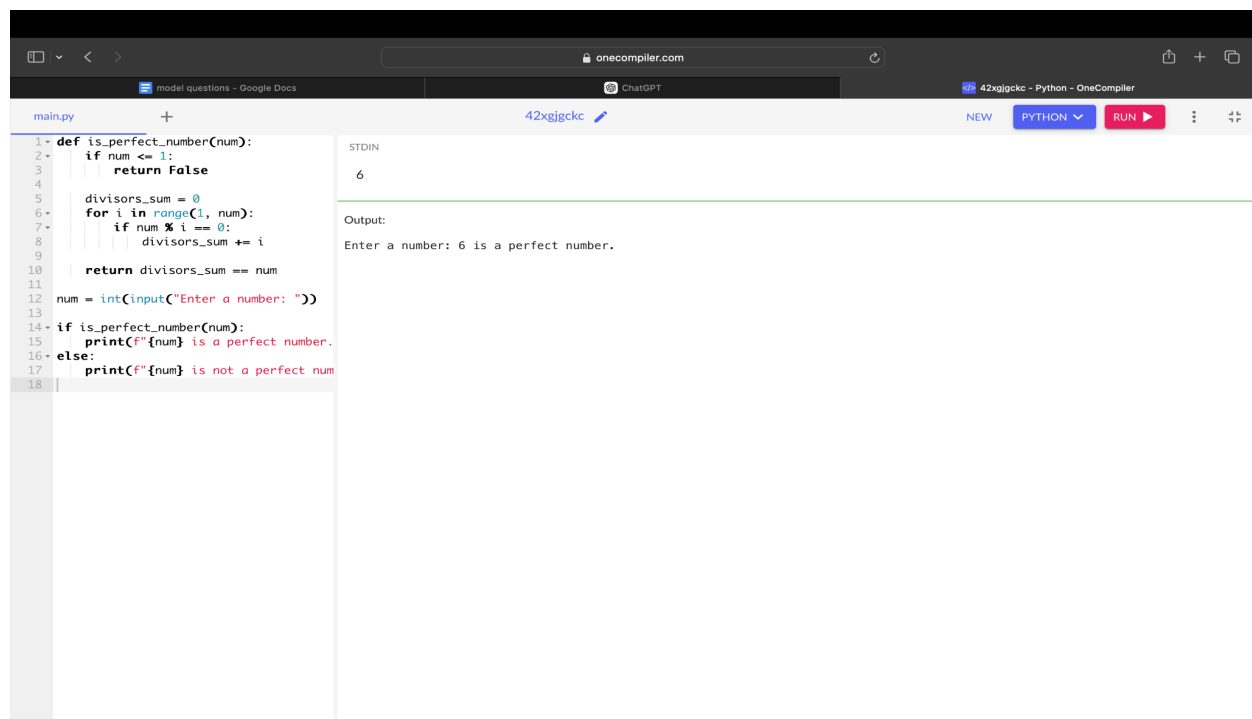


1. Write a program to print the given perfect number or not.

**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:**



The screenshot shows a web browser window with the URL `onecompiler.com`. The browser's address bar also displays `42xgjgckc`. The page has a dark theme. At the top, there are tabs for `model questions - Google Docs`, `ChatGPT`, and `42xgjgckc - Python - OneCompiler`. Below the tabs, there is a toolbar with buttons for `NEW`, `PYTHON` (with a dropdown arrow), `RUN` (with a play icon), and a settings icon. The main area is split into two panels. The left panel, titled `main.py`, contains the Python code from the previous block. The right panel, titled `STDIN`, shows the input `6`. Below the `STDIN` panel, the `Output:` section displays the program's 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:**

The screenshot shows a web browser window with the URL `onecompiler.com`. The page title is "42xgjcgc - Python - OneCompiler". The code editor on the left contains the following Python code:

```
1 def msd_lsd(number):  
2     num_str = str(abs(number))  
3     return num_str[0], num_str[-1]  
4  
5 try:  
6     number = int(input("Enter a number: "))  
7     msd, lsd = msd_lsd(number)  
8     print(f"MSD: {msd}, LSD: {lsd}")  
9 except ValueError:  
10     print("Invalid input. Please enter an integer.")  
11
```

The right side of the interface shows the execution results. Under the "STDIN" tab, the input "1256798" is shown. Under the "Output" tab, the output "Enter a number: MSD: 1, LSD: 8" is displayed.

