ASSIGNMENT 1

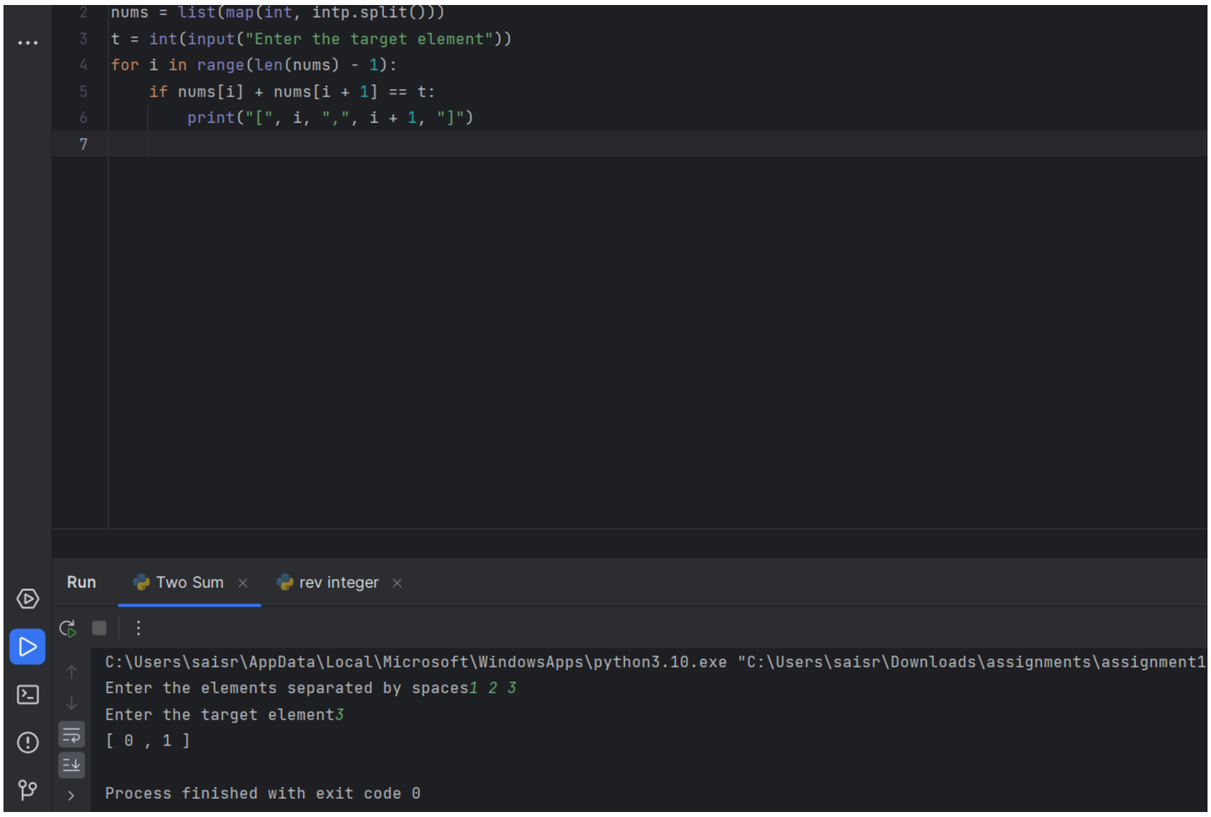
1. Two Sum

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.
You may assume that each input would have *exactly* one solution, and you may not use the *same*
element twice.
You can return the answer in any order.

Coding:

```
intp = input("Enter the elements separated by spaces")
nums = list(map(int, intp.split()))
t = int(input("Enter the target element"))
for i in range(len(nums) - 1):
    if nums[i] + nums[i + 1] == t:
        print("[", i, ",", i + 1, "]")
```

Output:



```
nums = list(map(int, intp.split()))
t = int(input("Enter the target element"))
for i in range(len(nums) - 1):
    if nums[i] + nums[i + 1] == t:
        print("[", i, ",", i + 1, "]")
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\saisr\Downloads\assignments\assignment1
Enter the elements separated by spaces1 2 3
Enter the target element3
[ 0 , 1 ]

Process finished with exit code 0
```

