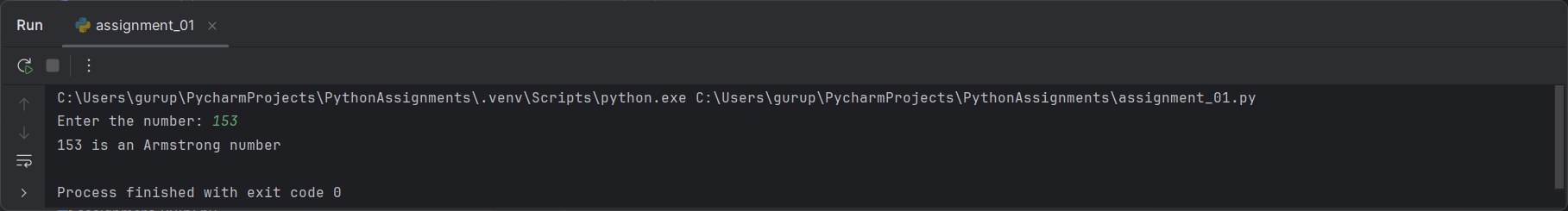
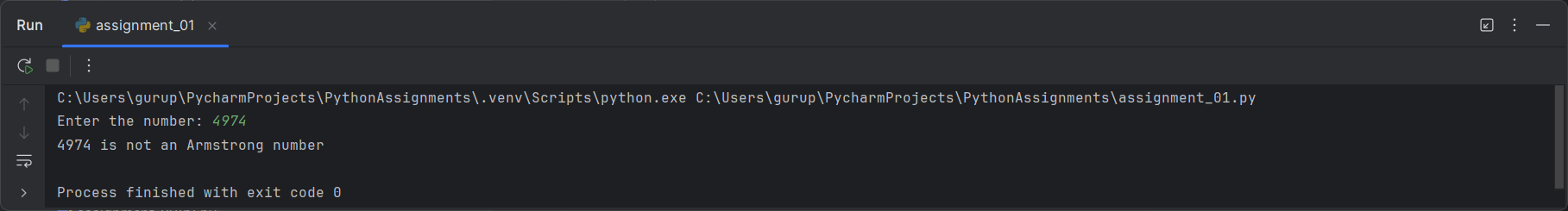
**Assignment Question 1:** Check whether input number is Armstrong number or not.

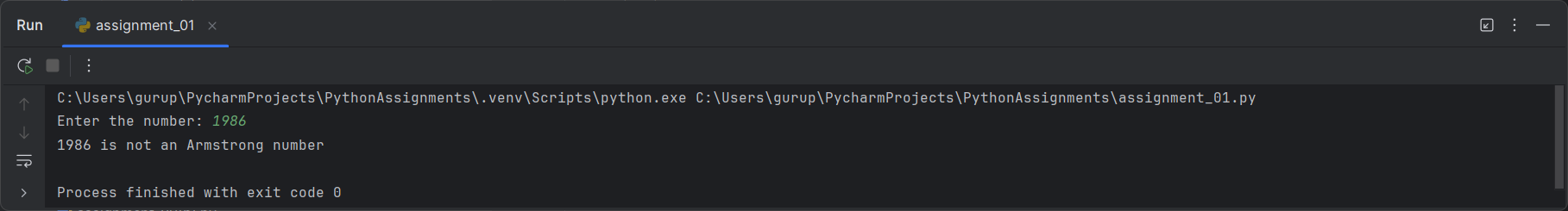
**Programme:**

num = input("Enter the number: ")  
s = 0  
n = int(num)  
  
while n > 0:  
 r = n % 10  
 s += r \*\* len(num)  
 n //= 10  
  
if int(num) == s:  
 print(num, "is an Armstrong number")  
else:  
 print(num, "is not an Armstrong number")

**Output:**

****

****

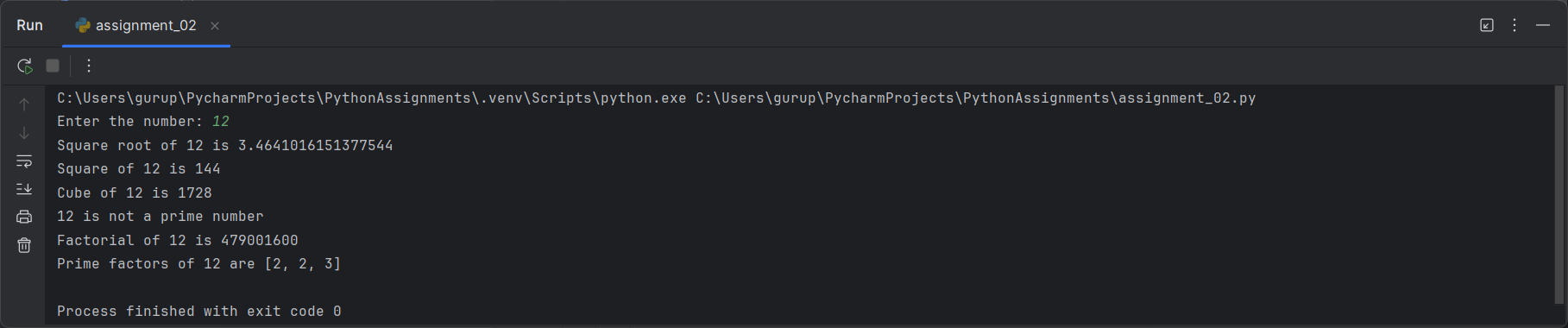
****

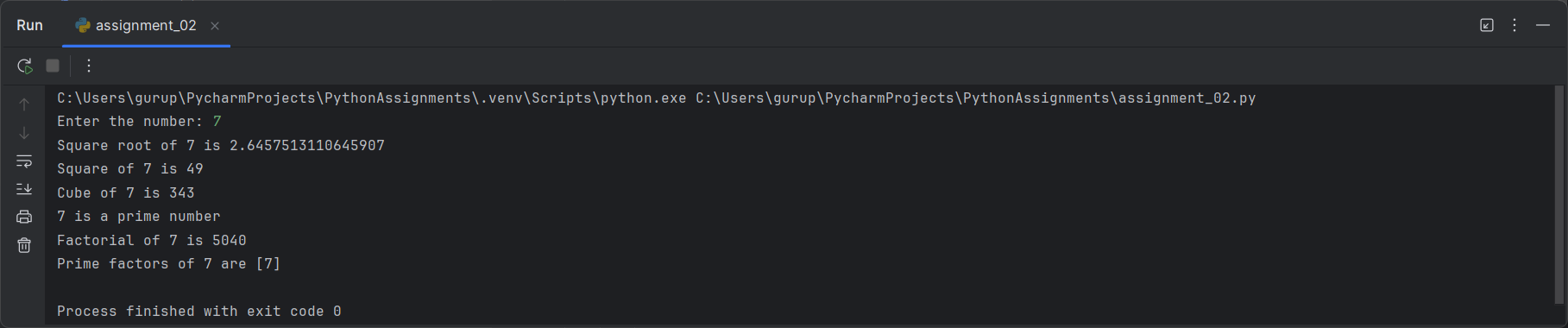
**Assignment Question 2:** Accept the number and Compute a) square root of number, b) Square of number, c) Cube of number d) check for prime, d) factorial of number e) prime factors.

