Code

sort_test.py

```python
import random
import time
import matplotlib.pyplot as plt
# all lists below will be used to plot the algorithm's running time
# store input size for bubble sort
bubble_sort_input_size: list[int] = []
# store elapsed time for bubble sort
bubble_sort_elapsed_time: list[int] = []
# store input size for insertion sort
insertion_sort_input_size: list[int] = []
# store elapsed time for insertion sort
insertion_sort_elapsed_time: list[int] = []

def bubble_sort(a: list[int]) -> None:  1 usage
    r"""Sorts in-place using Bubble Sort from class slides

    Sort by starting at the end of the list and working toward
    the beginning. The outer loop will consist of the elements
    that are already sorted. The inner loop is responsible for
    sorting by swapping elements to the left.

    Parameters
    ----------
    a : list of integers

```

```python
def bubble_sort(a: list[int]) -> None:  1 usage

    Returns
    -------
    None
    """

    start: int = time.perf_counter_ns()
    for i in range(0, len(a)):
        for j in range(len(a) - 1, i, -1):
            if a[j] < a[j - 1]:
                # swap elements if current element
                # is less than element on left
                temp = a[j]
                a[j] = a[j - 1]
                a[j - 1] = temp
    stop: int = time.perf_counter_ns()
    # do not time anything other than sort logic
    elapsed_time: int = stop - start
    # print results
    print(f'Elapsed time for input size {len(a)}: {elapsed_time}')
    # append the data to be used for plot
    bubble_sort_input_size.append(len(a))
    bubble_sort_elapsed_time.append(elapsed_time)
```

sort_test.py

```python


def insertion_sort(a: list[int]) -> None:  1 usage
    r"""Sorts in-place using Insertion Sort from class slides

    Sorts items by comparing the key element with values
    considered to be already sorted. The outer loop will consist
    of the sorted sequence while the inner loop is responsible
    for comparing items with the key (item to be added to sorted
    list)

    Parameters
    ----------
    a : list of integers

    Returns
    -------
    None
    """

    start: int = time.perf_counter_ns()
    for j in range(1, len(a)):
        key = a[j]
        # Insert a[j] into sorted sequence a[0...j-1]
        i = j - 1
        while i >= 0 and a[i] > key:
```

sort_test.py

```python
def insertion_sort(a: list[int]) -> None:  1 usage
        while i >= 0 and a[i] > key:
            a[i + 1] = a[i]
            i = i - 1
        a[i + 1] = key
    stop: int = time.perf_counter_ns()
    # Do not time anything other than sort logic
    elapsed_time = stop - start
    # print results
    print(f'Elapsed time for input size {len(a)}: {elapsed_time}')
    # append the data to be used for plot
    insertion_sort_input_size.append(len(a))
    insertion_sort_elapsed_time.append(elapsed_time)

def plot_time(x: list[int], y: list[int]) -> None:  2 usages
    r"""Plot the timing of an algorithm using Matplotlib

    Parameters
    ----------
    x: list[int]
        This axis represents the input size sent to the algorithm
    y: list[int]
        This axis represents the elapsed time (in nanoseconds)

    Returns
```

