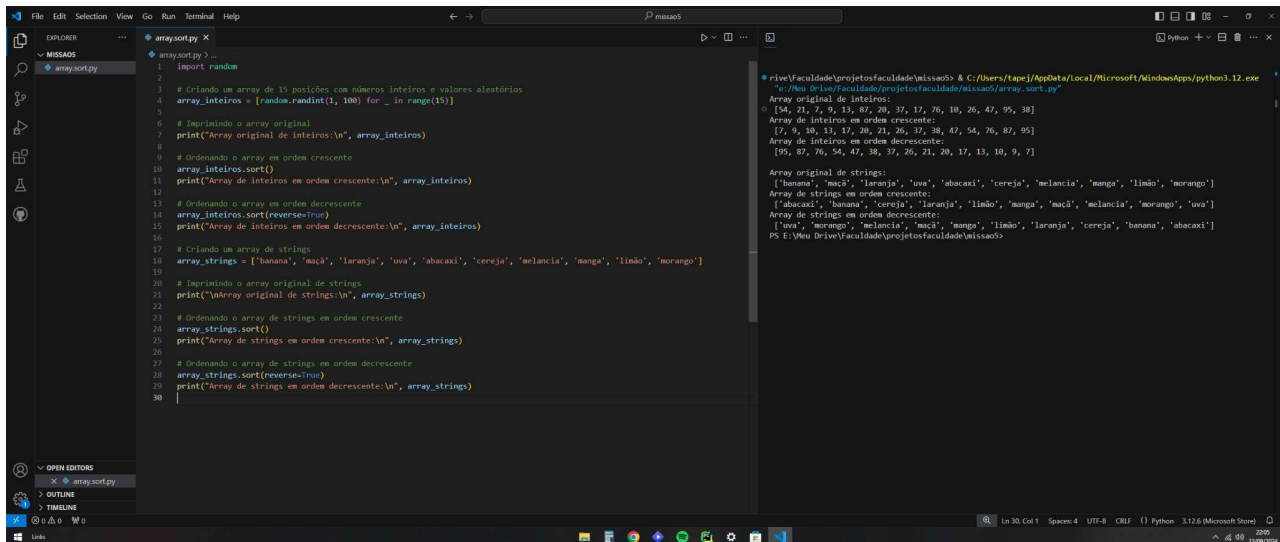


Missão Prática - Nível 5 – Mundo 1

Aluno: Tapejara Lira lung

Matrícula: 2024.06.08779-1

Microatividade 1: Ordenação de um array utilizando Python

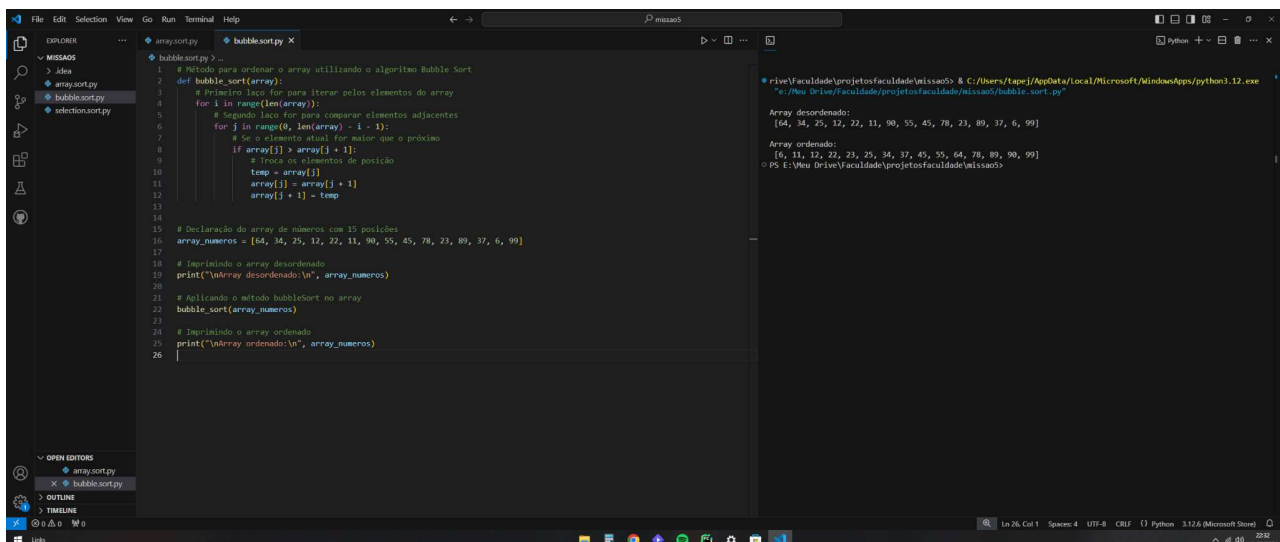


The screenshot shows a VS Code editor with a file named 'array.sort.py'. The code is as follows:

```
1 import random
2
3 # Criando um array de 15 posições com números inteiros e valores aleatórios
4 array_inteiros = [random.randint(1, 100) for _ in range(15)]
5
6 # Imprimindo o array original
7 print("Array original de inteiros:\n", array_inteiros)
8
9 # Ordenando o array em ordem crescente
10 array_inteiros.sort()
11 print("Array de inteiros em ordem crescente:\n", array_inteiros)
12
13 # Ordenando o array em ordem decrescente
14 array_inteiros.sort(reverse=True)
15 print("Array de inteiros em ordem decrescente:\n", array_inteiros)
16
17 # Criando um array de strings
18 array_strings = ['banana', 'maça', 'laranja', 'uva', 'abacaxi', 'cereja', 'melancia', 'manga', 'limão', 'morango']
19
20 # Imprimindo o array original de strings
21 print("\nArray original de strings:\n", array_strings)
22
23 # Ordenando o array de strings em ordem crescente
24 array_strings.sort()
25 print("Array de strings em ordem crescente:\n", array_strings)
26
27 # Ordenando o array de strings em ordem decrescente
28 array_strings.sort(reverse=True)
29 print("Array de strings em ordem decrescente:\n", array_strings)
30
```

The output in the terminal shows the original and sorted arrays for both integers and strings.