**Programme:**

import math  
  
num = int(input("Enter the number: "))  
  
# a) Square root of number  
sqrt = math.sqrt(num)  
print("Square root of", num, "is", sqrt)  
  
# b) Square of number  
sqr = num \*\* 2  
print("Square of", num, "is", sqr)  
  
# c) Cube of number  
cube = num \*\* 3  
print("Cube of", num, "is", cube)  
  
# d) Check for prime  
if num == 0 or num == 1:  
 print(num, "is not a prime number")  
elif num > 1:  
 for i in range(2, num):  
 if (num % i) == 0:  
 print(num, "is not a prime number")  
 break  
 else:  
 print(num, "is a prime number")  
  
# d) Factorial of number  
fact = 1  
  
if num == 0 or num == 1:  
 print("Factorial of", num, "is 1")  
elif num > 1:  
 for i in range(2, num + 1):  
 fact = fact \* i  
 print("Factorial of", num, "is", fact)  
  
# f) Prime factors  
factors = []  
n = num  
  
for i in range(2, n + 1):  
 while n % i == 0:  
 factors.append(i)  
 n //= i  
print("Prime factors of", num, "are", factors)

**Output:**



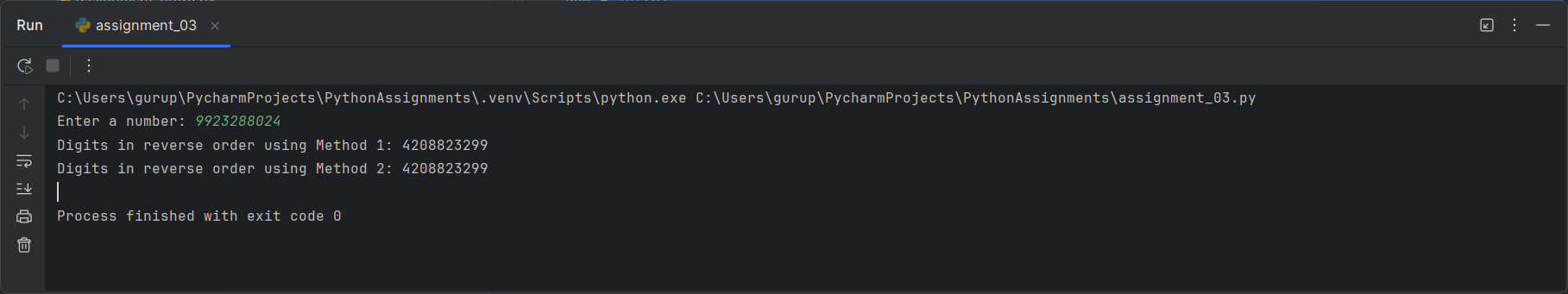
****

**Assignment Question 3:** Accept a number from the user and print digits of number in reverse order.

**Programme:**

# Accept input from the user  
n = input("Enter a number: ")  
  
# Method 1: Using string slicing function  
reversed\_n = n[::-1]  
print(f"Digits in reverse order using Method 1: {reversed\_n}")  
  
# Method 2: Using loops  
reversed\_num = 0  
num = int(n)  
  
while num > 0:  
 rem = num % 10  
 reversed\_num = (reversed\_num \* 10) + rem  
 num //= 10  
print(f"Digits in reverse order using Method 2: {reversed\_num}")

**Output:**

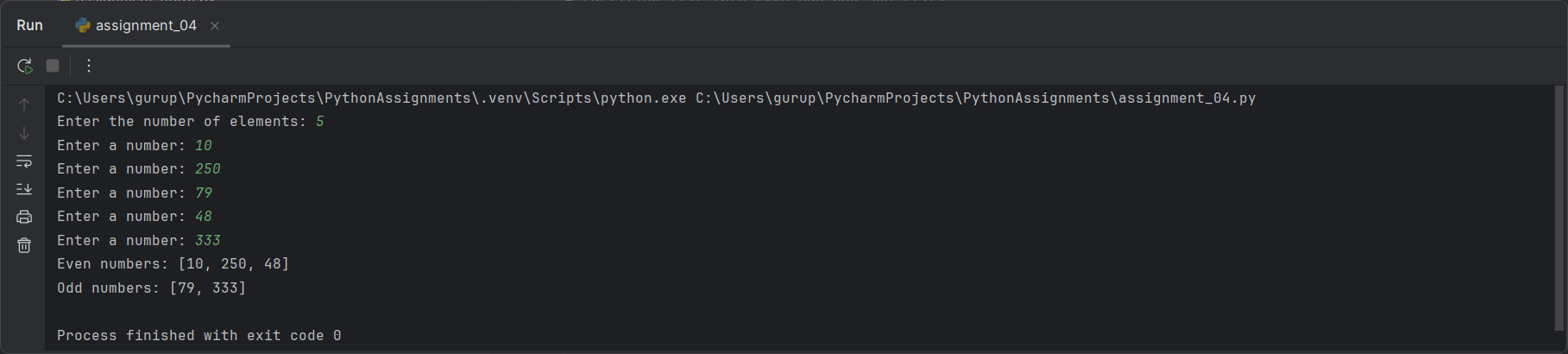
****

**Assignment Question 4:** Accept list of N integers and partition list into sub-lists of even and odd numbers.

**Programme:**

# Accept list of N integers from the user  
N = int(input("Enter the number of elements: "))  
numbers = []  
  
# Accept N integers and append to the list  
for \_ in range(N):  
 number = int(input("Enter a number: "))  
 numbers.append(number)  
  
# Partition list into even and odd sub-lists  
even\_numbers = [num for num in numbers if num % 2 == 0]  
odd\_numbers = [num for num in numbers if num % 2 != 0]  
  
# Output the sub-lists  
print("Even numbers:", even\_numbers)  
print("Odd numbers:", odd\_numbers)

**Output:**

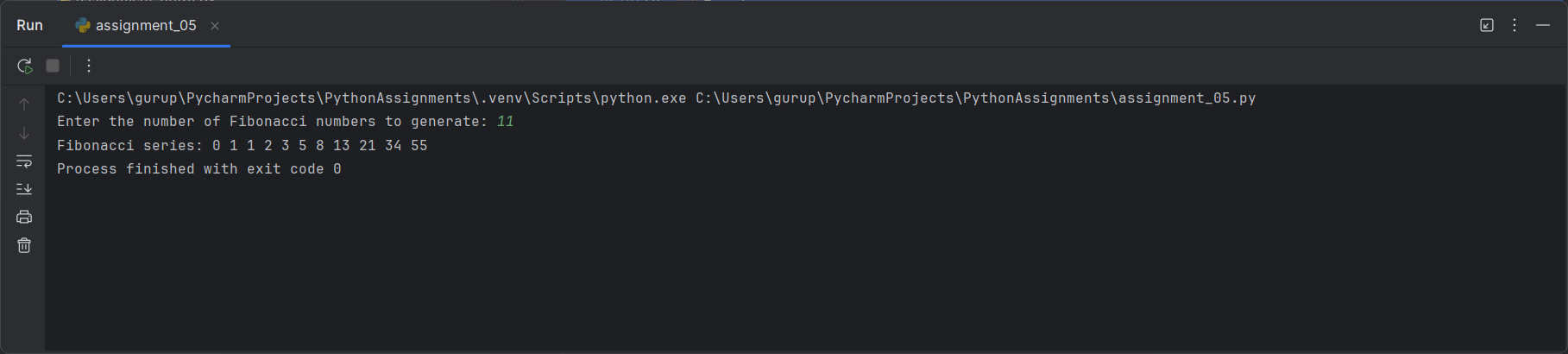
****

**Assignment Question 5:** Accept from the user the number of Fibonacci numbers to be generated and print the Fibonacci series.

