

1.

```
def perform_on_list lst
    return sorted lst
```

```
print perform_on_list
print perform_on_list 1
print perform_on_list 7 7 7 7
print perform_on_list -5 -1 -3 -2 -4
```

The screenshot shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'DAA\_ASSIGNMENTS' with a subdirectory 'day3' containing files '1.py' through '8.py'. The code editor shows the content of '1.py', which is a Python script defining a function 'perform\_on\_list' that returns a sorted list. The script also includes several print statements to test the function. The terminal at the bottom shows the execution of the script, displaying the output of the print statements: an empty list, a list with one element, a list with four elements, and a list with five elements.

```
def perform_on_list(lst):
    return sorted(lst)

print(perform_on_list([]))
print(perform_on_list([1]))
print(perform_on_list([7, 7, 7, 7]))
print(perform_on_list([-5, -1, -3, -2, -4]))
```

```
/usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/1.py
nandakishore@Otonautss-MacBook-Air daa_assignments % /usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/1.py
[]
[1]
[7, 7, 7, 7]
[-5, -4, -3, -2, -1]
nandakishore@Otonautss-MacBook-Air daa_assignments %
```

2.

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

```
print selection_sort 5 2 9 1 5 6
print selection_sort 10 8 6 4 2
print selection_sort 1 2 3 4 5
```

```
out.py 1.py 2.py x 3.py 4.py 5.py 7.py 6.py 8.py
day3 > 2.py > ...
1 def selection_sort(arr):
2     for i in range(len(arr)):
3         min_idx = i
4         for j in range(i + 1, len(arr)):
5             if arr[min_idx] > arr[j]:
6                 min_idx = j
7         arr[i], arr[min_idx] = arr[min_idx], arr[i]
8     return arr
9
10 print(selection_sort([5, 2, 9, 1, 5, 6]))
11 print(selection_sort([10, 8, 6, 4, 2]))
12 print(selection_sort([1, 2, 3, 4, 5]))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - []

/usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/1.py
nandakishore@Otonautss-MacBook-Air daa_assignments % /usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/1.py
[1]
[7, 7, 7, 7]
[-5, -4, -3, -2, -1]
nandakishore@Otonautss-MacBook-Air daa_assignments % /usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/2.py
[1, 2, 5, 5, 6, 9]
[2, 4, 6, 8, 10]
[1, 2, 3, 4, 5]
nandakishore@Otonautss-MacBook-Air daa_assignments %
```

3.

```
def bubble_sort arr
    n = len(arr)
    for i in range n
        swapped = False
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            break
    return arr
```

```
print bubble_sort 64 25 12 22 11
print bubble_sort 29 10 14 37 13
print bubble_sort 3 5 2 1 4
print bubble_sort 1 2 3 4 5
print bubble_sort 5 4 3 2 1
```

```
daa_assignments
out.py 1.py 2.py 3.py x 4.py 5.py 7.py 6.py 8.py
day3 > 3.py > ...
1 def bubble_sort(arr):
2     n = len(arr)
3     for i in range(n):
4         swapped = False
5         for j in range(0, n - i - 1):
6             if arr[j] > arr[j + 1]:
7                 arr[j], arr[j + 1] = arr[j + 1], arr[j]
8                 swapped = True
9         if not swapped:
10             break
11     return arr
12
13 print(bubble_sort([64, 25, 12, 22, 11]))
14 print(bubble_sort([29, 10, 14, 37, 13]))
15 print(bubble_sort([3, 5, 2, 1, 4]))
16 print(bubble_sort([1, 2, 3, 4, 5]))
17 print(bubble_sort([5, 4, 3, 2, 1]))

EXPLORER
DAA_ASSIGNMENTS
day3
1.py
2.py
3.py
4.py
5.py
6.py
7.py
8.py
binary_search.py
bubble_sort.py
champagne_tower.py
climb_stairs.py
count_indices.py
count_pairs.py
find_ways_to_move_out.py
first_palindromic_string.py
game_of_life.py
large_groups.py
max_element.py
merge_sort.py
rob_houses.py
sort_and_find_max.py
sum_of_squares_of_distinct_coun...
two_sums.py
unique_elements.py
unique_paths.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [] ... ^ x

nandakishore@Otonautss-MacBook-Air daa_assignments %
```