Microatividade 2: Utilização do algoritmo de ordenação “Buble Sort” em Python



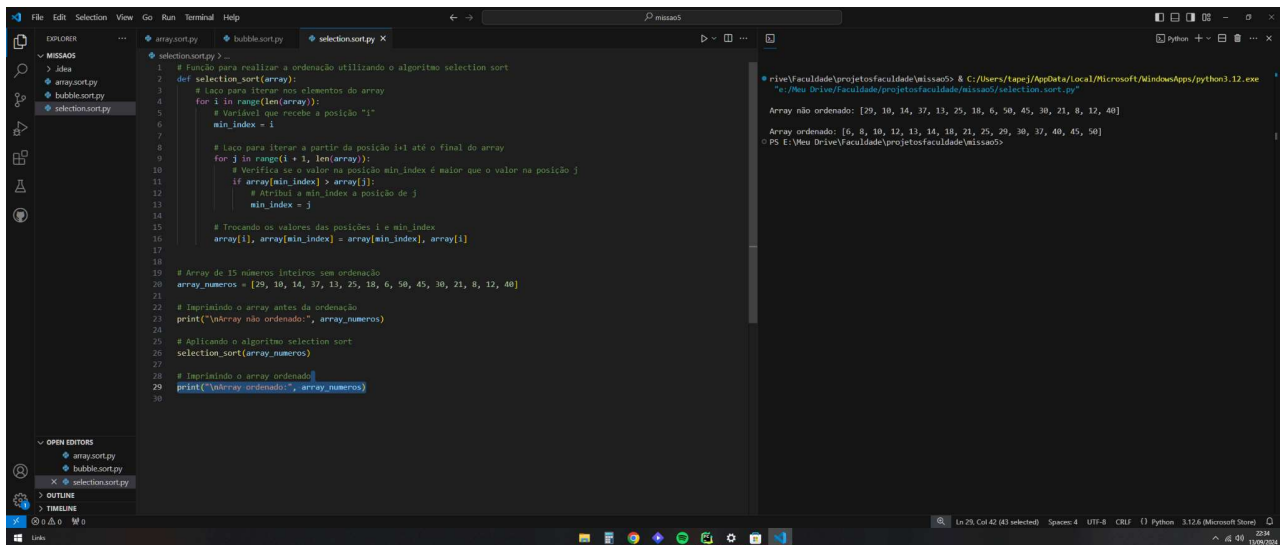
The screenshot shows a VS Code editor with a file named 'bubble.sort.py'. The code is as follows:

```
1 # Método para ordenar o array utilizando o algoritmo Bubble Sort
2 def bubble_sort(array):
3     # Primeiro laço for para iterar pelos elementos do array
4     for i in range(len(array)):
5         # Segundo laço for para comparar elementos adjacentes
6         for j in range(0, len(array) - i - 1):
7             # Se o elemento atual for maior que o próximo
8             if array[j] > array[j + 1]:
9                 # Troca os elementos de posição
10                 temp = array[j]
11                 array[j] = array[j + 1]
12                 array[j + 1] = temp
13
14
15 # Declaração do array de números com 15 posições
16 array_numeros = [64, 34, 25, 12, 22, 11, 90, 55, 45, 78, 23, 89, 37, 6, 99]
17
18 # Imprimindo o array desordenado
19 print("\nArray desordenado:\n", array_numeros)
20
21 # Aplicando o método bubbleSort no array
22 bubble_sort(array_numeros)
23
24 # Imprimindo o array ordenado
25 print("\nArray ordenado:\n", array_numeros)
26
```

The output in the terminal shows the original unsorted array and the sorted array after applying the bubble sort algorithm.

Microatividade 3:

