# ASSIGNMENT

# :Python Programming

Submitted by: Sarath Prasad TS RollNo: Mca 52

**Create a simple calculator in Python.**

**Program**

#Calculator

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

b=int(input("enter the input:"))

c=input("enter: + - / \* ")

if c=='+':

x=a+b

elif c=='-':

x=a-b

elif c=='/':

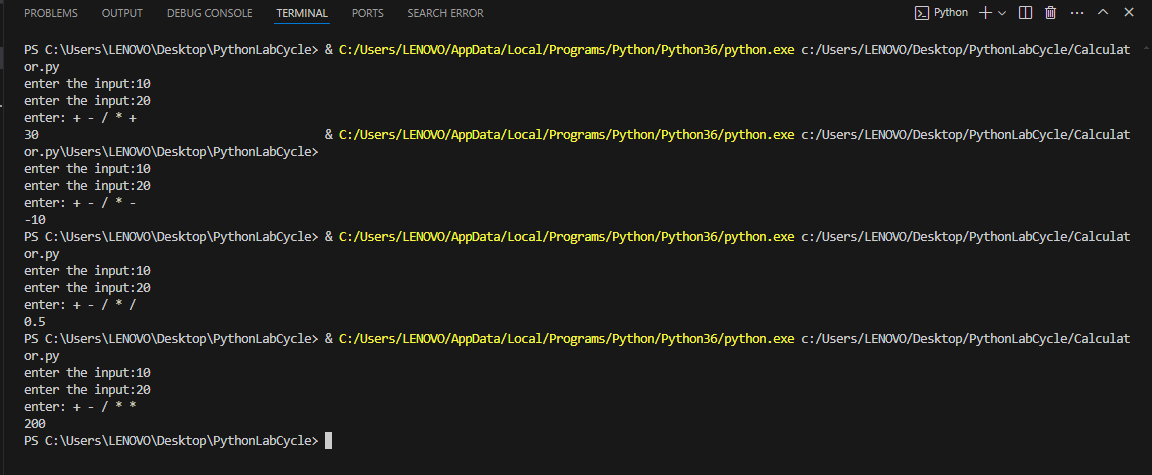
x=a/b

elif c=='\*':

x=a\*b

print(x)

**Output:-**



**1.2. An electric power distribution company charges domestic customers as follows: Consumption unit Rate of charge:**

**1. 0-200 Rs. 0.50 per unit**

**2. 201-400 Rs. 0.65 per unit in excess of 200**

**3. 401-600 Rs 0.80 per unit excess of 400**

**4. 601 and above Rs 1.00per unit excess of 600**

**5. If the bill exceeds Rs. 400, then a surcharge of 15% will be charged,and the minimum bill should be Rs. 100/-**

**Create a Python program based on the scenario mentioned above.**

**Program**

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

if a<=200:

price=a\*0.50

elif a<=400:

price=200\*0.50+(a-200)\*0.65

elif a<=600:

price=200\*0.5+200\*0.65+(a-400)\*0.85

else:

price=200\*0.5+200\*0.65+(a-600)\*1.00

if price<100:

price=100

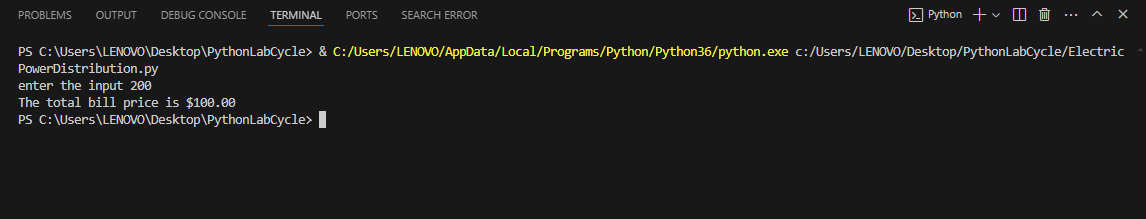
elif price>400:

surcharge=price\*0.15

price=price+surcharge

print(f"The total bill price is ${price:.2f}")

**Output:-**



* 1. **Print the pyramid of numbers using for loops.**

**Program**

#Print the pyramid of numbers using for loops.

