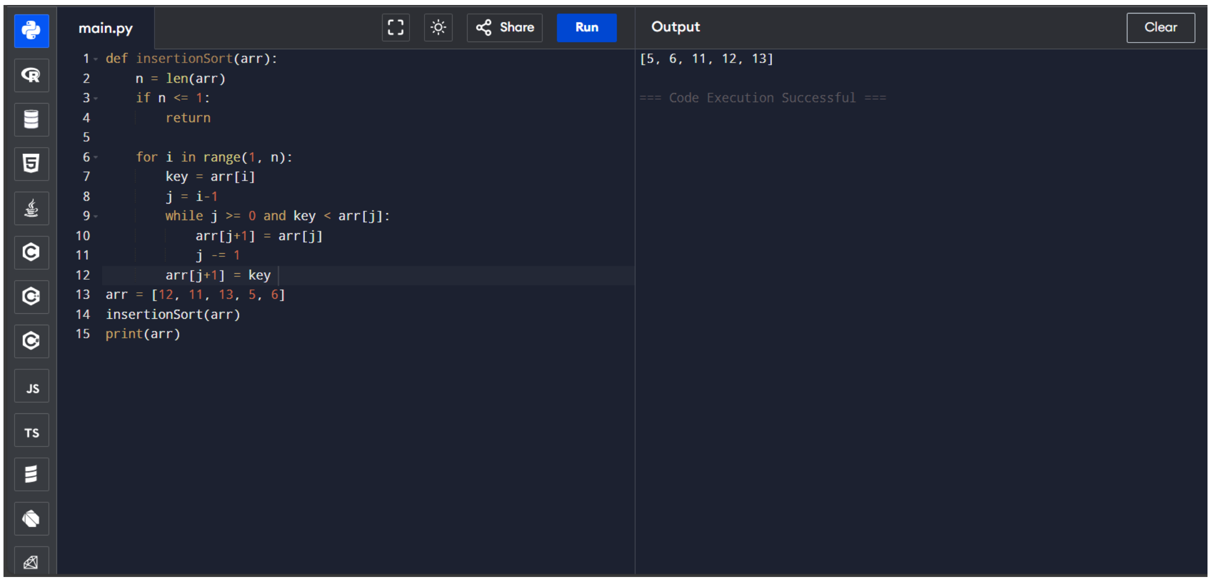
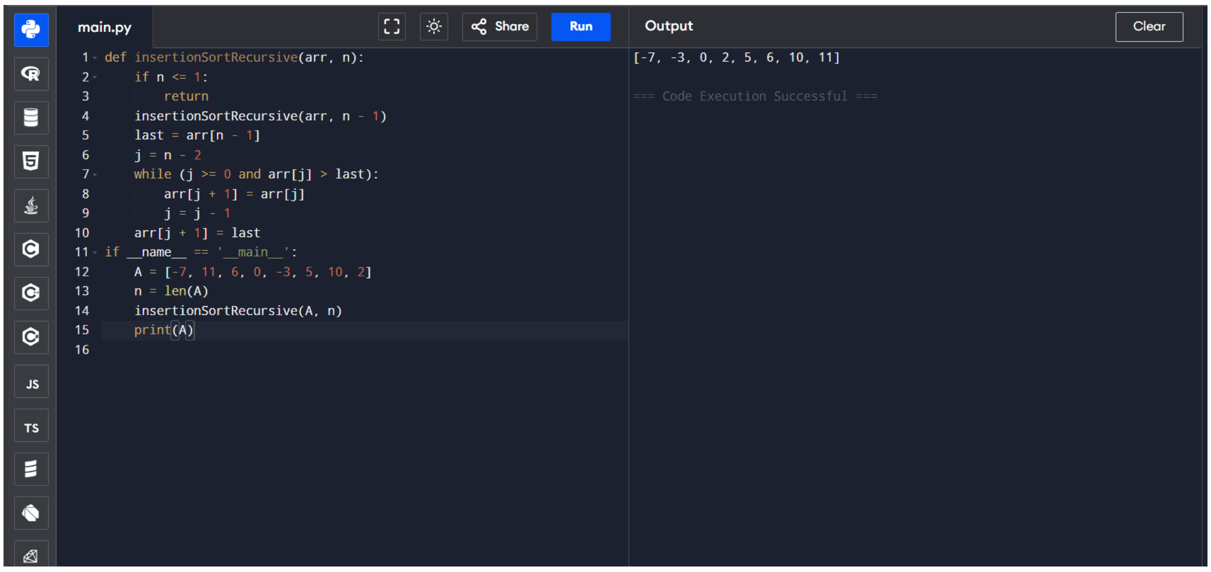
# DAY 9 PRACTICE PROGRAMS



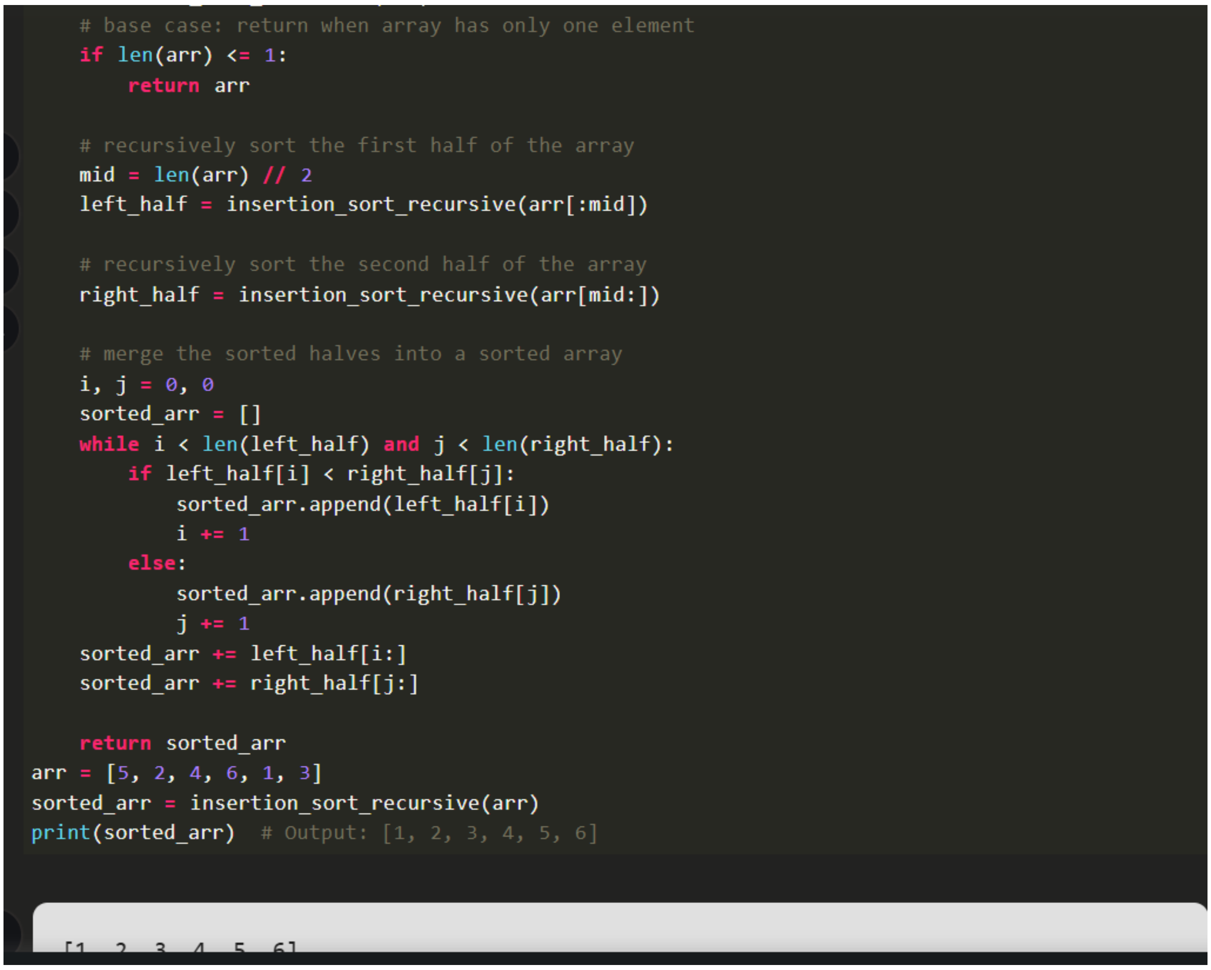
```
def insertionSort(arr):
    n = len(arr)
    if n <= 1:
        return

    for i in range(1, n):
        key = arr[i]
        j = i-1
        while j >= 0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print(arr)
```

```
[5, 6, 11, 12, 13]

=== Code Execution Successful ===
```



