array.sort.py

```
import random

# 1. Crie um array de 15 posições com números inteiros e valores aleatórios, não ordenados
array_inteiros = [random.randint(1, 100) for _ in range(15)]

# 2. Realize a ordenação dos dados do array utilizando o método “sort”
array_inteiros.sort()

# 3. Imprima, na tela, o conteúdo do array ordenado
print("Array ordenado crescente:", array_inteiros)

# 4. Utilize o método “sort” novamente, mas com parâmetros para ordenar de forma decrescente
array_inteiros.sort(reverse=True)

# 5. Imprima o array ordenado de forma decrescente
print("Array ordenado decrescente:", array_inteiros)

# 6. Crie um array de strings (nome, dataNascimento, cpf, rg)
array_strings = ["nome", "dataNascimento", "cpf", "rg"]

# 7. Ordene o array de strings de forma crescente
array_strings.sort()

# 8. Imprima o array ordenado de forma crescente
print("Array de strings ordenado crescente:", array_strings)
```

```
# 9. Ordene o array de strings de forma decrescente
array_strings.sort(reverse=True)

# 10. Imprima o array ordenado de forma decrescente
print("Array de strings ordenado decrescente:", array_strings)
```

---

bubble.sort.py

```
# Método Bubble Sort
def bubbleSort(array):
  # Laço externo para percorrer todo o array
  for i in range(len(array)):
    # Laço interno para comparar os elementos adjacentes
    for j in range(0, len(array) - i - 1):
      # Verifica se o elemento atual é maior que o próximo
      if array[j] > array[j + 1]:
        # Se for maior, troca os valores
        temp = array[j]
        array[j] = array[j + 1]
        array[j + 1] = temp

# Array de 15 números inteiros
array = [98, 12, 56, 89, 34, 73, 62, 21, 48, 94, 3, 77, 26, 15, 67]

# Aplicando o método bubbleSort ao array
```

```
bubbleSort(array)

# Imprimindo o array ordenado
print("Array ordenado:", array)
```

---

selection.sort.py

```
# Array de 15 números inteiros
array = [23, 45, 12, 78, 34, 56, 90, 67, 18, 9, 29, 100, 3, 62, 81]

# Laço externo para iterar pelos elementos do array
for i in range(len(array)):
    # A variável 'min_index' recebe o valor de 'i'
    min_index = i

    # Laço interno para comparar os elementos subsequentes ao atual
    for j in range(i + 1, len(array)):
        # Verifica se o valor na posição 'min_index' é maior que o valor na posição 'j'
        if array[min_index] > array[j]:
            # Se sim, atualiza 'min_index' para a posição 'j'
            min_index = j

    # Troca os valores: o valor na posição 'i' troca com o valor na posição 'min_index'
    array[i], array[min_index] = array[min_index], array[i]

# Imprime o array ordenado
```

```
print("Array ordenado:", array)
```

---

ler.txt.py

```
# Abrir o arquivo 'loremipsum.txt' utilizando o método 'open'
arquivo = open('loremipsum.txt', 'r')

# Ler todo o conteúdo do arquivo e imprimir na tela
conteudo = arquivo.read()
print("Conteúdo completo do arquivo:")
print(conteudo)

# Fechar o arquivo
arquivo.close()

# Reabrir o arquivo para ler novamente
arquivo = open('loremipsum.txt', 'r')

# Ler e imprimir apenas a primeira linha
primeira_linha = arquivo.readline()
print("\nPrimeira linha do arquivo:")
print(primeira_linha)

# Fechar o arquivo novamente
arquivo.close()
```

```
# Reabrir o arquivo para ler os primeiros 3 caracteres
arquivo = open('loremipsum.txt', 'r')

# Ler e imprimir os 3 primeiros caracteres
tres_primeiros = arquivo.read(3)
print("\nOs 3 primeiros caracteres do arquivo:")
print(tres_primeiros)

# Fechar o arquivo
arquivo.close()

# Utilizando a instrução 'with' para abrir e ler o conteúdo do arquivo
with open('loremipsum.txt', 'r') as arquivo:
    conteudo_com_with = arquivo.read()

print("\nConteúdo do arquivo usando 'with':")
print(conteudo_com_with)
```

---

escrever.txt.py

```
# Abrir (ou criar) o arquivo 'texto.txt' no modo de escrita ('w')
arquivo = open('texto.txt', 'w')

# Criar uma lista vazia
texto = list()
```

```
# Usar o método 'append' para adicionar frases à lista
texto.append("Eu sou fã de One Piece\n")
texto.append("Eu gosto muito de jogar Destiny 2\n")
texto.append("Quero acabar logo a faculdade\n")

# Escrever o conteúdo da lista no arquivo
for linha in texto:
    arquivo.write(linha)

# Fechar o arquivo após a escrita
arquivo.close()

print("Arquivo 'texto.txt' criado e preenchido com sucesso.")
```

---

kdd.py

```
import time

# Função Bubble Sort
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
# Função Selection Sort
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i+1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

# Função para ler o arquivo txt
def ler_arquivo(arquivo_nome):
    palavras = []
    try:
        with open(arquivo_nome, 'r', encoding='utf-8') as arquivo:
            for linha in arquivo:
                # Remover quebras de linha e separar palavras por espaços
                linha = linha.strip()
                palavras.extend(linha.split())
    except FileNotFoundError:
        print(f"Erro: O arquivo '{arquivo_nome}' não foi encontrado.")
    return palavras

# Função para salvar palavras ordenadas no arquivo
def salvar_arquivo(arquivo_nome, palavras):
    with open(arquivo_nome, 'w', encoding='utf-8') as arquivo:
        for palavra in palavras:
            arquivo.write(palavra + '\n')
```

```
# Lendo o conteúdo do arquivo loremipsum.txt
arquivo_txt = 'loremipsum.txt'  # Nome do arquivo de entrada
palavras = ler_arquivo(arquivo_txt)

# Verificar se há palavras no arquivo lido
if not palavras:
    print(f"O arquivo '{arquivo_txt}' está vazio ou não foi possível ler o conteúdo.")
else:
    # Ordenação com Bubble Sort
    palavras_bubble = palavras.copy()
    start_time = time.time()
    bubble_sort(palavras_bubble)
    end_time = time.time()
    print(f"Bubble Sort - Tempo de execução: {end_time - start_time:.6f} segundos")

    # Ordenação com Selection Sort
    palavras_selection = palavras.copy()
    start_time = time.time()
    selection_sort(palavras_selection)
    end_time = time.time()
    print(f"Selection Sort - Tempo de execução: {end_time - start_time:.6f}
segundos")

    # Ordenação com o método nativo sort()
    palavras_sort = palavras.copy()
    start_time = time.time()
    palavras_sort.sort()
    end_time = time.time()
```

```
    print(f"Sort (Método Nativo) - Tempo de execução: {end_time - start_time:.6f} segundos")


    # Salvando as palavras ordenadas no arquivo
    arquivo_saida = 'palavras_ordenadas.txt'
    salvar_arquivo(arquivo_saida, palavras_sort)
    print(f"As palavras ordenadas foram salvas no arquivo '{arquivo_saida}'.")
```