**Experiment 1**

**Aim:** To show uses of the operators in Python.

**Code:**

a = 40

b = 70

# arithemetic operators

print("Arithematic operators")

print("+: ", a+b)

print("-: ", a-b)

print("\*: ", a\*b)

print("/: ", a/b)

print("\*: ", a\*2)

print("//: ", a//b)

print("+: ", a+b)

print("%: ", a % b)

print("\n")

# relational operators

print("relational operators")

print(a > b)

print(a >= b)

print(a <= b)

print(a < b)

print(a == b)

print(a != b)

print("\n")

# logical operators

print("logical operators")

print(a and b)

print(a or b)

print(not a)

print("\n")

# bitwise operators

print("bitwise operators")

print(a & b)

print(a ^ b)

print(a | b)

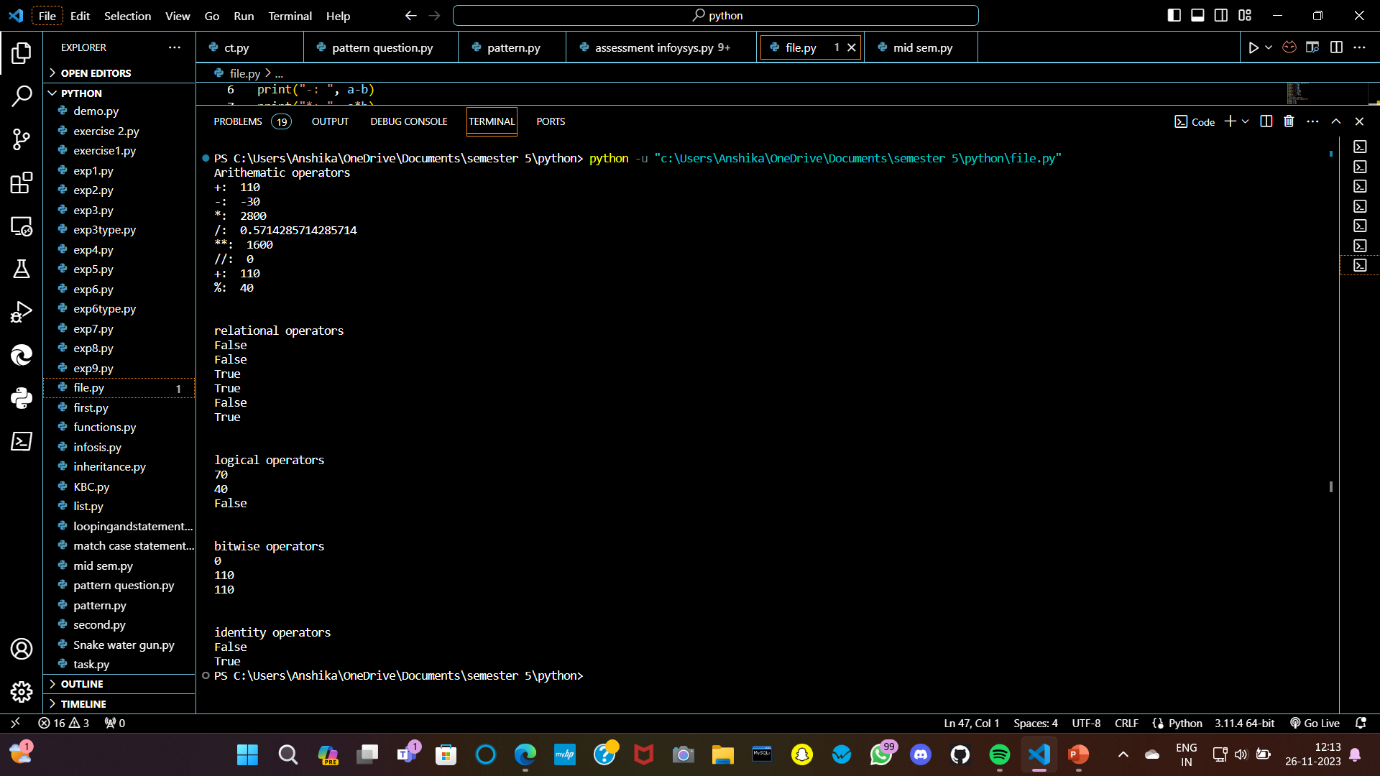
print("\n")

# identity operators

print("identity operators")

print(a is b)

print(a is not b)

**Output:**

**Experiment 2**

**Aim:** To display the star pattern using for loops.

**Code:**

num = int(input("enter a number"))