Utilização do algoritmo de ordenação “Selection Sort” em Python

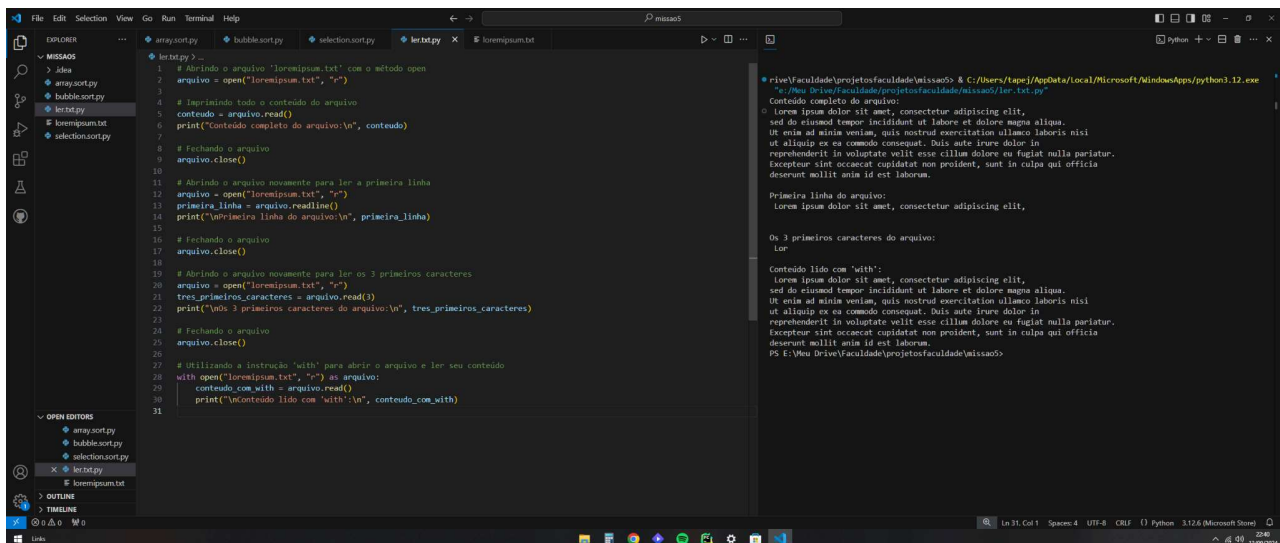


The screenshot shows the Visual Studio Code editor with a Python file named `selectionsort.py` open. The code implements the Selection Sort algorithm. It starts by defining a function `selection_sort(array)` that iterates through the array, finding the minimum element in each pass and swapping it with the element at the current position. The array is initialized with 15 integers: `[29, 10, 14, 37, 13, 25, 18, 6, 50, 45, 30, 21, 8, 12, 40]`. The output shows the array before and after sorting.

```
1 def selection_sort(array):
2     # Função para realizar a ordenação utilizando o algoritmo selection sort
3     # laço para iterar nos elementos do array
4     for i in range(len(array)):
5         # Variável que recebe a posição "i"
6         min_index = i
7
8         # laço para iterar a partir da posição i+1 até o final do array
9         for j in range(i + 1, len(array)):
10            # Verifica se o valor na posição min_index é maior que o valor na posição j
11            if array[min_index] > array[j]:
12                # Attribui a min_index a posição de j
13                min_index = j
14
15            # Trocando os valores das posições i e min_index
16            array[i], array[min_index] = array[min_index], array[i]
17
18 # Array de 15 números inteiros sem ordenação
19 array_numeros = [29, 10, 14, 37, 13, 25, 18, 6, 50, 45, 30, 21, 8, 12, 40]
20
21 # Imprimindo o array antes da ordenação
22 print("\narray não ordenado:", array_numeros)
23
24 # Aplicando o algoritmo selection sort
25 selection_sort(array_numeros)
26
27 # Imprimindo o array ordenado
28 print("\narray ordenado:", array_numeros)
```

Microatividade 4:

Leitura de dados a partir de um arquivo externo em Python

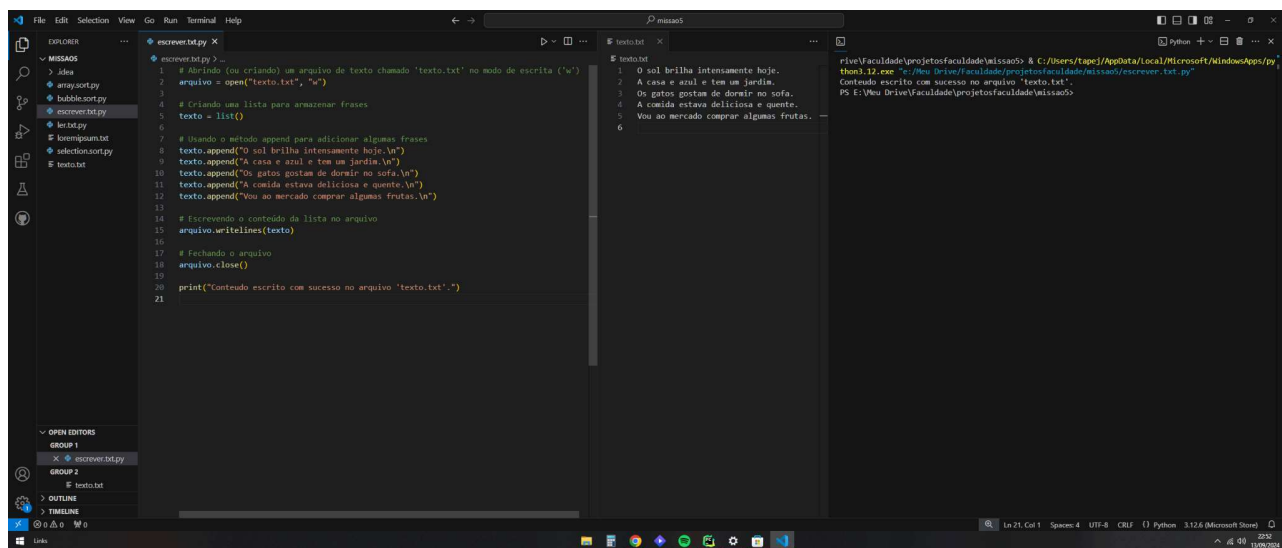


The screenshot shows the Visual Studio Code editor with a Python file named `ler.txt.py` open. The code demonstrates how to read data from a file named `loremipsum.txt`. It shows three methods: reading the entire file content, reading the first line, and reading the first three characters. The output shows the content of the file, the first line, and the first three characters.

```
1 # Abrindo o arquivo 'loremipsum.txt' com o método open
2 arquivo = open("loremipsum.txt", "r")
3
4 # Imprimindo todo o conteúdo do arquivo
5 conteudo = arquivo.read()
6 print("Conteúdo completo do arquivo:\n", conteudo)
7
8 # Fechando o arquivo
9 arquivo.close()
10
11 # Abrindo o arquivo novamente para ler a primeira linha
12 arquivo = open("loremipsum.txt", "r")
13 primeira_linha = arquivo.readline()
14 print("\nprimeira linha do arquivo:\n", primeira_linha)
15
16 # Fechando o arquivo
17 arquivo.close()
18
19 # Abrindo o arquivo novamente para ler os 3 primeiros caracteres
20 arquivo = open("loremipsum.txt", "r")
21 tres_primeiros_caracteres = arquivo.read(3)
22 print("\nOs 3 primeiros caracteres do arquivo:\n", tres_primeiros_caracteres)
23
24 # Fechando o arquivo
25 arquivo.close()
26
27 # Utilizando a instrução 'with' para abrir o arquivo e ler seu conteúdo
28 with open("loremipsum.txt", "r") as arquivo:
29     conteudo_com_with = arquivo.read()
30     print("\nConteúdo lido com 'with':\n", conteudo_com_with)
```

Microatividade 5:

Escrita de dados em um arquivo externo em Python



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays a project named 'MISSAOS' with files: 'ide', 'arraysort.py', 'bubble.sort.py', 'escrever.txt.py', 'leibniz.py', 'loempsum.de', 'selectionsort.py', and 'texto.txt'. The main editor area has two tabs: 'escrever.txt.py' and 'texto.txt'. The 'escrever.txt.py' tab is active, showing a Python script that creates a text file, appends several lines of text, and prints a success message. The 'texto.txt' tab shows the content of the file after execution. A terminal window on the right shows the command to run the script and the resulting output.