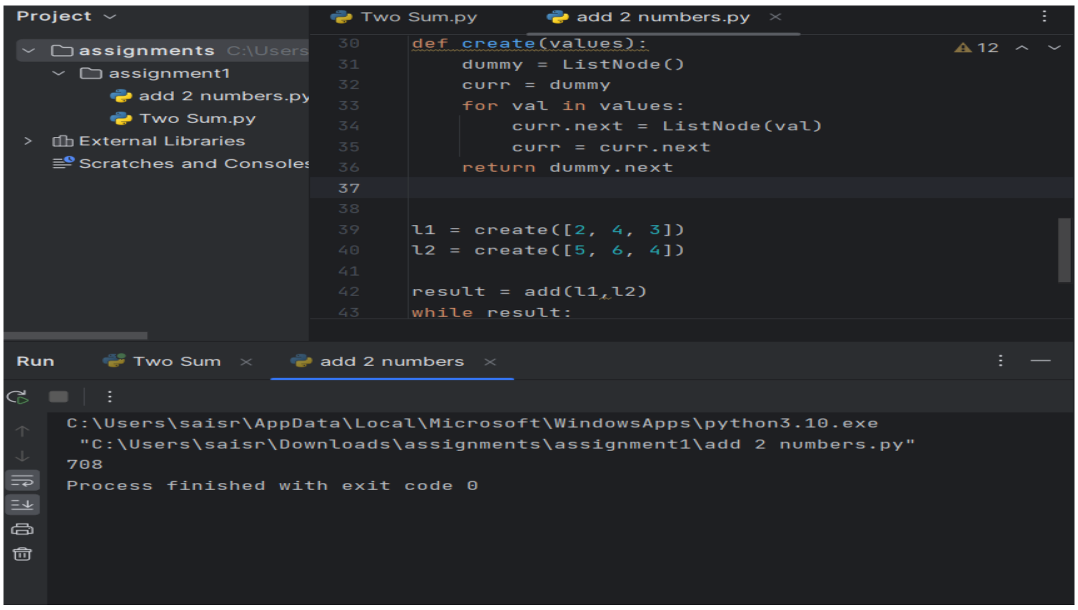
2.Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are
stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and
return the sum as a linked list.
You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Coding

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def add(l1,l2):
    dummy = ListNode()
    curr = dummy
    carry = 0

    while l1 or l2:
        val1 = l1.val if l1 else 0
        val2 = l2.val if l2 else 0

        total = val1 + val2 + carry
        carry = total // 10
        digit = total % 10

        curr.next = ListNode(digit)
        curr = curr.next

        l1 = l1.next if l1 else None
        l2 = l2.next if l2 else None

    if carry:
        curr.next = ListNode(carry)

    return dummy.next

def create(values):
    dummy = ListNode()
    curr = dummy
    for val in values:
        curr.next = ListNode(val)
        curr = curr.next
    return dummy.next


l1 = create([2, 4, 3])
l2 = create([5, 6, 4])

result = add(l1,l2)
while result:
    print(result.val,end="")
    result=result.next
```

Output



```
def create(values):
    dummy = ListNode()
    curr = dummy
    for val in values:
        curr.next = ListNode(val)
        curr = curr.next
    return dummy.next


l1 = create([2, 4, 3])
l2 = create([5, 6, 4])

