A black background with white text

AI-generated content may be incorrect.

Name: Shoaib Ahmed

Intern ID: TN/IN02/PY/027

Task no: 02

Internship domain: python language

**Task 1:**

1. Store 5 student names & print each.

**Code:**

students = ["Shoaib", "ali", "waleed", "ubaid", "abdullah"]

for student in students:

    print(student)

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. Reverse list without reverse ()

**Code:**

students = ["wali", "shoaib", "abdullah", "junaid", "ali"]

reversed\_students = students[::-1]

print("Reversed list:", reversed\_students)

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Task 2:**

1. Store 3 coordinates & unpack.

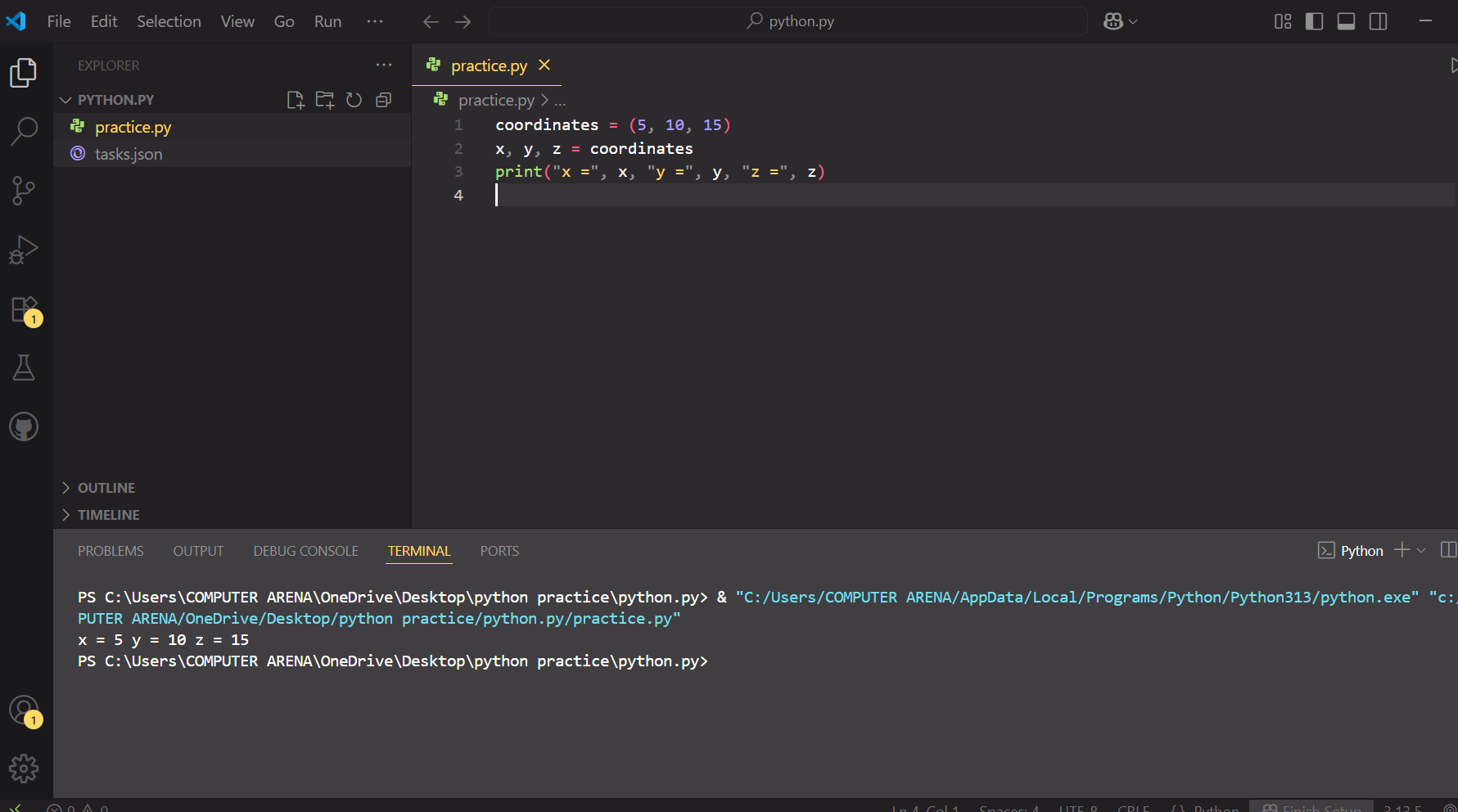
**Code:**

coordinates = (5, 10, 15)

x, y, z = coordinates

print("x =", x, "y =", y, "z =", z)

