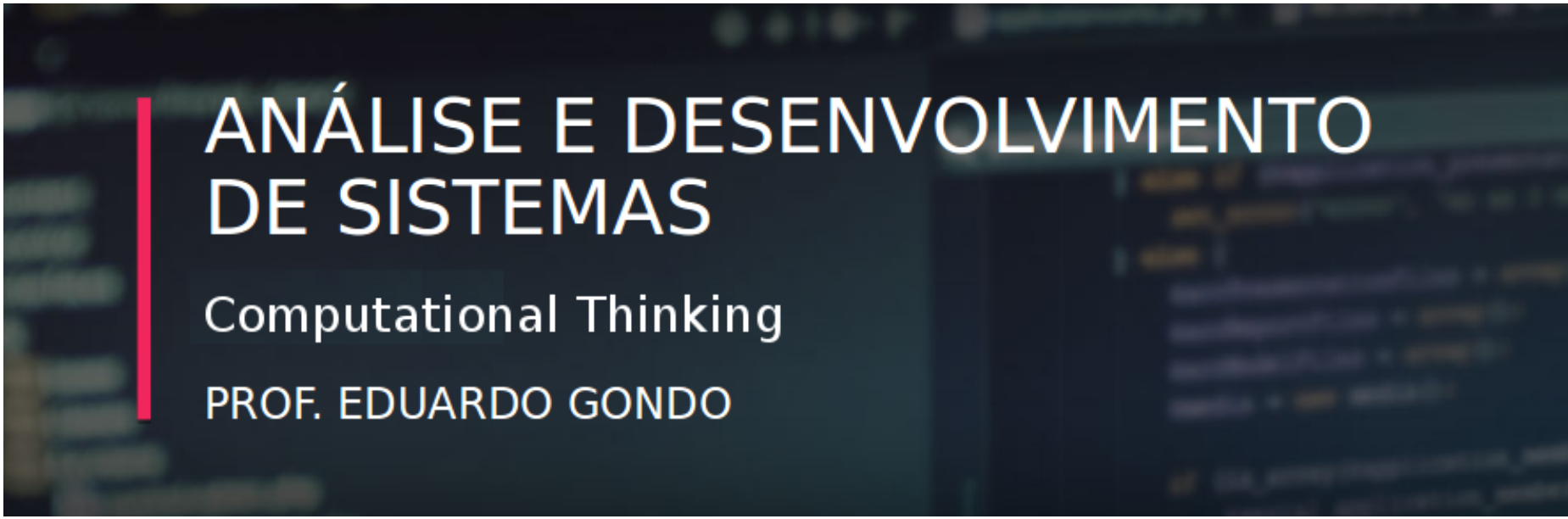
FIAP



Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

Martin Fowler

Introdução

- as tuplas representam conjunto de dados **imutáveis**
- ou seja, não é possível adicionar, alterar ou remover valores armazenados
- apenas podemos usar os valores armazenados
- exemplo: meses do ano, cores primárias, dias da semana, estações do ano, etc
- contudo, para alguns tipos de problemas, precisamos adicionar, alterar e excluir valores sobre esses conjuntos de dados
- para problemas deste tipo, usaremos as listas

Listas

- **listas** são usadas para, assim como as tuplas, representar conjunto de informações
- essas informações são organizadas de forma linear e acessadas através de um índice
- *listas* não são como *arrays* de outras linguagens
- veja abaixo alguns exemplos:

```
1  minhaLista = []                    #criando lista vazia
2  minhaLista.append("Java")          #adiciona no fim
3  minhaLista.append("Lógica")        #idem
4
5  print(minhaLista)                  #imprimindo lista
6
7  minhaLista.insert(0,"Banco")       #inserindo na posicao 0
8  print(minhaLista)
9
10 meusNumeros = [1, 4, 9]            #criando lista com 3 valores
11 meusNumeros.append(16)             #adiciona no fim
12 print(meusNumeros)                 #imprimindo lista
```