result = add(l1,l2)
while result:
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment1\add 2 numbers.py"
708
Process finished with exit code 0
```

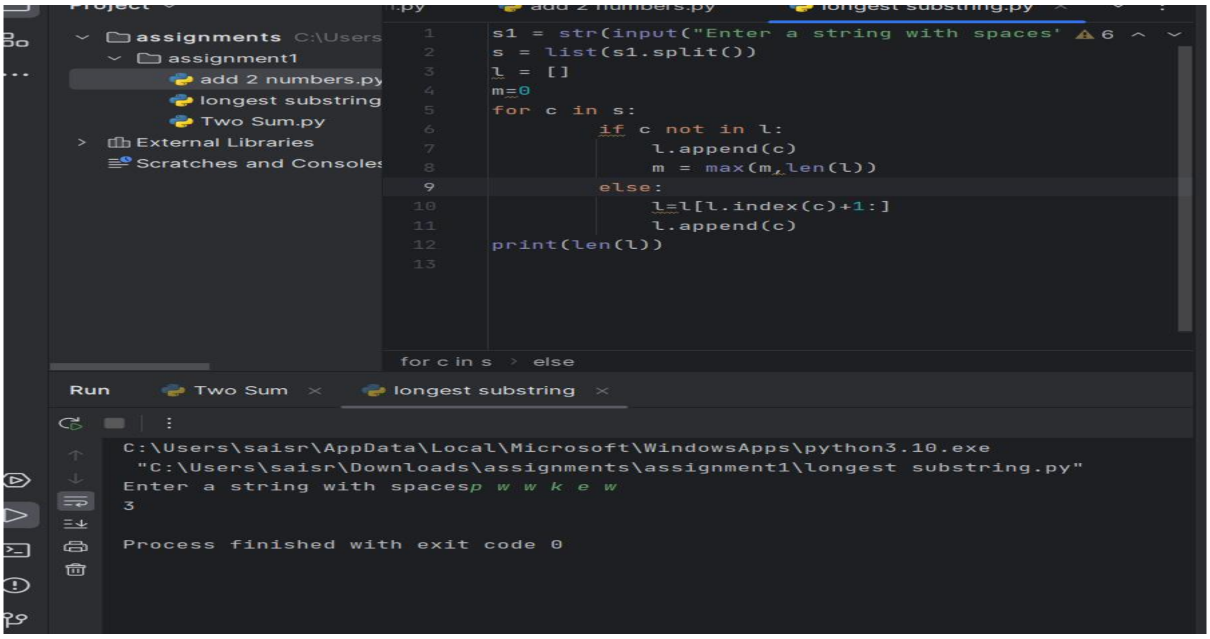
3. Longest Substring without Repeating Characters

Given a string s, find the length of the longest substring without repeating characters.

Coding

```
s1 = str(input("Enter a string with spaces"))
s = list(s1.split())
l = []
m=0
for c in s:
        if c not in l:
            l.append(c)
            m = max(m,len(l))
        else:
            l=l[l.index(c)+1:]
            l.append(c)
print(len(l))
```

Output:



```python
s1 = str(input("Enter a string with spaces'
s = list(s1.split())
l = []
m=0
for c in s:
        if c not in l:
            l.append(c)
            m = max(m,len(l))
        else:
            l=l[l.index(c)+1:]
            l.append(c)
print(len(l))
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment1\longest substring.py"
Enter a string with spacesp w w k e w
3

