WK02\Assignment02_binary_search.py

```
# Python 3.12

# Generate an ordered array of length <= 100; 1 and 10000 included as end point extremes
import random
a=[1]
for i in range(2000):
    if random.randint(0,10) == 5:
        a.append(1000+i)
    if len(a) == 99:
        break
a.append(10000)
print(a) # optional print statement to check the array values

# Randomly select the target from the list
target = a[random.randint(0,len(a)-1)] # 1 or 10000 to test ends, 0 or 99999 to test fail
print('\nTarget value:\t\t', target)
# store starting length of the array
length = len(a)
print('Size of array "a":\t', len(a))
# set a variable to keep track of the upper index of the array to be searching through
top = length
# set a variable to keep track of the lower index of the array to be searching through
floor = 0
# set initial guess to the index of the center most element, rounded down
guess = int(length*.5)
# initialize count variable to keep track of the number of iterations to find the target
count = 0

# infinite loop until a break is encountered
while True:
    # iterate the count variable
    count += 1
    # if statement to check if the value of the array "a" at index "guess" is equal to,
    # greater than or less than the target value

    if len(a) == 1: # special case for when there is only one element in the array
        if a[0] == target:
            guess = 0
            print('\nIndex ', guess, ', "a[guess]", holds ', a[guess], ' which should be equal to the "target" value, ', target, '. It took ', count, ' guesses to find the target.\n', sep='')
            break
        else:
            print('Target not found after', count, 'guesses.')
            break
    # check the special case where it takes the maximum number of tries to find
    # the target, where the floor and top variables are neighboring indexes
    elif floor == top - 1:
        if a[guess-1] == target: # confirm the found value is indeed the target
```

```
            guess = guess - 1
            print('\nIndex ', guess, ', "a[guess]", holds ', a[guess], ' which should be equal to the "target" value, ', target, '. It took ', count, ' guesses to find the target.\n', sep='')
            break
        else:
            print('Target not found after', count, 'guesses.')
            break
    elif a[guess] == target:
        print('\nIndex ', guess, ', "a[guess]", holds ', a[guess], ' which should be equal to the "target" value, ', target, '. It took ', count, ' guesses to find the target.\n', sep='')
        break
    elif a[guess] < target:
        floor = guess # update the floor value so future guesses don't go below it
        length = len(a[guess:top]) # calculate the remaining number of values to check
        guess = int(guess+(length//2)) # update guess to a value between guess and top
    else:
        top = guess # update the top value so future guesses don't go above it
        length = len(a[floor:guess]) # calculate the remaining number of values to check
        guess = int(guess-(length//2)) # update guess to a value between guess and floor
```