def Piramid(rows):

for i in range(rows):

print(" "\*(rows-i-1),end="")

for j in range(2\*i+1):

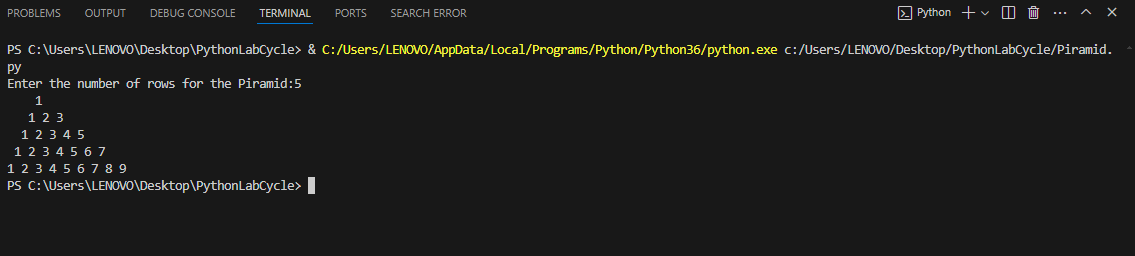
print(f"{j+1}",end=" ")

print()

rows=int(input("Enter the number of rows for the Piramid:"))

Piramid(rows)

**Output:-**



* 1. **Write a program to find the number and sum of all integers greater than 100 and less than 200 that are divisible by 7.**

**Program**

i=100

sum=0

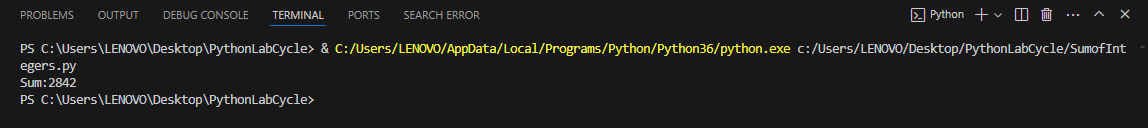
for i in range(200):

if i%7==0:

sum+=i

print(f"Sum:{sum}")

**Output:-**



* 1. **Write a recursive function to calculate the sum of numbers from 0 to 10**

**Program**

sum=0

n=0

def CalSum(n):

if n==0:

return 0

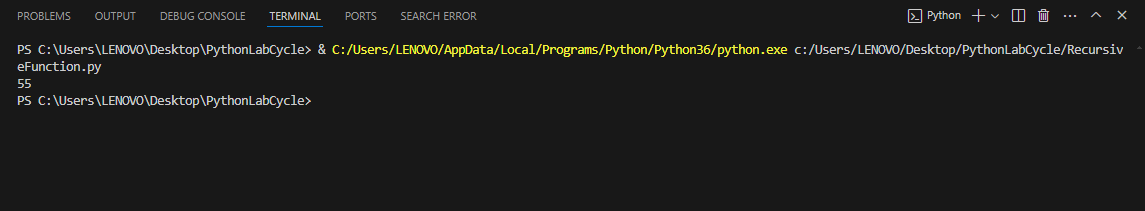
else:

return n+CalSum(n-1)

result=CalSum(10)

print(result)

**Output:-**



* 1. **Write a Python program to reverse the digits of a given number and add them to the original. If the sum is not a palindrome, repeat this procedure.**

**Program**

def reverse\_number(number):

return int(str(number)[::-1])

def is\_palindrome(number):

return str(number) == str(number)[::-1]

def reverse\_and\_add\_to\_palindrome(number):

while True:

reversed\_num = reverse\_number(number)

number += reversed\_num

if is\_palindrome(number):