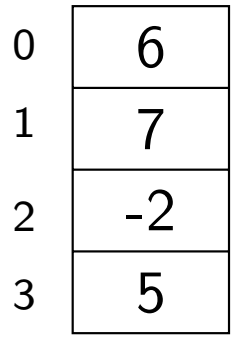
Considerações do exemplo

- duas formas de criar uma lista: sem nenhum elemento ou com alguns valores
- ex: `[]` ou `["ana", "bia", "celi"]`
- método `append(elem)` adiciona `elem` no final da lista
- ex: `conjunto.append("Fiap")`
- antes de falar do insert, a função `len(lista)` retorna a quantidade de elementos da lista
- método `insert(pos, elem)` insere o `elem` na posição `pos`
- se `pos` for maior que o tamanho da *lista*, então `elem` é adicionado no final dela
- senão todos os elementos da lista a partir da posição `pos` são deslocados uma posição para a "direita" para adicionarmos `elem` em `pos`

Representação gráfica

Tuplas ou Listas para qualquer um deles sempre faça um DESENHO.

```
1  numeros = []
2  numeros.append(6)
3  numeros.append(7)
4  numeros.append(-2)
5  numeros.append(5)
```




```
1  cor = []
2  cor.append('azul')
3  cor.append('vermelho')
4  cor.append('amarelo')
```

Mais alguns recursos

- vimos como criar, inserir e imprimir na *lista*
- agora veremos como acessar, alterar e até excluir um elemento dela

```
1   frutas = ["uva", "kiwi", "      #criando lista com quatro
        maca", "caqui"]                 elementos
2   print(frutas[2])                #acessando um elemento
3   frutas[2] = "laranja"           #alterando elemento
4   print(frutas)                   #imprimindo lista
5
6   remove = frutas.pop(1)          #removendo elemento
7   print(frutas)                   #imprimindo lista
8
9   del frutas[0]                   #removendo elemento
10  print(frutas)                   #imprimindo lista
11
12  frutas[2] = "banana"            #erro, não existe posição 2
```

Mais considerações

- acessar ou alterar um elemento da *lista*: use o nome da lista mais um índice (igual às *tuplas*)
- seja *ind* um índice da *lista* então $0 \leq ind < len(frutas)$
- python aceita índices negativos, mas vamos trabalhar o mais próximo das outras linguagens de programação
- ex: `x = frutas[0]` ou `frutas[0] = "abacaxi"`
- o método `pop(<ind>)` retorna e remove o elemento que está na posição `<ind>` da *lista*
- se você não passar nenhum argumento para o `pop` ele retorna e remove o último elemento da *lista*
- observe as impressões das linhas 4 e 7 do nosso exemplo
- também é possível remover um elemento da lista usando `del`

Listas - percorrendo os elementos

para percorrer todos os elementos de uma lista podemos fazer:

```
1  sports = ["Futebol","Vôlei","Basquete","Golf","Judô","
       Beisebol","Poker"]
2  i = 0
3  while i < len(sports):
4    print(sports[i])
5    i = i + 1
6
7  for c in sports:
8    print(c)
```

Em Python podemos usar o comando `while` combinado com a função `len` ou usarmos o comando `for`. O comando for percorre a lista do início até o último elemento.

Concatenando Listas

```
1  individual = ["natação","tênis","judô"]
2  coletivo = ["futebol","vôlei","basquete"]
3  esporte = individual + coletivo
4  print(esporte)
5
6  outros = ("Boxe","Golfe")
7  erro = coletivo + outros
8
9  for esp in outros:
10     individual.append(esp)
11 print(individual)
```

- na linha 3, criamos uma nova *lista* com a união das *listas* individual e coletivo
- na linha 6, temos uma tupla e a união de uma tupla com uma lista não é permitida (linha 7)
- podemos inserir os elementos da tupla em uma lista, essa operação é permitida

Exemplo de uso

Problema 1: Considere uma turma de n alunos onde desejamos calcular a média das notas do N2020 e saber quantos alunos estão acima ou iguais a média e quantos alunos estão abaixo da média. Escreva um algoritmo que lê um inteiro n representando a quantidade de alunos e cada uma das n notas. Depois calcula e exibe a média da turma e as quantidades de alunos que estão abaixo da média ou que estão, pelo menos, na média da sala.