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: "))
```

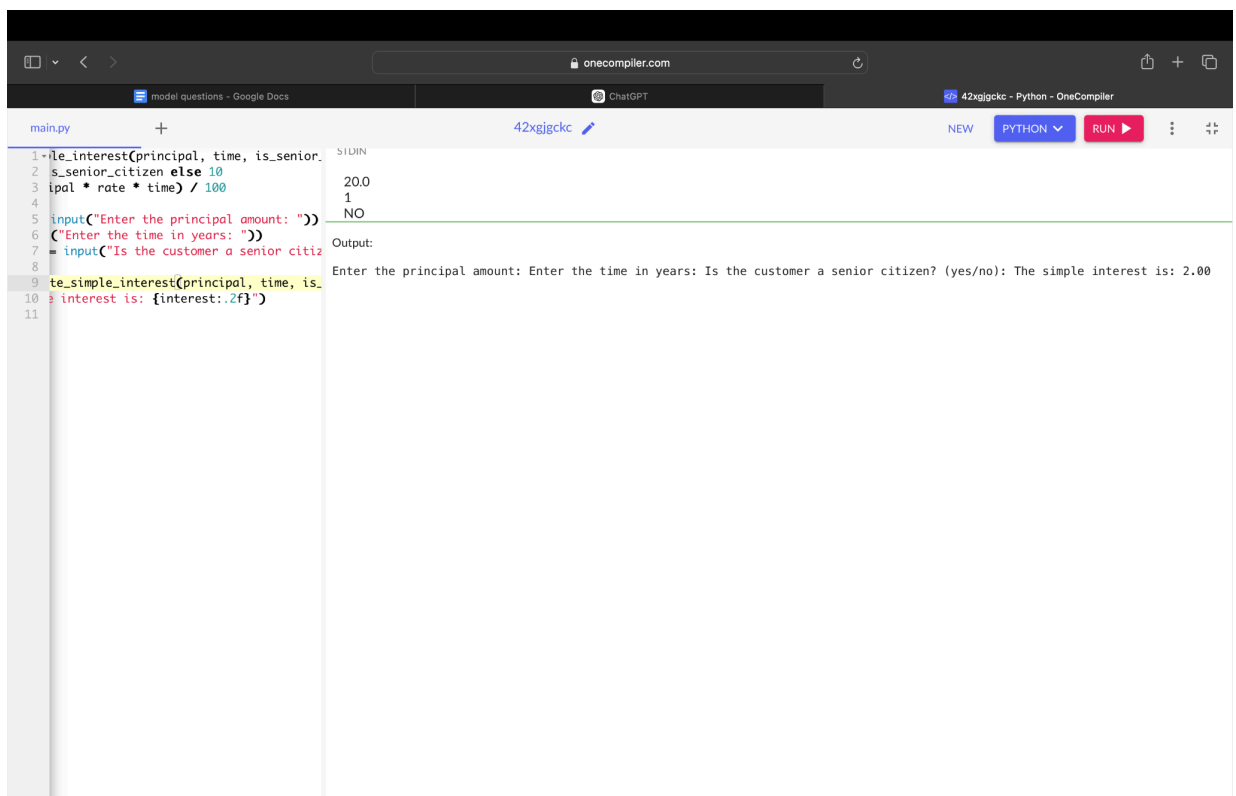
```
time = float(input("Enter the time in years: "))
```

