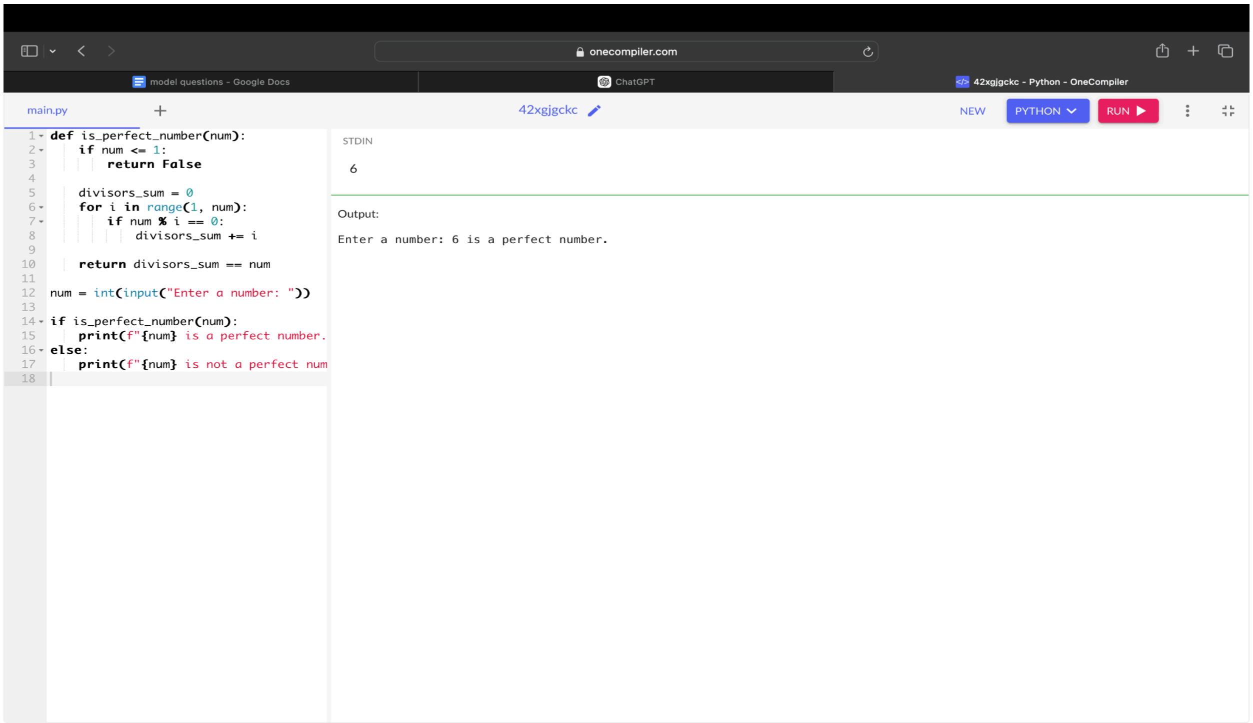
1. Write a program to print the given perfect number or not.

PROGRAM:

```
def is_perfect_number(num):
    if num <= 1:
        return False
    divisors_sum = 0
    for i in range(1, num):
        if num % i == 0:
            divisors_sum += i
    return divisors_sum == num
num = int(input("Enter a number: "))
if is_perfect_number(num):
    print(f"{num} is a perfect number.")
else:
    print(f"{num} is not a perfect number.")
```

OUTPUT:



onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN

```
def is_perfect_number(num):
    if num <= 1:
        return False

    divisors_sum = 0
    for i in range(1, num):
        if num % i == 0:
            divisors_sum += i

    return divisors_sum == num

num = int(input("Enter a number: "))

if is_perfect_number(num):
    print(f"{num} is a perfect number.
else:
    print(f"{num} is not a perfect num
```

STDIN

```
6
```

Output:

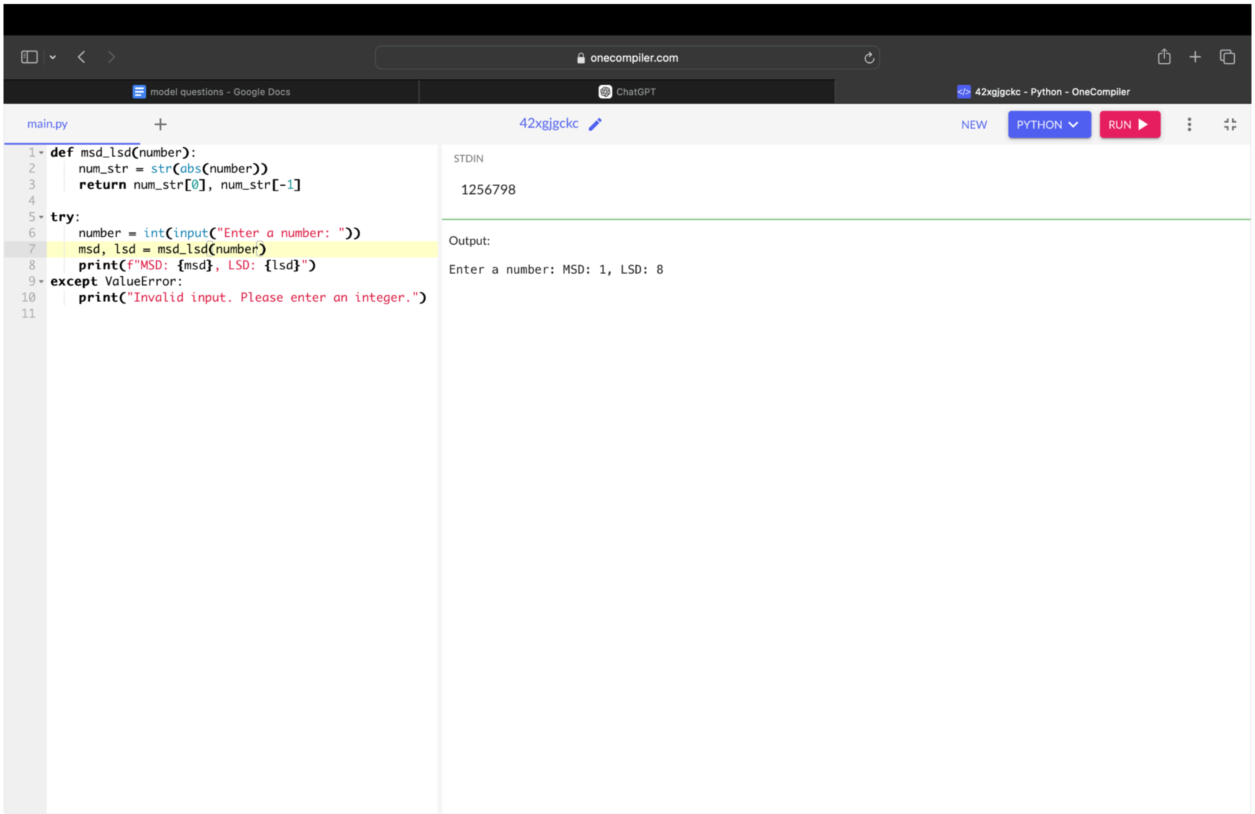
```
Enter a number: 6 is a perfect number.
```

2. Write the python program to display the most & least significant digit of a number.

PROGRAM:

```
def msd_lsd(number):
    num_str = str(abs(number))
    return num_str[0], num_str[-1]
try:
    number = int(input("Enter a number: "))
    msd, lsd = msd_lsd(number)
    print(f"MSD: {msd}, LSD: {lsd}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
42xgjgckc
NEW
PYTHON
RUN

```
def msd_lsd(number):
    num_str = str(abs(number))
    return num_str[0], num_str[-1]

try:
    number = int(input("Enter a number: "))
    msd, lsd = msd_lsd(number)
    print(f"MSD: {msd}, LSD: {lsd}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

STDIN

```
1256798
```

Output:

```
Enter a number: MSD: 1, LSD: 8
```

3. Write a program using a function to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered a 12 percent rate of interest; for all other customers, the ROI is 10 percent.