**Output:**



2. Swap vars using tuple assignment.

**Code:**

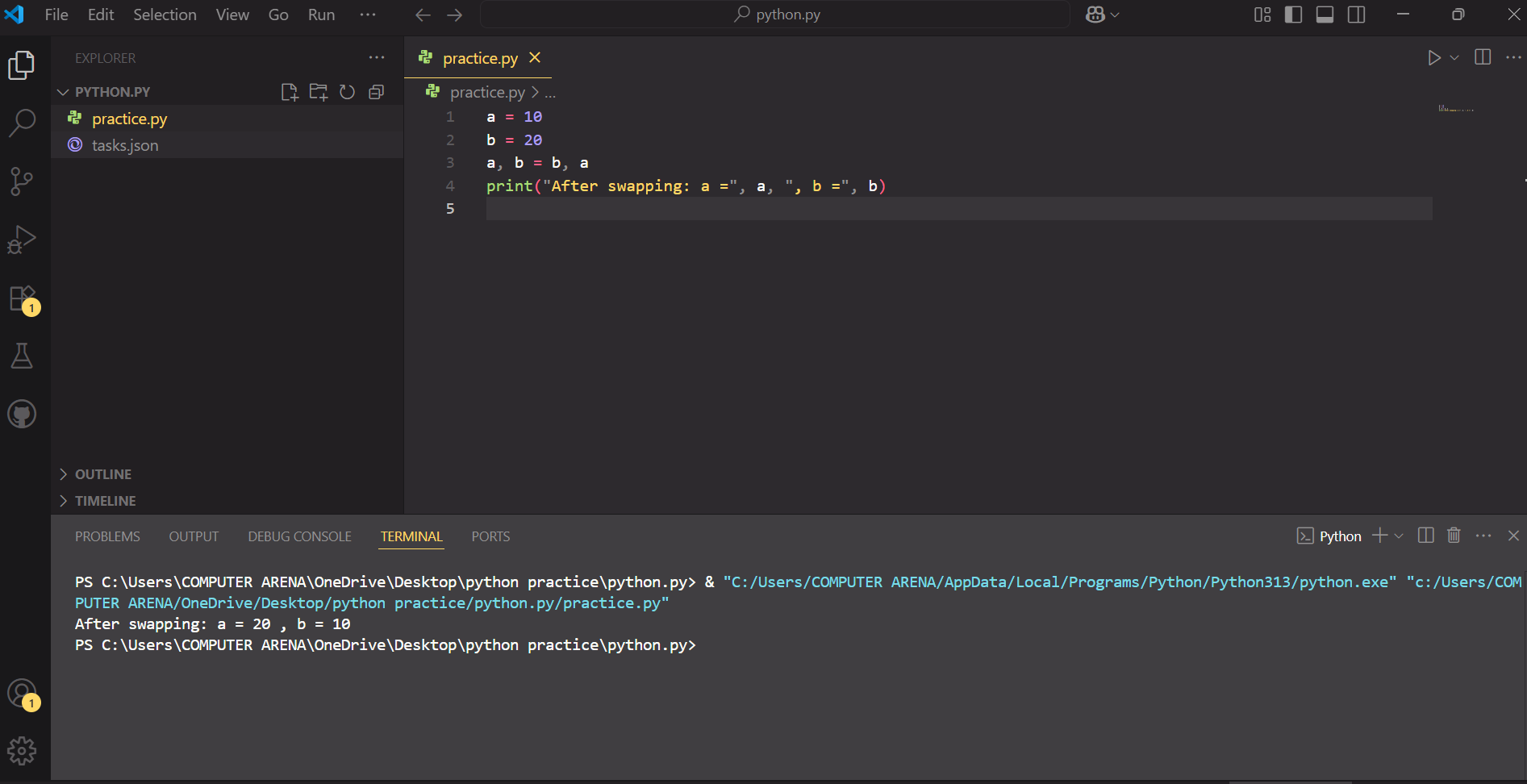
a = 10

b = 20

a, b = b, a

print("After swapping: a =", a, ", b =", b)

**Output:**



**Task 3:**

1. Remove duplicates from list.

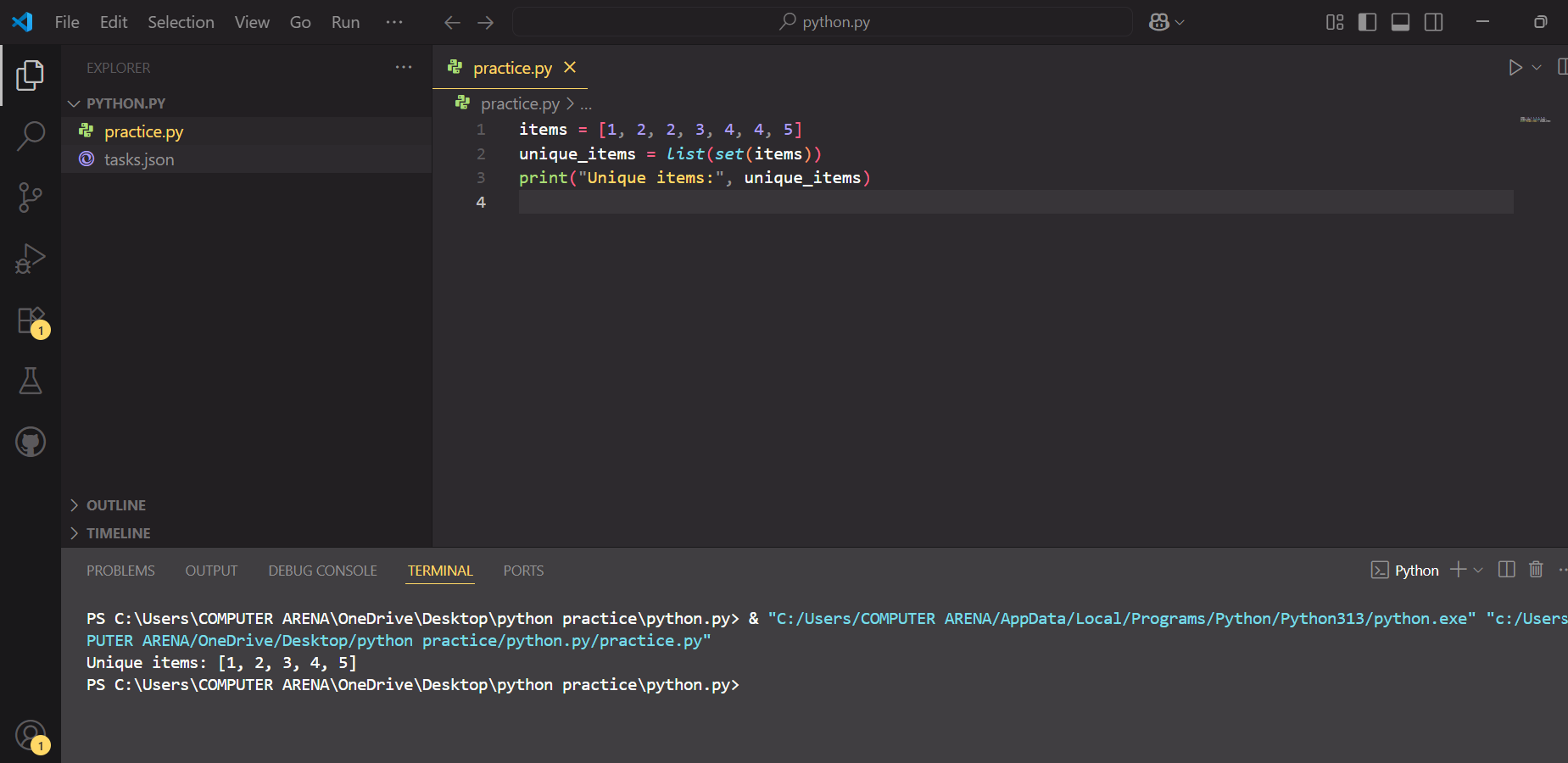
**Code:**

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

unique\_items = list(set(items))

print("Unique items:", unique\_items)