**Programme:**

# Accept the number of Fibonacci terms from the user  
n = int(input("Enter the number of Fibonacci numbers to generate: "))  
  
# Initializing the first two Fibonacci numbers  
a, b = 0, 1  
  
# Print Fibonacci series  
print("Fibonacci series: ", end="")  
for \_ in range(n):  
 print(a, end=" ")  
 a, b = b, a + b # Update a and b for the next term

**Output:**

****

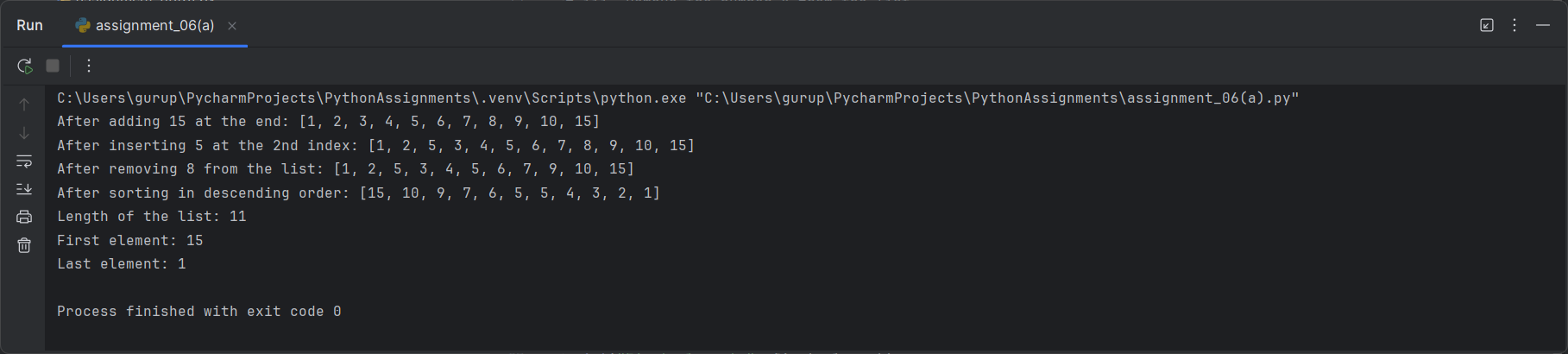
**Assignment Question 6A:** Create a list of integers from 1 to 10 and perform:

1. Add the number 15 at the end of the list.
2. Insert the number 5 at the 2nd index.
3. Remove the number 8 from the list.
4. Sort the list in descending order.
5. Find the length of the list.
6. Retrieve the first and last elements of the list.

**Programme:**

# Create a list of integers from 1 to 10  
my\_list = list(range(1, 11))  
  
# i. Add the number 15 at the end of the list  
my\_list.append(15)  
print("After adding 15 at the end:", my\_list)  
  
# ii. Insert the number 5 at the 2nd index  
my\_list.insert(2, 5)  
print("After inserting 5 at the 2nd index:", my\_list)  
  
# iii. Remove the number 8 from the list  
my\_list.remove(8)  
print("After removing 8 from the list:", my\_list)  
  
# iv. Sort the list in descending order  
my\_list.sort(reverse=True)  
print("After sorting in descending order:", my\_list)  
  
# v. Find the length of the list  
length\_of\_list = len(my\_list)  
print("Length of the list:", length\_of\_list)  
  
# vi. Retrieve the first and last elements of the list  
first\_element = my\_list[0]  
last\_element = my\_list[-1]  
print("First element:", first\_element)  
print("Last element:", last\_element)

**Output:**

****

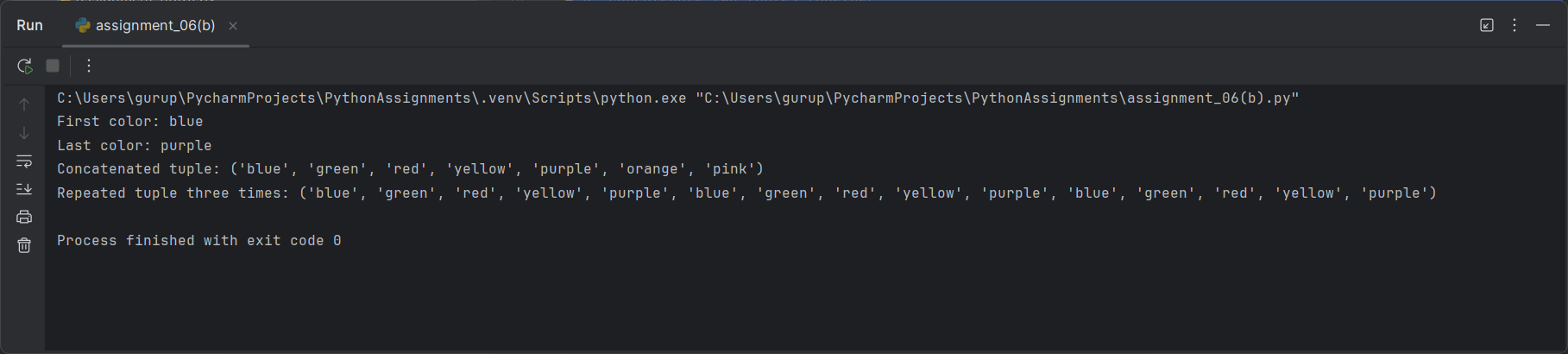
**Assignment Question 6B:** Create a tuple of your favorite colors and perform:

1. Access the first and last elements of the tuple.
2. Concatenate two tuples together.
3. Repeat a tuple three times and display the result.

**Programme:**

# Create a tuple of favorite colors  
favorite\_colors = ("blue", "green", "red", "yellow", "purple")  
  
# a. Access the first and last elements of the tuple  
first\_color = favorite\_colors[0]  
last\_color = favorite\_colors[-1]  
print("First color:", first\_color)  
print("Last color:", last\_color)  
  
# b. Concatenate two tuples together  
another\_colors = ("orange", "pink")  
concatenated\_tuple = favorite\_colors + another\_colors  
print("Concatenated tuple:", concatenated\_tuple)  
  
# c. Repeat the tuple three times and display the result  
repeated\_tuple = favorite\_colors \* 3  
print("Repeated tuple three times:", repeated\_tuple)

**Output:**

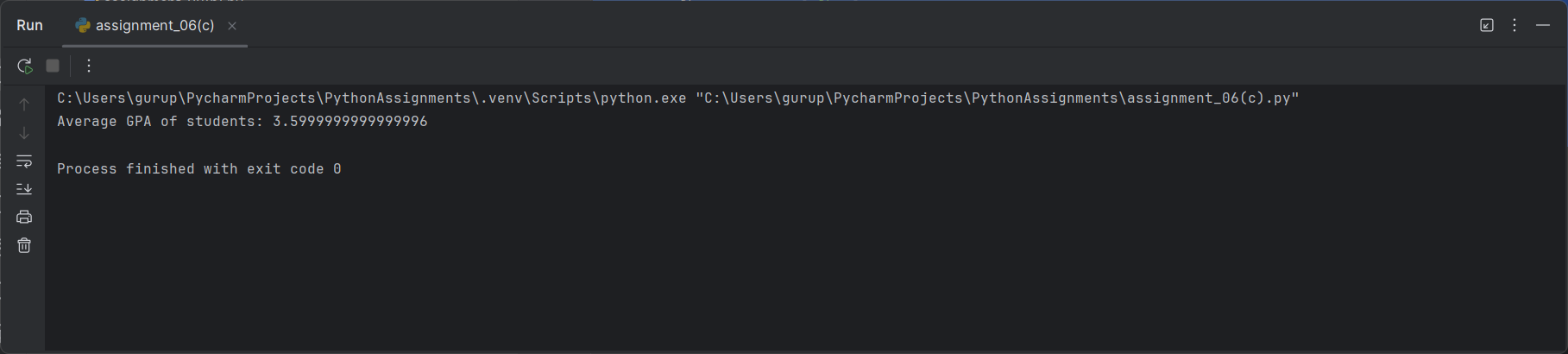
****

**Assignment Question 6C:** Write a program that takes a list of dictionaries where each dictionary represents a student with keys name, age, gpa. The program should compute the average GPA of the students.