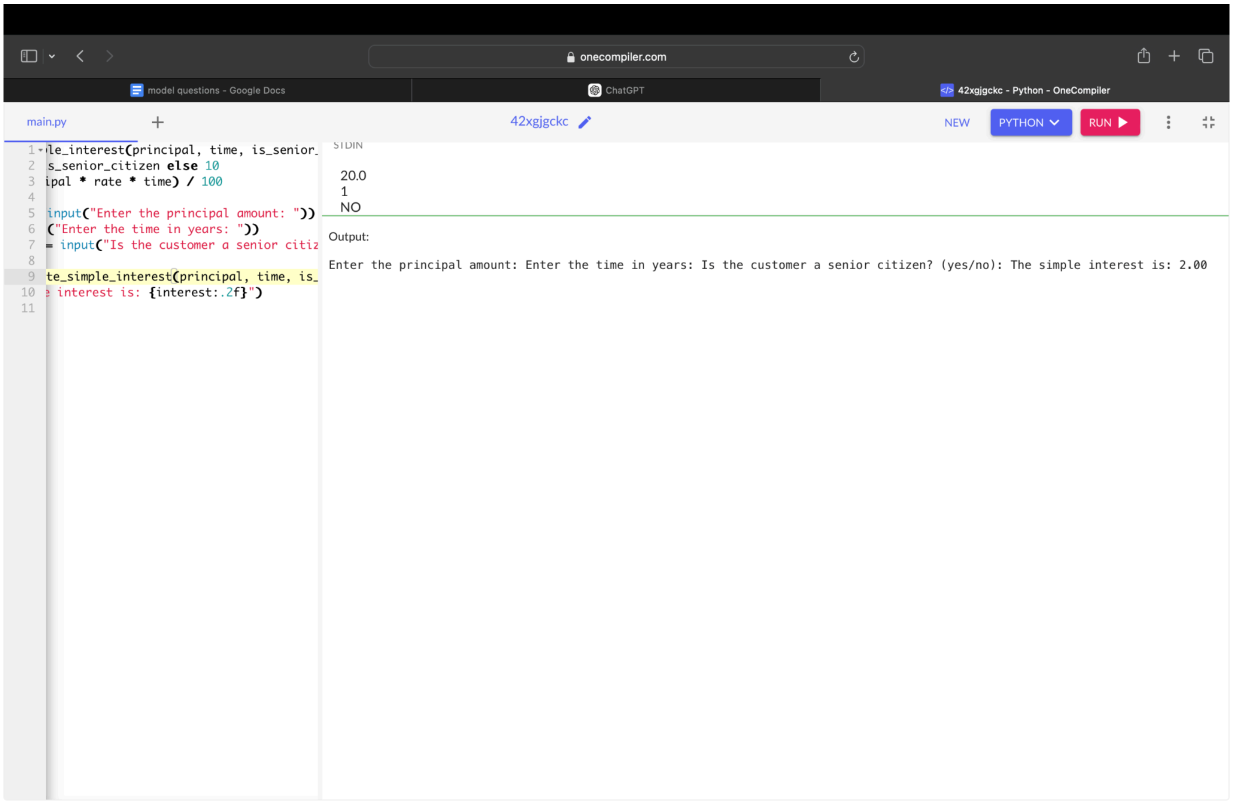
PROGRAM:

```
def calculate_simple_interest(principal, time, is_senior_citizen):
    rate = 12 if is_senior_citizen else 10
    return (principal * rate * time) / 100


principal = float(input("Enter the principal amount: "))
time = float(input("Enter the time in years: "))
is_senior_citizen = input("Is the customer a senior citizen? (yes/no): ").strip().lower() == 'yes'


interest = calculate_simple_interest(principal, time, is_senior_citizen)
print(f"The simple interest is: {interest:.2f}")
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN

```
le_interest(principal, time, is_senior_
s_senior_citizen else 10
ipal * rate * time) / 100

input("Enter the principal amount: "))
("Enter the time in years: "))
= input("Is the customer a senior citiz

te_simple_interest(principal, time, is_
e interest is: {interest:.2f}")

```

STDIN

