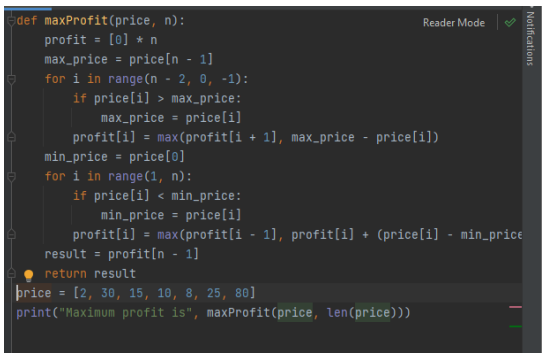
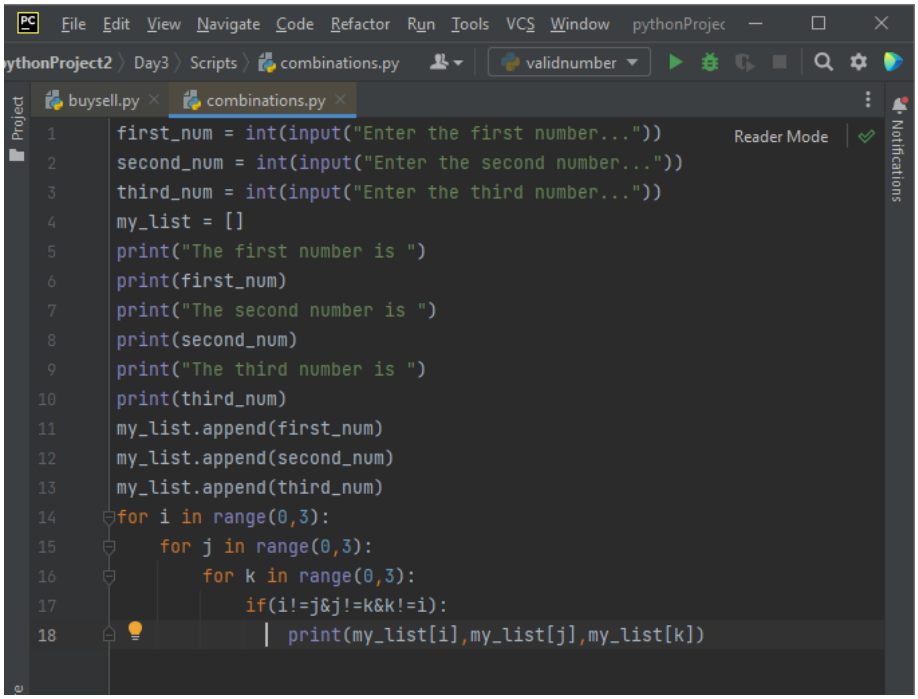
DAY 2 PROGRAMS

1.



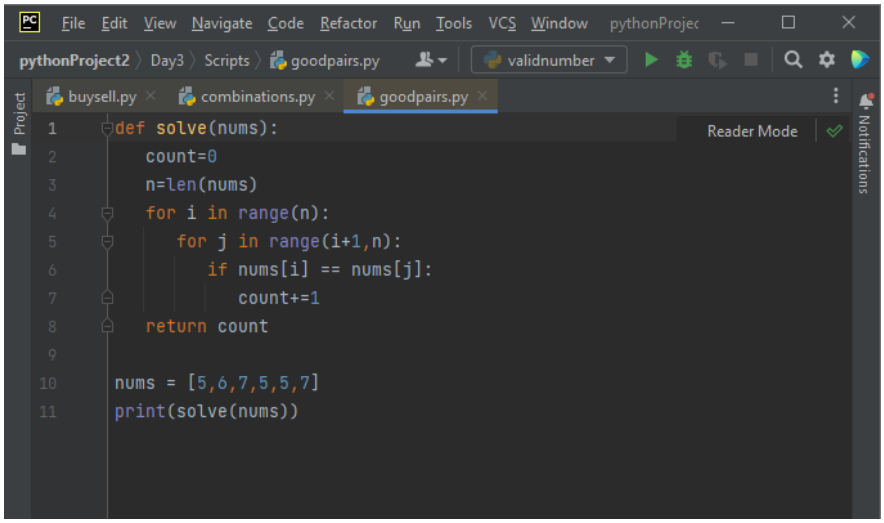
```
def maxProfit(price, n):
    profit = [0] * n
    max_price = price[n - 1]
    for i in range(n - 2, 0, -1):
        if price[i] > max_price:
            max_price = price[i]
        profit[i] = max(profit[i + 1], max_price - price[i])
    min_price = price[0]
    for i in range(1, n):
        if price[i] < min_price:
            min_price = price[i]
        profit[i] = max(profit[i - 1], profit[i] + (price[i] - min_price
    result = profit[n - 1]
    return result
price = [2, 30, 15, 10, 8, 25, 80]
print("Maximum profit is", maxProfit(price, len(price)))
```