return number

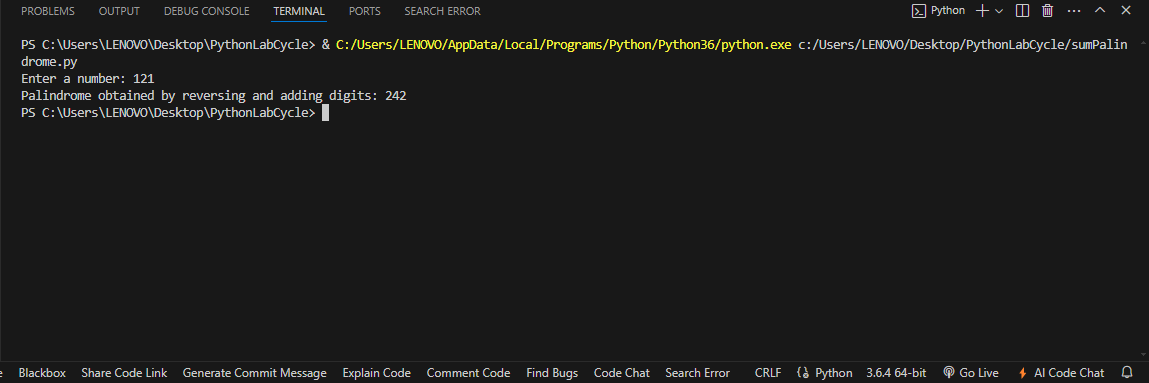
# Test the function

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

result = reverse\_and\_add\_to\_palindrome(num)

print("Palindrome obtained by reversing and adding digits:", result)

**Output:-**



**1.7. Write a menu-driven program that performs the following operations on strings**

**1. Check if the String is a Substring of Another String**

**2. Count Occurrences of Character**

**3. Replace a substring with another substring**

**4. Convert to Capital Letters**

**Program**

def isSubstring(Mstring,Substring):

return Substring in Mstring

def CountOccurance(String, Char):

return String.count(Char)

def ReplaceSubstring(String,old,new):

return String.replace(old,new)

def ConvertToCapital(String):

return String.upper()

def main():

while True:

print("\nMenu")

print("1.Check if the String is a Substring of Another String")

print("2.Count Occurrences of Character")

print("3.Replace a substring with another substring")

print("4.Convert to Capital Letters")

print("5.Exit")

choice=input("enter your choice:")

if choice=="1":

Mstring=input("Enter the main string:")

Substring=input("Ente the Substring to check:")

if isSubstring(Mstring,Substring):

print(f"'{Substring}'is a Substring of '{Mstring}'")

else:

print(f"'{Substring}'is not a Substring of '{Mstring}'")

elif choice=="2":

String=input("enter the string")

Char=input("enter the character to count")

if len(Char)!=1:

print("enter a single character")

else:

count=CountOccurance(String,Char)

print(f"the character'{Char}occurs{count} times in {String}")

elif choice=="3":

Mstring=input("Enter the main string:")

oldSubstring=input("Ente the Substring to change :")

newSubstring=input("enter the New Substring")

newString=ReplaceSubstring(Mstring,oldSubstring,newSubstring)

print(f"the new string after replacing '{oldSubstring}' with '{newSubstring}' is {newString}")

elif choice=="4":

String=input("Enter the string to convert to capital letters:")

print(f"the string in capital letters is {ConvertToCapital(String)}")