**Programme:**

# List of dictionaries where each dictionary represents a student  
students = [  
 {"name": "Alice", "age": 20, "gpa": 3.5},  
 {"name": "Bob", "age": 22, "gpa": 3.8},  
 {"name": "Charlie", "age": 21, "gpa": 3.9},  
 {"name": "David", "age": 23, "gpa": 3.2}  
]  
  
# Compute the average GPA  
total\_gpa = 0  
num\_students = len(students)  
  
# Summing the GPAs of all students  
for student in students:  
 total\_gpa += student["gpa"]  
  
# Calculating the average GPA  
average\_gpa = total\_gpa / num\_students if num\_students > 0 else 0  
  
# Print the average GPA  
print("Average GPA of students:", average\_gpa)

**Output:**

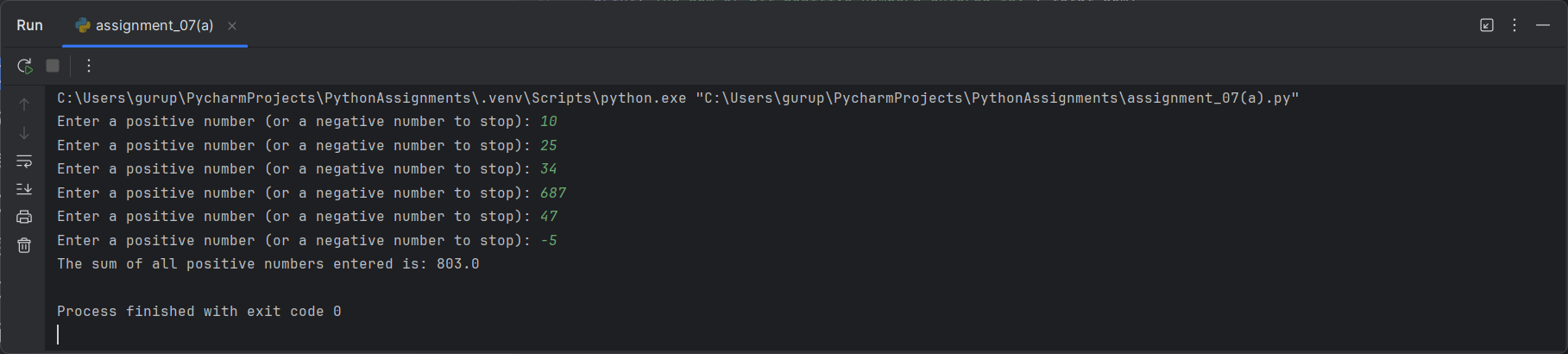
****

**Assignment Question 7A:** Write a program that uses a loop to repeatedly ask the user to enter positive numbers. The loop will come to an end when a negative number is entered. After collecting all the positive numbers, the program will compute their sum and display the result to the user.

**Programme:**

# Initialize the sum of positive numbers to 0  
total\_sum = 0  
  
# Start a loop to ask for positive numbers  
while True:  
 # Ask the user for input  
 number = float(input("Enter a positive number (or a negative number to stop): "))  
  
 # Check if the number is negative, then break the loop  
 if number < 0:  
 break  
  
 # Add the positive number to the total sum  
 total\_sum += number  
  
# Display the sum of positive numbers  
print("The sum of all positive numbers entered is:", total\_sum)

**Output:**



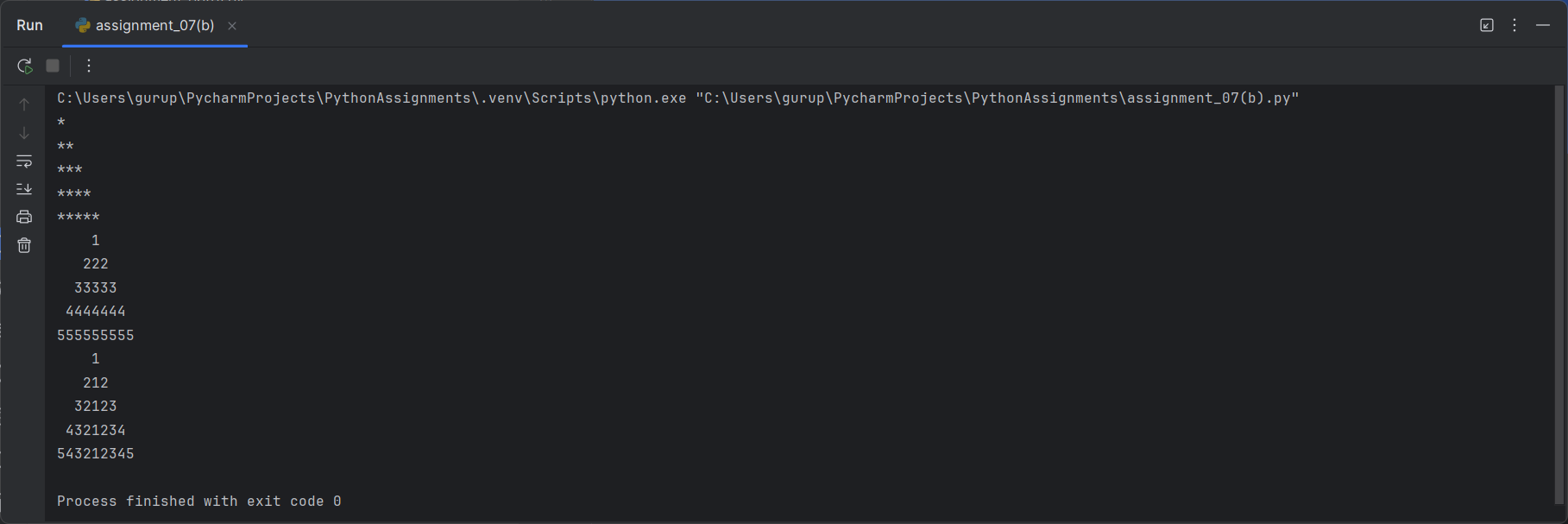
**Assignment Question 7B:** Write programs to print following patterns:

**Programme:**

# Left Pattern  
n = 5  
for i in range(1, n + 1):  
 print("\*" \* i)  
  
# Middle Pattern  
n = 5 # Number of rows  
for i in range(1, n + 1):  
 print(" " \* (n - i) + (str(i) \* (2 \* i - 1)))  
  
# Right Pattern  
n = 5 # Number of rows  
for i in range(1, n + 1):  
 left = "".join(str(j) for j in range(i, 0, -1)) # Decreasing part  
 right = "".join(str(j) for j in range(2, i + 1)) # Increasing part  
 print(" " \* (n - i) + left + right)

**Output:**

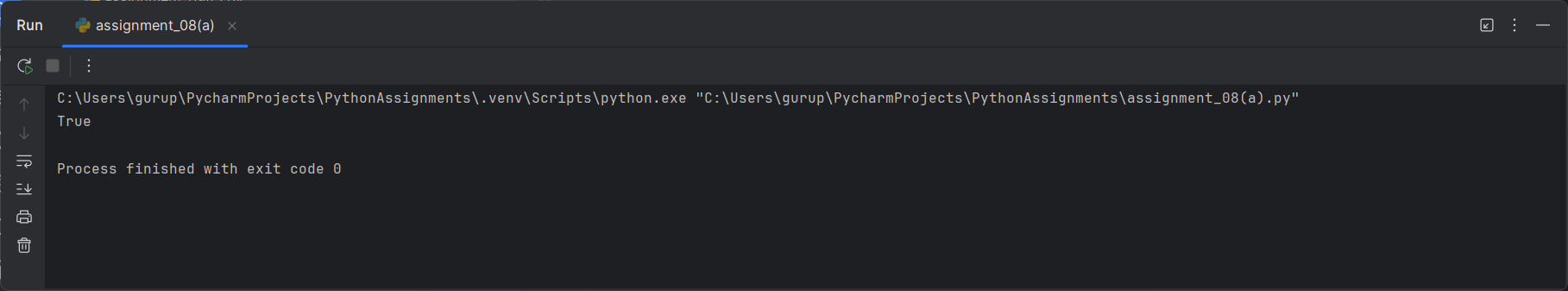


**Assignment Question 8A:** Write a function ball\_collide that takes two balls as parameters and computes if they are colliding. Your function should return a boolean representing whether or not the balls are 29 colliding. 9 hint: represent a ball on a plane as a tuple of (x, y, r), r being the radius if (distance between two balls centers) <= (sum of their radii) then (they are colliding)