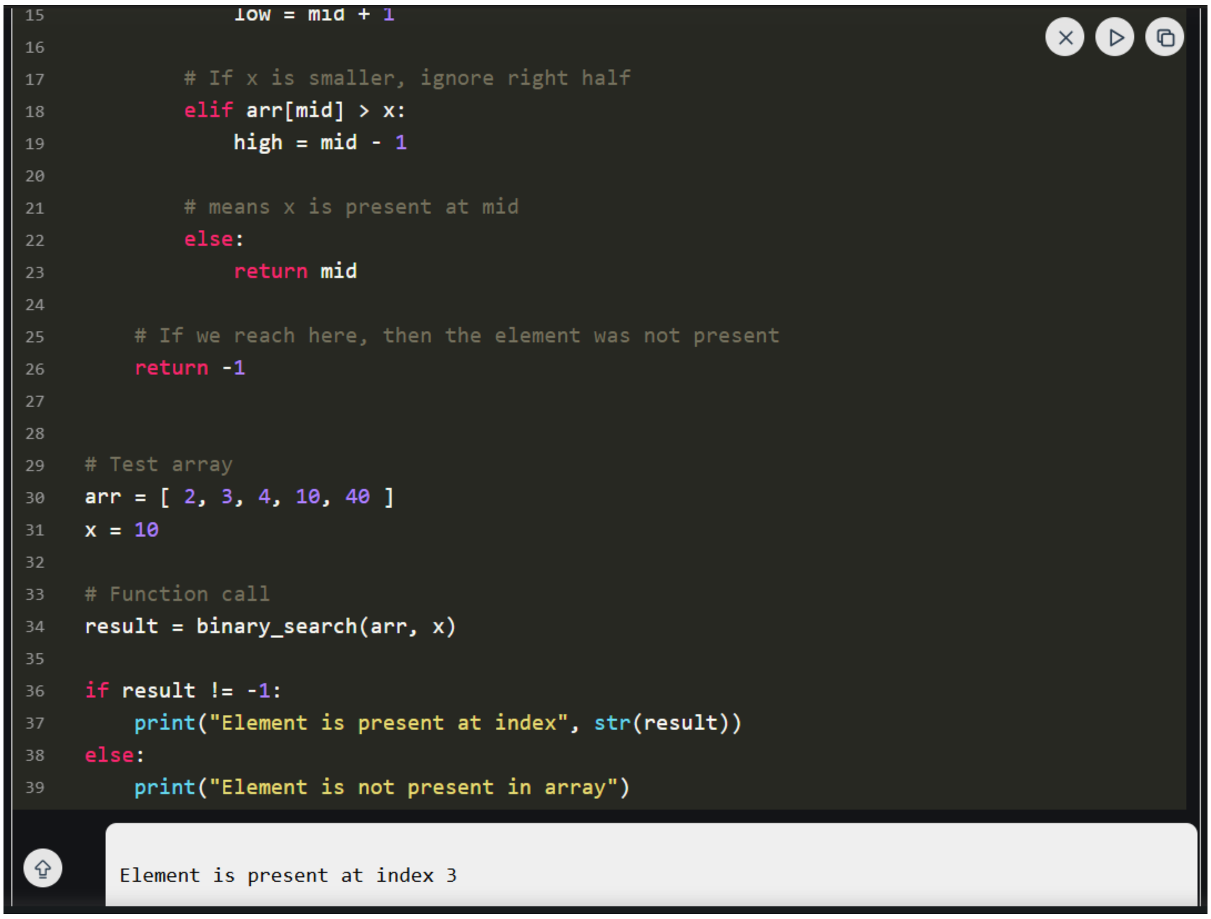
```
def insertionSortRecursive(arr, n):
    if n <= 1:
        return
    insertionSortRecursive(arr, n - 1)
    last = arr[n - 1]
    j = n - 2
    while (j >= 0 and arr[j] > last):
        arr[j + 1] = arr[j]
        j = j - 1
    arr[j + 1] = last
if __name__ == '__main__':
    A = [-7, 11, 6, 0, -3, 5, 10, 2]
    n = len(A)
    insertionSortRecursive(A, n)
    print(A)
```

```
[-7, -3, 0, 2, 5, 6, 10, 11]

=== Code Execution Successful ===
```