4.

```
def insertion_sort arr
    for i in range 1 len arr
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

print insertion_sort 3 1 4 1 5 9 2 6 5 3
print insertion_sort 5 5 5 5 5
print insertion_sort 2 3 1 3 2 1 1 3
```



```
1 def insertion_sort(arr):
2     for i in range(1, len(arr)):
3         key = arr[i]
4         j = i - 1
5         while j >= 0 and key < arr[j]:
6             arr[j + 1] = arr[j]
7             j -= 1
8         arr[j + 1] = key
9     return arr
10
11 print(insertion_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3]))
12 print(insertion_sort([5, 5, 5, 5, 5]))
13 print(insertion_sort([2, 3, 1, 3, 2, 1, 1, 3]))
```

5.

```
def find_kth_missing_positive arr k
    missing =
    current = 1
    i = 0
    while len missing < k
        if i < len(arr) and arr[i] == current:
            i += 1
        else:
            missing.append(current)
```

```

current += 1
return missing - 1

```

```

print find_kth_missing_positive 2 3 4 7 11 5
print find_kth_missing_positive 1 2 3 4 2

```

```

day3 > 4.py > ...
1 def insertion_sort(arr):
2     for i in range(1, len(arr)):
3         key = arr[i]
4         j = i - 1
5         while j >= 0 and key < arr[j]:
6             arr[j + 1] = arr[j]
7             j -= 1
8         arr[j + 1] = key
9     return arr
10
11 print(insertion_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3]))
12 print(insertion_sort([5, 5, 5, 5, 5]))
13 print(insertion_sort([2, 3, 1, 3, 2, 1, 1, 3]))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

nandakishore@Otonautss-MacBook-Air daa\_assignments %

```

6.
def find_peak_element nums
    left right = 0 len nums - 1
    while left < right
        mid = (left + right) // 2
        if nums[mid] > nums[mid + 1]:
            right = mid
        else:
            left = mid + 1
    return left

```

```

print find_peak_element 1 2 3 1
print find_peak_element 1 2 1 3 5 6 4

```



7.

```
def str_str haystack needle
    return haystack     needle
```

```
print str_str "sadbutsad" "sad"
print str_str "leetcode" "leeto"
```



8.

```
def find_substrings words
    result =
    for i word in enumerate words
        for j other in enumerate words
            if i != j and word in other:
                result.append(word)
                break
    return result
```

```
print find_substrings "mass" "as" "hero" "superhero"
print find_substrings "leetcode" "et" "code"
print find_substrings "blue" "green" "bu"
```

The screenshot shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a directory named 'DAA\_ASSIGNMENTS' containing a subdirectory 'day3' with files '1.py' through '8.py'. The code editor shows the content of '8.py', which contains the following Python code:

```
1 def find_substrings(words):
2     result = []
3     for i, word in enumerate(words):
4         for j, other in enumerate(words):
5             if i != j and word in other:
6                 result.append(word)
7                 break
8     return result
9
10 print(find_substrings(["mass", "as", "hero", "superhero"]))
11 print(find_substrings(["leetcode", "et", "code"]))
12 print(find_substrings(["blue", "green", "bu"]))
```

The terminal at the bottom shows the output of the script:

```
nandakishore@Otonautss-MacBook-Air daa_assignments % /usr/local/bin/python3 /Users/nandakishore/Documents/daa_assignments/day3/8.py
['as', 'hero']
['et', 'code']
[]
```

The status bar at the bottom indicates the current line and column: 'Ln 12, Col 48'. The file encoding is 'UTF-8' and the line ending is 'LF'. The Python version is '3.12.4 64-bit'.