```
20.0
1
NO
```

Output:

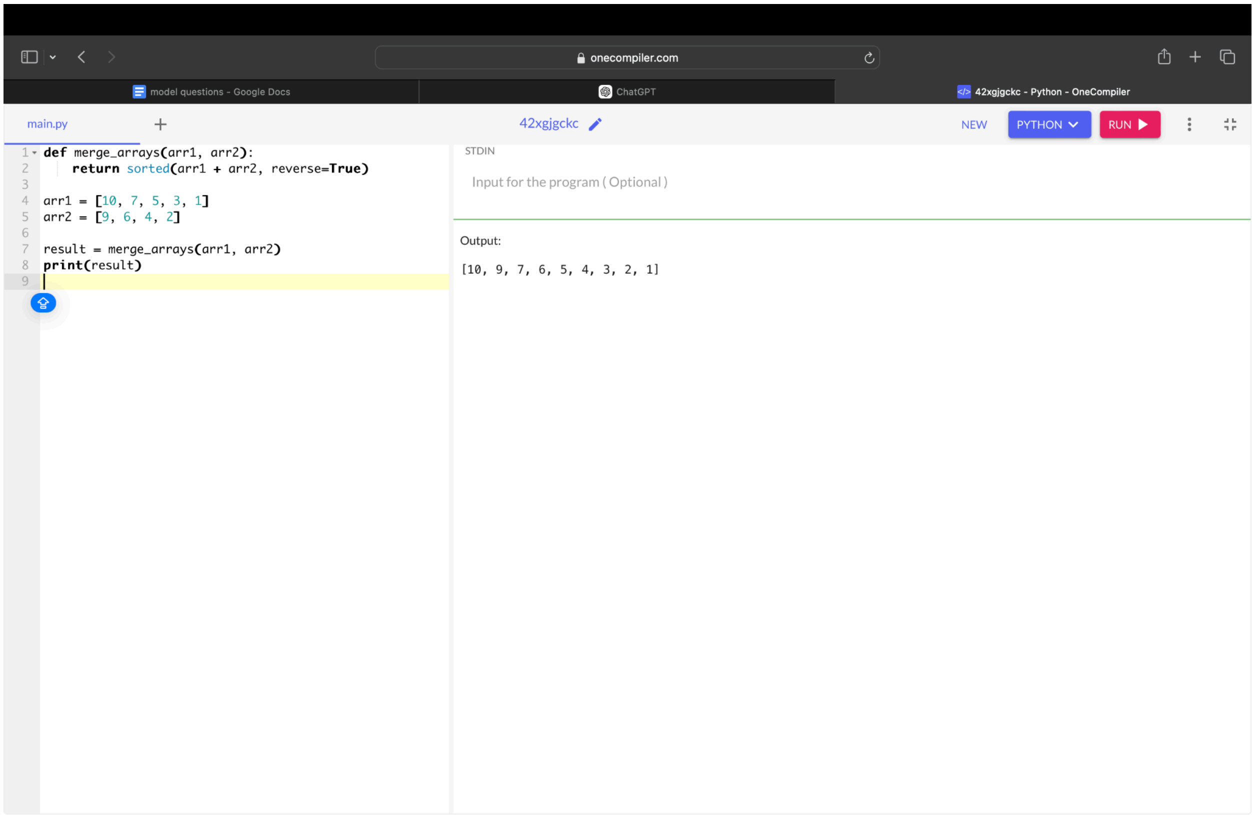
```
Enter the principal amount: Enter the time in years: Is the customer a senior citizen? (yes/no): The simple interest is: 2.00
```

4. Write a python program to merge two sorted arrays in non ascending order.

PROGRAM:

```
def merge_arrays(arr1, arr2):
    return sorted(arr1 + arr2, reverse=True)
arr1 = [10, 7, 5, 3, 1]
arr2 = [9, 6, 4, 2]
result = merge_arrays(arr1, arr2)
print(result)
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
42xgjgckc
NEW
PYTHON
RUN
STDIN
Input for the program (Optional)

```
def merge_arrays(arr1, arr2):
    return sorted(arr1 + arr2, reverse=True)

arr1 = [10, 7, 5, 3, 1]
arr2 = [9, 6, 4, 2]

result = merge_arrays(arr1, arr2)
print(result)
```

Output:

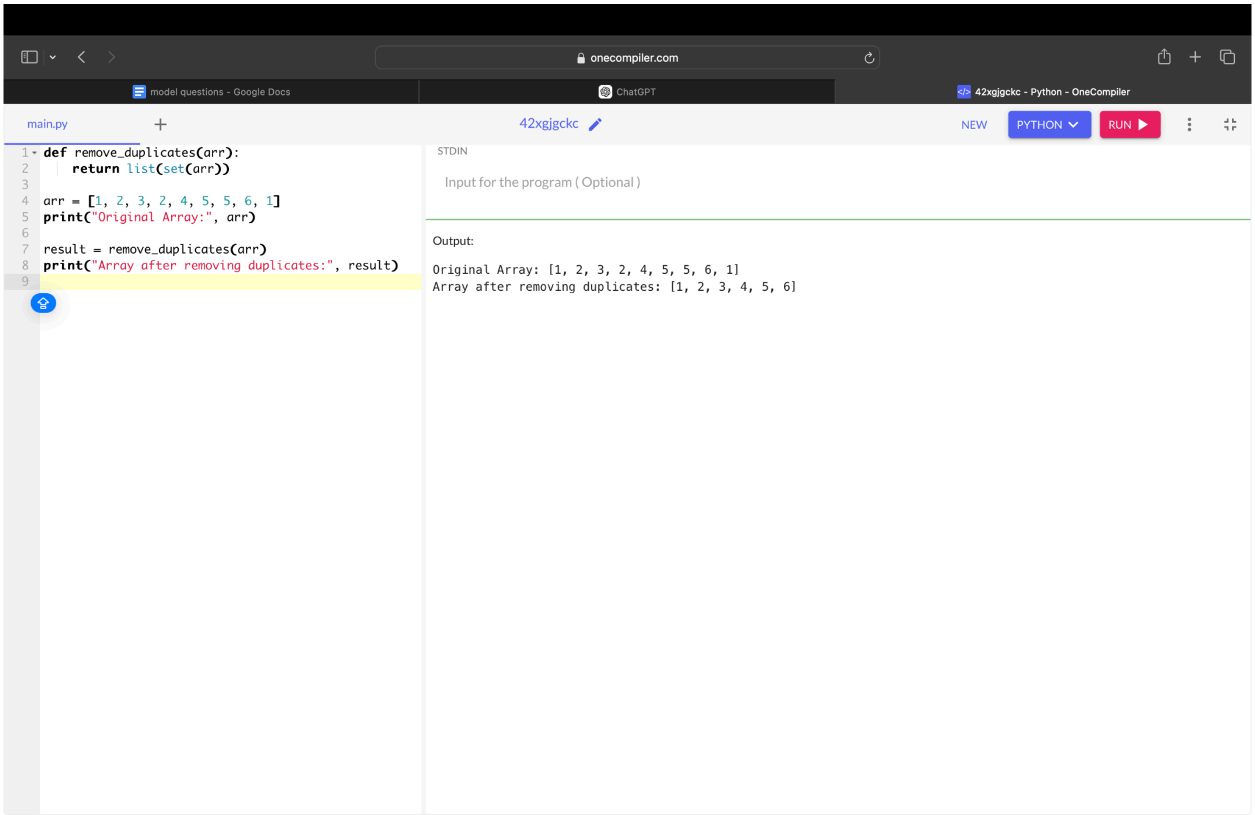
```
[10, 9, 7, 6, 5, 4, 3, 2, 1]
```

5. Program to remove duplicates present in 1D array.

PROGRAM:

```
def remove_duplicates(arr):
    return list(set(arr))
arr = [1, 2, 3, 2, 4, 5, 5, 6, 1]
print("Original Array:", arr)
result = remove_duplicates(arr)
print("Array after removing duplicates:", result)
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN
STDIN
Input for the program (Optional)
Output:

```
def remove_duplicates(arr):
    return list(set(arr))

arr = [1, 2, 3, 2, 4, 5, 5, 6, 1]
print("Original Array:", arr)

result = remove_duplicates(arr)
print("Array after removing duplicates:", result)
```