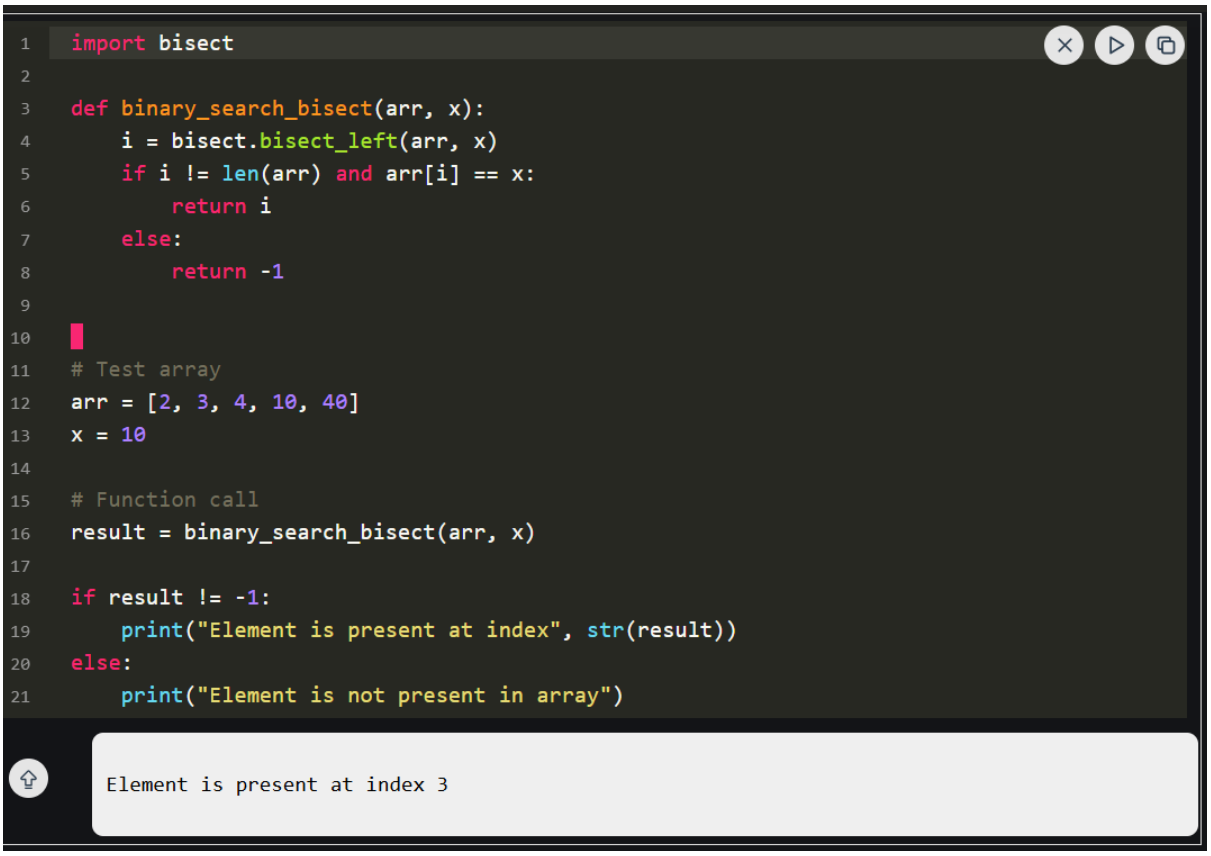
```python
    # base case: return when array has only one element
    if len(arr) <= 1:
        return arr

    # recursively sort the first half of the array
    mid = len(arr) // 2
    left_half = insertion_sort_recursive(arr[:mid])

    # recursively sort the second half of the array
    right_half = insertion_sort_recursive(arr[mid:])

    # merge the sorted halves into a sorted array
    i, j = 0, 0
    sorted_arr = []
    while i < len(left_half) and j < len(right_half):
        if left_half[i] < right_half[j]:
            sorted_arr.append(left_half[i])
            i += 1
        else:
            sorted_arr.append(right_half[j])
            j += 1
    sorted_arr += left_half[i:]
    sorted_arr += right_half[j:]

    return sorted_arr
arr = [5, 2, 4, 6, 1, 3]
sorted_arr = insertion_sort_recursive(arr)
print(sorted_arr)  # Output: [1, 2, 3, 4, 5, 6]
```

```
[1  2  3  4  5  6]
```



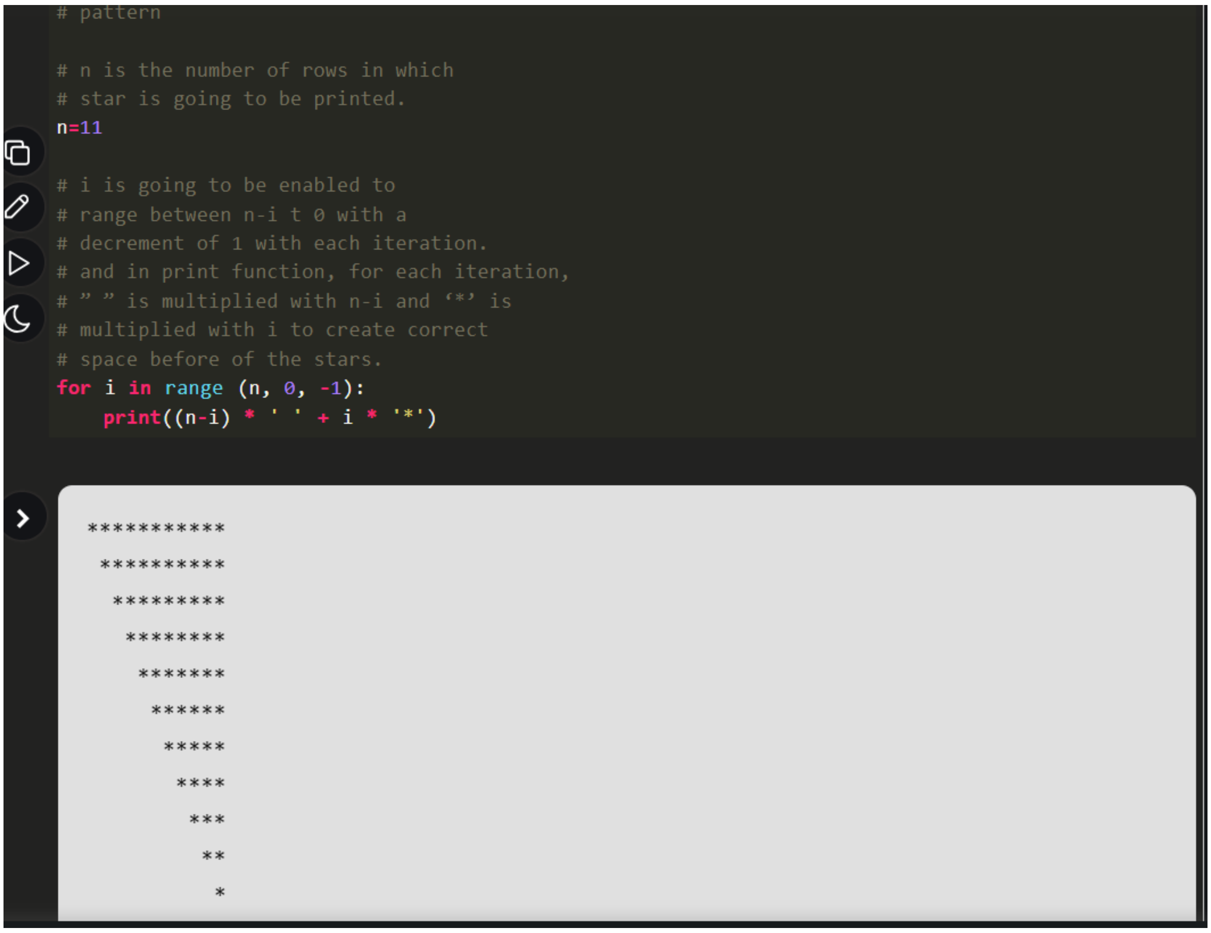
```python
            low = mid + 1

        # If x is smaller, ignore right half
        elif arr[mid] > x:
            high = mid - 1

        # means x is present at mid
        else:
            return mid

    # If we reach here, then the element was not present
    return -1


# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = 10

# Function call
result = binary_search(arr, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