Vamos dividir esse problema em etapas:

a) ler o inteiro n (qtd de alunos) e as n notas de cada aluno
b) calcular a média das notas
c) introduzir as listas e fazer a contagem dos alunos segundo seu aproveitamento

Solução comentada

Fazendo passo a passo

Solução comentada

Fazendo passo a passo

```
1  n = int(input("Qtd alunos:"))
2  i = 1
3  while i <= n:
4    nota = float(input("Nota:"))
5    i = i + 1
```

#entrada de dados

Calculando a média

Solução comentada

Fazendo passo a passo

```
1  n = int(input("Qtd alunos:"))
2  i = 1
3  while i <= n:
4    nota = float(input("Nota:"))          #entrada de dados
5    i = i + 1
```

Calculando a média

```
1  n = int(input("Qtd alunos:"))
2  i = 1
3  soma = 0
4  while i <= n:
5    nota = float(input("Nota:"))          #entrada da notas
6    soma = soma + nota                    #somatória das notas
7    i = i + 1
8
9  media = soma / n                        #cálculo da média
10 print("média:", media)
```

Solução comentada

- com a média podemos comparar com as notas dos alunos
- usaremos uma *lista* para armazenar essas notas:

```
1  n = int(input("Qtd alunos:"))
2  i = 1
3  soma = 0
4  conjunto = []
5  while i <= n:
6    nota = float(input("Nota:"))
7    soma = soma + nota
8    conjunto.append(nota)
9    i = i + 1
10
11 media = soma / n
```

```
#declarando uma lista
    vazia

#colocando as notas
    dentro da lista
```

Solução comentada (continuação)

```
12  acima = 0
13  abaixo = 0                              #criando contadores
14  for nota in conjunto:
15      if nota < media:                    #percorrendo as notas
16          abaixo = abaixo + 1                 e fazendo a
17      else:                                   contagem de acordo
18          acima = acima + 1                    com a média
19
20  print("Media", media)
21  print(acima, "alunos foram bem")
22  print(abaixo,"alunos não foram tão  #mostrando resultados
        bem)
```

- sem as listas a solução desse algoritmo não seria possível

Estatística

PROBLEMA 2 Já vimos como calcular a média aritmética de vários números. Sua tarefa agora é escrever uma função que recebe uma lista contendo número reais e retorna o maior número e o menor número da lista. O retorno do seu método deverá ser na forma de tuplas.

```
1  def maiorMenor(conjunto):
2      max = conjunto[0]
3      min = conjunto[0]
4      i = 1
5      while i < len(conjunto):
6          if conjunto[i] < min:
7              min = conjunto[i]
8          if conjunto[i] > max:
9              max = conjunto[i]
10         i = i + 1
11
12     return (min, max)
13
14 lista = [3.2, -0.1, 8.9, 5.4, -3.1, 6.2, 10]
15 resultado = maiorMenor(lista)
```

Contagem

- para as listas existe o método `count(valor)`
- ele verifica a quantidade de ocorrências de `valor` dentro da lista
- veja abaixo um exemplo:

```
1  frutas = ["maçã", "figo", "pêra", "figo", "caqui", "figo",
       "limão"]
2  x = frutas.count("figo")
3  print(x)
```

Contagem

PROBLEMA 3 Escreva uma função que se reproduz o mesmo funcionamento do método count. Sua função recebe como parâmetros uma lista, um valor e retorna a quantidade de ocorrências de valor na lista.

```
1  def contagem(lista, valor):
2      ocorrencias = 0
3      for info in lista:
4          if info == valor:
5              ocorrencias = ocorrencias + 1
6
7      return ocorrencias
8
9  frutas = ["maçã", "figo", "pêra", "figo", "caqui", "figo",
       "limão"]
10 resultado = contagem(frutas, "figo")
11 print(resultado)
```