```
Original Array: [1, 2, 3, 2, 4, 5, 5, 6, 1]
Array after removing duplicates: [1, 2, 3, 4, 5, 6]
```

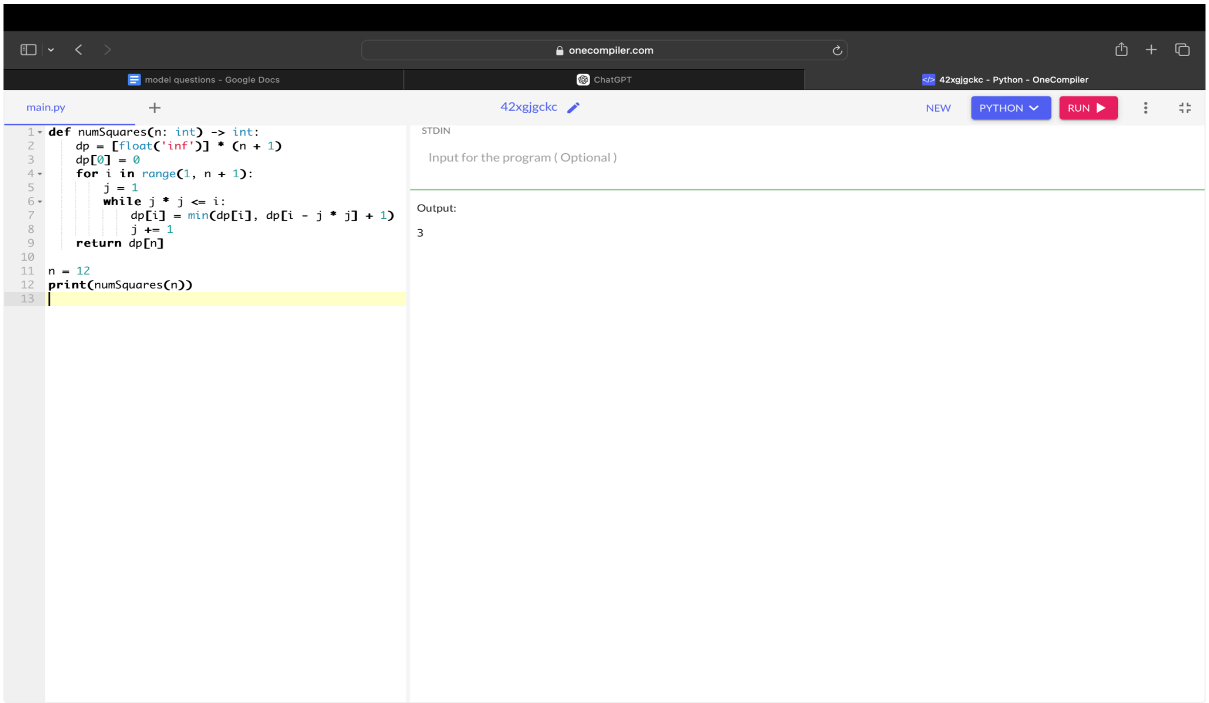
6. Given an integer n, return the least number of perfect square numbers that sum to n. A perfect square is an integer that is the square of an integer, in other words, it is the product of some integer with itself. For example, 1, 4, 9 & 16 are perfect squares while 3 and 11 or not.

PROGRAM:

```
def numSquares(n: int) -> int:
    dp = [float('inf')] * (n + 1)
    dp[0] = 0
    for i in range(1, n + 1):
        j = 1
        while j * j <= i:
            dp[i] = min(dp[i], dp[i - j * j] + 1)
            j += 1
    return dp[n]

n = 12
print(numSquares(n))
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN
STDIN
Input for the program (Optional)
Output:

```
def numSquares(n: int) -> int:
    dp = [float('inf')] * (n + 1)
    dp[0] = 0
    for i in range(1, n + 1):
        j = 1
        while j * j <= i:
            dp[i] = min(dp[i], dp[i - j * j] + 1)
            j += 1
    return dp[n]

n = 12
print(numSquares(n))
```