**Programme:**

import math  
  
  
def ball\_collide(ball1, ball2):  
 # Unpack the balls into (x, y, r) format  
 x1, y1, r1 = ball1  
 x2, y2, r2 = ball2  
  
 # Calculate the distance between the centers of the balls  
 distance = math.sqrt((x2 - x1) \* 2 + (y2 - y1) \* 2)  
  
 # Check if the distance is less than or equal to the sum of the radii  
 return distance <= (r1 + r2)  
  
  
# Example usage  
ball1 = (0, 0, 5) # Ball 1 at (0, 0) with radius 5  
ball2 = (3, 4, 3) # Ball 2 at (3, 4) with radius 3  
print(ball\_collide(ball1, ball2)) # Output: True or False

**Output:**

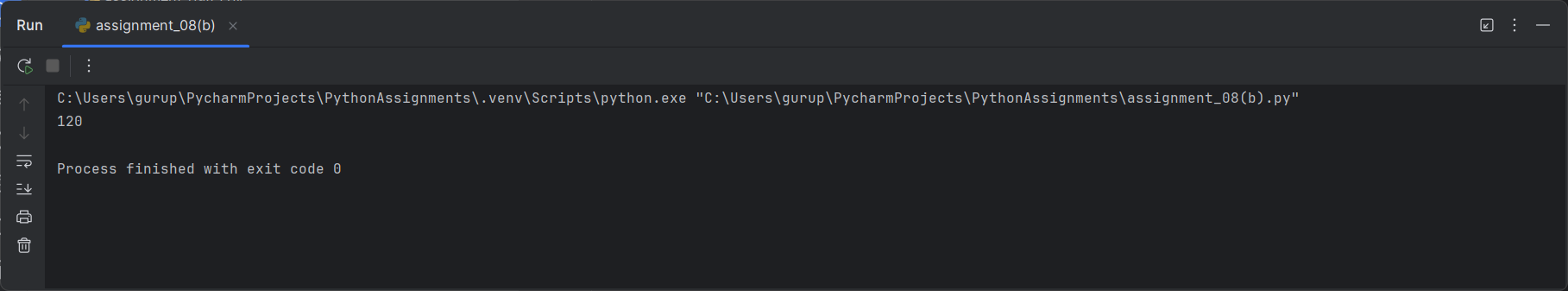


**Assignment Question 8B:** Implement a python Program for factorial of number by using recursion.

**Programme:**

def factorial(n):  
 if n == 1:  
 return 1  
 else:  
 return n \* factorial(n - 1)  
  
  
# Example usage  
print(factorial(5)) # Output: 120

**Output:**

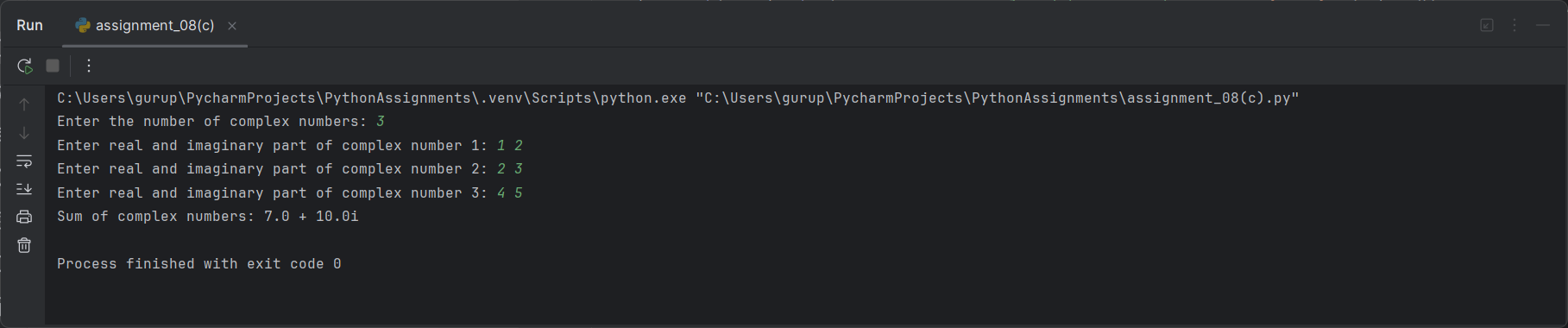


**Assignment Question 8C:** Define a function which takes TWO objects representing complex numbers and returns new complex number with an addition of two complex number. Define a suitable class ‘Complex’ to represent the complex number. develop a program to read N (N>=2) complex numbers and to compute the addition of N complex numbers.

**Programme:**

class Complex:  
 def \_\_init\_\_(self, real, imag):  
 self.real = real  
 self.imag = imag  
  
 def \_\_add\_\_(self, other):  
 return Complex(self.real + other.real, self.imag + other.imag)  
  
 def \_\_str\_\_(self):  
 return f"{self.real} + {self.imag}i"  
  
  
# Read N complex numbers and compute their sum  
N = int(input("Enter the number of complex numbers: "))  
total = Complex(0, 0)  
  
for i in range(N):  
 r, i = map(float, input(f"Enter real and imaginary part of complex number {i + 1}: ").split())  
 total += Complex(r, i)  
  
print("Sum of complex numbers:", total)

**Output:**



**Assignment Question 9A:** Write a python program to define a module to find Fibonacci Numbers and import the module to another program.

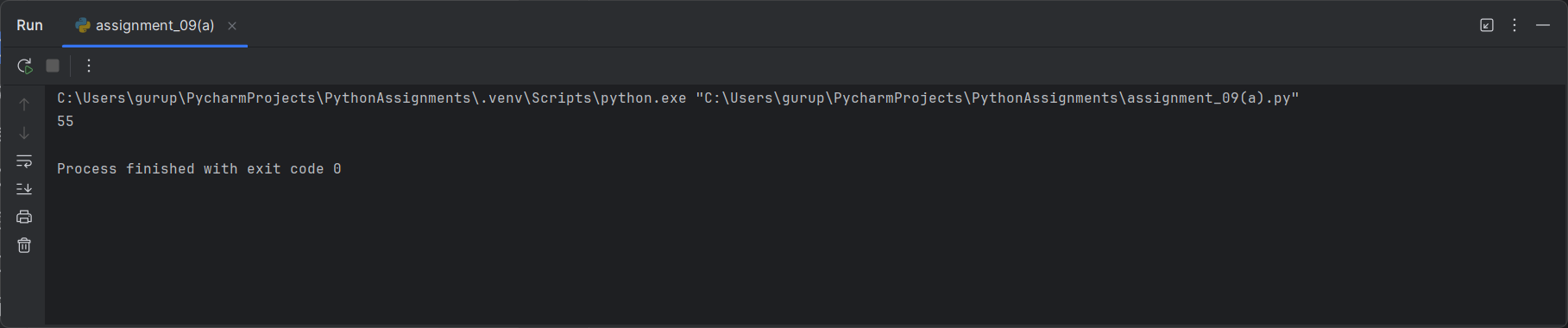
**Programme:**

**fibonacci.py module**

def fibonacci(n):  
 if n <= 1:  
 return n  
 else:  
 return fibonacci(n - 1) + fibonacci(n - 2)  
  