```
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:**



The screenshot shows a web browser window with the URL `onecompiler.com`. The browser has several tabs open, including "model questions - Google Docs", "ChatGPT", and "42xgjgckc - Python - OneCompiler". The active tab shows a Python file named `main.py` with the following code:

```
1 def calculate_simple_interest(principal, time, is_senior_citizen):
2     rate = 12 if is_senior_citizen else 10
3     return (principal * rate * time) / 100
4
5 principal = float(input("Enter the principal amount: "))
6 time = float(input("Enter the time in years: "))
7 is_senior_citizen = input("Is the customer a senior citizen? (yes/no): ").strip().lower() == 'yes'
8
9 interest = calculate_simple_interest(principal, time, is_senior_citizen)
10 print(f"The simple interest is: {interest:.2f}")
11
```

On the right side of the IDE, there is a "STDIN" input area with the following values:

```
20.0
1
NO
```

Below the "STDIN" area, there is an "Output:" section showing the program's output:

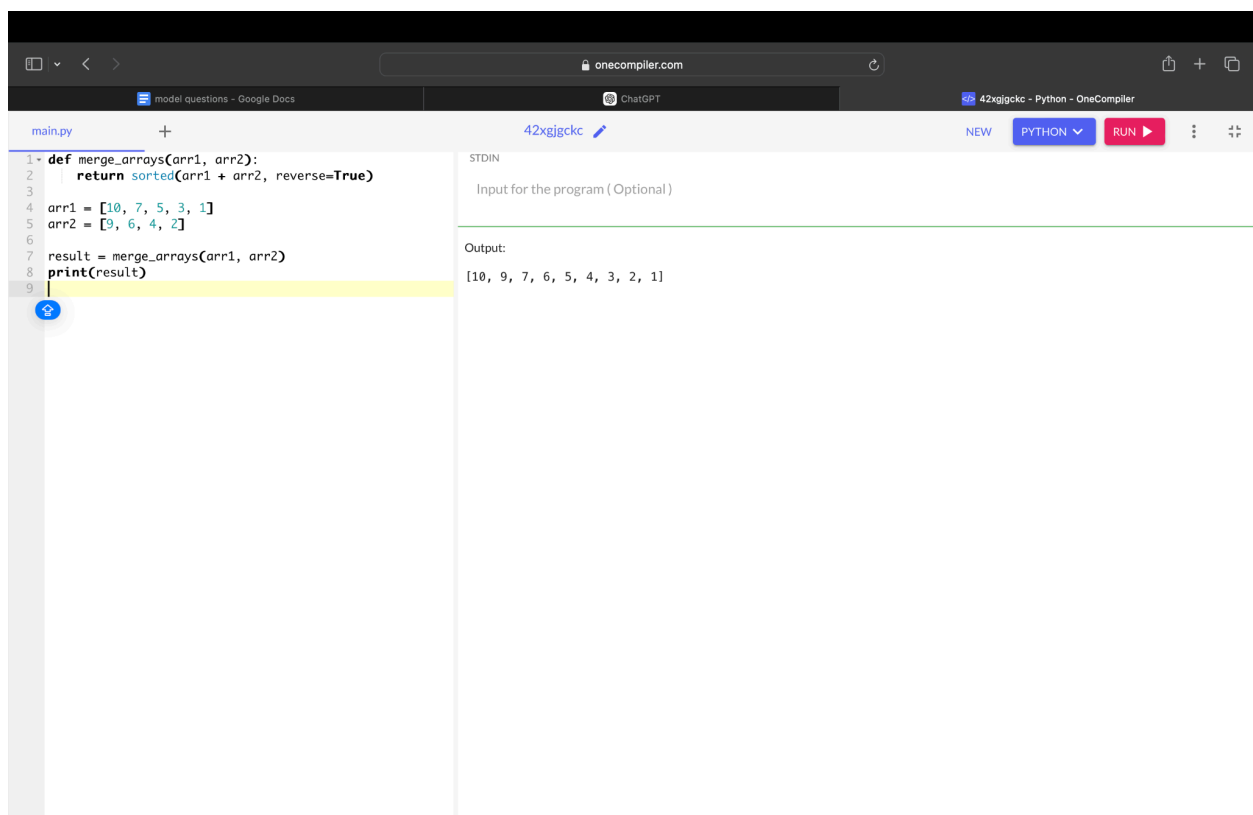
```
Enter the principal amount: Enter the time in years: Is the customer a senior citizen? (yes/no): The simple interest is: 2.00
```