Process finished with exit code 0
```

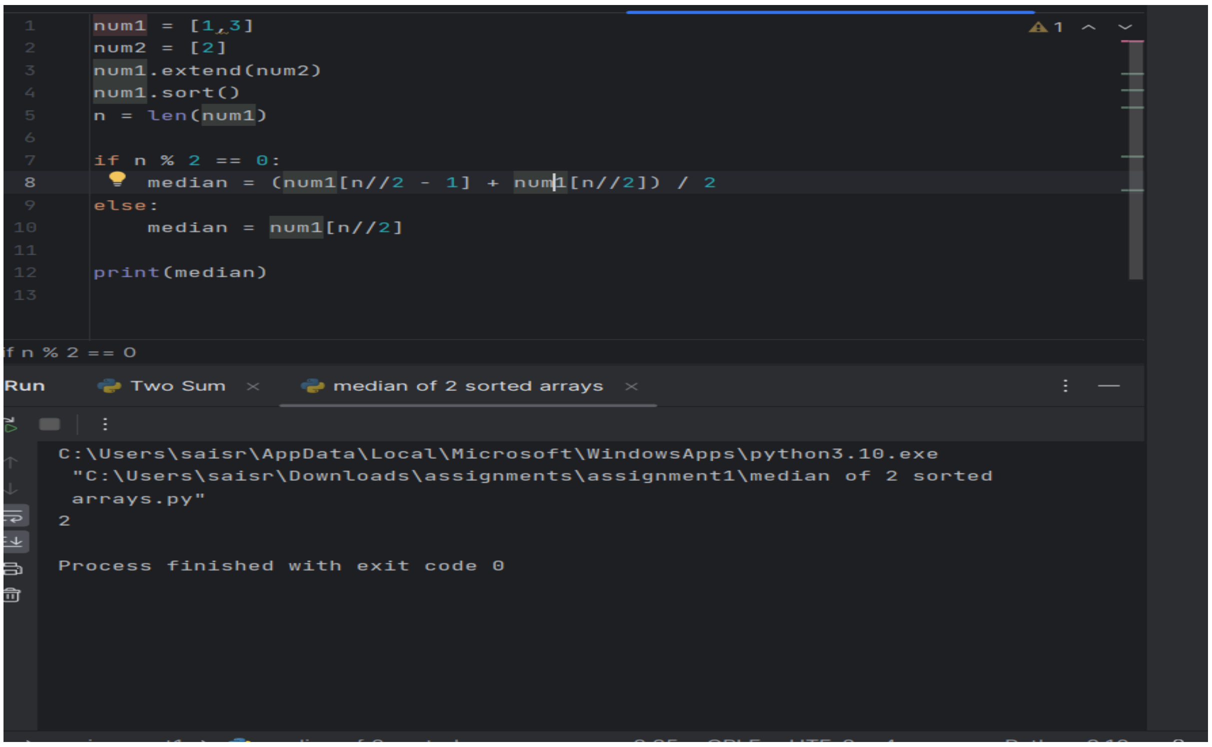
4. Median of Two Sorted Arrays

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the
two sorted arrays.

The overall run time complexity should be O(log (m+n))

Coding:

```python
num1 = [1,3]
num2 = [2]
num1.extend(num2)
num1.sort()
n = len(num1)

if n % 2 == 0:
    median = (num1[n//2 - 1] + num1[n//2]) / 2
else:
    median = num1[n//2]

print(median)
```



```
num1 = [1,3]
num2 = [2]
num1.extend(num2)
num1.sort()
n = len(num1)

if n % 2 == 0:
    median = (num1[n//2 - 1] + num1[n//2]) / 2
else:
    median = num1[n//2]

print(median)
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment1\median of 2 sorted
 arrays.py"
2

Process finished with exit code 0
```

5. Longest Palindromic Substring

Given a string s, return *the longest palindromic substring* in s.

Example 1:

Input: s = "babad"

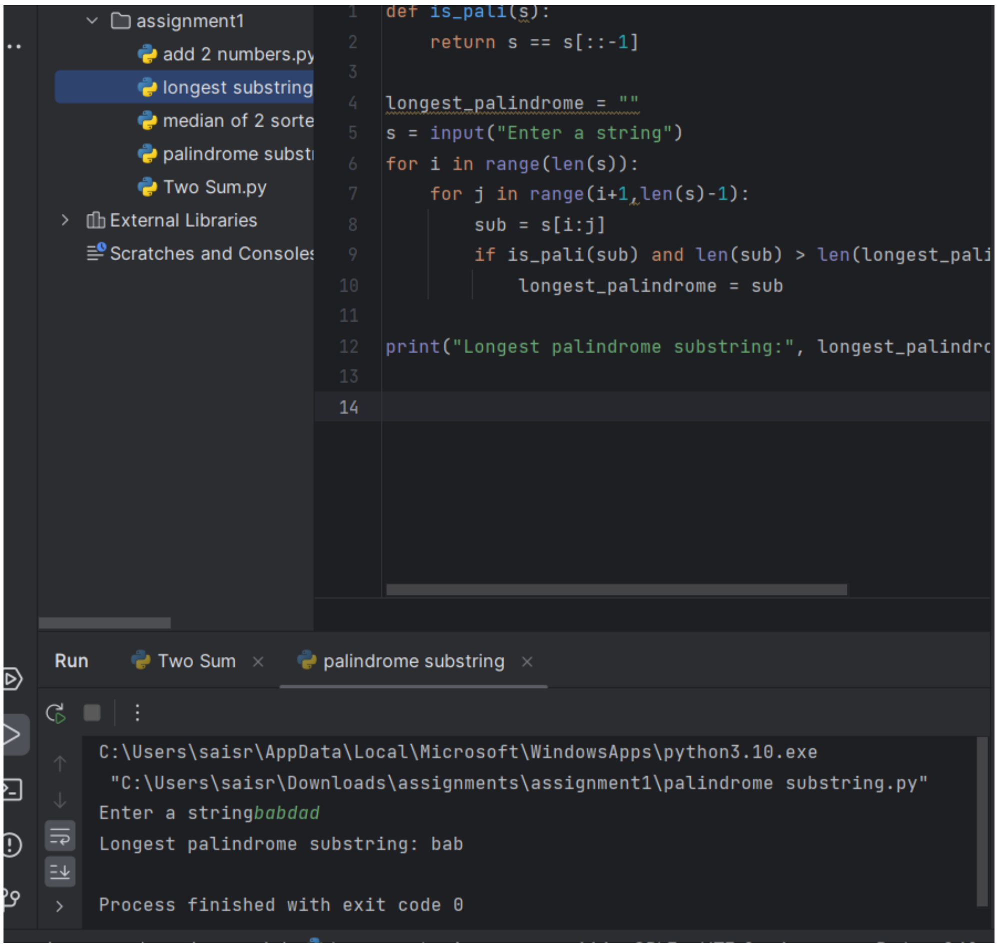
Output: "bab"

Explanation: "aba" is also a valid answer.

Coding