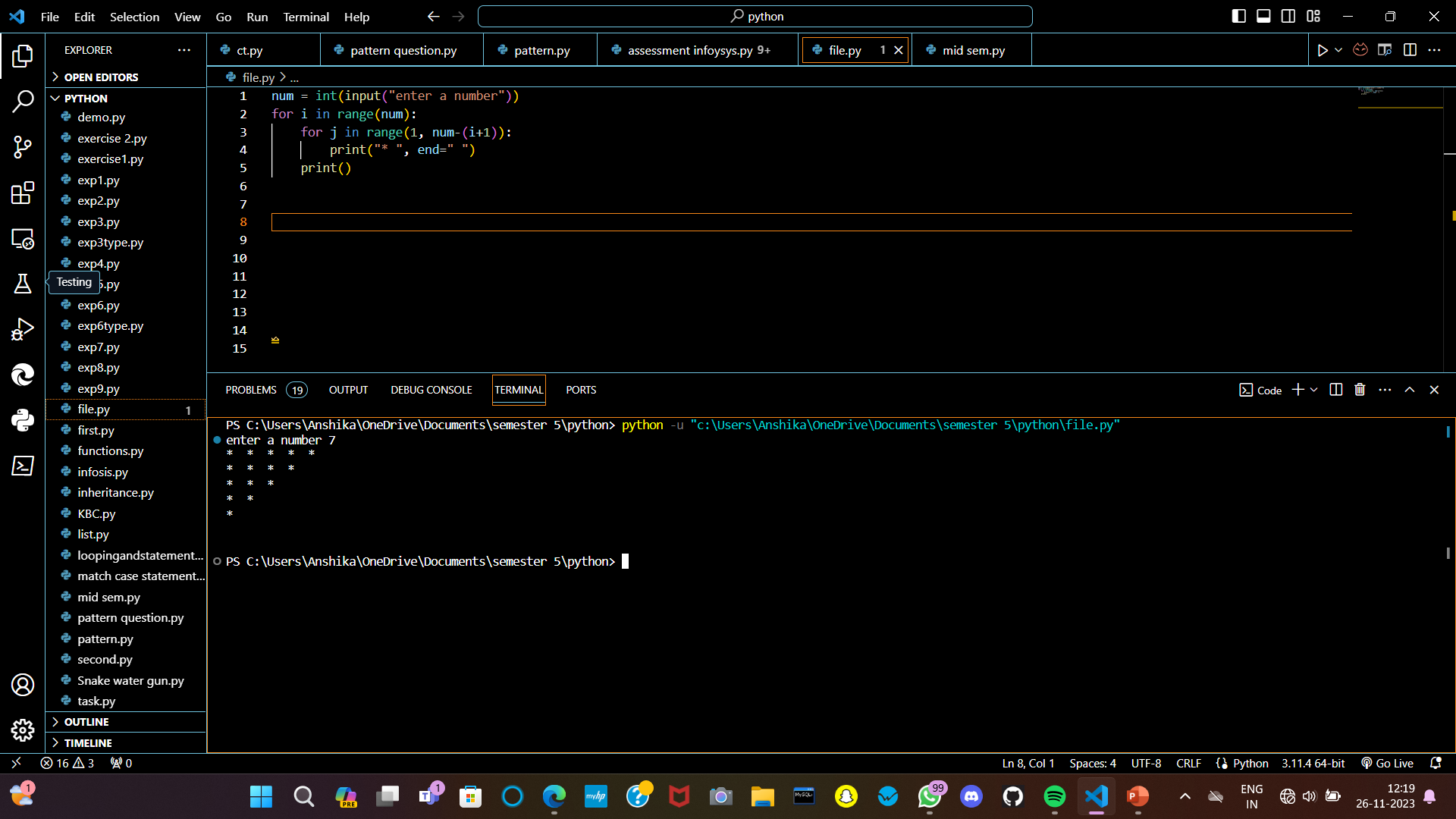
for i in range(num):

for j in range(1, num-(i+1)):

print("\* ", end=" ")

print()

**Output:**

****

**Experiment 3**

**Aim:** To perform using operations using inbuilt methods.

**Code:**

a = " Hello Gautam Buddha University"

print("String operations\n")

print("Original String : ", a)

print("Coverts all to lowercase : ", a.lower())

print("converts all to uppercase : ", a.upper())

print("Swaps the cases of alphabets : ", a.swapcase())

print("Checks whether the string is alphabet or not : ", a.isalpha())

print("Checks whether the string is digit or not : ", a.isdigit())

print("First index of the letter in the given string : ", a.index("r"))