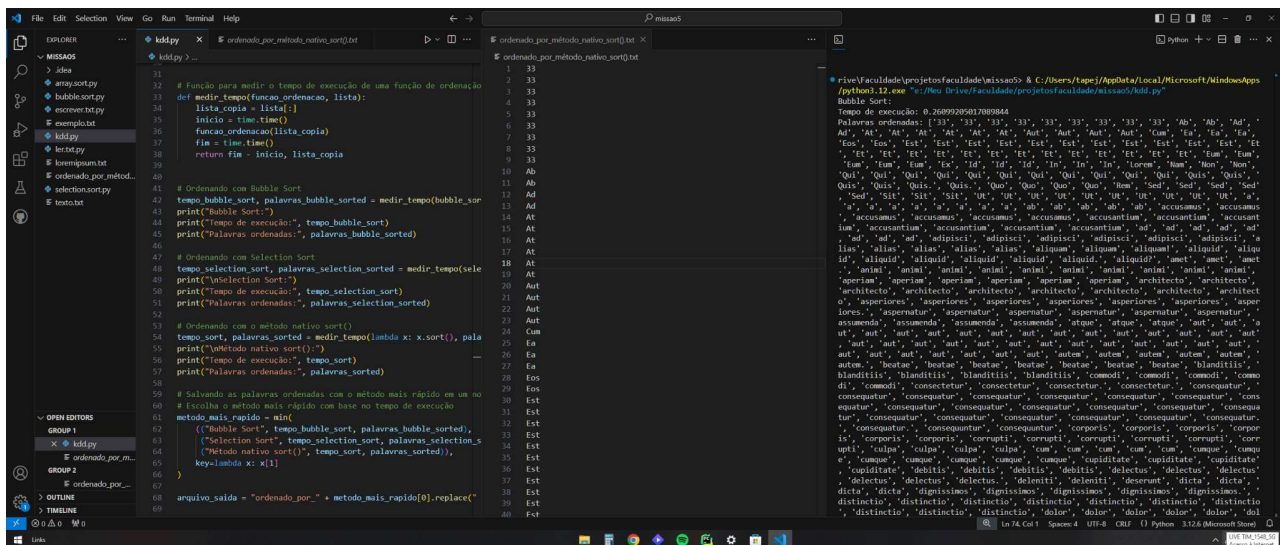
```
1 # Abrindo (ou criando) um arquivo de texto chamado 'texto.txt' no modo de escrita ('w')
2 arquivo = open("texto.txt", "w")
3
4 # Criando uma lista para armazenar frases
5 texto = list()
6
7 # Usando o método append para adicionar algumas frases
8 texto.append("O sol brilha intensamente hoje.\n")
9 texto.append("A casa é azul e tem um jardim.\n")
10 texto.append("Os gatos gostam de dormir no sofá.\n")
11 texto.append("A comida estava deliciosa e quente.\n")
12 texto.append("Vou ao mercado comprar algumas frutas.\n")
13
14 # Escrevendo o conteúdo da lista no arquivo
15 arquivo.writelines(texto)
16
17 # Fechando o arquivo
18 arquivo.close()
19
20 print("Conteúdo escrito com sucesso no arquivo 'texto.txt'.")
21
```

```
1
2 O sol brilha intensamente hoje.
3 A casa é azul e tem um jardim.
4 Os gatos gostam de dormir no sofá.
5 A comida estava deliciosa e quente.
6 Vou ao mercado comprar algumas frutas.
7
```

```
rive\Faculdade\projetos\faculdade\missao5> & C:\Users\tapei\AppData\Local\Microsoft\WindowsApps\python3.12.exe "E:\Meu Drive\Faculdade\projetos\faculdade\missao5\escrever.txt.py"
Conteúdo escrito com sucesso no arquivo 'texto.txt'.
PS E:\Meu Drive\Faculdade\projetos\faculdade\missao5>
```

Missão Prática:

Colocando tudo em ordem e guardando



The image shows a VS Code editor with a Python script named `ordenado_por_metodo_nativo_sort.py` and a terminal window. The script implements a function `medir_tempo(funcao_ordenacao, lista)` to measure the execution time of a sorting function. It compares the 'Bubble Sort' and 'Selection Sort' methods with the native `list.sort()` method. The terminal output shows the results of the sorting process, including the time taken for each method and the sorted list of words.

```
1 33
2 33
3 33
4 33
5 33
6 33
7 33
8 33
9 33
10 Ab
11 Ab
12 Ad
13 Ad
14 At
15 At
16 At
17 At
18 Aut
19 Aut
20 Aut
21 Aut
22 Aut
23 Aut
24 Cam
25 Ea
26 Ea
27 Ea
28 Eos
29 Eos
30 Est
31 Est
32 Est
33 Est
34 Est
35 Est
36 Est
37 Est
38 Est
39 Est
40 Est
41 Est
42 Est
43 Est
44 Est
45 Est
46 Est
47 Est
48 Est
49 Est
50 Est
51 Est
52 Est
53 Est
54 Est
55 Est
56 Est
57 Est
58 Est
59 Est
60 Est
61 Est
62 Est
63 Est
64 Est
65 Est
66 Est
67 Est
68 Est
69 Est
70 Est
71 Est
72 Est
73 Est
74 Est
75 Est
76 Est
77 Est
78 Est
79 Est
80 Est
81 Est
82 Est
83 Est
84 Est
85 Est
86 Est
87 Est
88 Est
89 Est
90 Est
91 Est
92 Est
93 Est
94 Est
95 Est
96 Est
97 Est
98 Est
99 Est
100 Est
```