**Output:**



2. Find intersection of two sets.

**Code:**

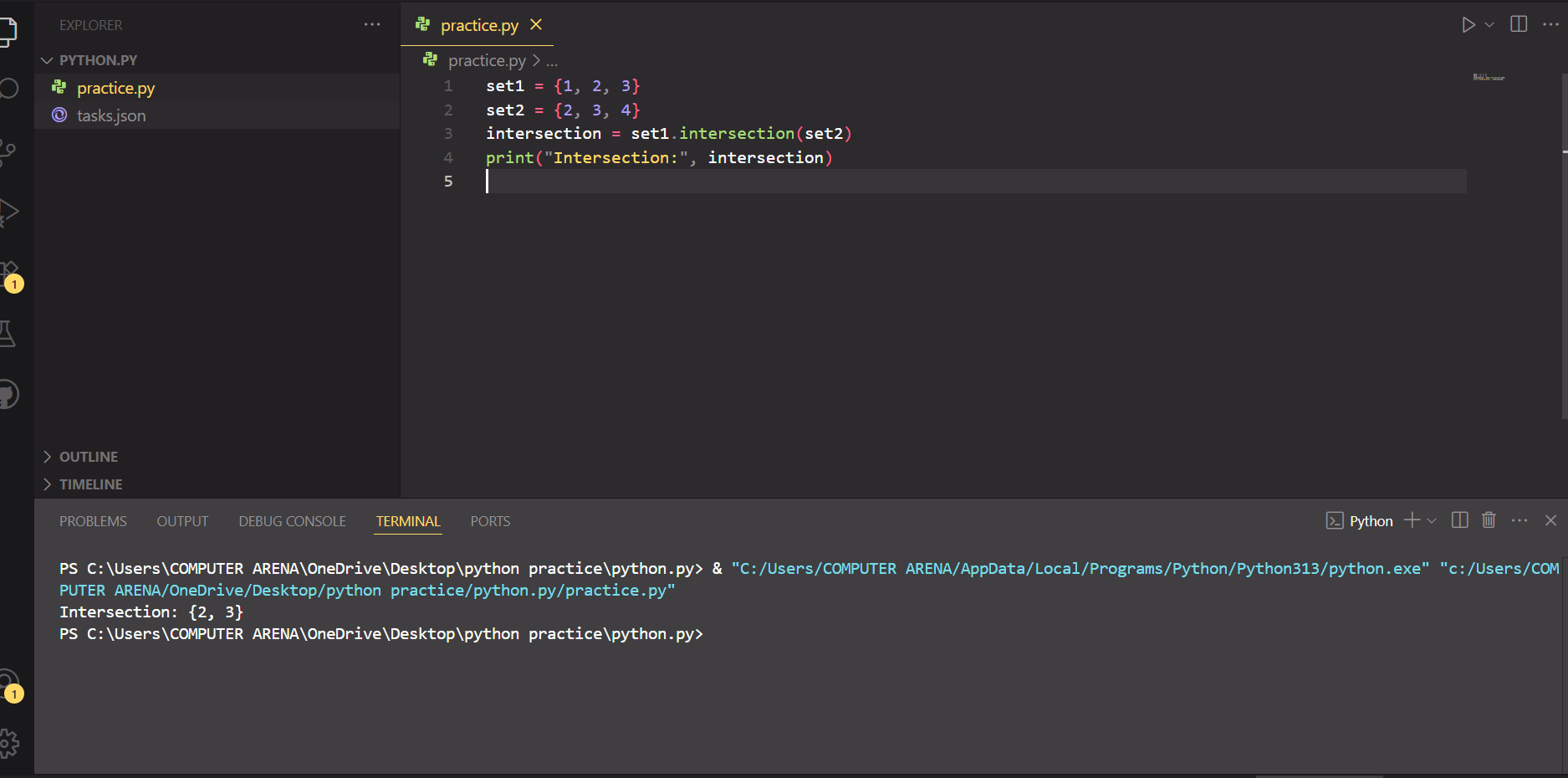
set1 = {1, 2, 3}

set2 = {2, 3, 4}

intersection = set1.intersection(set2)

print("Intersection:", intersection)

**Output:**



**Task 4:**

1. Student record CRUD in dict.

**Code:**

students = {}

*def* create(*name*, *marks*):

    students[name] = marks

*def* read(*name*):

    return students.get(name, "Not found")

*def* update(*name*, *marks*):

    if name in students:

        students[name] = marks

*def* delete(*name*):

    students.pop(name, None)

create ("shoaib", 90)

print(read("shoaib"))

update ("shoaib", 90)

delete("shoaib")

print(read("shoaib")) # *Should print 'Not found'*

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

2. Count word frequency in sentence.

**Code:**

sentence = "this is a test this is only a test"

words = sentence.split()