**main.py**

import fibonacci  
  
n = 10 # Example: Find the 10th Fibonacci number  
print(fibonacci.fibonacci(n)) # Output: 55

**Output:**

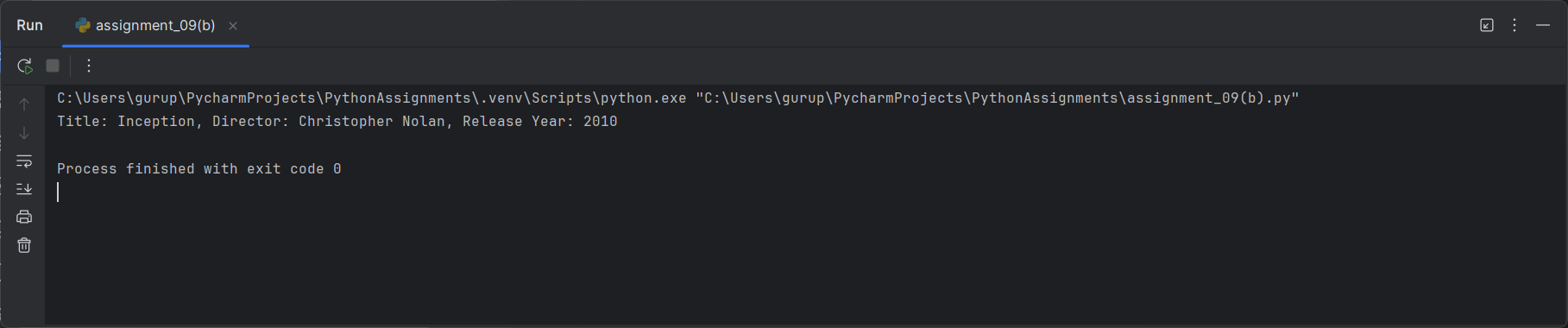


**Assignment Question 9B:** Create a class Movie with attributes title, director, and release\_year. The class should have An \_\_init\_\_() method to initialize the attributes.A method get\_movie\_details() that returns a string with the movie’s title, director, and release year.Create an instance of Movie with the title "Inception", director "Christopher Nolan", and release year 2010.Call the get\_movie\_details() method and print the result.

**Programme:**

class Movie:  
 def \_\_init\_\_(self, title, director, release\_year): # Corrected the constructor  
 self.title = title  
 self.director = director  
 self.release\_year = release\_year  
  
 def get\_movie\_details(self):  
 return f"Title: {self.title}, Director: {self.director}, Release Year: {self.release\_year}"  
  
  
# Example usage  
movie = Movie("Inception", "Christopher Nolan", 2010)  
print(movie.get\_movie\_details())

**Output:**

****

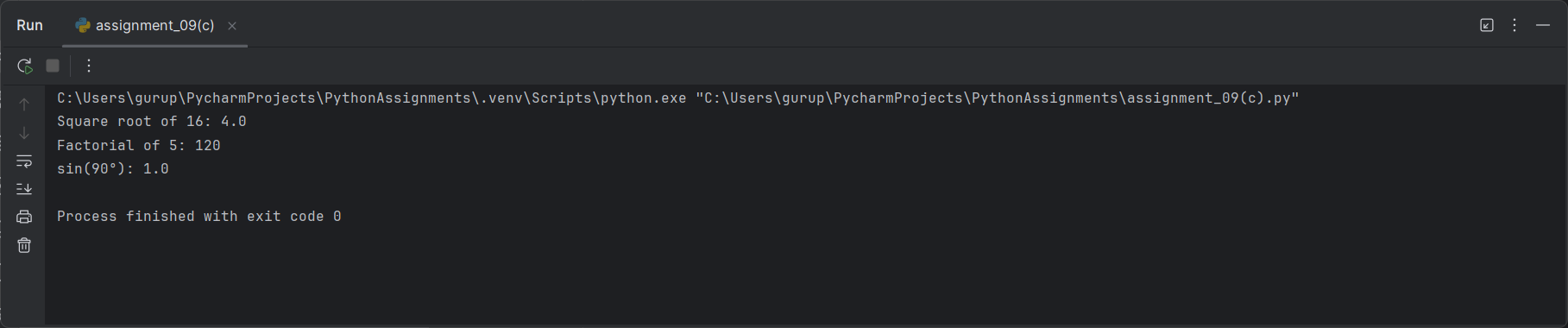
**Assignment Question 9C:** Write a program using the math module to calculate:

* Square root of a number
* Factorial of a number
* Value of sin(90°)

**Programme:**

import math  
  
# Square root of a number  
sqrt\_result = math.sqrt(16)  
print("Square root of 16:", sqrt\_result) # Output: 4.0  
  
# Factorial of a number  
factorial\_result = math.factorial(5)  
print("Factorial of 5:", factorial\_result) # Output: 120  
  
# Value of sin(90°)  
sin\_result = math.sin(math.radians(90))  
print("sin(90°):", sin\_result) # Output: 1.0

**Output:**

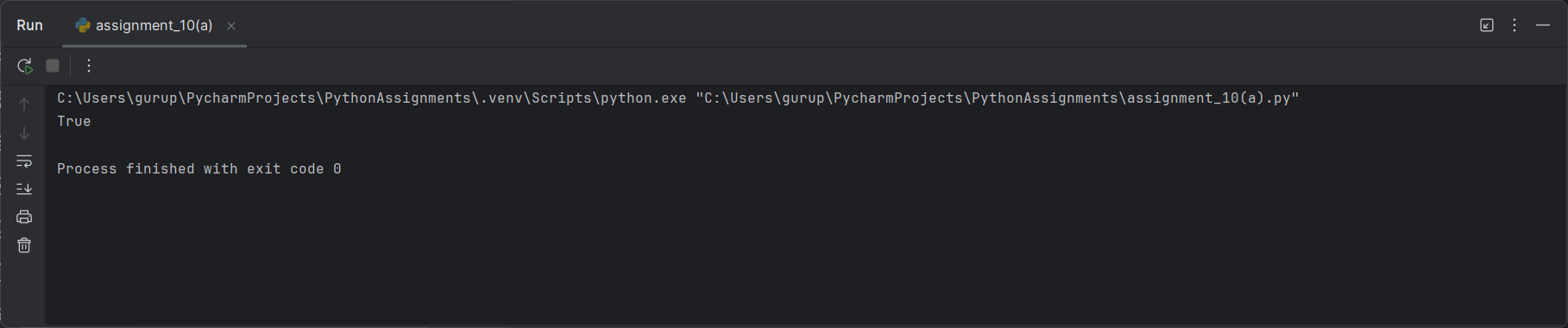


**Assignment Question 10A:** Write a program to validate an email address using RegEx.

**Programme:**

import re  
  
  
def validate\_email(email):  
 pattern = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'  
 return re.match(pattern, email) is not None  
  