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:**



The screenshot shows a web browser window with the URL `onecompiler.com`. The browser tabs include "model questions - Google Docs", "ChatGPT", and "42xgigckc - Python - OneCompiler". The main content area displays a Python program in a file named `main.py`. The code defines a function `merge_arrays` that takes two arrays, concatenates them, and sorts the result in descending order. It then defines two arrays, `arr1 = [10, 7, 5, 3, 1]` and `arr2 = [9, 6, 4, 2]`, calls the function, and prints the result. The output of the program is shown in a box on the right, displaying the sorted array `[10, 9, 7, 6, 5, 4, 3, 2, 1]`.

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

STDIN  
Input for the program (Optional)

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:**

The screenshot shows a web browser window with the URL `onecompiler.com`. The browser's address bar and tabs are visible at the top. The main content area is divided into two panels. The left panel, titled `main.py`, contains the following Python code:

```
1 def remove_duplicates(arr):  
2     return list(set(arr))  
3  
4 arr = [1, 2, 3, 2, 4, 5, 5, 6, 1]  
5 print("Original Array:", arr)  
6  
7 result = remove_duplicates(arr)  
8 print("Array after removing duplicates:", result)  
9
```

The right panel, titled `42xgigckc`, shows the execution results. It has a section for `STDIN` with the text "Input for the program ( Optional )". Below this, the `Output:` section displays the following text:

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

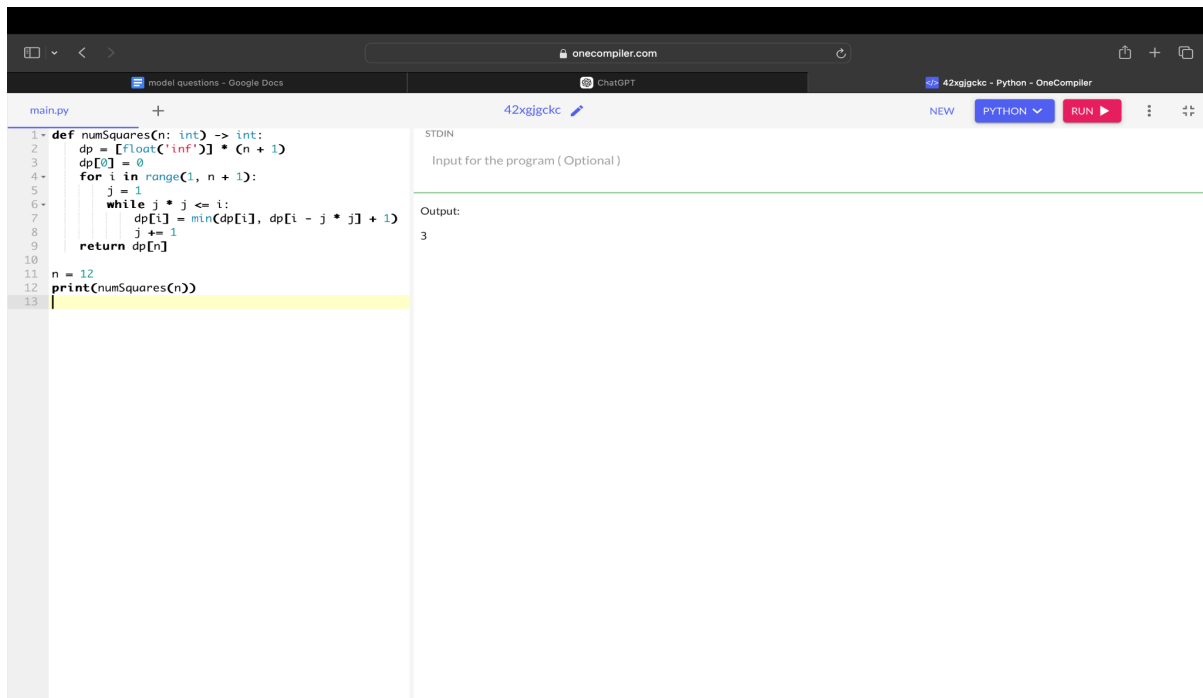
6. Given an integer  $n$ , return the least number of perfect square numbers that sum to  $n$ . A perfect square is an integer that is the square of an integer, in other words, it is the product of some integer with itself. For example, 1, 4, 9 & 16 are perfect squares while 3 and 11 or not.