Criando uma lista como vetores

PROBLEMA 4: Algumas vezes precisamos criar uma lista com uma capacidade definida mas sem valores específicos. Vamos criar algumas dessas listas como exemplo.

```python
#PYTHON
textos = [""] * 10
numeros = [0] * 15
n = 100
logicos = [False] * n

print(textos)
print(numeros)
print(logicos)
```

```java
//JAVA
String[] textos = new String[10];
Integer[] nums = Integer[15];
int n = 100;
Boolean[] logicos = new Boolean[n];

for (n = 0; n < nums.length; n++) {
    System.out.println(nums[n]);
}
```

IMPORTANTE: no Python podemos aumentar a capacidade da lista, mas no Java isso não é possível

for do Java no Python

- podemos fazer um `for` no Python parecido com o que temos no Java e nas outras linguagens
- usando o `range(n)` cria um "intervalo" de números de 0 a $n-1$
- veja o exemplo:

```
1  for x in range(10):                #range(10) cria
2      print(x)                       #números de 0 a 9
3
4  for i in range(1, 100):            #range(1, 100) cria
5      print(i)                       #números de 1 a 99
6
7  nums = [0] * 100                   #percorrendo o vetor
8  for i in range(len(nums)):         #parecido com outras
9      print(nums[i])                     linguagens
```

Mais exemplos

Problema 5: Crie uma função em Python que recebe uma lista contendo números inteiros e aumente todos os valores da lista em uma unidade.

```
1  def aumenta(lista):
2    i = 0
3    while i < len(lista):
4       lista[i] = lista[i] + 1
5       i = i + 1
6
7  teste = [1, 4, 6, 19, 30, 45]
8  aumenta(teste)
9  print(teste)
```

Separando os elementos da lista

- da mesma forma que podemos extrair uma substring, podemos extrair uma sublista
- veja um exemplo:

```
1  lista = [4, 6, 0, -2, 20, 40, 41, 43]
2  novaLista = lista[1:5]
3  print(novaLista)
4  print(lista)
```

- mas se quisermos algo mais específico?

Separando os elementos da lista

- da mesma forma que podemos extrair uma substring, podemos extrair uma sublista
- veja um exemplo:

```
1  lista = [4, 6, 0, -2, 20, 40, 41, 43]
2  novaLista = lista[1:5]
3  print(novaLista)
4  print(lista)
```

- mas se quisermos algo mais específico?
- por exemplo, separar os valores que estão nos índices pares dos valores que estão nos índices ímpares

Separando os elementos da lista

PROBLEMA 5: Crie uma função em Python que recebe uma lista contendo números inteiros e separa essa lista em duas outras listas. Uma contém os valores dos índices pares e a outra os valores dos índices ímpares da lista original.

```
1  def separa(lista):
2    i = 0
3    pares = []
4    impares = []
5    while i < len(lista):
6       if i % 2 == 0:
7      pares.append(lista[i])
8       else
9      impares.append(lista[i])
10      i = i + 1
11   return [pares, impares]
12
13 teste = [7, -34, 26, 1, 3, 5, 8, 0]
14 separados = separa(teste)
15 print(separados[0])
16 print(separados[1])
```

Referência Bibliográfica

- Puga e Rissetti - Lógica de Programação e Estrutura de Dados
- Ascêncio e Campos - Fundamentos da Programação de Computadores
- Forbelone e Eberspacher - Lógica de programação: a construção de algoritmos e estruturas de dados
- Documentação do Python - https://docs.python.org/3.8/
- Python Programming For Beginners: Learn The Basics Of Python Programming (Python Crash Course, Programming for Dummies) (English Edition). Kindle
- Python: 3 Manuscripts in 1 book: - Python Programming For Beginners - Python Programming For Intermediates - Python Programming for Advanced (English Edition). Kindle

Copyleft