```
def is_pali(s):
    return s == s[::-1]

longest_palindrome = ""
s = input("Enter a string")
for i in range(len(s)):
    for j in range(i+1,len(s)-1):
        sub = s[i:j]
        if is_pali(sub) and len(sub) > len(longest_palindrome):
            longest_palindrome = sub

print("Longest palindrome substring:", longest_palindrome)
```

Output:



```python
def is_pali(s):
    return s == s[::-1]

longest_palindrome = ""
s = input("Enter a string")
for i in range(len(s)):
    for j in range(i+1,len(s)-1):
        sub = s[i:j]
        if is_pali(sub) and len(sub) > len(longest_pali
            longest_palindrome = sub

print("Longest palindrome substring:", longest_palindro
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe
 "C:\Users\saisr\Downloads\assignments\assignment1\palindrome substring.py"
Enter a stringbabdad
Longest palindrome substring: bab

Process finished with exit code 0
```

6. Zigzag Conversion
The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows
like this: (you may want to display this pattern in a fixed font for better legibility)

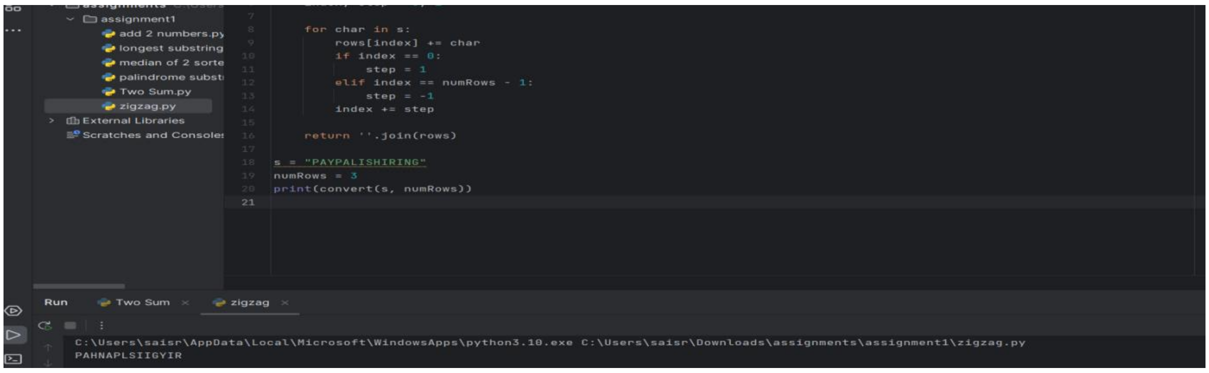
P A H N
A P L S I I G
Y I R

And then read line by line: "PAHNAPLSIIGYIR"
Write the code that will take a string and make this conversion given a number of rows:
string convert(string s, int numRows);

Coding:

```python
def convert(s, numRows):
    if numRows == 1 or numRows >= len(s):
        return s

    rows = [''] * numRows
    index, step = 0, 1

    for char in s:
        rows[index] += char
        if index == 0:
            step = 1
        elif index == numRows - 1:
            step = -1
        index += step

    return ''.join(rows)

s = "PAYPALISHIRING"
numRows = 3
print(convert(s, numRows))
```

Output:



```
    for char in s:
        rows[index] += char
        if index == 0:
            step = 1
        elif index == numRows - 1:
            step = -1
        index += step

    return ''.join(rows)

s = "PAYPALISHIRING"
numRows = 3
print(convert(s, numRows))
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\saisr\Downloads\assignments\assignment1\zigzag.py
PAHNAPLSIIGYIR
```

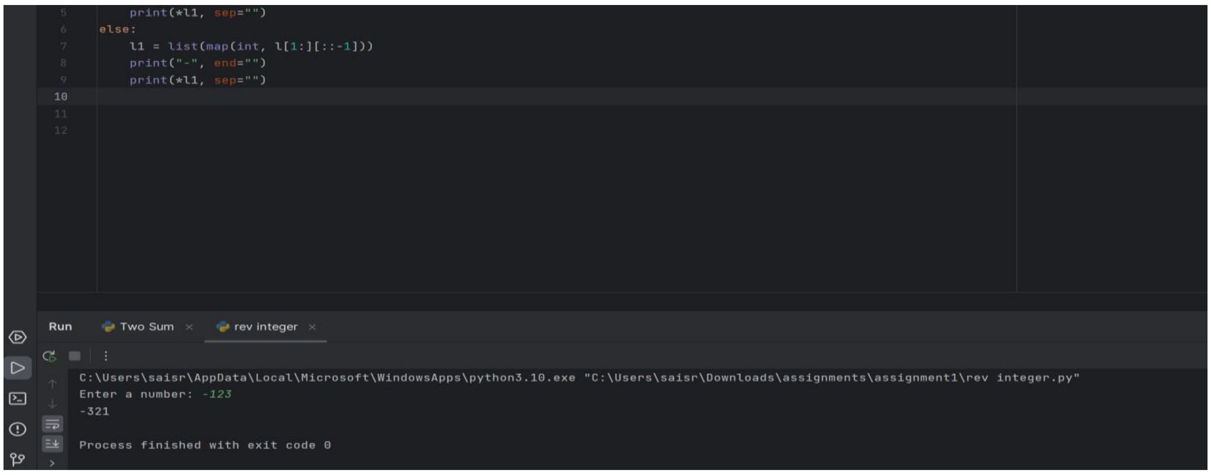
7. Reverse Integer

Given a signed 32-bit integer x, return x *with its digits reversed*. If reversing x causes the value
to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.
Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Coding:

```
x = input("Enter a number: ")
l = list(x)
if l[0] != "-":
    l1 = list(map(int, l[::-1]))
    print(*l1, sep="")
else:
    l1 = list(map(int, l[1:][::-1]))
    print("-", end="")
    print(*l1, sep="")
```

Output:



```
    print(*l1, sep="")
else:
    l1 = list(map(int, l[1:][::-1]))
    print("-", end="")
    print(*l1, sep="")
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\saisr\Downloads\assignments\assignment1\rev integer.py"
Enter a number: -123
-321