### **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:**



The screenshot shows a web browser window with the URL `onecompiler.com`. The page displays a Python code editor with the following code:

```
1 def numSquares(n: int) -> int:
2     dp = [float('inf')] * (n + 1)
3     dp[0] = 0
4     for i in range(1, n + 1):
5         j = 1
6         while j * j <= i:
7             dp[i] = min(dp[i], dp[i - j * j] + 1)
8             j += 1
9     return dp[n]
10
11 n = 12
12 print(numSquares(n))
13
```

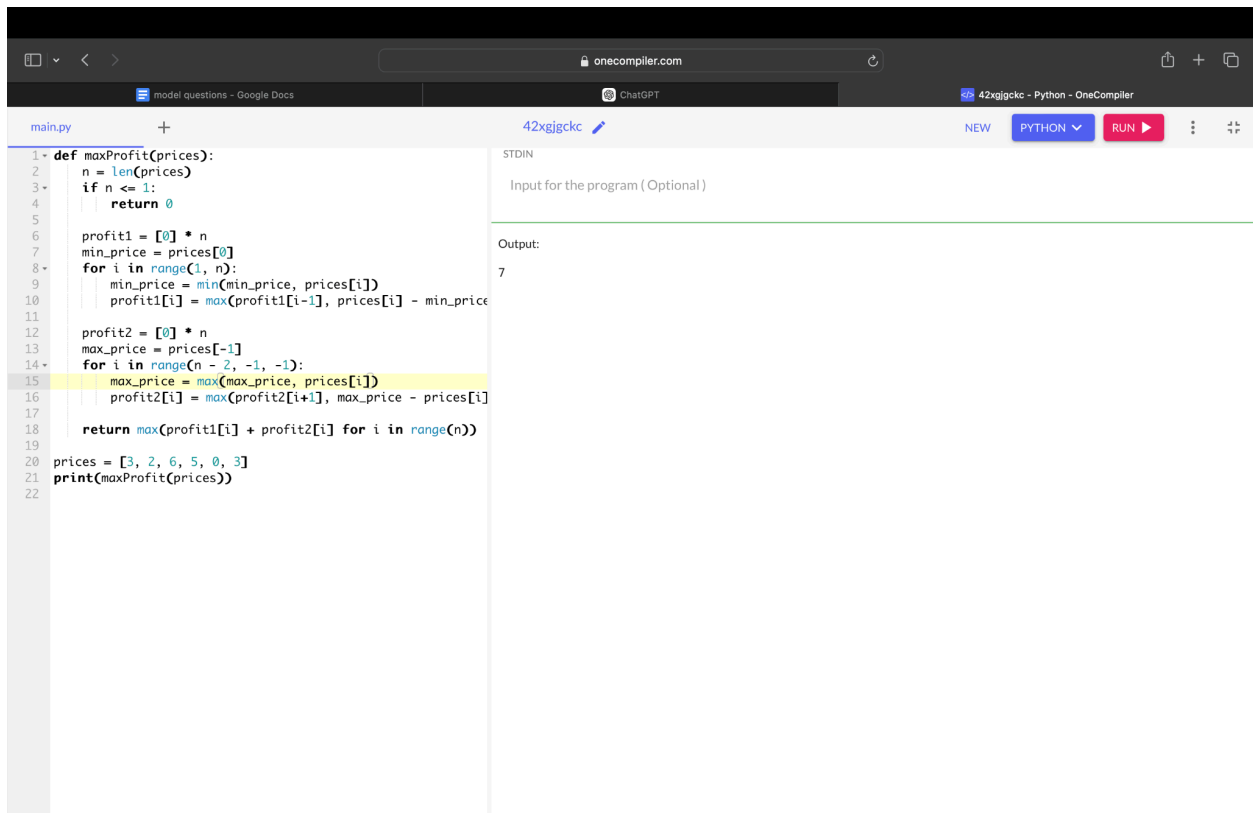
The code is executed, and the output is displayed as `3`. The interface includes a top bar with navigation icons, a search bar, and a language selector set to `PYTHON`. The output area is labeled `Output:` and shows the result `3`.