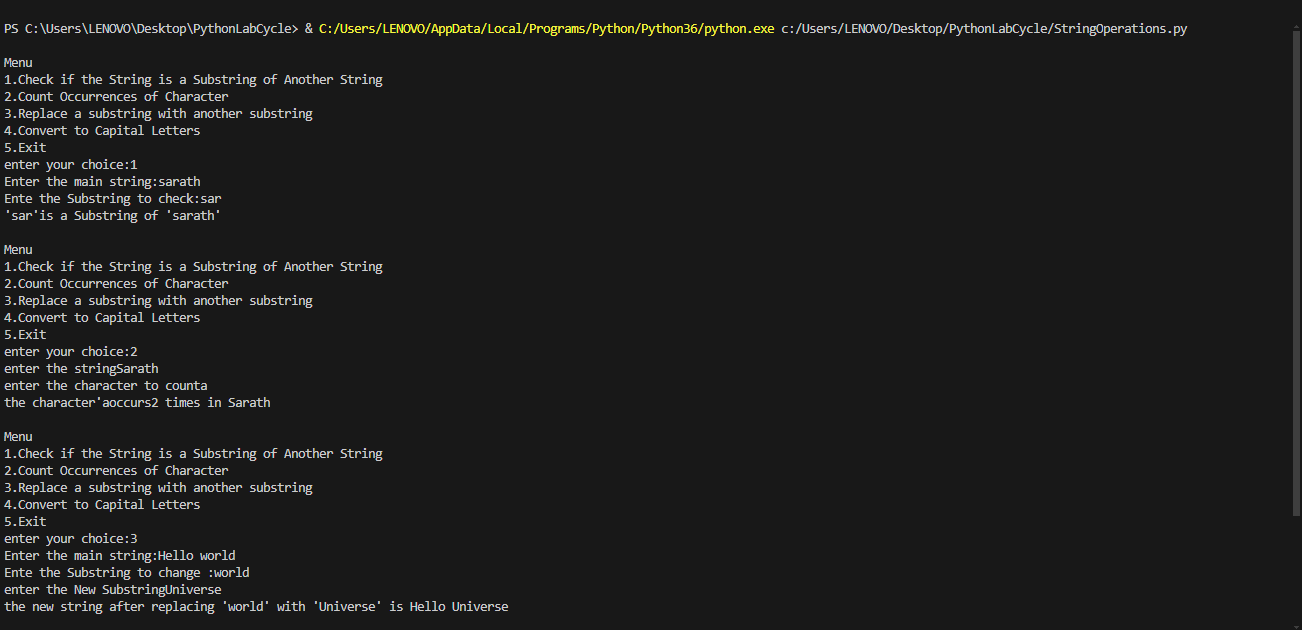
else:

exit()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



* 1. **Write a function to find the factorial of a number but also store the factorials** **calculated in a dictionary.**

**Program**

dfactorials = {}

def factorial(n):

if n < 0:

raise ValueError("No negative values allowed")

if n in dfactorials:

return dfactorials[n]

if n == 0 or n == 1:

dfactorials[n] = 1

else:

dfactorials[n] = n \* factorial(n - 1)

return dfactorials[n]

def main():

while True:

print("\nMenu")

print("1. Calculate factorial")

print("2. EXIT")

choice = input("Enter your choice: ")

if choice == "1":

try:

number = int(input("Enter the number: "))

result = factorial(number)

print(f"Factorial of {number} is {result}")

print("Factorial Dictionary:", dfactorials)

except ValueError as e:

print(e)

elif choice == "2":

print("Exiting the program.")

break

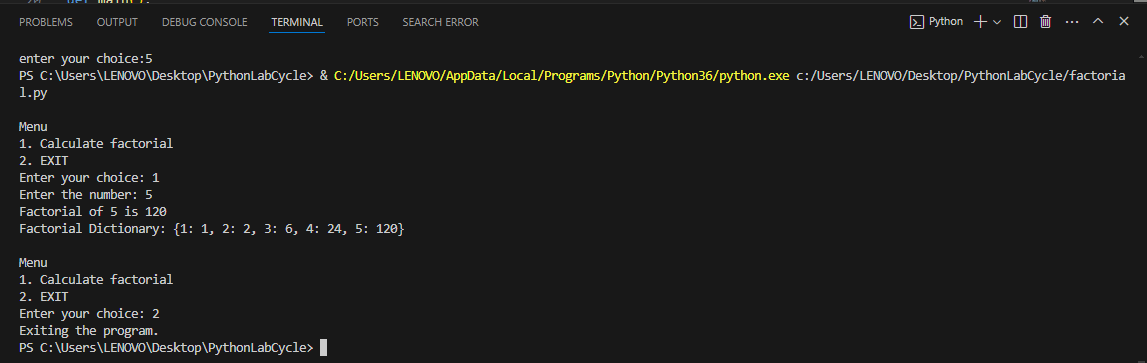
else:

print("Invalid choice. Please enter a valid option (1 or 2).")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