Process finished with exit code 0
```

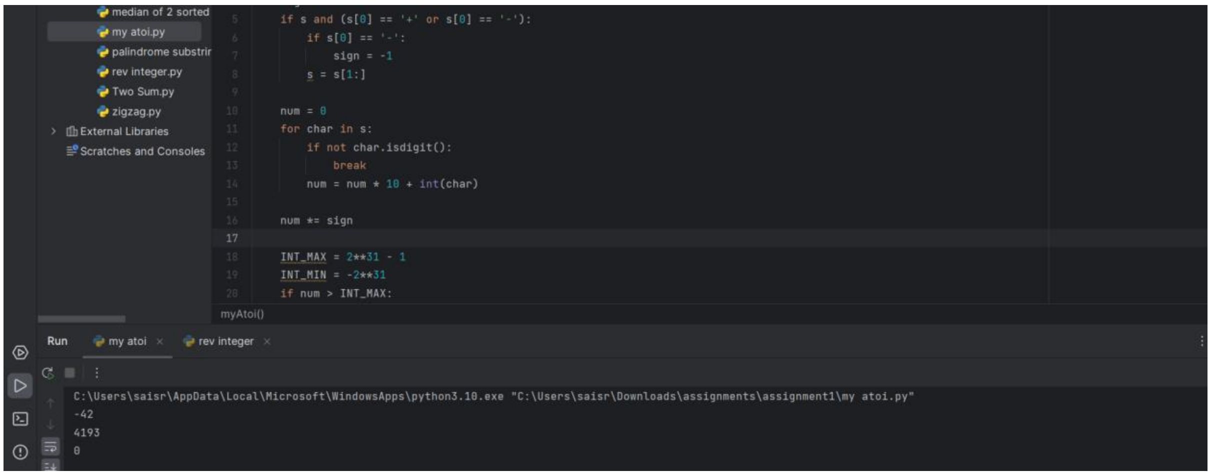
8. String to Integer (atoi)

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

Coding:

```
def myAtoi(s: str) -> int:
    s = s.lstrip()

    sign = 1
    if s and (s[0] == '+' or s[0] == '-'):
        if s[0] == '-':
            sign = -1
        s = s[1:]

    num = 0
    for char in s:
        if not char.isdigit():
            break
        num = num * 10 + int(char)

    num *= sign

    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    if num > INT_MAX:
        return INT_MAX
    elif num < INT_MIN:
        return INT_MIN
    else:
        return num

s = "-42"
print(myAtoi(s))

s = "4193 with words"
print(myAtoi(s))

s = "words and 987"
print(myAtoi(s))  # Output: 0
```

Output:



```
    if s and (s[0] == '+' or s[0] == '-'):
        if s[0] == '-':
            sign = -1
        s = s[1:]

    num = 0
    for char in s:
        if not char.isdigit():
            break
        num = num * 10 + int(char)

    num *= sign

    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    if num > INT_MAX:
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\saisr\Downloads\assignments\assignment1\my atoi.py"
-42
4193
0
```

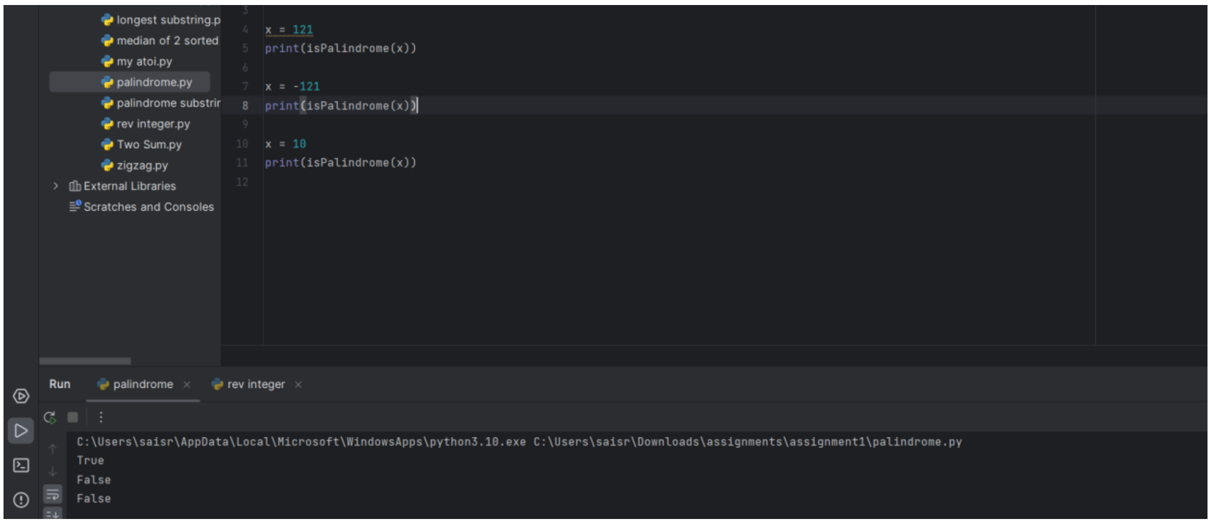
9. Palindrome Number

Given an integer x, return true *if* x *is a palindrome, and* false *otherwise*

Coding