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:



The screenshot shows the OneCompiler Python IDE interface. The left pane displays a Python script named `main.py` with line numbers 1 to 22. The script defines a `maxProfit` function that calculates the maximum profit from a list of stock prices. It uses two arrays, `profit1` and `profit2`, to store intermediate results. The right pane shows the output of the program, which is the number 7. The browser address bar indicates the URL is `onecompiler.com`.

```
1 def maxProfit(prices):
2     n = len(prices)
3     if n <= 1:
4         return 0
5
6     profit1 = [0] * n
7     min_price = prices[0]
8     for i in range(1, n):
9         min_price = min(min_price, prices[i])
10        profit1[i] = max(profit1[i-1], prices[i] - min_price)
11
12    profit2 = [0] * n
13    max_price = prices[-1]
14    for i in range(n-2, -1, -1):
15        max_price = max(max_price, prices[i])
16        profit2[i] = max(profit2[i+1], max_price - prices[i])
17
18    return max(profit1[i] + profit2[i] for i in range(n))
19
20 prices = [3, 2, 6, 5, 0, 3]
21 print(maxProfit(prices))
22
```

Output:

7

8. Given an  $m \times n$  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:

The screenshot shows a web browser window with the URL `onecompiler.com`. The browser tabs include "model questions - Google Docs", "ChatGPT", and "42xgjc - Python - OneCompiler". The interface has a top bar with "NEW", "PYTHON", and "RUN" buttons. The main area is split into two panes. The left pane, titled "main.py", contains the following Python code:

```

1 def calculate_sums(matrix):
2     m, n = len(matrix), len(matrix[0])
3     row_sums = [sum(row) for row in matrix]
4     column_sums = [sum(matrix[i][j] for i in range(m)) for j in range(n)]
5     diagonal_sum_1 = sum(matrix[i][i] for i in range(min(m, n)))
6     diagonal_sum_2 = sum(matrix[i][n-1-i] for i in range(min(m, n)))
7
8     print("Row sums:", row_sums)
9     print("Column sums:", column_sums)
10    print("Primary diagonal sum:", diagonal_sum_1)
11    print("Secondary diagonal sum:", diagonal_sum_2)
12
13 # Example usage:
14 matrix = [
15     [1, 2, 3],
16     [4, 5, 6],
17     [7, 8, 9]
18 ]
19 calculate_sums(matrix)
20

```

The right pane, titled "42xgjc", shows the output of the program. It includes a section for "STDIN" with the text "Input for the program ( Optional )" and a section for "Output" with the following results:

```

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

```