```
Element is present at index 3
```



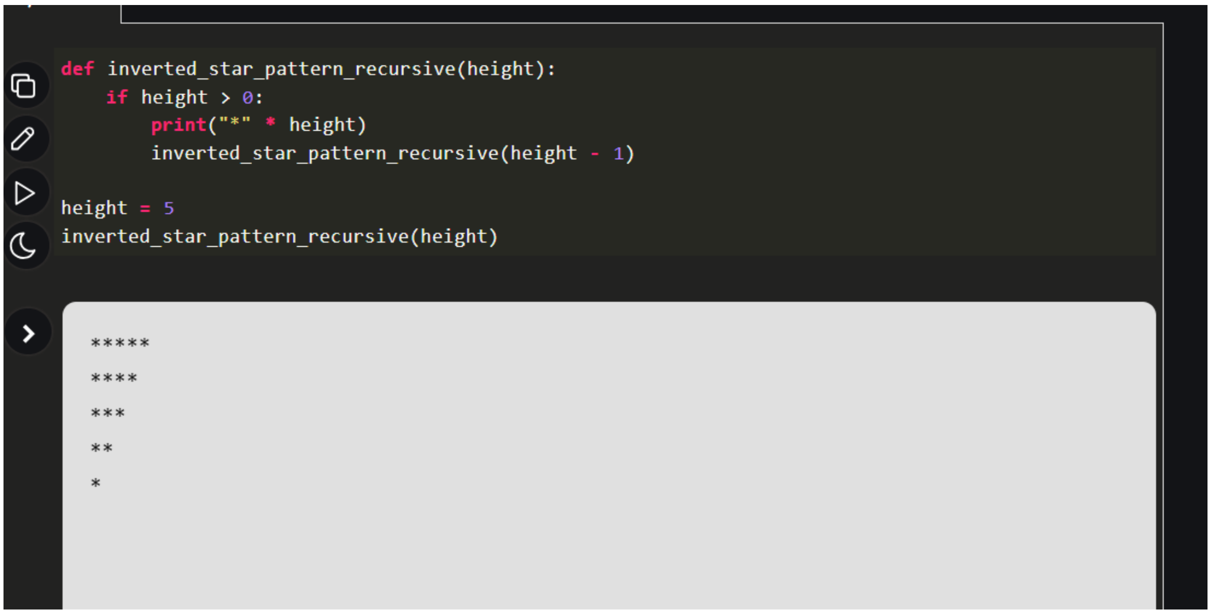
```python
import bisect

def binary_search_bisect(arr, x):
    i = bisect.bisect_left(arr, x)
    if i != len(arr) and arr[i] == x:
        return i
    else:
        return -1


# Test array
arr = [2, 3, 4, 10, 40]
x = 10

# Function call
result = binary_search_bisect(arr, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

```
Element is present at index 3
```



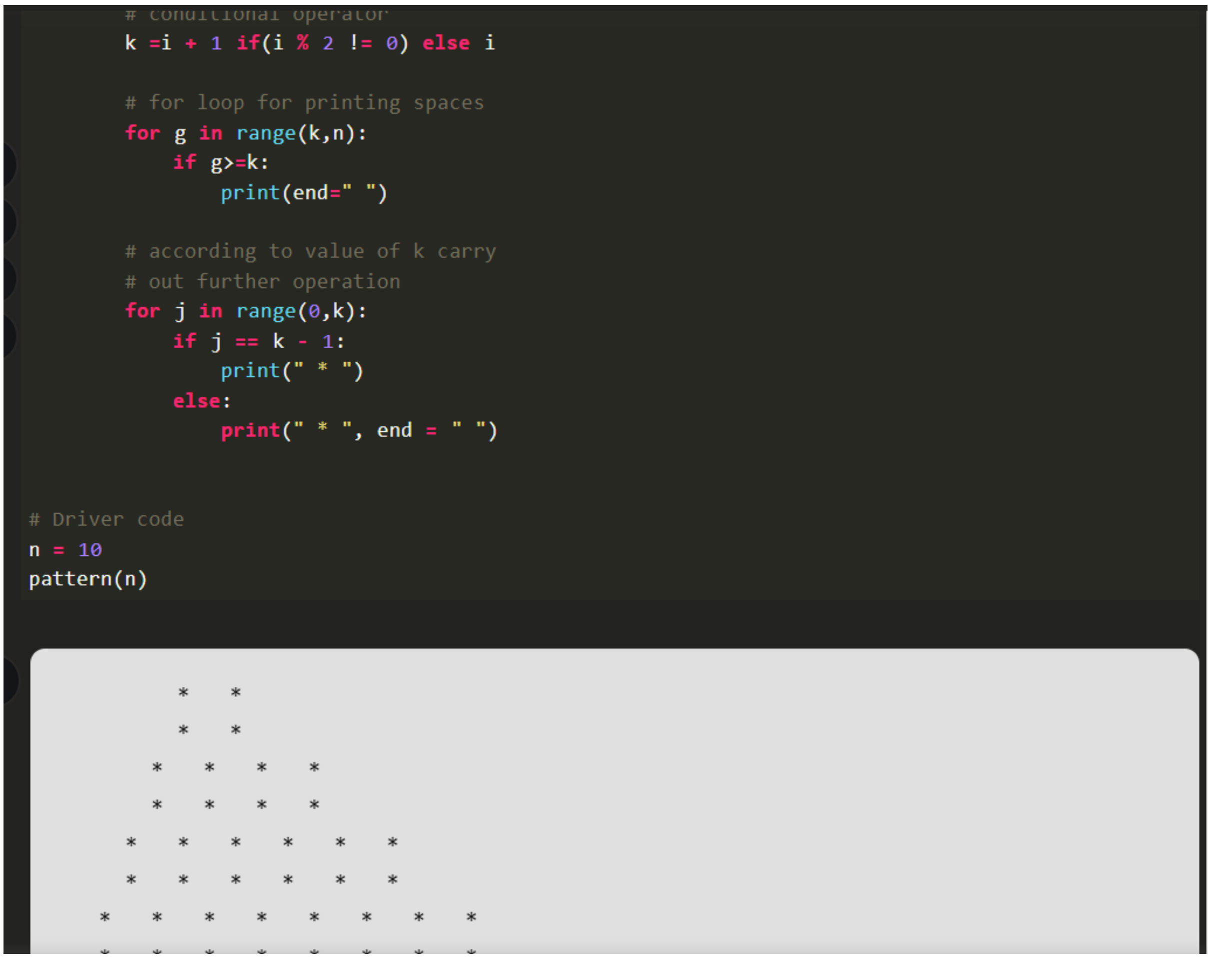
```
# pattern

# n is the number of rows in which
# star is going to be printed.
n=11

# i is going to be enabled to
# range between n-i t 0 with a
# decrement of 1 with each iteration.
# and in print function, for each iteration,
# ” ” is multiplied with n-i and ‘*’ is
# multiplied with i to create correct
# space before of the stars.
for i in range (n, 0, -1):
    print((n-i) * ' ' + i * '*')
```

```
***********
 **********
  *********
   ********
    *******
     ******
      *****
       ****
        ***
         **
          *
```