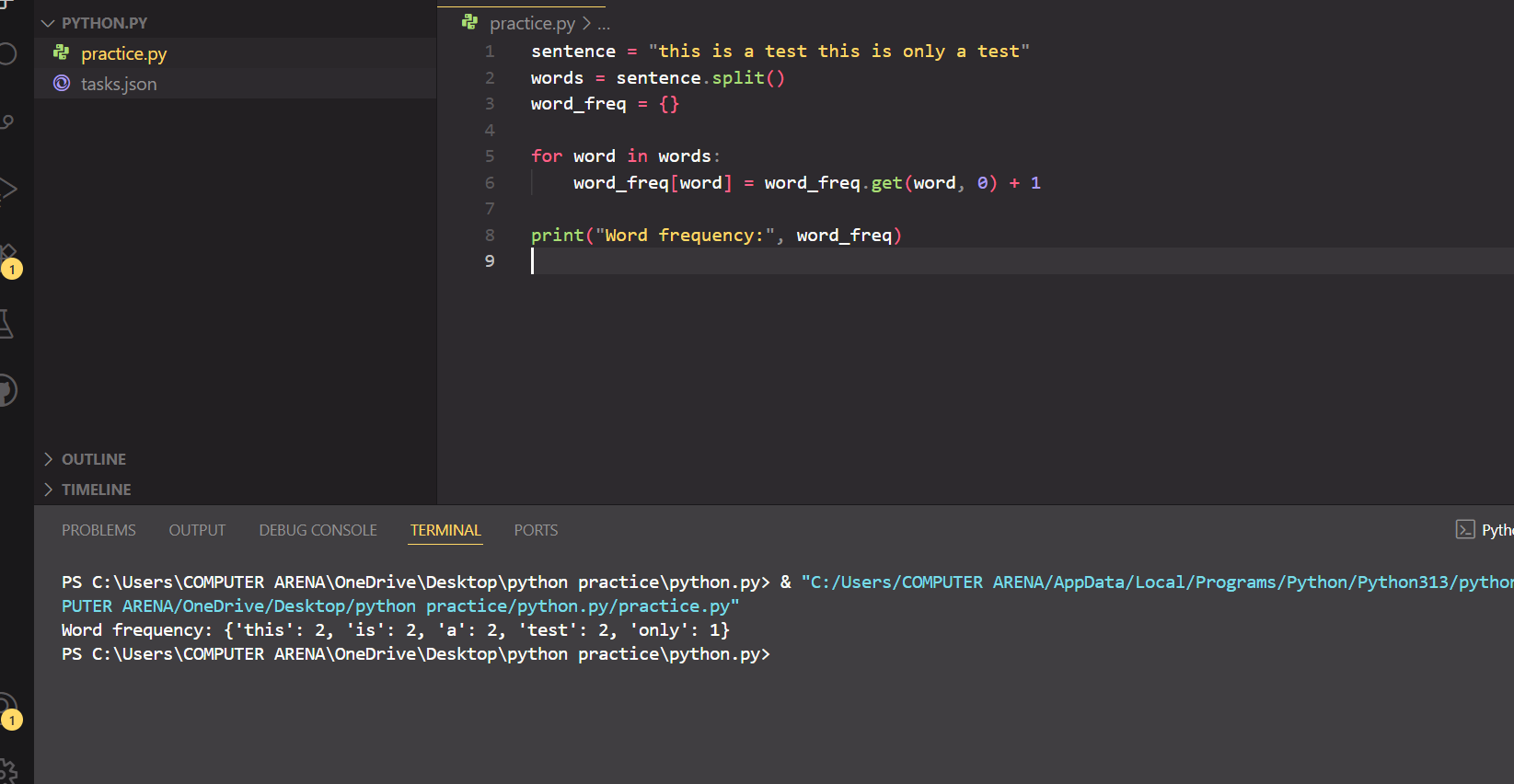
word\_freq = {}

for word in words:

word\_freq[word] = word\_freq.get(word, 0) + 1

print("Word frequency:", word\_freq)

**Output:**



**Task 5:**

1. Write calc (a, b,op).

**Code:**

def calc(a, b, op):

if op == '+':

return a + b

elif op == '-':

return a - b

elif op == '\*':

return a \* b

elif op == '/':

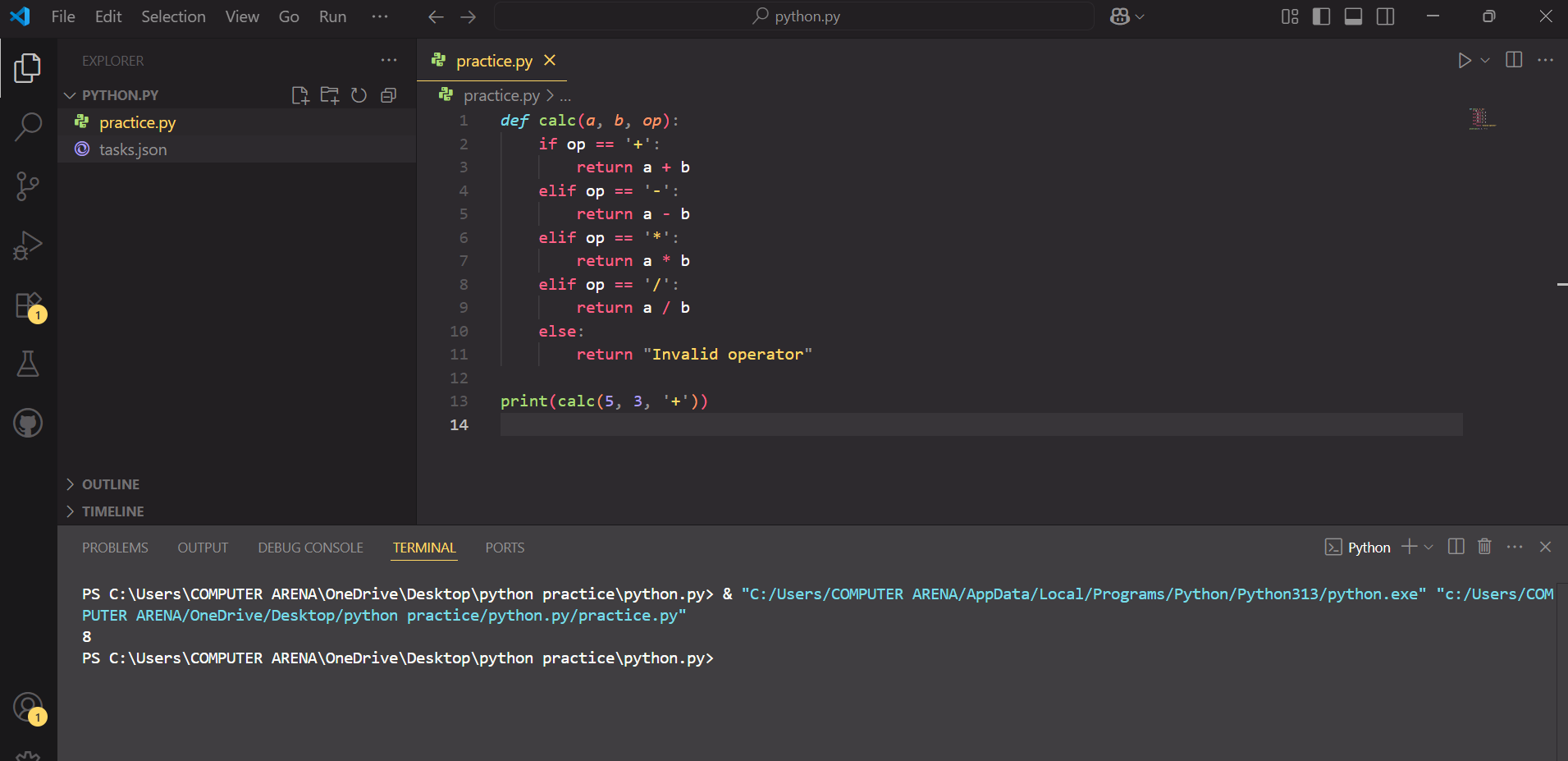
return a / b

else:

return "Invalid operator"

print(calc(5, 3, '+'))

**Output:**



2. Write factorial(n) recursive.

**Code:**

def factorial(n):