```
def isPalindrome(x: int) -> bool:
    return str(x) == str(x)[::-1]

x = 121
print(isPalindrome(x))

x = -121
print(isPalindrome(x))

x = 10
print(isPalindrome(x))
```

Output:



```
x = 121
print(isPalindrome(x))

x = -121
print(isPalindrome(x))

x = 10
print(isPalindrome(x))
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\saisr\Downloads\assignments\assignment1\palindrome.py
True
False
False
```

10. Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for
'.' and '*' where:

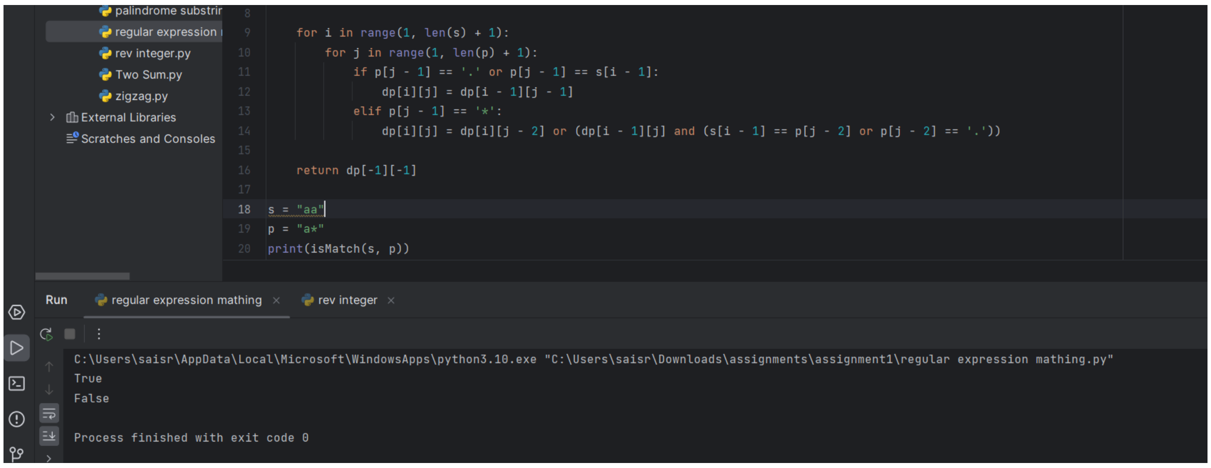
- '.' Matches any single character.
- '*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial)

Coding:

```python
def isMatch(s: str, p: str) -> bool:
    dp = [[False] * (len(p) + 1) for _ in range(len(s) + 1)]
    dp[0][0] = True

    for j in range(1, len(p) + 1):
        if p[j - 1] == '*':
            dp[0][j] = dp[0][j - 2]

    for i in range(1, len(s) + 1):
        for j in range(1, len(p) + 1):
            if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            elif p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] and (s[i - 1] ==
p[j - 2] or p[j - 2] == '.'))

    return dp[-1][-1]

s = "aa"
p = "a*"
print(isMatch(s, p))

s = "mississippi"
p = "mis*is*p*."
print(isMatch(s, p))
```

Output:



```python

    for i in range(1, len(s) + 1):
        for j in range(1, len(p) + 1):
            if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            elif p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] and (s[i - 1] == p[j - 2] or p[j - 2] == '.'))

    return dp[-1][-1]

s = "aa"
p = "a*"
print(isMatch(s, p))
```

```
C:\Users\saisr\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\saisr\Downloads\assignments\assignment1\regular expression mathing.py"
True
False

Process finished with exit code 0
```