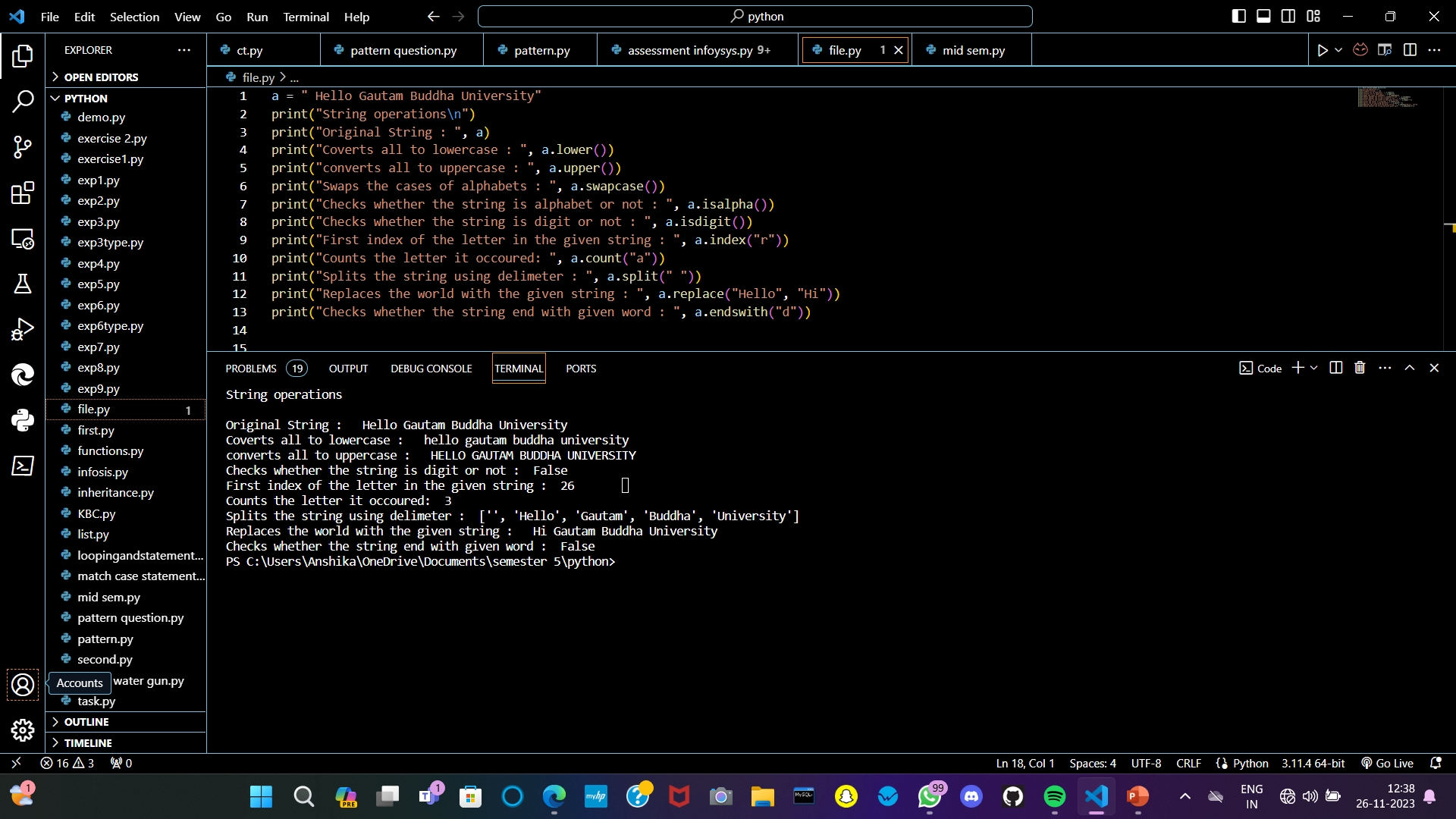
print("Counts the letter it occoured: ", a.count("a"))

print("Splits the string using delimeter : ", a.split(" "))

print("Replaces the world with the given string : ", a.replace("Hello", "Hi"))

print("Checks whether the string end with given word : ", a.endswith("d"))

**Output:**

****

**Experiment 4**

**Aim:** Program to print list in six different ways.

**Code:**

# Python program to print list

a = [1, 2, 3, 4, 5]

# printing the list using loop

for x in range(len(a)):

print(a[x])

print("\n")