  
# Example usage  
email = "example@domain.com"  
print(validate\_email(email)) # Output: True

**Output:**

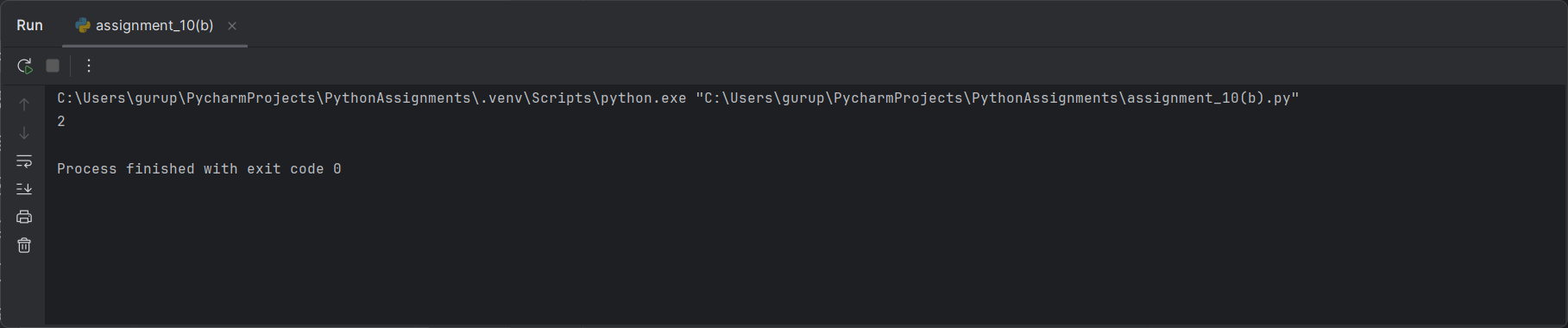
****

**Assignment Question 10B:** Write a program to find and count all occurrences of a pattern in a string using re.findall().

**Programme:**

import re  
  
  
def count\_pattern\_occurrences(text, pattern):  
 return len(re.findall(pattern, text))  
  
  
# Example usage  
text = "Hello world! Hello everyone!"  
pattern = "Hello"  
print(count\_pattern\_occurrences(text, pattern)) # Output: 2

**Output:**

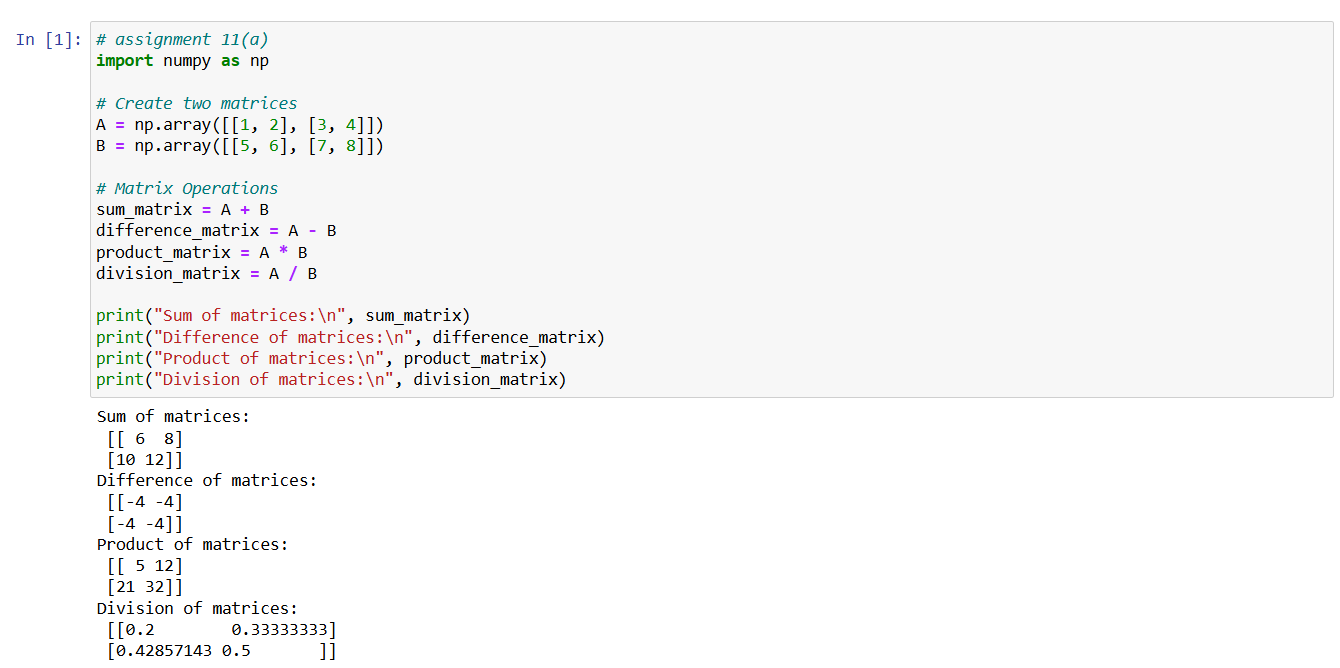


**Assignment Question 11A:** Write a Python numpy program to create two matrices and perform addition, subtraction, multiplication and division operation on matrix.

**Programme:**

import numpy as np  
  
# Create two matrices  
A = np.array([[1, 2], [3, 4]])  
B = np.array([[5, 6], [7, 8]])  
  
# Matrix Operations  
sum\_matrix = A + B  
difference\_matrix = A - B  
product\_matrix = A \* B  
division\_matrix = A / B  
  
print("Sum of matrices:\n", sum\_matrix)  
print("Difference of matrices:\n", difference\_matrix)  
print("Product of matrices:\n", product\_matrix)  
print("Division of matrices:\n", division\_matrix)