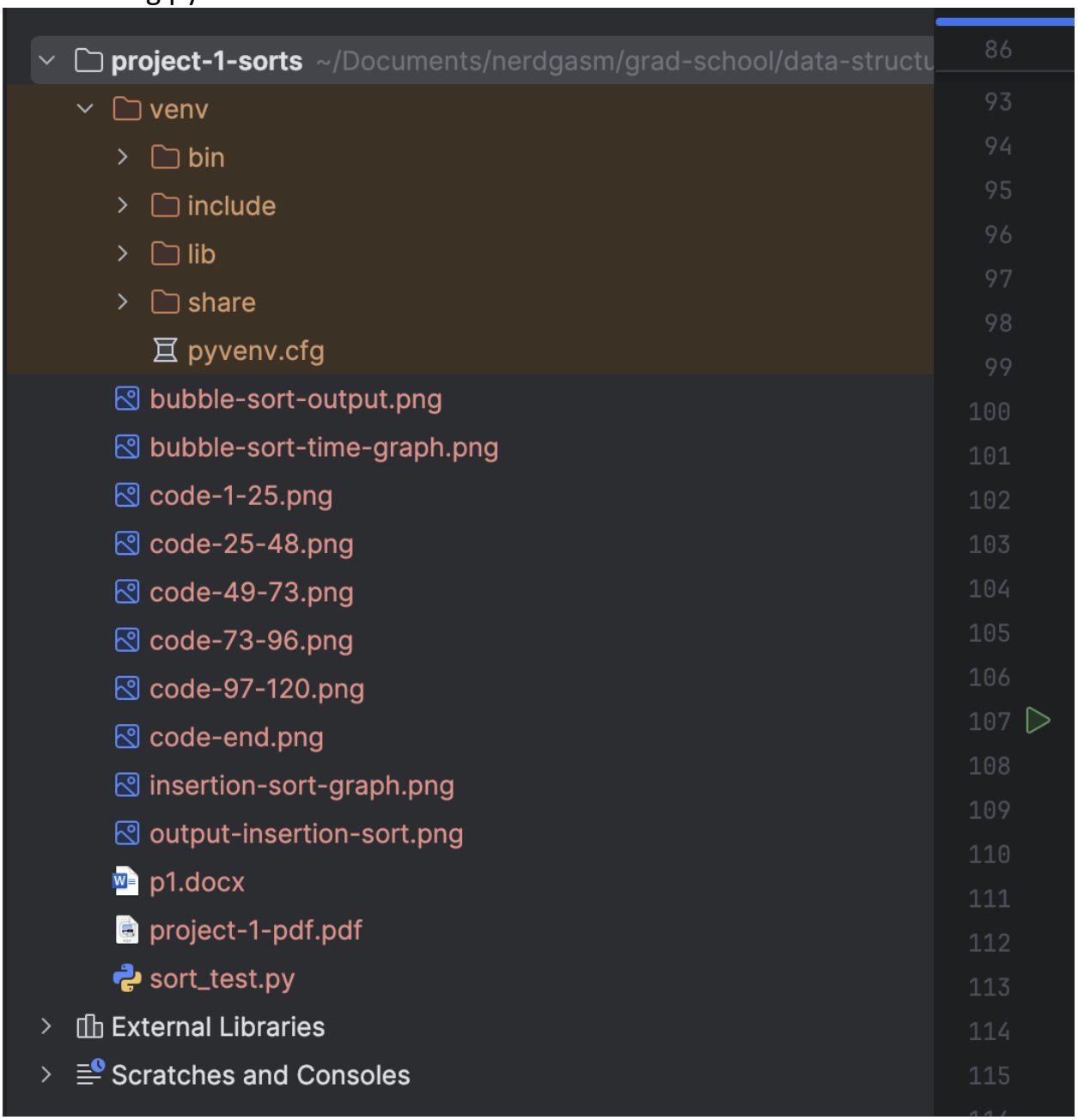
sort_test.py

```python
def plot_time(x: list[int], y: list[int]) -> None:  2 usages
    -------
    None
    """
    plt.plot( *args: x, y)
    plt.xlabel('Input Size')
    plt.ylabel('Time (ns)')
    plt.show()



if __name__ == "__main__":
    print("Beginning Bubble Sort on 10 lists")
    # generate lists for bubble sort
    # note these will be sorted in-place so do not use the same lists for insertion-sort
    for i in range(1, 11):
        # we want to run the sorts on 10k, 20k... 100k elements
        x: int = i * 10000
        input_list: list[int] = [random.randrange( start: 1,  stop: 1000,  step: 1) for i in range(x)]
        bubble_sort(input_list)
    # Once the sort has finished plot its graph (should be quadratic)
    plot_time(bubble_sort_input_size, bubble_sort_elapsed_time)

    print("Beginning Insertion Sort on 10 lists")
    for i in range(1, 11):
```

sort_test.py

```python
if __name__ == "__main__":
    print("Beginning Bubble Sort on 10 lists")
    # generate lists for bubble sort
    # note these will be sorted in-place so do not use the same lists for insertion-sort
    for i in range(1, 11):
        # we want to run the sorts on 10k, 20k... 100k elements
        x: int = i * 10000
        input_list: list[int] = [random.randrange( start: 1,  stop: 1000,  step: 1) for i in range(x)]
        bubble_sort(input_list)
    # Once the sort has finished plot its graph (should be quadratic)
    plot_time(bubble_sort_input_size, bubble_sort_elapsed_time)

    print("Beginning Insertion Sort on 10 lists")
    for i in range(1, 11):
        # we want to run the sorts on 10k, 20k... 100k elements
        x: int = i * 10000
        input_list: list[int] = [random.randrange( start: 1,  stop: 1000,  step: 1) for i in range(x)]
        insertion_sort(input_list)
    plot_time(insertion_sort_input_size, insertion_sort_elapsed_time)
```