# using the sep parameter in print()

# printing the list using \* operator separated by comma

print("using the sep parameter")

print(\*a)

# printing the list using \* and sep operator

print("printing lists separated by commas")

print(\*a, sep = ", ")

print("printing lists in new line")

print(\*a, sep = "\n")

print("\n")

# convert a list to a string for display

print("convert a list to astring for display")

a =["Geeks", "for", "Geeks"]

# print the list using join function()

print(' '.join(a))

# print the list by converting a list of

# integers to string

a = [1, 2, 3, 4, 5]

print(str(a)[1:-1])

print("\n")

# using map() function

print("using map() function")

a = [1, 2, 3, 4, 5]

print(' '.join(map(str, a)))

print("in new line")

print('\n'.join(map(str, a)))

print("\n")

# using list comprehension

print("use list comprehension")

a = [1, 2, 3, 4, 5]

[print(i, end=' ') for i in a]

print("\nIn new line")

[print(i) for i in a]

print("\n")

# using indexing and slicing

print('using indexing and slicing')

list = [1, 2, 3, 4, 5, 6]

#method 1

print(list[:])

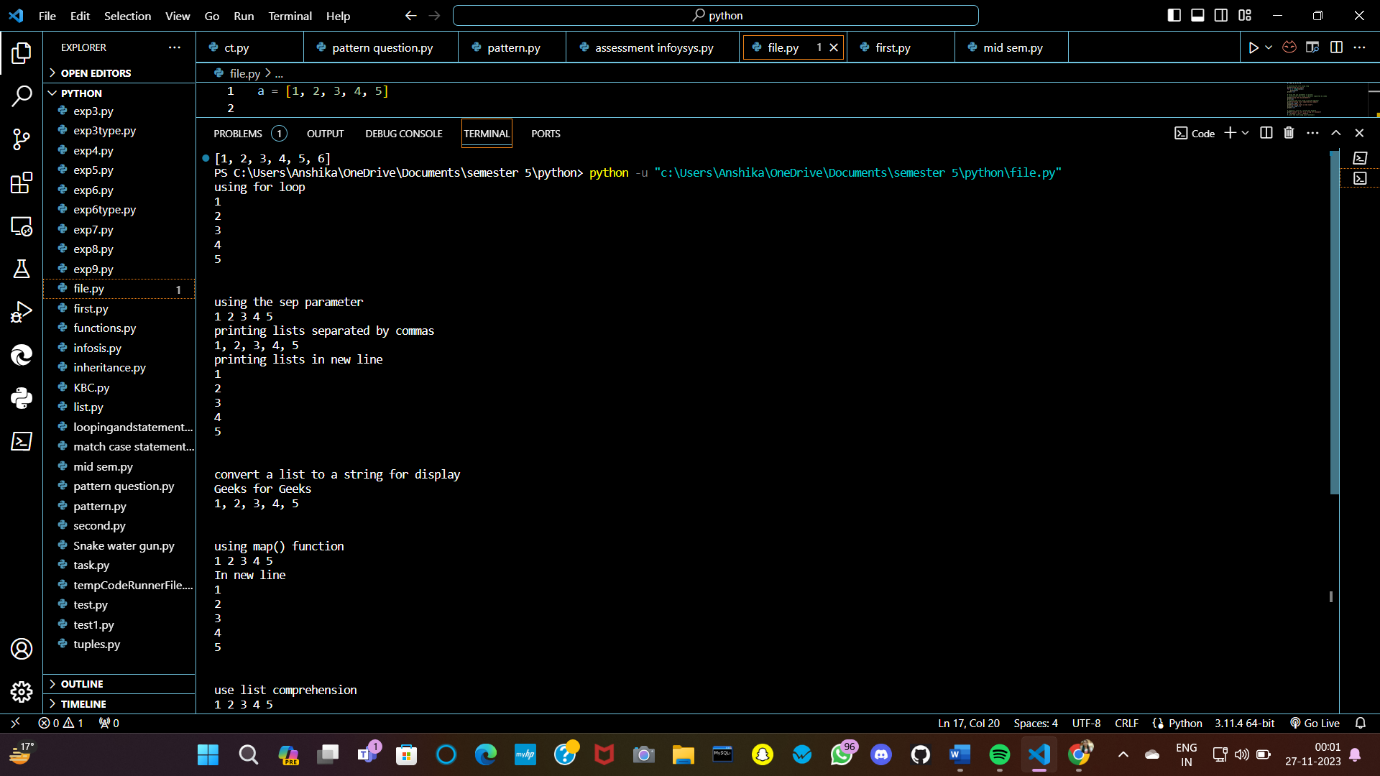
#method 2

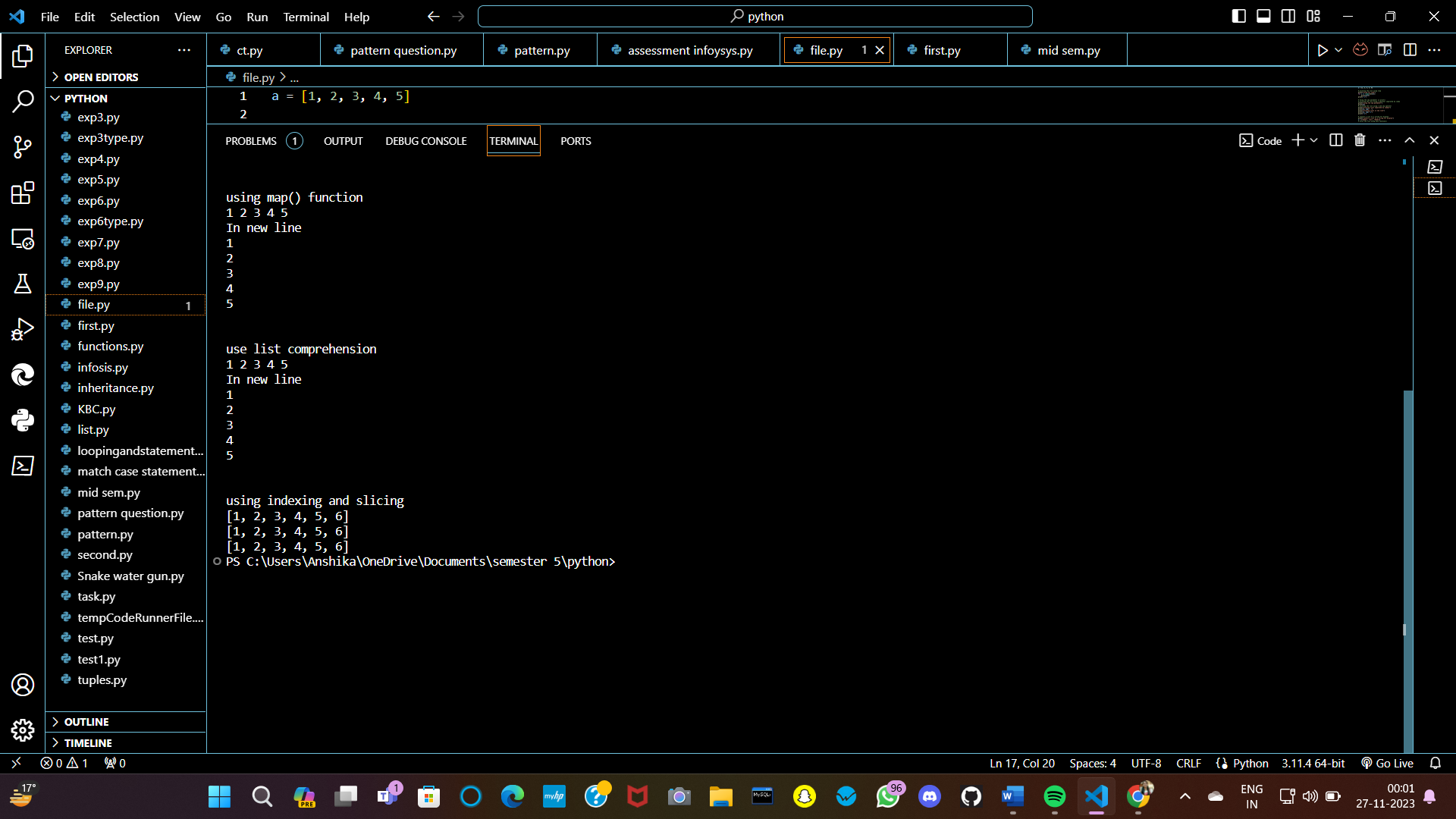
print(list[0:])

#method 3

print(list[0:len(list)])

