binary_array_search.py

```
'''
Author   :    Hameed Abdul
Date     :    3/16/18
Purpose  :    Implement a personal interpertation of
Binary Array Search from lecture
'''

from heap_sort import heap_sort
import numpy as np

def test_cases():
    # Random Arrays to test
    rand_arr1 = np.random.randint(256, size=8)
    rand_arr2 = np.random.randint(256, size=8)

    # Given Test Cases
    sorted_arr = [0, 1, 2, 3, 4, 5, 7, 9, 10, 12, 14, 18, 20]
    unsorted_arr = [3, 0, 1, 10, 18, 4, 7, 20, 15, 9, 2, 12, 14, 5]

    print(heap_sort(sorted_arr))
    binary_search_array(sorted_arr, 188, sorted_arr)
    binary_search_array(sorted_arr, 188, sorted_arr)

    print(heap_sort(unsorted_arr))
    binary_search_array(unsorted_arr, 4, unsorted_arr)

def binary_search_array(curr_arr, value, full_arr):
    '''
    :param value: Query value
    :param curr_arr: Input Array that is assumed to be unsorted
    :return: Sorted array and Index of value within array(Index will be based on SORTED output array)
    '''

    # Exit if array is empty
    if len(curr_arr) == 0:
        print("Value not in list")
        return None

    # Heap sort input array
    heap_sort(curr_arr)

    # Size and Middle value
    size = len(curr_arr)
    middle = size // 2

    if curr_arr[middle] == value:
        print("Found value at index :", full_arr.index(value))
        return middle

    # Recursively search right or left children until either value is found or list is empty
    if value < curr_arr[middle]:
        binary_search_array(curr_arr[:middle], value, full_arr)
    elif value > curr_arr[middle]:
        binary_search_array(curr_arr[middle + 1:], value, full_arr)

test_cases()
```