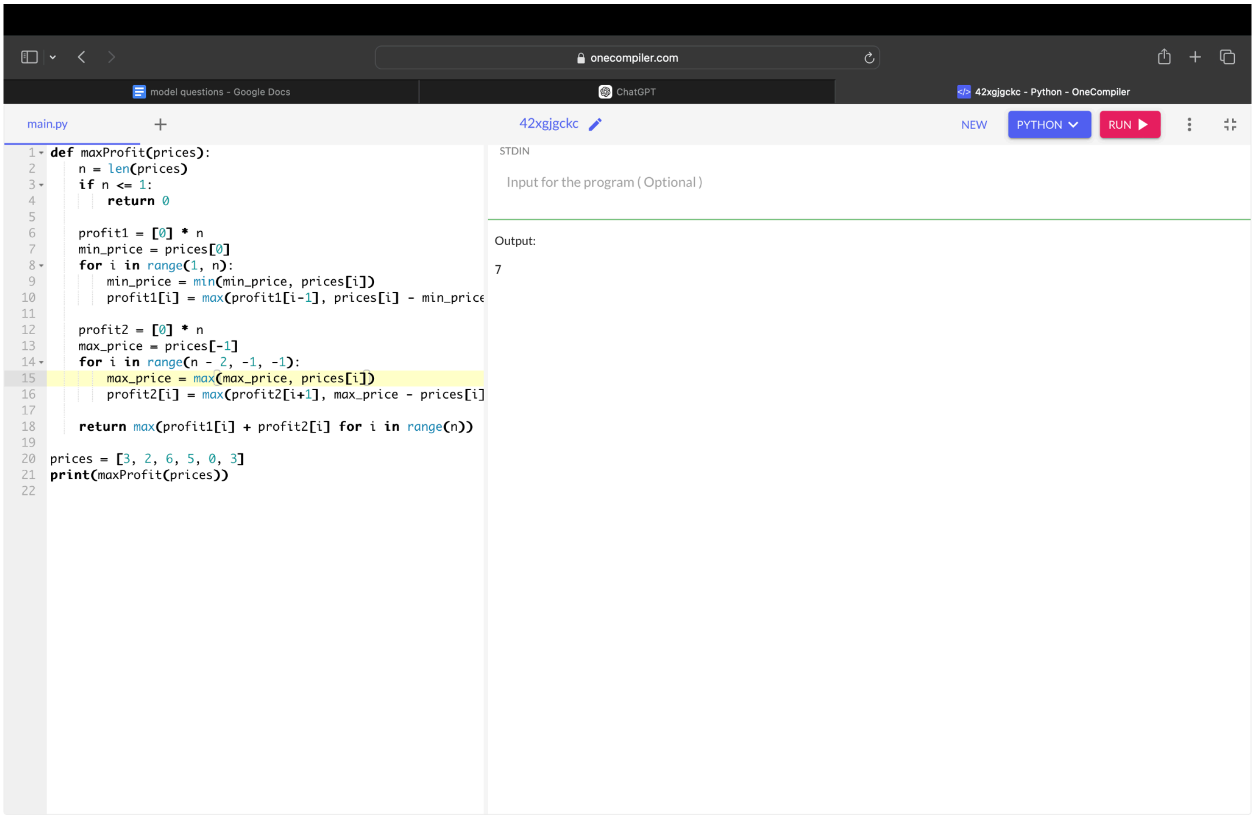
```
3
```

7. In daily share trading, a buyer buys shares in the morning & sells them on the same day. If the trader is allowed to make at most 2 transactions can only start after the first one is completed(Buy → Sell → Buy → Sell). Given stock prices throughout the day. Find out the maximum profit that a share trader could have made.

PROGRAM:

```
def maxProfit(prices):
    n = len(prices)
    if n <= 1:
        return 0
    profit1 = [0] * n
    min_price = prices[0]
    for i in range(1, n):
        min_price = min(min_price, prices[i])
        profit1[i] = max(profit1[i-1], prices[i] - min_price)
    profit2 = [0] * n
    max_price = prices[-1]
    for i in range(n - 2, -1, -1):
        max_price = max(max_price, prices[i])
        profit2[i] = max(profit2[i+1], max_price - prices[i])
    return max(profit1[i] + profit2[i] for i in range(n))
prices = [3, 2, 6, 5, 0, 3]
print(maxProfit(prices))
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN
STDIN
Input for the program (Optional)

```
def maxProfit(prices):
    n = len(prices)
    if n <= 1:
        return 0

    profit1 = [0] * n
    min_price = prices[0]
    for i in range(1, n):
        min_price = min(min_price, prices[i])
        profit1[i] = max(profit1[i-1], prices[i] - min_price

    profit2 = [0] * n
    max_price = prices[-1]
    for i in range(n - 2, -1, -1):
        max_price = max(max_price, prices[i])
        profit2[i] = max(profit2[i+1], max_price - prices[i]

    return max(profit1[i] + profit2[i] for i in range(n))

prices = [3, 2, 6, 5, 0, 3]
print(maxProfit(prices))
```

Output:

```
7
```

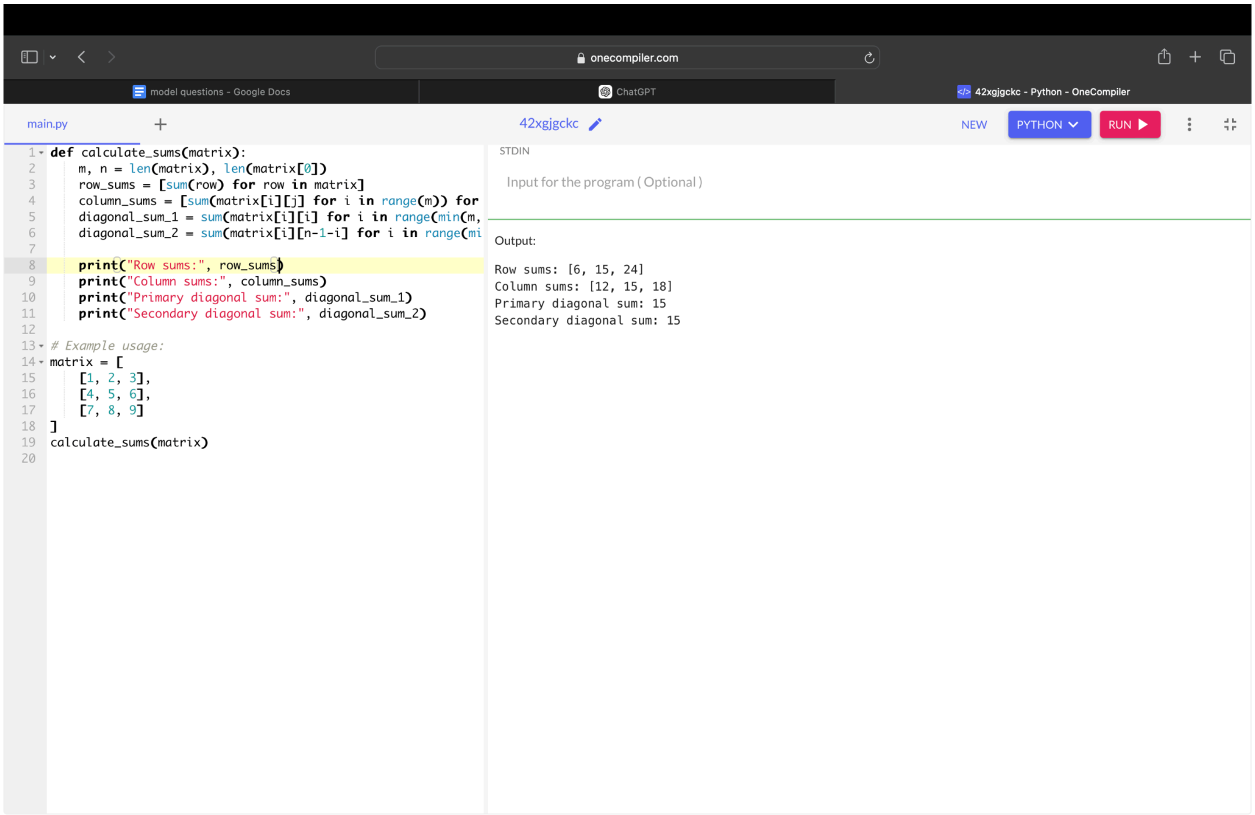
8. Given an m x n matrix. Find the row sum, column sum & diagonal sum of elements.

PROGRAM:

```
def calculate_sums(matrix):
    m, n = len(matrix), len(matrix[0])
    row_sums = [sum(row) for row in matrix]
    column_sums = [sum(matrix[i][j] for i in range(m)) for j in range(n)]
    diagonal_sum_1 = sum(matrix[i][i] for i in range(min(m, n)))
    diagonal_sum_2 = sum(matrix[i][n-1-i] for i in range(min(m, n)))
    print("Row sums:", row_sums)
    print("Column sums:", column_sums)
    print("Primary diagonal sum:", diagonal_sum_1)
    print("Secondary diagonal sum:", diagonal_sum_2)


matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
calculate_sums(matrix)
```

OUTPUT:


onecompiler.com
model questions - Google Docs
ChatGPT
42xgjgckc - Python - OneCompiler
main.py
+
42xgjgckc
NEW
PYTHON
RUN
STDIN
Input for the program (Optional)

```
def calculate_sums(matrix):
    m, n = len(matrix), len(matrix[0])
    row_sums = [sum(row) for row in matrix]
    column_sums = [sum(matrix[i][j] for i in range(m)) for
    diagonal_sum_1 = sum(matrix[i][i] for i in range(min(m,
    diagonal_sum_2 = sum(matrix[i][n-1-i] for i in range(mi

    print("Row sums:", row_sums)
    print("Column sums:", column_sums)
    print("Primary diagonal sum:", diagonal_sum_1)
    print("Secondary diagonal sum:", diagonal_sum_2)

# Example usage:
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
calculate_sums(matrix)
```

Output:

```
Row sums: [6, 15, 24]
Column sums: [12, 15, 18]
Primary diagonal sum: 15
Secondary diagonal sum: 15
```