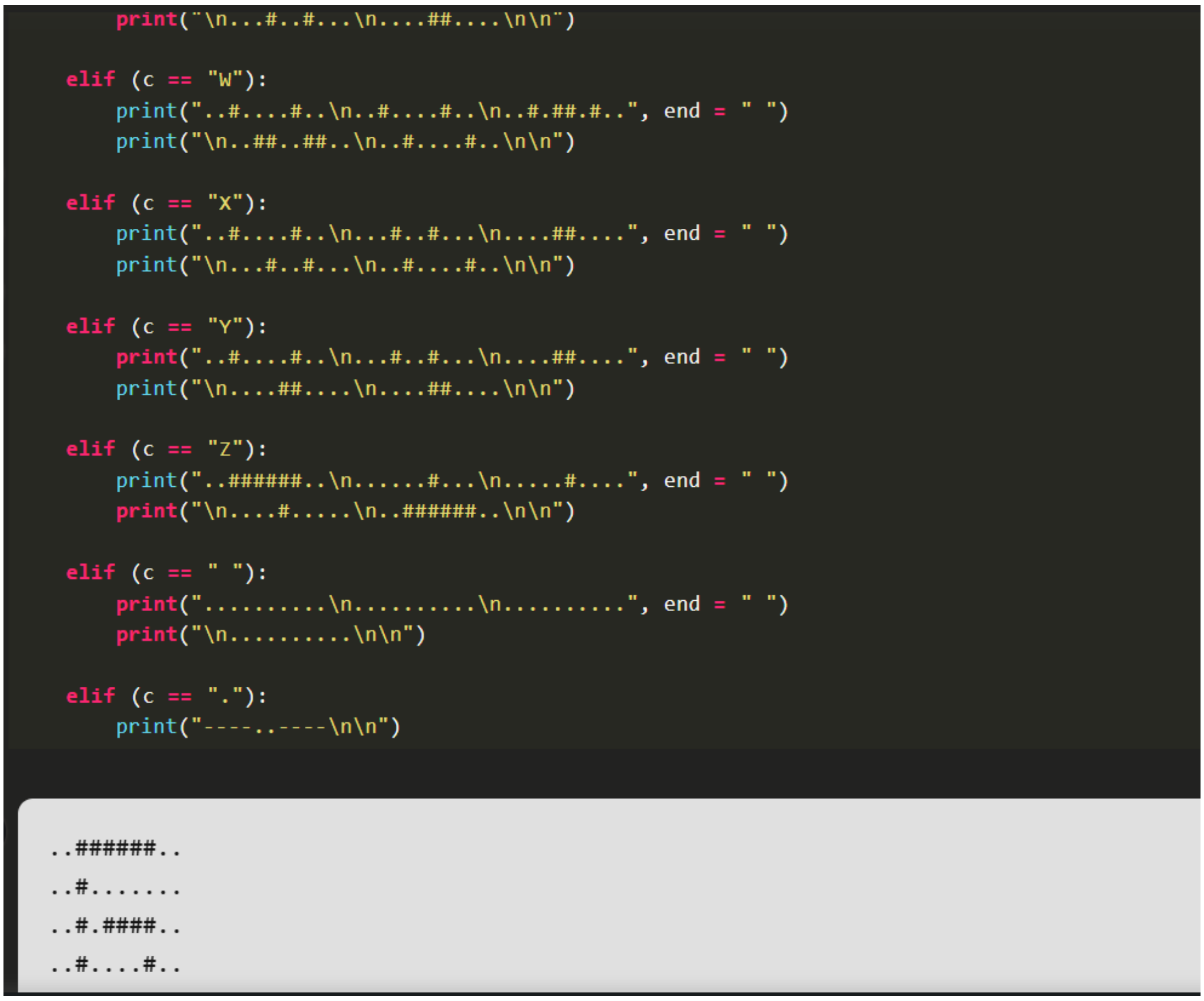
```
def inverted_star_pattern_recursive(height):
    if height > 0:
        print("*" * height)
        inverted_star_pattern_recursive(height - 1)

height = 5
inverted_star_pattern_recursive(height)
```

```
*****
****
***
**
*
```



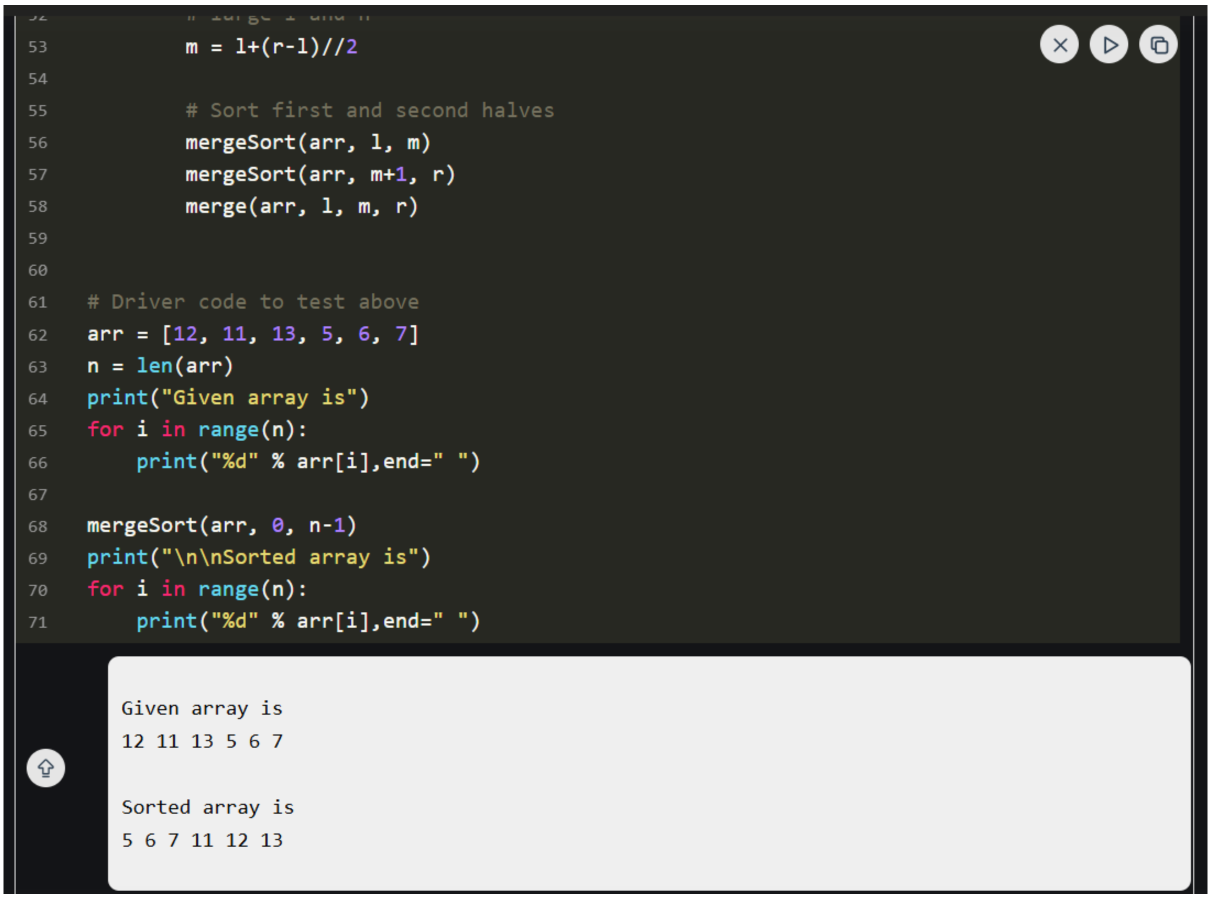
```python
        # conditional operator
        k =i + 1 if(i % 2 != 0) else i

        # for loop for printing spaces
        for g in range(k,n):
            if g>=k:
                print(end=" ")

        # according to value of k carry
        # out further operation
        for j in range(0,k):
            if j == k - 1:
                print(" * ")
            else:
                print(" * ", end = " ")


# Driver code
n = 10
pattern(n)
```

```
         *   *
         *   *
       *   *   *   *
       *   *   *   *
     *   *   *   *   *   *
     *   *   *   *   *   *
   *   *   *   *   *   *   *   *
   *   *   *   *   *   *   *   *
```