**1.9. Perform various set operations**

**1. Set Union**

**2. Set Intersection**

**3. Set Difference**

**Program**

def set\_union(set1, set2):

return set1.union(set2)

def set\_intersection(set1, set2):

return set1.intersection(set2)

def set\_difference(set1, set2):

return set1.difference(set2)

def main():

while True:

print("\nMenu:")

print("1. Set Union")

print("2. Set Intersection")

print("3. Set Difference")

print("4. Exit")

choice = input("Enter your choice: ")

if choice in ["1", "2", "3"]:

set1 = set(map(int, input("Enter the elements of the first set (separated by space): ").split()))

set2 = set(map(int, input("Enter the elements of the second set (separated by space): ").split()))

if choice == "1":

result = set\_union(set1, set2)

print(f"The union of {set1} and {set2} is {result}")

elif choice == "2":

result = set\_intersection(set1, set2)

print(f"The intersection of {set1} and {set2} is {result}")

elif choice == "3":

result = set\_difference(set1, set2)

print(f"The difference of {set1} and {set2} is {result}")

elif choice == "4":

print("Exiting the program.")

break

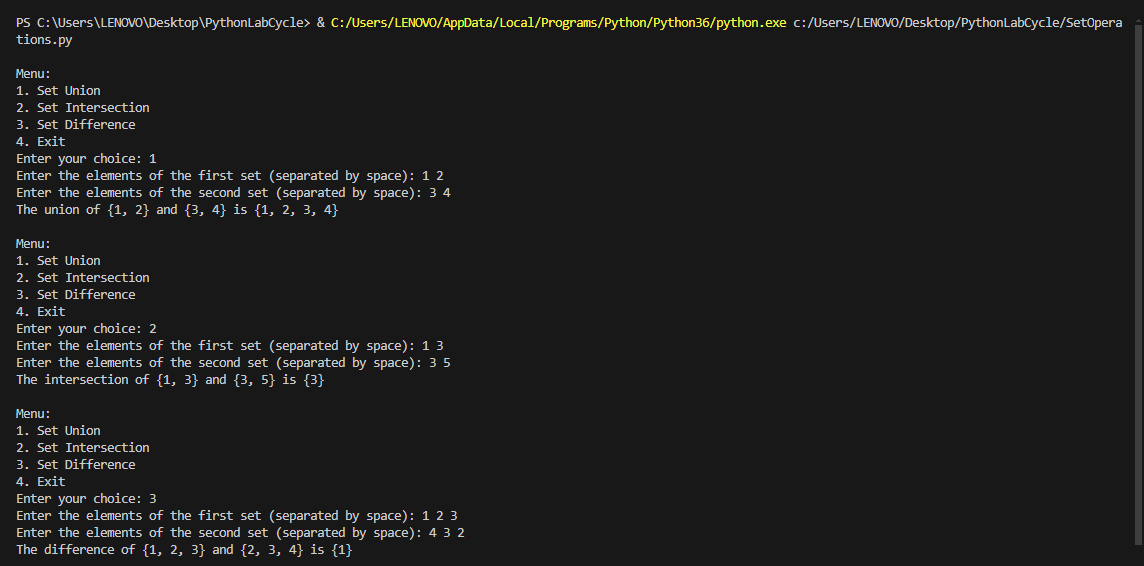
else:

print("Invalid choice. Please enter a valid option (1-4).")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



* 1. **Create a dictionary to store the name, roll\_no, and total\_mark of N students.** **Now print the details of the student with the highest total\_mark.**

**Program**

def get\_student\_data():

name = input("Enter the student's name: ")

roll\_no = input("Enter the student's roll number: ")

total\_marks = float(input("Enter the student's total marks: "))

return {"name": name, "roll\_no": roll\_no, "total\_marks": total\_marks}

def main():

students = []

N = int(input("Enter the number of students: "))

for \_ in range(N):

student\_data = get\_student\_data()

students.append(student\_data)

if students:

highest\_marks\_student = max(students, key=lambda x: x['total\_marks'])

print("\nStudent with the highest total marks:")

print(f"Name: {highest\_marks\_student['name']}")