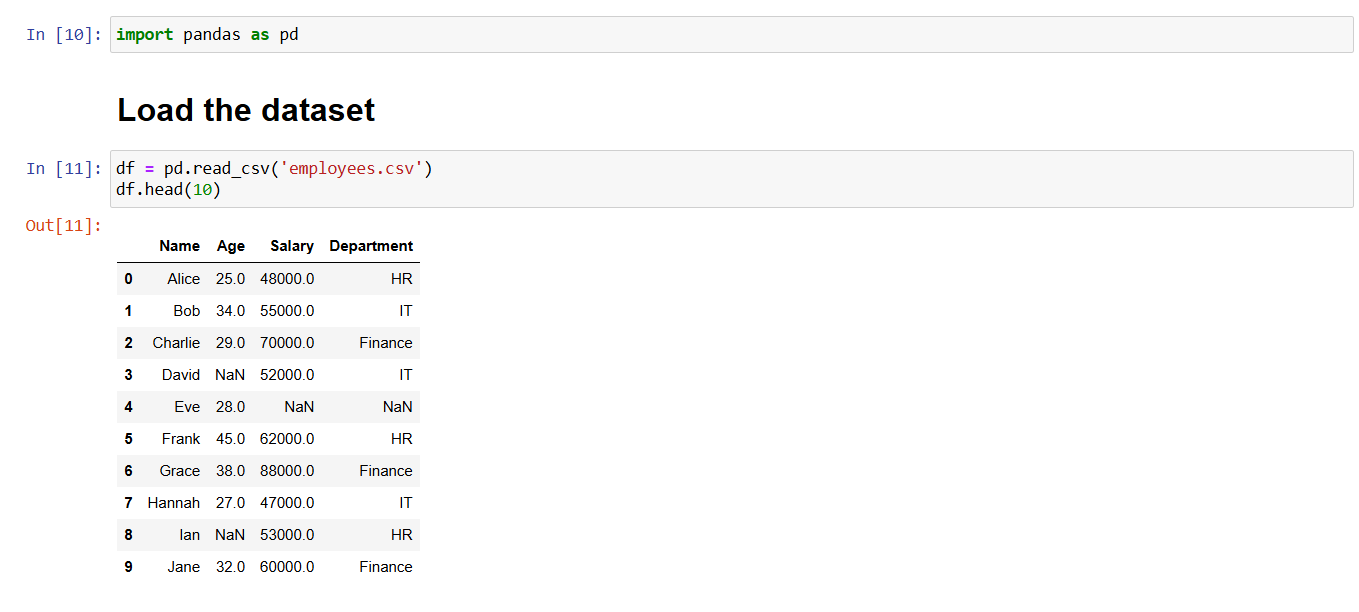
**Output:**

****

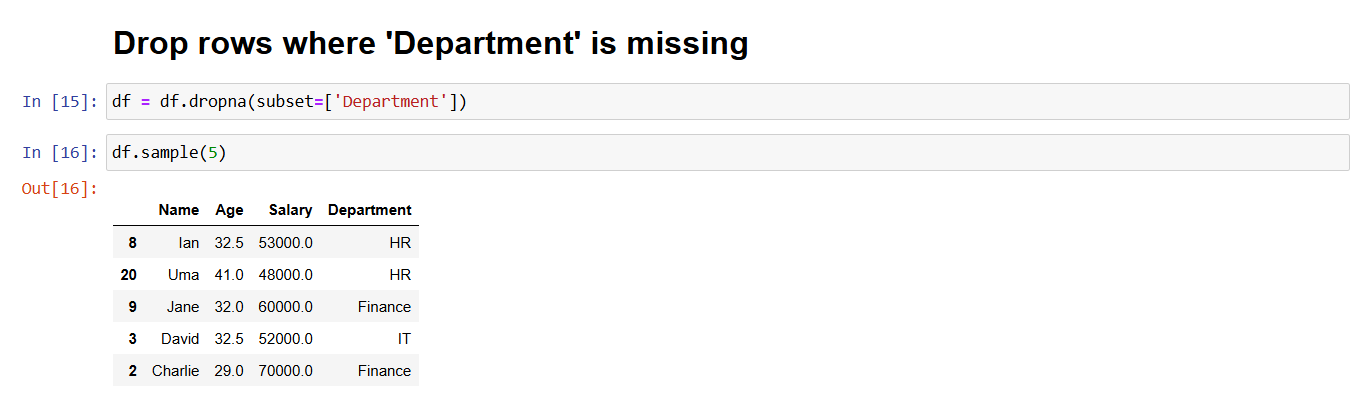
**Assignment Question 11B:** Load the dataset "employees.csv"( https://gist.github.com/kevin336/acbb2271e66c10a5b73aacf82ca82784#file-employees-csv) into a Pandas DataFrame.

1. Display the first 10 rows.
2. The dataset contains columns Age, Salary, and Department. Some rows have missing values. Write a script to:
3. Replace missing values in Age with the mean age.
4. Replace missing values in Salary with the median salary.
5. Drop rows where Department is missing.
6. Create a new column in a DataFrame that categorizes employees based on their salary:
7. Low (Salary < 50,000)
8. Medium (50,000 ≤ Salary < 100,000)
9. High (Salary ≥ 100,000)
10. Group the DataFrame by the Department column and calculate the average salary for each department.

**Programme and Output:**











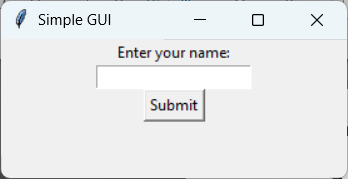
**Assignment Question 12A:** Create a Python GUI program with the following widgets:

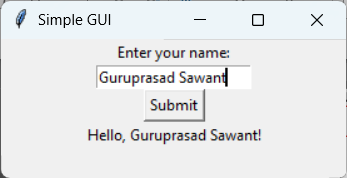
1. A Label to display "Enter your name".
2. An Entry widget for user input.
3. A Button to submit the input.

**Programme:**

import tkinter as tk  
  
  
def submit\_input():  
 name = entry.get()  
 label\_result.config(text=f"Hello, {name}!")  
  
  
# Create the main window  
window = tk.Tk()  
window.title("Simple GUI")  
  
# Create widgets  
label = tk.Label(window, text="Enter your name:")  
entry = tk.Entry(window)  
button = tk.Button(window, text="Submit", command=submit\_input)  
label\_result = tk.Label(window, text="")  
  
# Layout the widgets  
label.pack()  
entry.pack()  
button.pack()  
label\_result.pack()  
  
# Run the GUI  
window.mainloop()

**Output:**

****

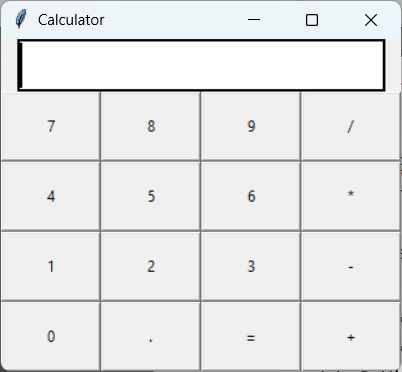
****

**Assignment Question 12B:** Build a calculator GUI that supports basic arithmetic operations (+, -, \*, /) using buttons and an Entry widget to display results.

**Programme:**

import tkinter as tk  
  
  
def on\_button\_click(value):  
 current = entry.get()  
 entry.delete(0, tk.END)  
 entry.insert(0, current + value)  
  
  
def on\_calculate():  
 try:  
 result = eval(entry.get())  
 entry.delete(0, tk.END)  
 entry.insert(0, result)  
 except Exception:  
 entry.delete(0, tk.END)  
 entry.insert(0, "Error")  
  
  
# Create main window  
window = tk.Tk()  
window.title("Calculator")  
  
# Entry widget for displaying expressions  
entry = tk.Entry(window, width=16, font=('Arial', 24), borderwidth=2, relief='solid')  
entry.grid(row=0, column=0, columnspan=4)  
  
# Button definitions  
buttons = [  
 ('7', 1, 0), ('8', 1, 1), ('9', 1, 2), ('/', 1, 3),  
 ('4', 2, 0), ('5', 2, 1), ('6', 2, 2), ('\*', 2, 3),  
 ('1', 3, 0), ('2', 3, 1), ('3', 3, 2), ('-', 3, 3),  
 ('0', 4, 0), ('.', 4, 1), ('=', 4, 2), ('+', 4, 3)  
]  
  
# Adding buttons to the grid  
for (text, row, col) in buttons:  
 button = tk.Button(window, text=text, width=10, height=3,  
 command=lambda t=text: on\_button\_click(t) if t != '=' else on\_calculate())  
 button.grid(row=row, column=col)  
  
# Run the main event loop  
window.mainloop()

**Output:**



**Assignment Question 13:** Using a dataset of your choice, create:

1. Plot a line chart showing the trend of a company's revenue over the years. Label the axes and add a title.
2. A scatterplot showing the relationship between two numerical columns (e.g., Salary and Experience).
3. A heatmap to display the correlation matrix of the numerical columns.

**Programme:**

# assignment 13  
import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
  
# Load the dataset (for demonstration, we'll use a sample dataset)  
df = pd.DataFrame({  
 'Year': [2015, 2016, 2017, 2018, 2019],  
 'Revenue': [50, 55, 60, 65, 70],  
 'Salary': [30000, 35000, 40000, 45000, 50000],  
 'Experience': [1, 2, 3, 4, 5]  
})  
  
# Line chart (Revenue over years)  
plt.plot(df['Year'], df['Revenue'], marker='o')  
plt.title("Company Revenue Over Years")  
plt.xlabel("Year")  
plt.ylabel("Revenue")  
plt.show()  
  
# Scatter plot (Salary vs Experience)  
plt.scatter(df['Experience'], df['Salary'])  
plt.title("Salary vs Experience")  
plt.xlabel("Experience")  
plt.ylabel("Salary")  
plt.show()  
  
# Heatmap (Correlation matrix)  
sns.heatmap(df.corr(), annot=True)  
plt.title("Correlation Matrix")  
plt.show()

**Output:**