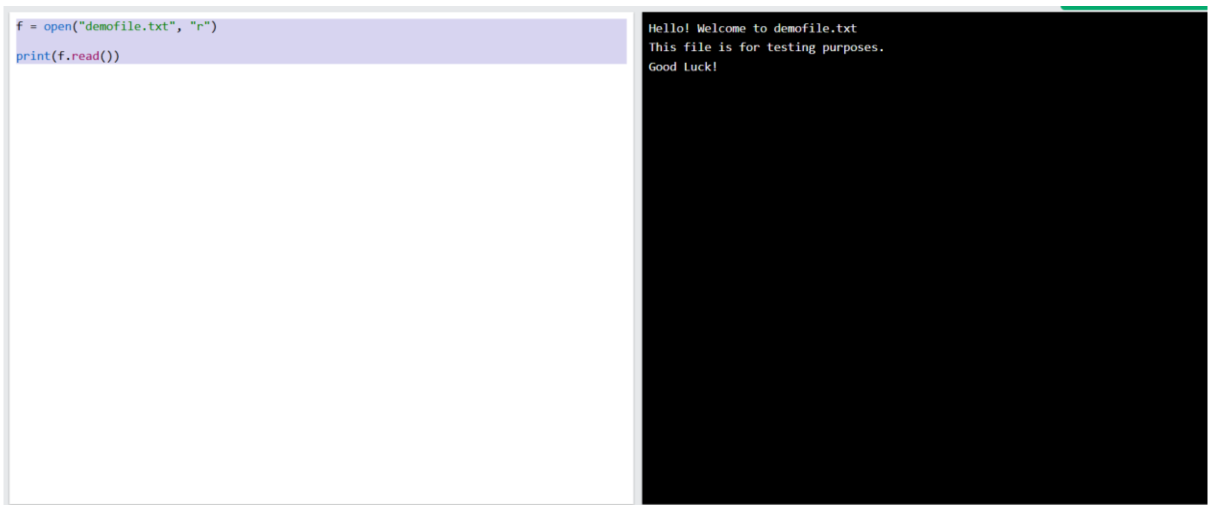
```python
        print("\n...#..#...\n....##....\n\n")

    elif (c == "W"):
        print("..#....#..\n..#....#..\n..#.##.#..", end = " ")
        print("\n..##..##..\n..#....#..\n\n")

    elif (c == "X"):
        print("..#....#..\n...#..#...\n....##....", end = " ")
        print("\n...#..#...\n..#....#..\n\n")

    elif (c == "Y"):
        print("..#....#..\n...#..#...\n....##....", end = " ")
        print("\n....##....\n....##....\n\n")

    elif (c == "Z"):
        print("..######..\n......#...\n.....#....", end = " ")
        print("\n....#.....\n..######..\n\n")

    elif (c == " "):
        print("..........\n..........\n..........", end = " ")
        print("\n..........\n\n")

    elif (c == "."):
        print("----..----\n\n")
```

```
..######..
..#.......
..#.####..
..#....#..
```



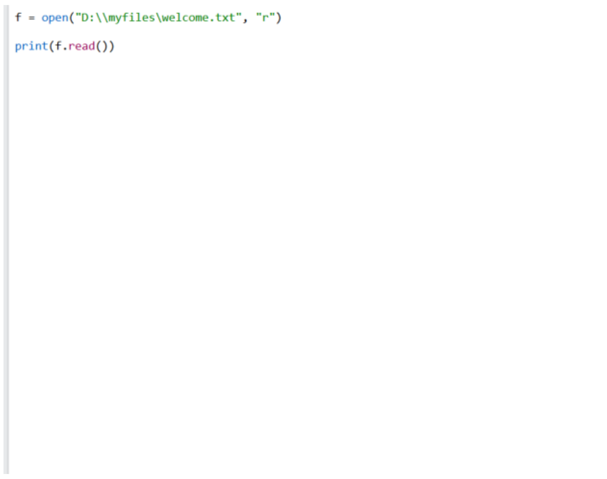
```python
        m = l+(r-l)//2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)


# Driver code to test above
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i],end=" ")

mergeSort(arr, 0, n-1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i],end=" ")
```

```
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

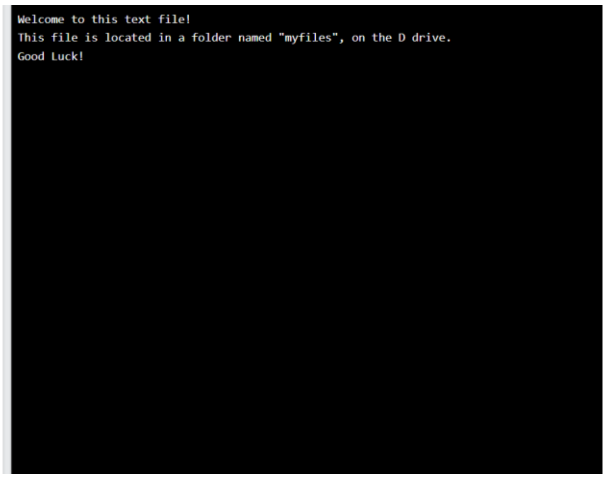