IDE showing python file and venv



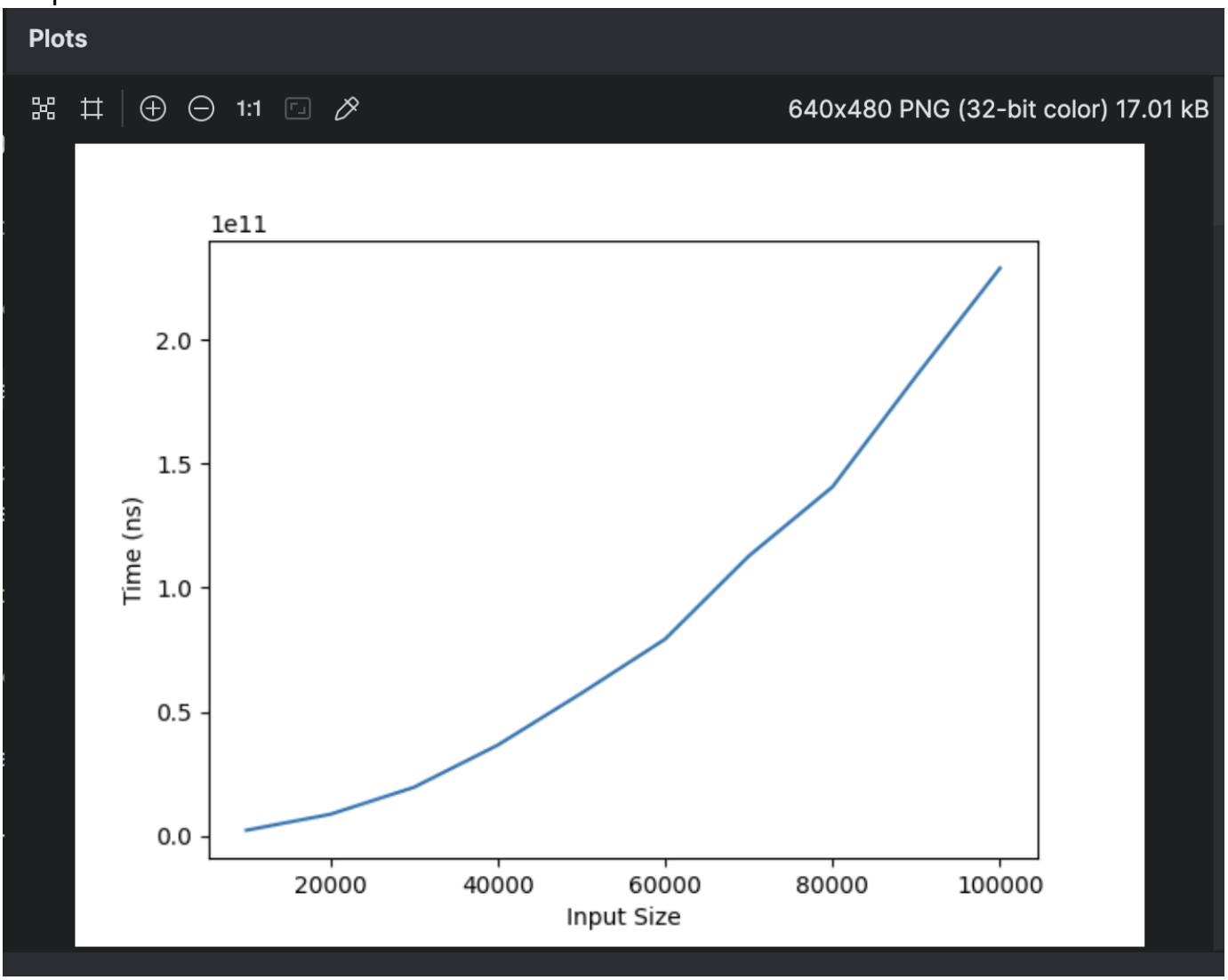
Output of experiment For Bubble Sort

```
Run    sort_test

Beginning Bubble Sort on 10 lists
Elapsed time for input size 10000: 2287958125
Elapsed time for input size 20000: 8713757167
Elapsed time for input size 30000: 19597288750
Elapsed time for input size 40000: 36577694542
Elapsed time for input size 50000: 57497697208
Elapsed time for input size 60000: 79294952667
Elapsed time for input size 70000: 112796306625
Elapsed time for input size 80000: 140671398917
Elapsed time for input size 90000: 185358264333
Elapsed time for input size 100000: 228805673875
```

Output of experiment for Insertion Sort

```
Beginning Insertion Sort on 10 lists
Elapsed time for input size 10000: 812344541
Elapsed time for input size 20000: 3255235292
Elapsed time for input size 30000: 7442371583
Elapsed time for input size 40000: 13504252958
Elapsed time for input size 50000: 20386126792
Elapsed time for input size 60000: 30666097125
Elapsed time for input size 70000: 40006711500
Elapsed time for input size 80000: 52390396250
Elapsed time for input size 90000: 66243761625
Elapsed time for input size 100000: 83835616167
```

Graph for Bubble Sort



Graph for Insertion Sort