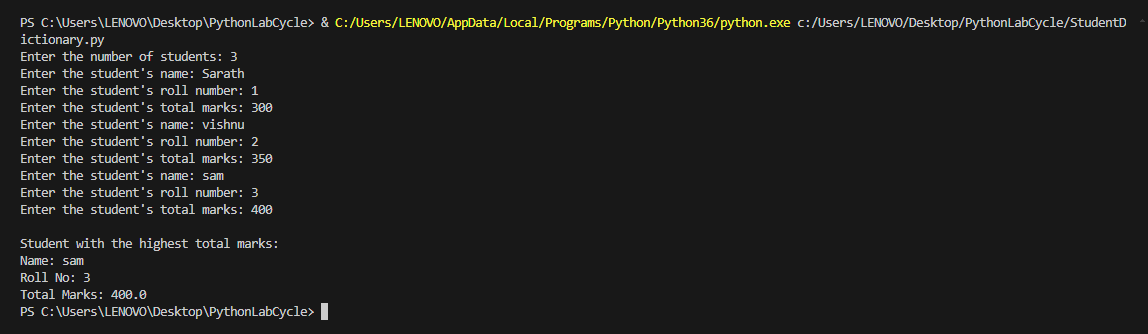
print(f"Roll No: {highest\_marks\_student['roll\_no']}")

print(f"Total Marks: {highest\_marks\_student['total\_marks']}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



* 1. **Write a Python program to copy the contents of a file into another file, line by line.**

**Program**

def copy\_file(source\_file, destination\_file):

try:

# Open source file in read mode

with open(source\_file, 'r') as src:

# Open destination file in write mode

with open(destination\_file, 'w') as dest:

# Read each line from source file and write to destination file

for line in src:

dest.write(line)

print(f"Contents of '{source\_file}' copied to '{destination\_file}' successfully.")

except FileNotFoundError:

print("One of the files does not exist.")

except IOError as e:

print(f"Error occurred while copying the file: {e}")

def main():

source\_file = input("Enter the source file name: ")

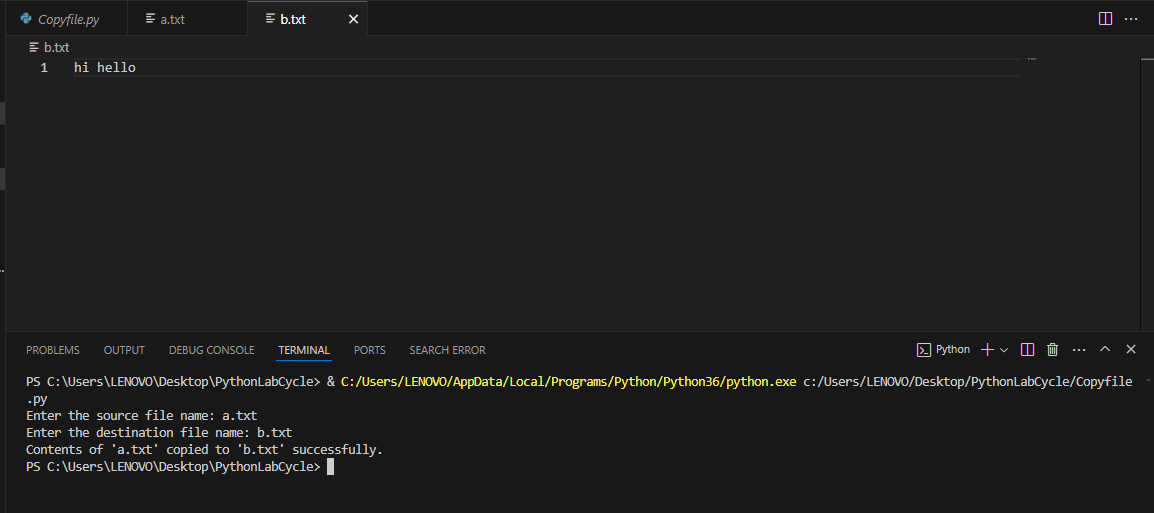
destination\_file = input("Enter the destination file name: ")

copy\_file(source\_file, destination\_file)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



**1.12. Use the OS module to perform**

**1. Create a directory**

**2. Directory Listing**

**3. Search for “.py” files**

**4. Remove a particular file**

**Program**

import os

def create\_directory(directory\_name):

"""Create a directory with the given name."""