**Output:**





**Experiment 5**

**Aim:** To implement the stack and queue using a list.

**Code:**

class queue\_implementation:

def \_\_init\_\_(self, list1=[]):

self.list1=list1

def enqueue(self, n):

self.list1.append(n)

def display(self):

if(len(self, list1)==0):

print("empty queue")

else:

for i in range(len(self.list1)-1, -1, -1):

print(self.list1[i], end=" ")

print("\n")

def dequeue(self):

if(len(self.list) == 0):

print("Empty queue")

else:

self.list1.pop(0)

class stack\_implementation:

def \_\_init\_\_(self, list1=[]):

self.list1 = list1

def push(self, n):

self.list1.append(n)

def display\_stack(self):

for i in range(len(self.list1) -1, -1, -1):

print(self.list1[i], end=" ")

print("\n")

def pop\_stack(self):

if(len(self.list1) == 0):

print("Stack is empty")

else:

self.list1.pop()

obj1 = stack\_implementation ()

obj1.push(1)

obj1.display\_stack()

obj1.push(2)

obj1.push(3)

obj1.display\_stack()

obj1.pop\_stack()

obj1.display\_stack()

obj2 = queue\_implementation()

obj2.enqueue(3)

obj2.enqueue(4)

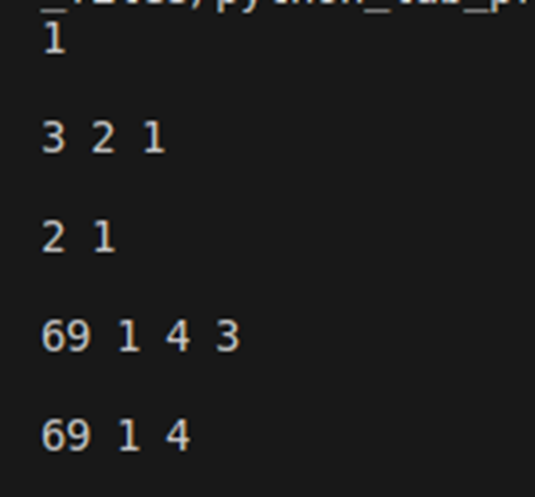
obj2.enqueue(1)

obj2.enqueue(69)

obj2.display()

obj2.dequeue()

obj2.display()

**Output:**

**Experiment 6**

**Aim:** To use the dictionary and make a dictionary of faculty and students and store them separately in lists.

**Code:**

faculty\_dict ={

'faculty\_id1': {'name': 'ABC','department': 'Computer Science'},

'faculty\_id2': {'name': 'EFG', 'department': 'Mathematics'}

}

students\_dict = {

'student\_id1': {'name': 'PQR', 'major': 'Physics'},

'student\_id2': {'name': 'XYZ', 'major': 'History'}

}

faculty\_list = list(faculty\_dict.values())

students\_list = list(students\_dict.values())

print("Faculty List: ")

for faculty in faculty\_list:

print(f"Name: {faculty['name']}, Department: {faculty['department']}")

print("\nStudents List: ")

for student in students\_dict:

print(f"Nmae: {student['name']}, Major: {student['major']}")

**Output:**

**Experiment 7**

**Aim:** To show the use of lambda expression.

**Code:**

A = lambda x:x+6

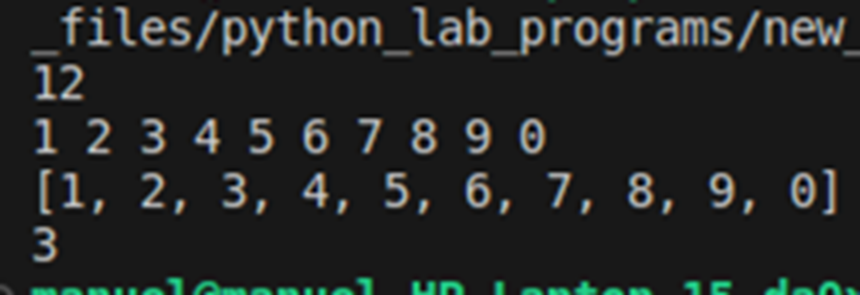
print(A(6))

list1=list(map(int, input().split()))

print(list1)

y=lambda x, z: z if A(z) +3>x else 6

print(y(6,3))

**Output:**

**Experiment 8**

**Aim:** To demonstrate the use of File writing and reading in text file.

**Code:**

with open("file.txt", "w") as f:

while(1==1):

line=input("enter the lines: ")

f.write(line)

f.write("\n")

choice=input("are you done(Y/N)")

if(choice.lower() == "y"):

break

else:

pass

f.close()