```python
f = open("demofile.txt", "r")

print(f.read())
```

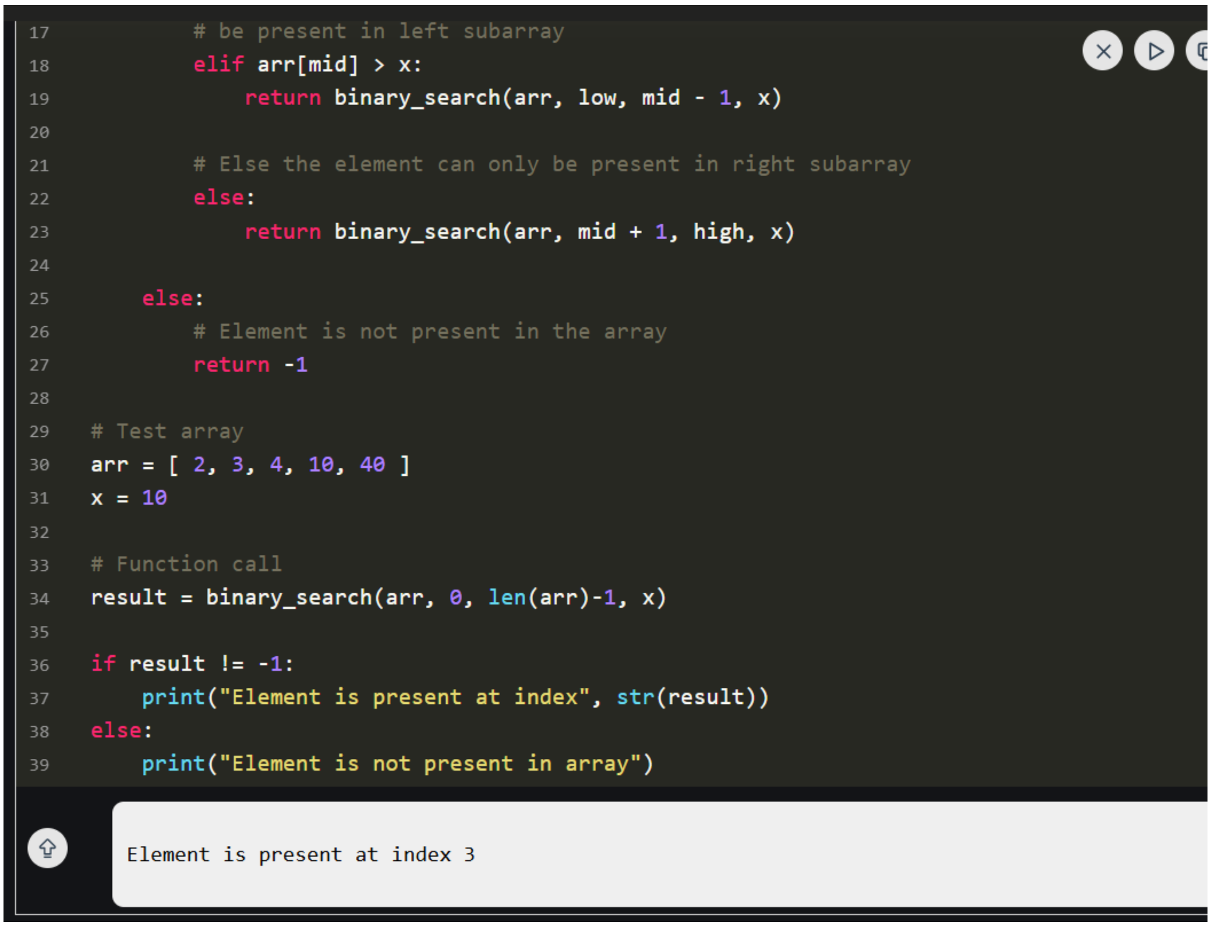
```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```



```python
f = open("D:\\myfiles\welcome.txt", "r")

print(f.read())
```

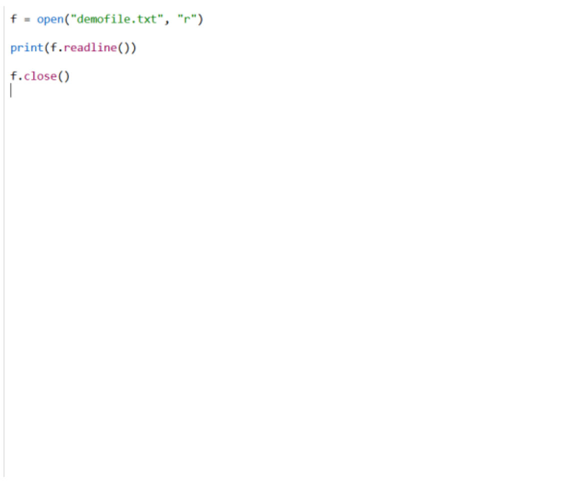


```
Welcome to this text file!
This file is located in a folder named "myfiles", on the D drive.
Good Luck!
```

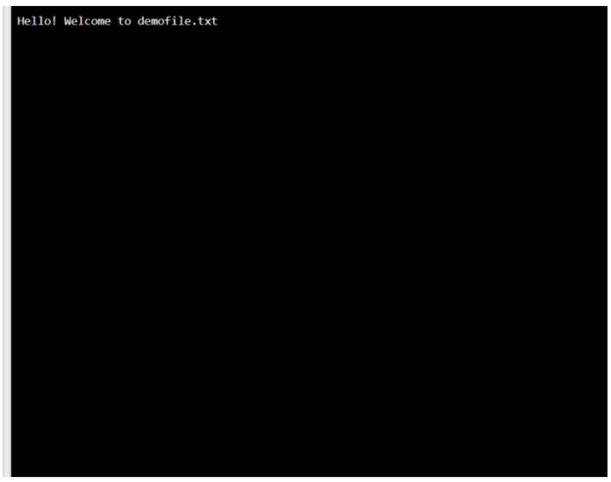


```python
        # be present in left subarray
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)

        # Else the element can only be present in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)

    else:
        # Element is not present in the array
        return -1

# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = 10

# Function call
result = binary_search(arr, 0, len(arr)-1, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

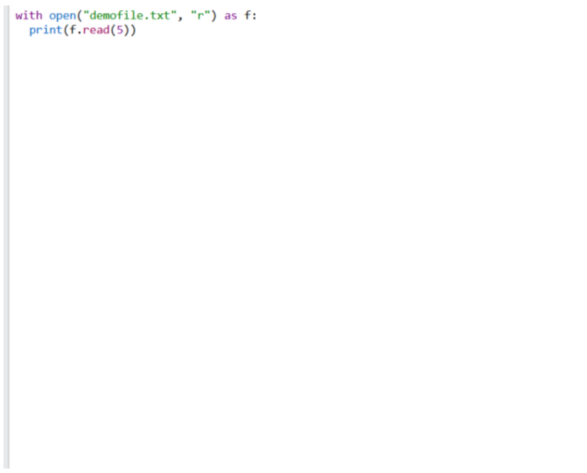
```
Element is present at index 3
```



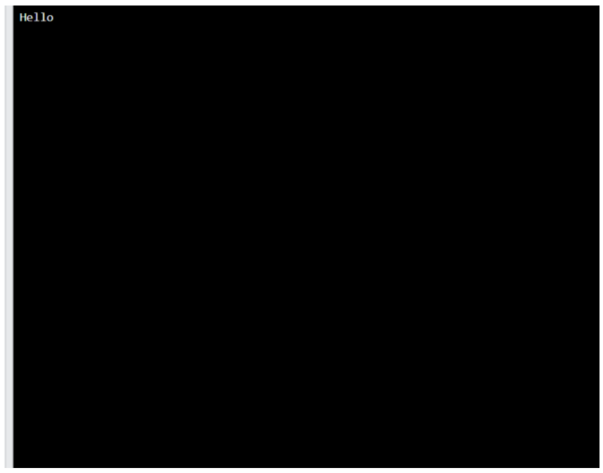
```
f = open("demofile.txt", "r")