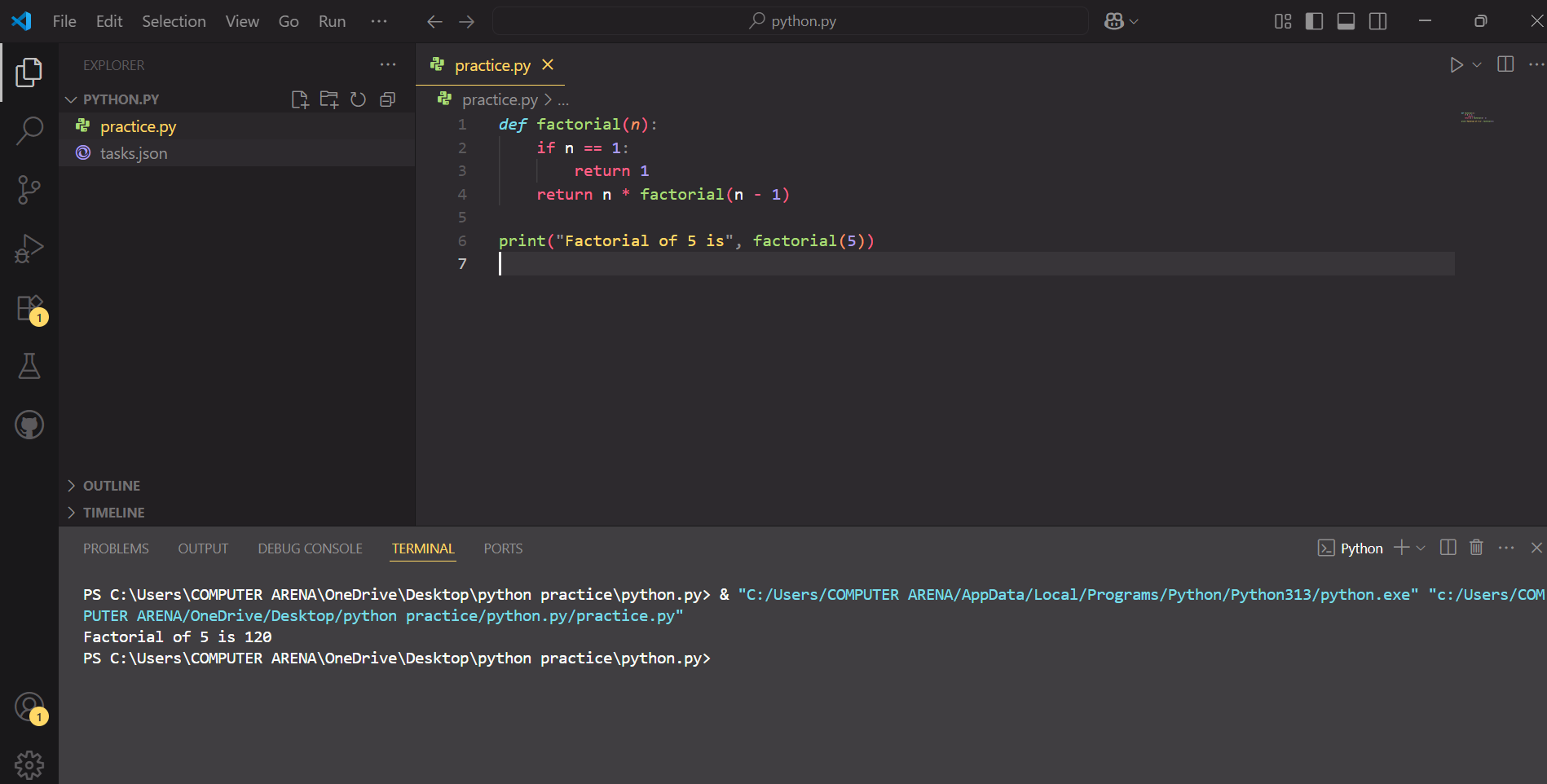
if n == 1:

return 1

return n \* factorial(n - 1)

print("Factorial of 5 is", factorial(5))