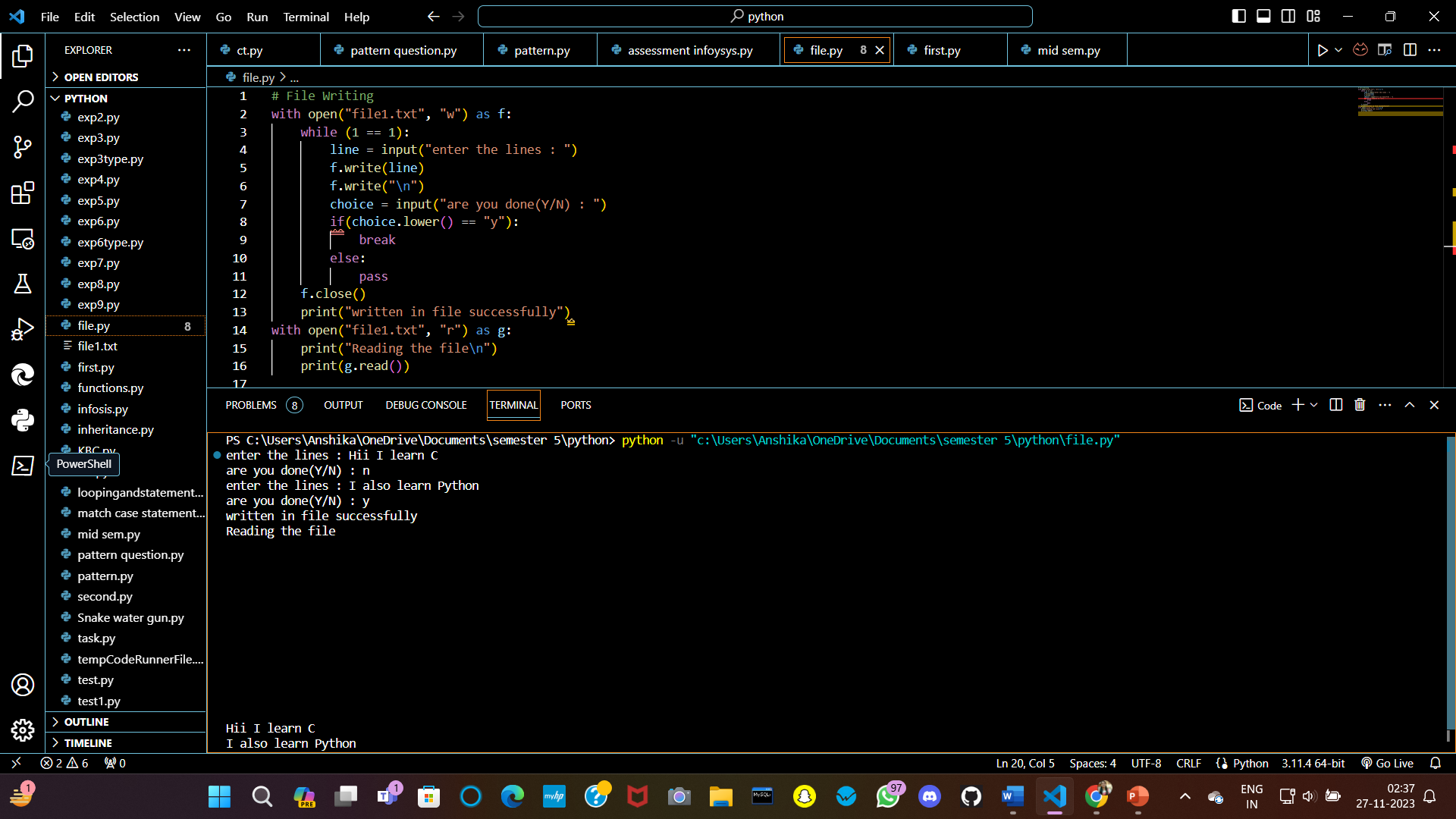
print("Written in file successfully")

with open("file1.txt","r") as g:

print("Reading the file\n")

print(g.read())

**Output:**

****



**Experiment 9**

**Aim:** To demonstrate error handling.

**Code:**

try:

a=int(input("enter the number: "))

print(a/2)

print(a/0)

except(ArithmeticError, ValueError):

print("An error Occoured\n")

**Output:**

****



**Experiment 10**

**Aim:** To demonstrate Multiple inheritance using classes  
**Code:**

class Employee:

def \_\_init\_\_(self, name):

self.name = name

def show(self):

print(f"the name is {self.name}")

class Dancer:

def \_\_init\_\_(self, dance):

self.dance = dance

def show(self):

print(f"the dance is {self.dance}")

class DancerEmployee(Employee, Dancer):

def \_\_init\_\_(self, name, dance):

self.name = name

self.dance = dance

o = DancerEmployee("ABC", "Kathak")