print(f.readline())

f.close()
```



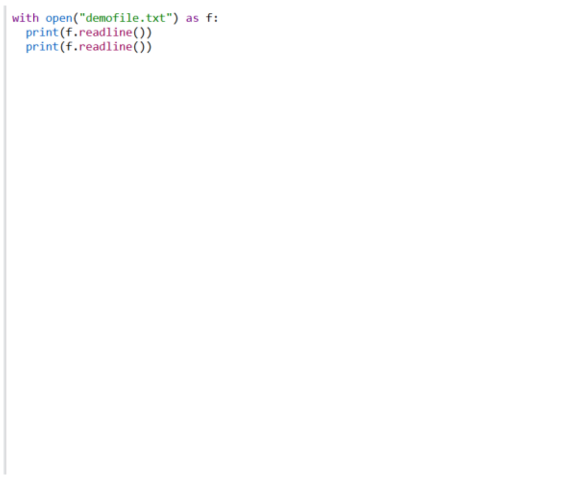
```
Hello! Welcome to demofile.txt
```



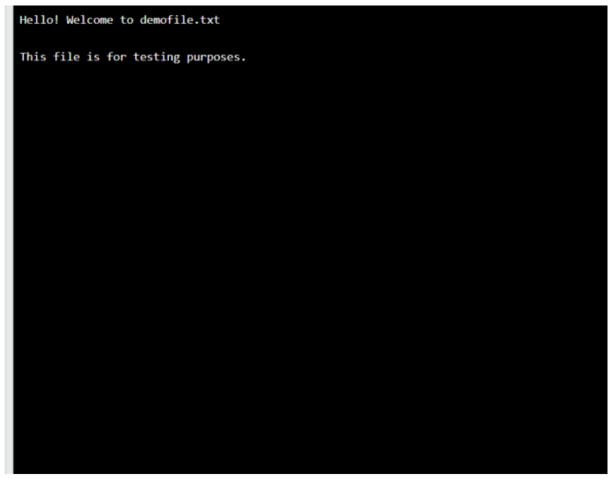
```
with open("demofile.txt", "r") as f:
  print(f.read(5))
```



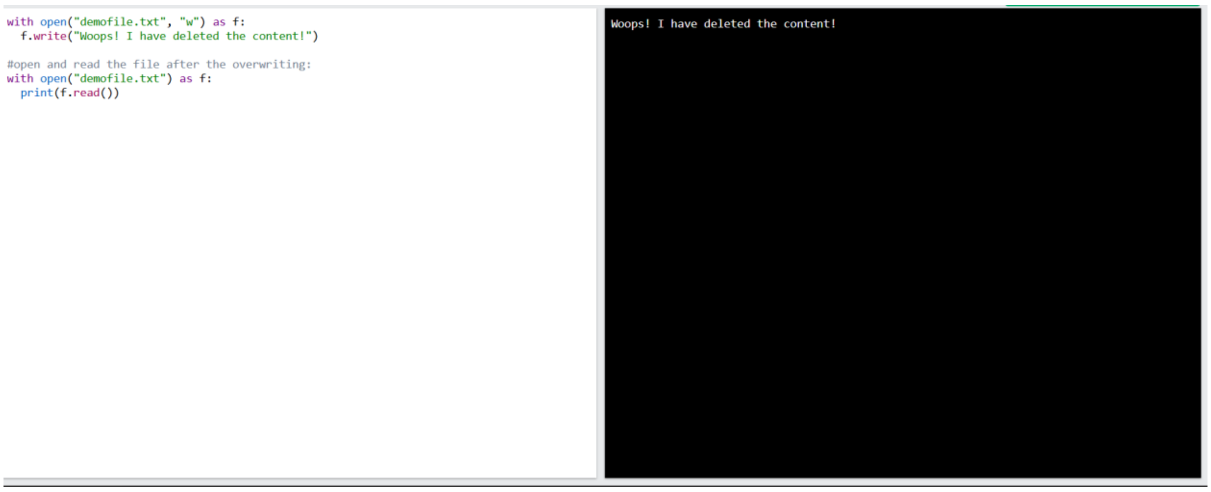
```
Hello
```



```
with open("demofile.txt") as f:
  print(f.readline())
  print(f.readline())
```

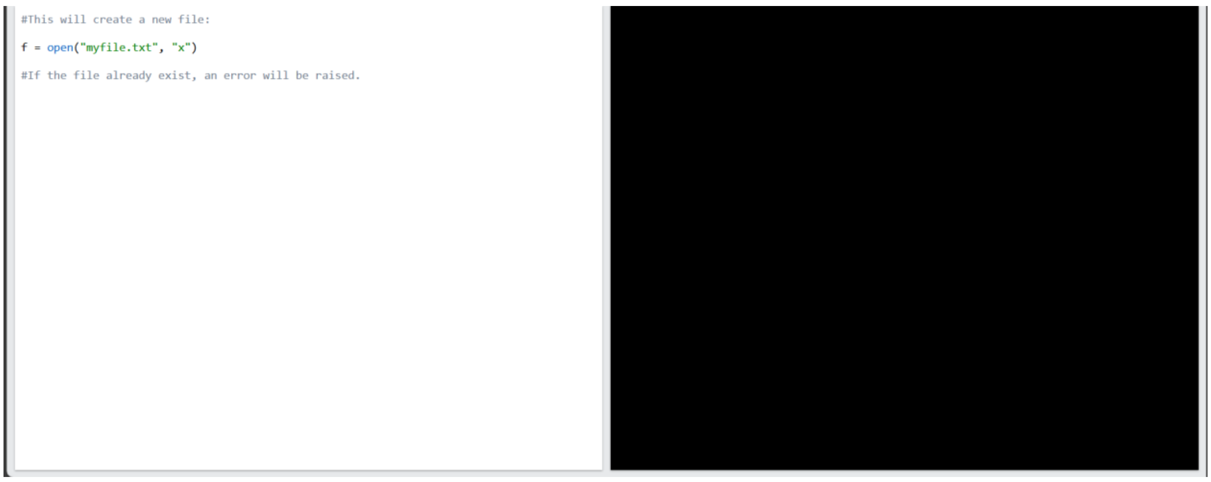


```
Hello! Welcome to demofile.txt

This file is for testing purposes.
```



```
with open("demofile.txt", "w") as f:
  f.write("Woops! I have deleted the content!")

#open and read the file after the overwriting:
with open("demofile.txt") as f:
  print(f.read())
```

```
Woops! I have deleted the content!
```



```
#This will create a new file:

f = open("myfile.txt", "x")

#If the file already exist, an error will be raised.
```