try:

os.makedirs(directory\_name, exist\_ok=True)

print(f"Directory '{directory\_name}' created successfully.")

except Exception as e:

print(f"Error creating directory '{directory\_name}': {e}")

def list\_directory(directory\_name):

"""List the contents of the given directory."""

try:

contents = os.listdir(directory\_name)

print(f"Contents of '{directory\_name}':")

for item in contents:

print(item)

except FileNotFoundError:

print(f"Directory '{directory\_name}' does not exist.")

except Exception as e:

print(f"Error listing directory '{directory\_name}': {e}")

def search\_py\_files(directory\_name):

"""Search for '.py' files in the given directory."""

try:

py\_files = [f for f in os.listdir(directory\_name) if f.endswith('.py')]

print(f"'.py' files in '{directory\_name}':")

for py\_file in py\_files:

print(py\_file)

except FileNotFoundError:

print(f"Directory '{directory\_name}' does not exist.")

except Exception as e:

print(f"Error searching for '.py' files in '{directory\_name}': {e}")

def remove\_file(file\_path):

"""Remove the specified file."""

try:

os.remove(file\_path)

print(f"File '{file\_path}' removed successfully.")

except FileNotFoundError:

print(f"File '{file\_path}' does not exist.")

except Exception as e:

print(f"Error removing file '{file\_path}': {e}")

def main():

while True:

print("\nMenu:")

print("1. Create a directory")

print("2. List the contents of a directory")

print("3. Search for '.py' files in a directory")

print("4. Remove a particular file")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == "1":

directory\_name = input("Enter the directory name to create: ")

create\_directory(directory\_name)

elif choice == "2":

directory\_name = input("Enter the directory name to list: ")

list\_directory(directory\_name)

elif choice == "3":

directory\_name = input("Enter the directory name to search for '.py' files: ")

search\_py\_files(directory\_name)

elif choice == "4":

file\_path = input("Enter the file path to remove: ")

remove\_file(file\_path)

elif choice == "5":

print("Exiting the program.")

break

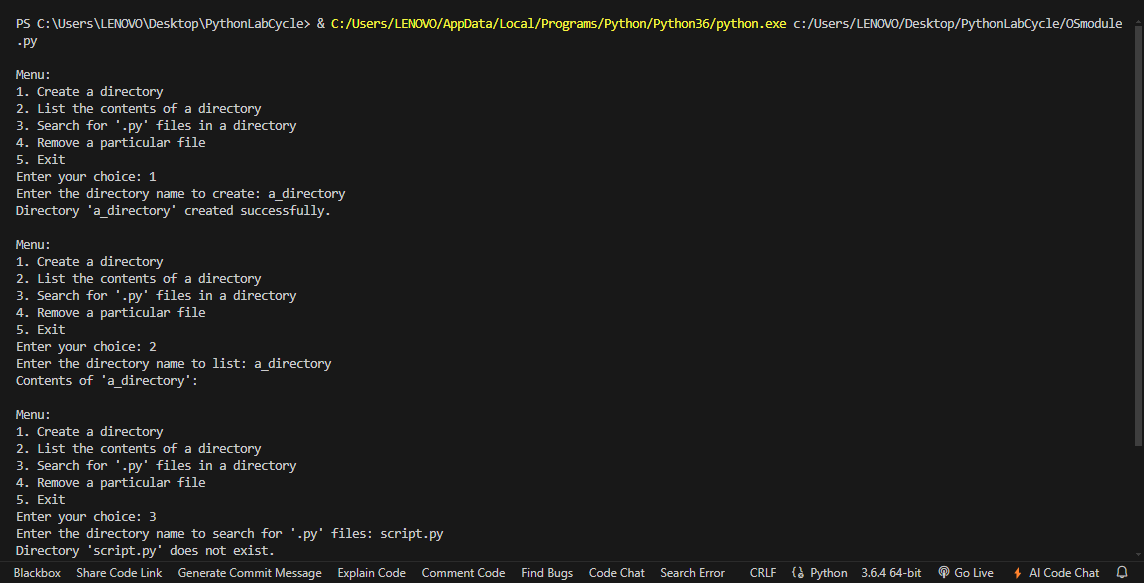
else:

print("Invalid choice. Please enter a valid option (1-5).")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**



