**estrutura de dados**

**27/02**

<https://dontpad.com/estruturas2025>

LINK DA AULA

https://meet.google.com/rgp-haew-jrm

**debug: ferramenta de debug**

**break point,** representam pontos de paradas para poder testar seu código melhor, podendo marca onde irá ter essas “pausa” em cada linha selecionada

**step over**, irá pular as linha a linha sem ir para alguma chamada de uma função que irá levá lo a alguma parte do código

**step into;** irá linha a linha porem agora ele ira nas funções sendo encaminhado para a linha de código onde está essa função.

debug, ou bug no sistema vem dos “primórdios da computação onde se tinha válvulas que representa 0 ou 1 no equipamentos, e literalmente os “bugs” insetos iam até essas válvulas porque estavam quentes e acontecia de queima elas e acabava tendo “bug”.

#Criando um array de 5 posicoes vazias

class PersonalArray:

SIZE = 5

insertPosition = 0

elements = [None] \* SIZE

#funcao que serve para definir se o array esta vazio ou nao

def isEmpty(self):

return self.size() == 0

#Funcao que serve para retornar o numero de elementos ja armazenados no array

def size(self):

return self.insertPosition

#Funcao que serve para definir se precisamos de mais memoria

def isMemoryFull(self):

return self.insertPosition == len(self.elements)

#Cria uma funcao para inserir na lista

def append(self, newElement):

if self.isMemoryFull():

self.updateMemory()

self.elements[self.insertPosition] = newElement

self.insertPosition += 1

#funcao para aumenta a memoria quando necesario

def updateMemory(self):

newArray = [None] \* (self.size() + self.SIZE)

for position in range(self.insertPosition - 1):

newArray[position] = self.elements[position]

self.elements = newArray

newArray = None

#funçao para limpar as pociçoes

def clear(self):

for position in range(len(self.elements)):

self.elements[position] = None

self.elements = [None] \* self.SIZE

self.insertPosition = 0

#funçao remover o ultimo elemento da lista "array"

def remove(self):

for remove

array = PersonalArray()

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

print (array.isEmpty())

print (array.size())

print (array.isMemoryFull())

print("\ncom o metodo clear\n")

array.clear()

print (array.isEmpty())

print (array.size())

print (array.isMemoryFull())

**Codigo 2**

class PersonalArray:

SIZE = 5

insertPosition = 0

elements = [None] \* SIZE

# Função que serve para definir se o array está vazio ou não

def isEmpty(self):

return self.size() == 0

# Função que serve para retornar o número de elementos já armazenados no array

def size(self):

return self.insertPosition

# Função que serve para definir se precisamos de mais memória

def isMemoryFull(self):

return self.insertPosition == len(self.elements)

# Função para inserir na lista

def append(self, newElement):

if self.isMemoryFull():

self.updateMemory()

self.elements[self.insertPosition] = newElement

self.insertPosition += 1

# Função para aumentar a memória quando necessário

def updateMemory(self):

newArray = [None] \* (self.size() + self.SIZE)

for position in range(self.insertPosition):

newArray[position] = self.elements[position]

self.elements = newArray

newArray = None

# Função para limpar as posições

def clear(self):

for position in range(len(self.elements)):

self.elements[position] = None

self.elements = [None] \* self.SIZE

self.insertPosition = 0

# Função para remover o último elemento da lista "array"

def remove(self):

# Remove o último elemento usando o índice -1

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1 # Decrementa a posição de inserção

#funcao para retorna um lemento de uma posiçao qualquer da lista

def elementAt(self, position):

return self.elements[position]

#funcao para remover um elemento da lista com base em uma posicao

def removePosition(self, position):

print("remove alguém")

#funcao para retornar um elemento de uma posicao qualquer da lista ???????????????????

def elementAt(self, position):

return self.elements[position]

# Testando a classe

array = PersonalArray()

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

print("ultimo elemento que sera eliminado:")

print(array.elementAt(array.size() -1))

print("\nAntes de remover:")

print("Elementos:", array.elements)

print("Tamanho:", array.size())

array.remove() # Remove o último elemento

print("\nDepois de remover:")

print("Elementos:", array.elements)

print("Tamanho:", array.size())

print("\nCom o método clear:\n")

array.clear()

print("Elementos:", array.elements)

print("Tamanho:", array.size())

**codigo outro**

class PersonalArray:

SIZE = 5

insertPosition = 0

elements = [None] \* SIZE

# Função que serve para definir se o array está vazio ou não

def isEmpty(self):

return self.size() == 0

# Função que serve para retornar o número de elementos já armazenados no array

def size(self):

return self.insertPosition

# Função que serve para definir se precisamos de mais memória

def isMemoryFull(self):

return self.insertPosition == len(self.elements)

# Função para inserir na lista

def append(self, newElement):

if self.isMemoryFull():

self.updateMemory()

self.elements[self.insertPosition] = newElement

self.insertPosition += 1

# Função para aumentar a memória quando necessário

def updateMemory(self):

newArray = [None] \* (self.size() + self.SIZE)

for position in range(self.insertPosition):

newArray[position] = self.elements[position]

self.elements = newArray

newArray = None

# Função para limpar as posições

def clear(self):

for position in range(len(self.elements)):

self.elements[position] = None

self.elements = [None] \* self.SIZE

self.insertPosition = 0

# Função para remover o último elemento da lista "array"

def remove(self):

# Remove o último elemento usando o índice -1

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1 # Decrementa a posição de inserção

def remove\_at(self, position):

if not self.isEmpty() and 0 <= position < self.insertPosition:

# Reorganiza os elementos após a posição removida

for i in range(position, self.insertPosition - 1):

self.elements[i] = self.elements[i + 1]

self.elements[self.insertPosition - 1] = None

self.insertPosition -= 1 # Atualiza a posição de inserção

array = PersonalArray()

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

array.append("fusca")

array.append("kombi")

array.append("kwid")

array.append("Ferrari")

array.append("Hilux")

print("\nAntes de remover:")

print("Elementos:", array.elements)

print("Tamanho:", array.size())

#array.remove() #metodo para remover o ultimo da lista

array.remove\_at(1) #metodo para remover uma posição específica

print("\nDepois de remover:")

print("Elementos:", array.elements)

print("Tamanho:", array.size())

print("\nCom o método clear:\n")

array.clear()

print("Elementos:", array.elements)

print("Tamanho:", array.size())