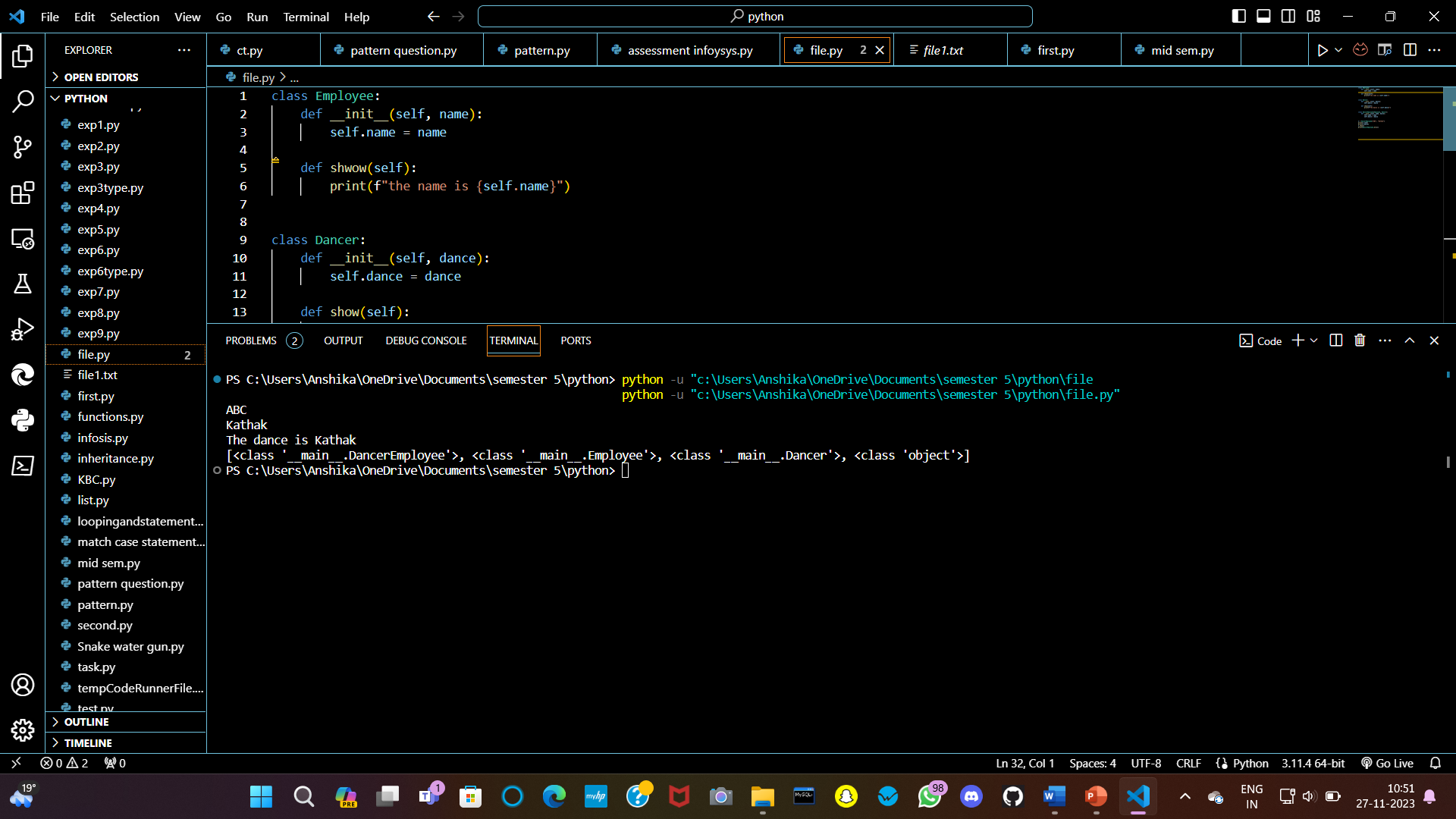
print(o.name)

print(o.dance)

o.show()

print(DancerEmployee.mro())

**Output:**

****

**Experiment 11**

**Aim:** To use NumPy and Pandas to generate a list.

**Code:**

import pandas as pd

import numpy as np

# Creating empty series

ser = pd.Series()

print("Pandas Series: ", ser)

# simple array

data = np.array(['g', 'e', 'e', 'k', 's'])

ser = pd.Series(data)

print("Pandas Series:\n", ser)

**Output:**