* 1. **Create a simple banking application by using inheritance.**

**Program**

class BankAccount:

def \_\_init\_\_(self, account\_number, account\_holder, balance=0.0):

self.account\_number = account\_number

self.account\_holder = account\_holder

self.balance = balance

def deposit(self, amount):

if amount > 0:

self.balance += amount

print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")

else:

print("Deposit amount must be positive.")

def withdraw(self, amount):

if amount > 0 and amount <= self.balance:

self.balance -= amount

print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

else:

print("Withdrawal amount must be positive and within the available balance.")

def get\_balance(self):

return self.balance

def display(self):

print(f"Account Number: {self.account\_number}")

print(f"Account Holder: {self.account\_holder}")

print(f"Balance: ${self.balance:.2f}")

class SavingsAccount(BankAccount):

def \_\_init\_\_(self, account\_number, account\_holder, balance=0.0, interest\_rate=0.01):

super().\_\_init\_\_(account\_number, account\_holder, balance)

self.interest\_rate = interest\_rate

def add\_interest(self):

interest = self.balance \* self.interest\_rate

self.balance += interest

print(f"Interest added: ${interest:.2f}. New balance: ${self.balance:.2f}")

class CheckingAccount(BankAccount):

def \_\_init\_\_(self, account\_number, account\_holder, balance=0.0, overdraft\_limit=500.0):

super().\_\_init\_\_(account\_number, account\_holder, balance)

self.overdraft\_limit = overdraft\_limit

def withdraw(self, amount):

if amount > 0 and amount <= self.balance + self.overdraft\_limit:

self.balance -= amount

print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

else:

print("Withdrawal amount must be positive and within the available balance and overdraft limit.")

def main():

accounts = {}

while True:

print("\nMenu:")

print("1. Create Savings Account")

print("2. Create Checking Account")

print("3. Deposit")

print("4. Withdraw")

print("5. Check Balance")

print("6. Add Interest to Savings Account")

print("7. Display Account Details")

print("8. Exit")

choice = input("Enter your choice: ")

if choice == "1":

acc\_num = input("Enter account number: ")

acc\_holder = input("Enter account holder name: ")

acc\_balance = float(input("Enter initial balance: "))

interest\_rate = float(input("Enter interest rate (e.g., 0.01 for 1%): "))

accounts[acc\_num] = SavingsAccount(acc\_num, acc\_holder, acc\_balance, interest\_rate)

print("Savings account created successfully.")

elif choice == "2":

acc\_num = input("Enter account number: ")

acc\_holder = input("Enter account holder name: ")

acc\_balance = float(input("Enter initial balance: "))

overdraft\_limit = float(input("Enter overdraft limit: "))

accounts[acc\_num] = CheckingAccount(acc\_num, acc\_holder, acc\_balance, overdraft\_limit)

print("Checking account created successfully.")

elif choice == "3":

acc\_num = input("Enter account number: ")

if acc\_num in accounts:

amount = float(input("Enter amount to deposit: "))

accounts[acc\_num].deposit(amount)

else:

print("Account not found.")

elif choice == "4":

acc\_num = input("Enter account number: ")

if acc\_num in accounts:

amount = float(input("Enter amount to withdraw: "))

accounts[acc\_num].withdraw(amount)

else:

print("Account not found.")

elif choice == "5":

acc\_num = input("Enter account number: ")

if acc\_num in accounts:

balance = accounts[acc\_num].get\_balance()

print(f"Account balance: ${balance:.2f}")

else:

print("Account not found.")

elif choice == "6":

acc\_num = input("Enter account number: ")

if acc\_num in accounts and isinstance(accounts[acc\_num], SavingsAccount):

accounts[acc\_num].add\_interest()

else:

print("Savings account not found.")

elif choice == "7":

acc\_num = input("Enter account number: ")

if acc\_num in accounts:

accounts[acc\_num].display()

else:

print("Account not found.")

elif choice == "8":

print("Exiting the program.")

break

else:

print("Invalid choice. Please enter a valid option.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:-**