2.



```
first_num = int(input("Enter the first number..."))
second_num = int(input("Enter the second number..."))
third_num = int(input("Enter the third number..."))
my_list = []
print("The first number is ")
print(first_num)
print("The second number is ")
print(second_num)
print("The third number is ")
print(third_num)
my_list.append(first_num)
my_list.append(second_num)
my_list.append(third_num)
for i in range(0,3):
    for j in range(0,3):
        for k in range(0,3):
            if(i!=j&j!=k&k!=i):
                print(my_list[i],my_list[j],my_list[k])
```

3.



```
def solve(nums):
    count=0
    n=len(nums)
    for i in range(n):
        for j in range(i+1,n):
            if nums[i] == nums[j]:
                count+=1
    return count

nums = [5,6,7,5,5,7]
print(solve(nums))
```

4.

```python
def add_binary_nums(x, y):
    max_len = max(len(x), len(y))

    x = x.zfill(max_len)
    y = y.zfill(max_len)

    # initialize the result
    result = ''

    # initialize the carry
    carry = 0

    # Traverse the string
    for i in range(max_len - 1, -1, -1):
        r = carry
        r += 1 if x[i] == '1' else 0
        r += 1 if y[i] == '1' else 0
        result = ('1' if r % 2 == 1 else '0') + result
        carry = 0 if r < 2 else 1  # Compute the carry.
    if carry != 0: result = '1' + result
    return result.zfill(max_len)
print(add_binary_nums('1101', '100'))
```

5.

```python
def minJumps(arr, l, h):
    if (h == l):
        return 0
    if (arr[l] == 0):
        return float('inf')
    min = float('inf')
    for i in range(l + 1, h + 1):
        if (i < l + arr[l] + 1):
            jumps = minJumps(arr, i, h)
            if (jumps != float('inf') and
                    jumps + 1 < min):
                min = jumps + 1

    return min

arr = [1, 3, 6, 3, 2, 3, 6, 8, 9, 5]
n = len(arr)
print('Minimum number of jumps to reach',
      'end is', minJumps(arr, 0, n - 1))
```

6.

```
def reverse(s):
    str = ""
    for i in s:
        str = i + str
    return str

s = "12345"

print("The original string is : ", end="")
print(s)

print("The reversed string(using loops) is : ", end="")
print(reverse(s))
```

7.

```
from itertools import permutations
a=permutations ([1,2,3],2)
for i in a:
  print(i)
```

8.

```
s1=input("enter a string")
s2=input("enter a string")
if(sorted(s1)==sorted(s2)):
    print("anagram")
else:
    print("not anagram")
```

10.

```python
def editDistanceHelper(i, j, str1, str2):
    if i == 0:
        return j
    if j == 0:
        return i
    ans = 1 + min(
        {
            editDistanceHelper(i, j - 1, str1, str2),  # Insert
            editDistanceHelper(i - 1, j, str1, str2),  # Remove
            editDistanceHelper(i - 1, j - 1, str1, str2),  # Replace
        }
    )
    if str1[i - 1] == str2[j - 1]:
        ans = min(ans, editDistanceHelper(i - 1, j - 1, str1, str2))

    return ans


def editDistance(str1, str2):
    n = len(str1)
    m = len(str2)
    ans = editDistanceHelper(n, m, str1, str2)
    return ans
print(editDistance("insertion", "execution"))
```