**Output:**

****

**Task 6:**

1. Use random & datetime in script.

**Code:**

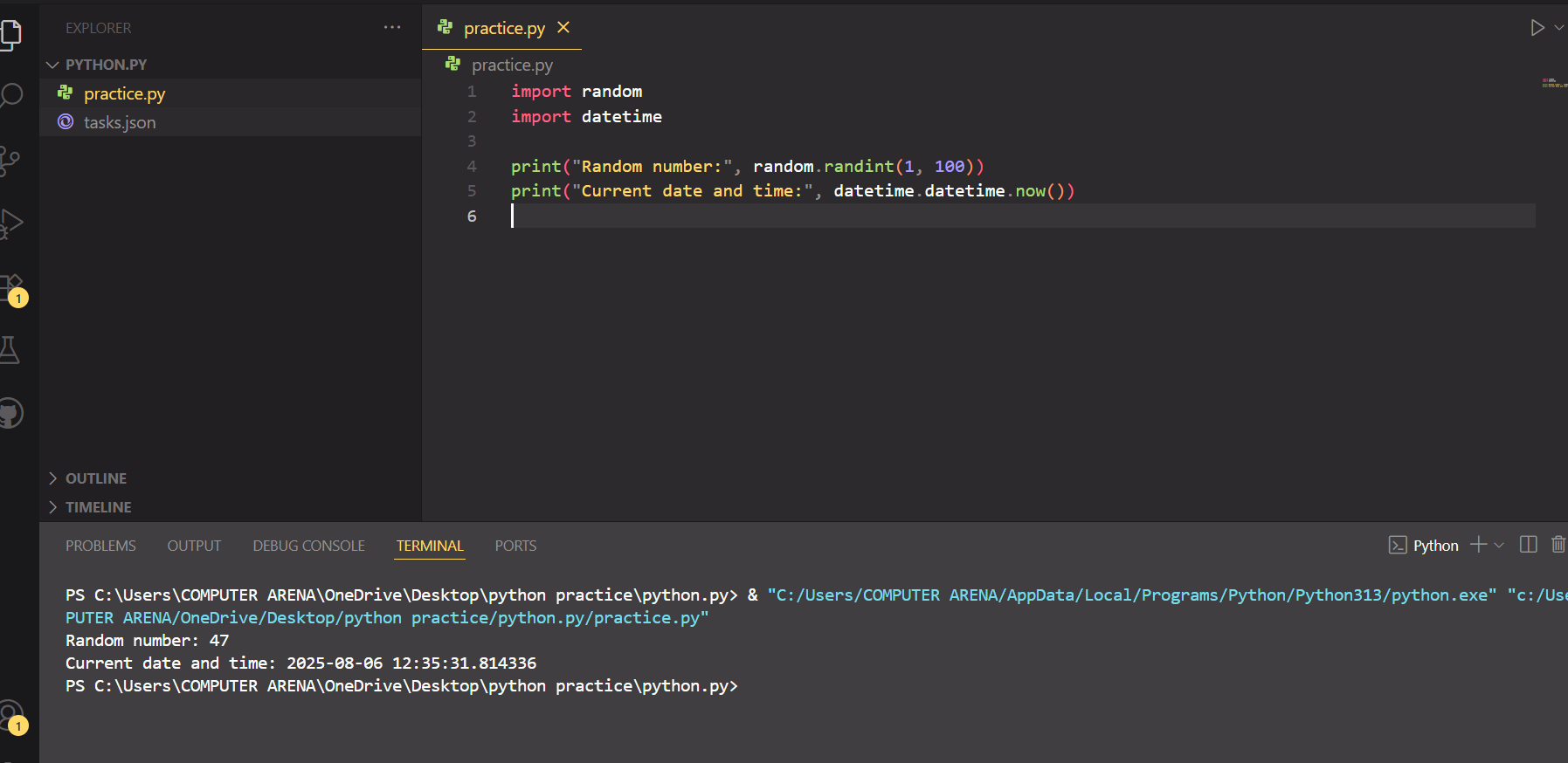
import random

import datetime

print("Random number:", random.randint(1, 100))

print("Current date and time:", datetime.datetime.now())

**Output:**



2. Create math\_utils module & import.

**Code:**

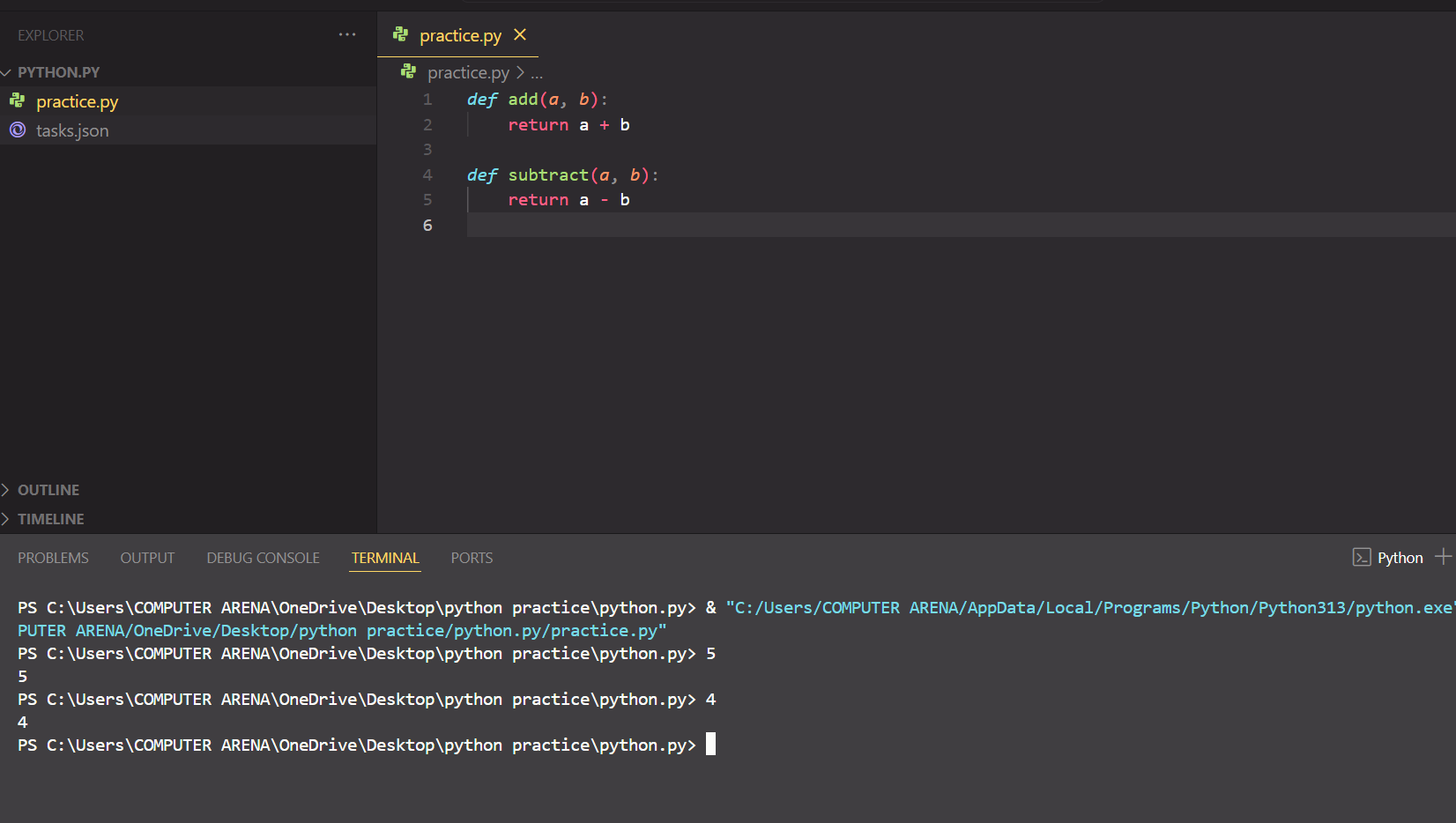
def add(a, b):

return a + b

def subtract(a, b):

return a – b

**Output:**



**Task 7:**

1. Safe int input loop.

**Code:**

while True:

try:

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

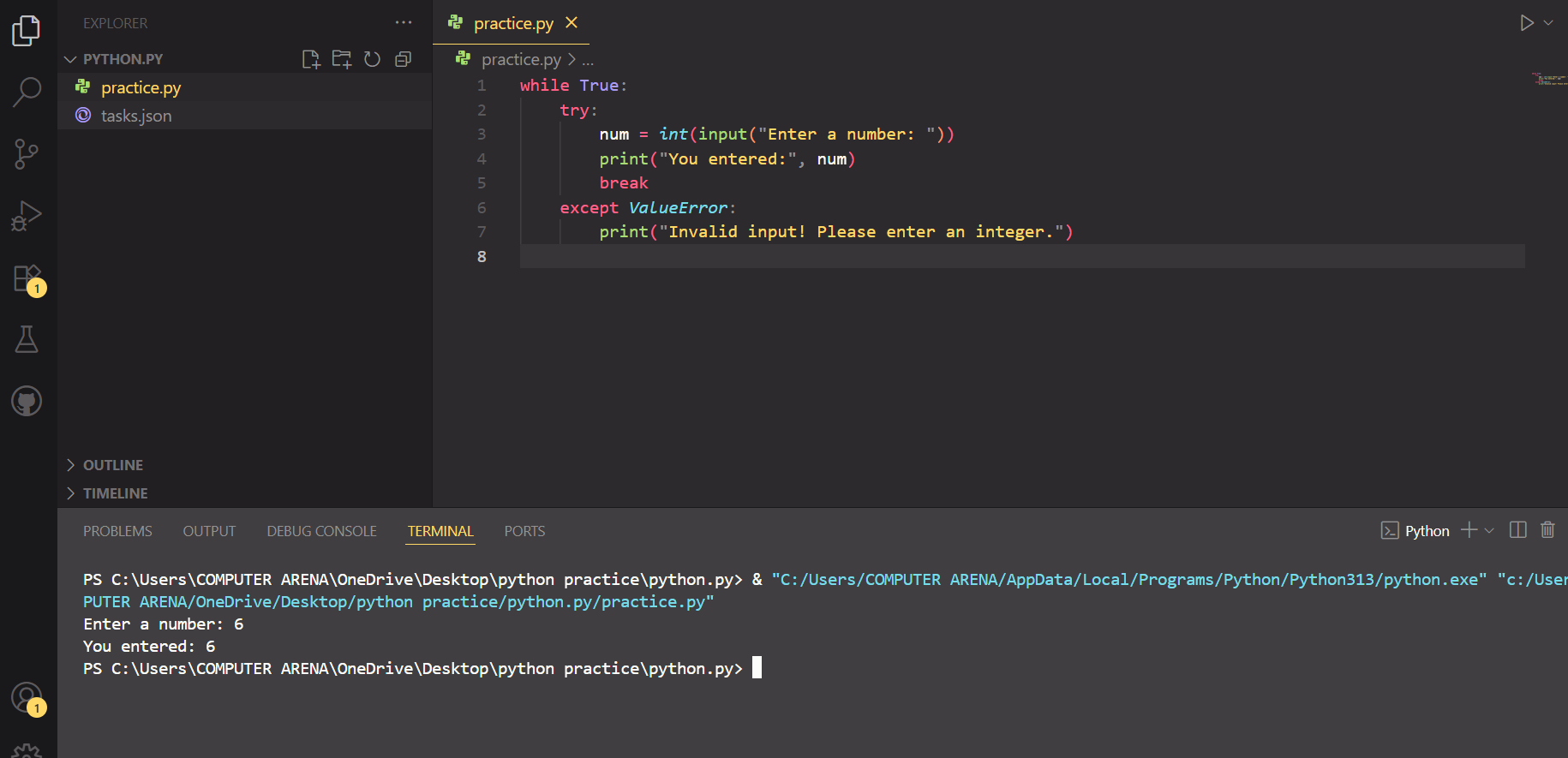
print("You entered:", num)

break

except ValueError:

print("Invalid input! Please enter an integer.")

**Output:**



2. File open with error message.

**Code:**

try:

with open("myfile.txt", "r") as file:

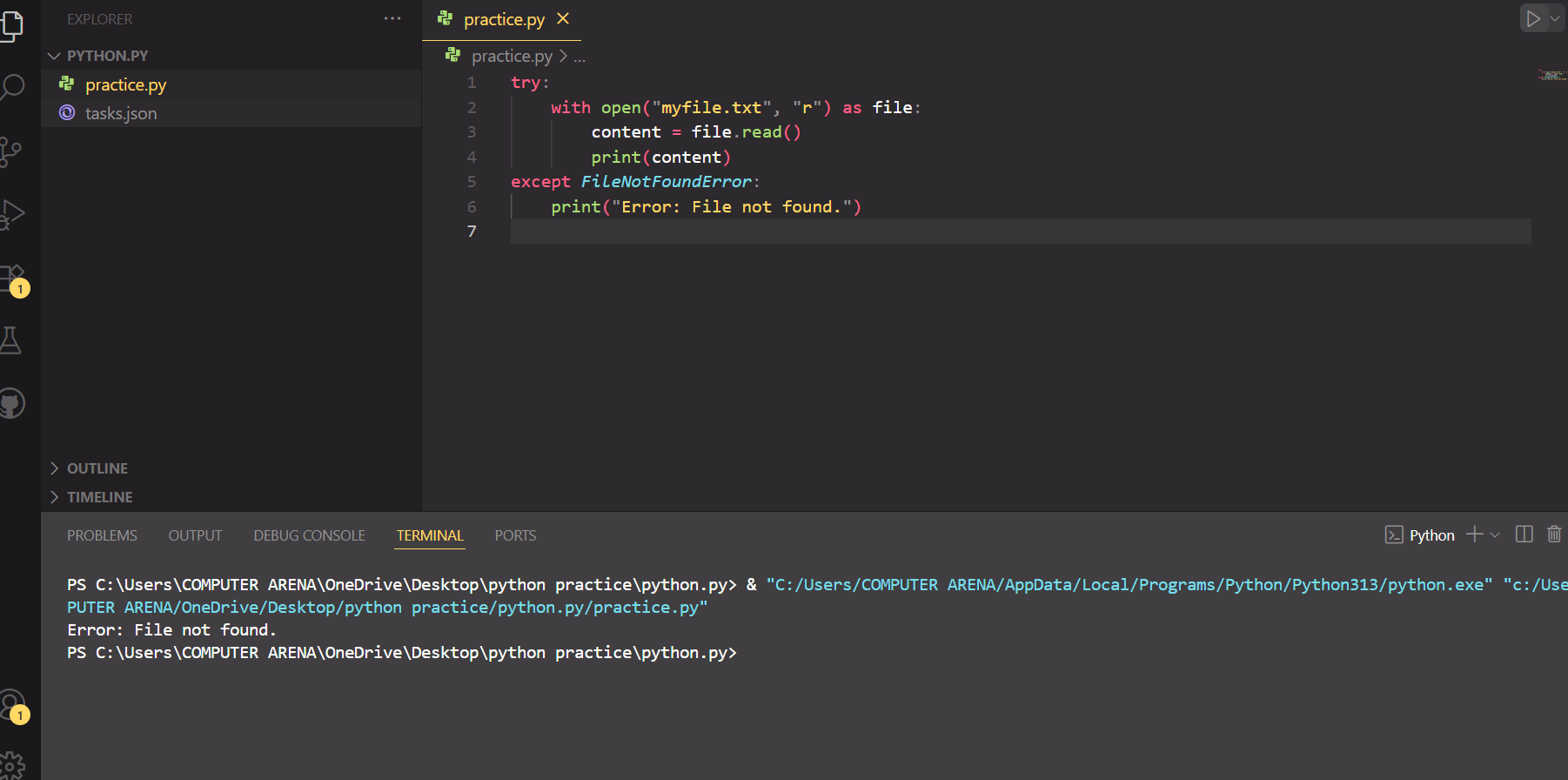
content = file.read()

print(content)

except FileNotFoundError:

print("Error: File not found.")

**Output:**



**Task 8: hard**

Phonebook App: CRUD contacts dict <-> JSON file storage**.**

**Code:**

import json

*# JSON file to store contact data*

filename = "contacts.json"

*# Load contacts from file (or start with empty list if file doesn't exist)*

try:

    with open(filename, "r") as file:

        contacts = json.load(file)

except *FileNotFoundError*:

    contacts = []

*# Save contacts to file*

*def* save\_contacts():

    with open(filename, "w") as file:

        json.dump(contacts, file, *indent*=4)

*# Add a new contact*

*def* add\_contact(*name*, *phone*):

    contacts.append({"name": name, "phone": phone})

    save\_contacts()

*# View all contacts*

*def* view\_contacts():

    for contact in contacts:

        print(*f*"Name: {contact['name']}, Phone: {contact['phone']}")

*# Update contact*

*def* update\_contact(*name*, *new\_phone*):

    for contact in contacts:

        if contact["name"] == name:

            contact["phone"] = new\_phone

            save\_contacts()

            return

    print("Contact not found.")

*# Delete contact*

*def* delete\_contact(*name*):

*global* contacts

    contacts = [c for c in contacts if c["name"] != name]

    save\_contacts()

*# Example usage*

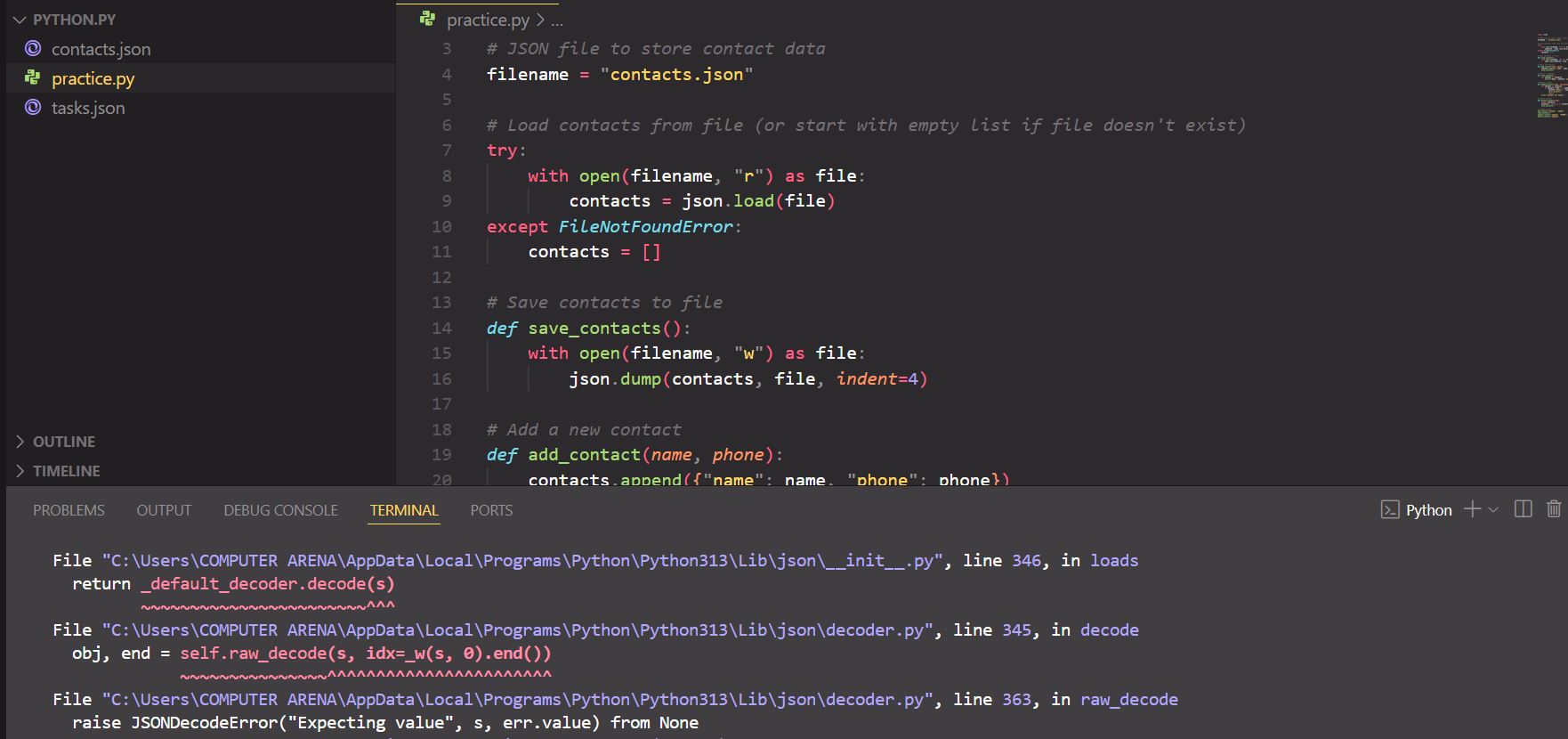
add\_contact ("shoaib", "12345")

view\_contacts()

update\_contact ("shoaib", "67890")

delete\_contact("shoaib")

**Output:**

****

**Learning from the task:**

From these tasks, I learned how to work with Python’s core data structures like lists, tuples, sets, and dictionaries, and how to use them in real-world applications. I understood how to perform operations like reversing a list, unpacking tuples, removing duplicates with sets, and managing student records using dictionaries. I practiced writing functions, including a calculator and recursive factorial, and got introduced to working with modules like random and datetime. I also explored creating and importing custom modules. Exception handling was covered to make programs more robust against user and file errors. Lastly, through the Phonebook app, I learned how to implement CRUD operations and store data using JSON files, which is a key step toward building real-world Python applications with persistent storage.