1. **"Given a string s containing just the characters '(', ')', '{', '}', '[' and**

**']', determine if the input string is valid using Stack.**

**An input string is valid if:**

**I.Open brackets must be closed by the same type of brackets.**

**II.Open brackets must be closed in the correct order.**

**III.Every close bracket has a corresponding open bracket of the same type.**

**Example 2:**

**Input: s = ""()[]{}""Output: true**

**Example 3:**

**Input: s = ""(]""**

**Output: false**

**An input string is valid if:**

**I.Open brackets must be closed by the same type of brackets.**

**II.Open brackets must be closed in the correct order.**

**III.Every close bracket has a corresponding open bracket of the same type.**

**Example 1:**

**Input: s = ""()""**

**Output: true**

**Example 2:**

**Input: s = ""()[]{}""**

**Output: true**

**Example 3:**

**Input: s = ""(]""**

**Ou tput: false"**

Code:

def isValid(s: str) -> bool:

stack = []

bracket\_map = {')': '(', '}': '{', ']': '['}

for char in s:

if char in bracket\_map:

if stack and stack[-1] == bracket\_map[char]:

stack.pop()

else:

return False

else:

stack.append(char)

return not stack

# Test cases

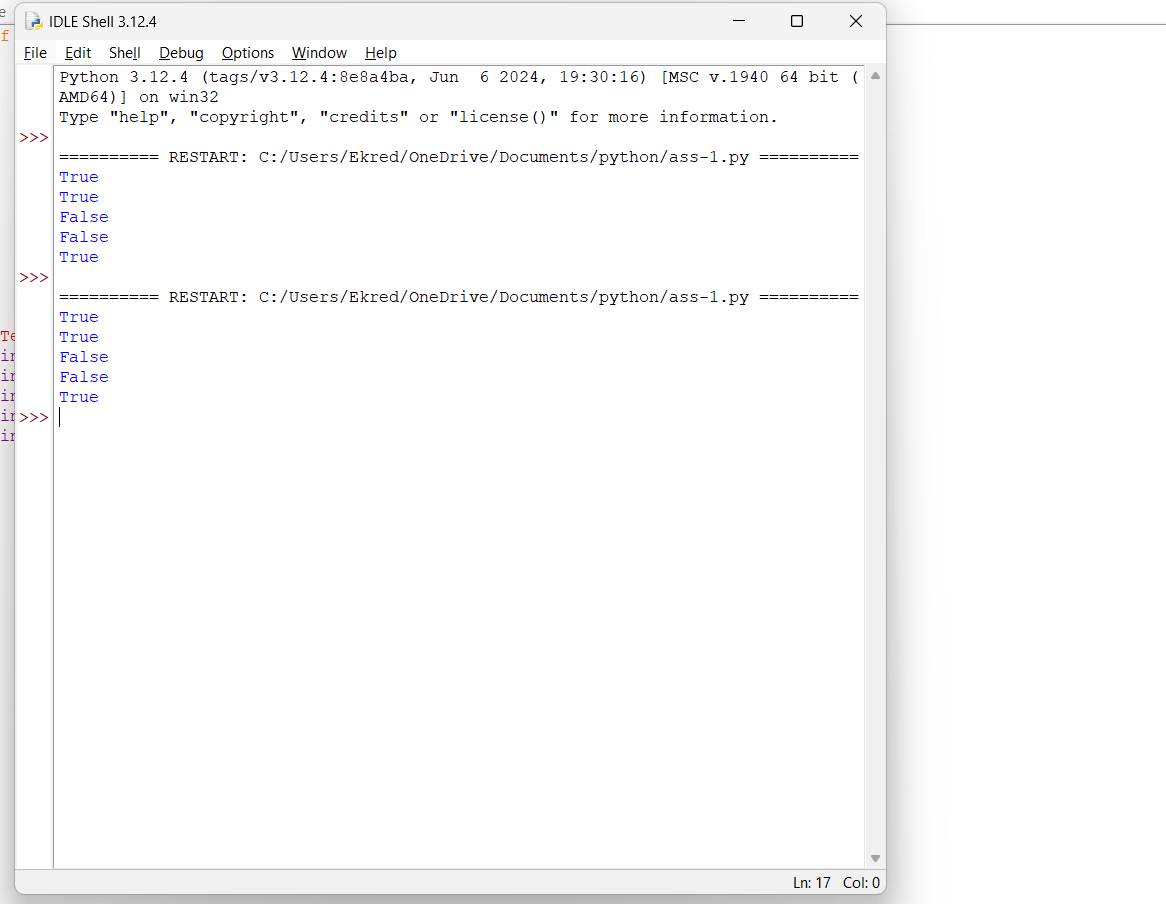
print(isValid("()")) # Output: True

print(isValid("()[]{}")) # Output: True

print(isValid("(]")) # Output: False

print(isValid("([)]")) # Output: False

print(isValid("{[]}")) # Output: True



**2) Find the year of the given Anniversary is a leap year or not. If leap year then print the**

**next Anniversary, if not leap year then print the previous Anniversary.**

**Sample Input:xx**

**Enter Date: Sample Output:**

**Given Anniversary Year: Non Leap Year. Anniversary Date: 04/1x  1/1946**

**Code:**

def is\_leap\_year(year):

# Function to check if a year is a leap year

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

return True

return False

def anniversary\_date(day, month, year):

# Function to determine the next or previous anniversary year based on leap year status

if is\_leap\_year(year):

next\_anniversary\_year = year + 1

return f"Given Anniversary Year: Leap Year. Anniversary Date: {month}/{day}/{next\_anniversary\_year}"

else:

previous\_anniversary\_year = year - 1

return f"Given Anniversary Year: Non Leap Year. Anniversary Date: {month}/{day}/{previous\_anniversary\_year}"

# Sample Input

day = int(input("Enter Day: "))

month = int(input("Enter Month: "))

year = int(input("Enter Year: "))

# Sample Output

print(anniversary\_date(day, month, year))

 **3) Write a program to print all the Non-Prime numbers between A and B?**

**Sample Input: A = 12 B = 19**

**Sample Output:**

**14, 15, 16, 18**

Code:

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

def non\_prime\_numbers(A, B):

non\_primes = []

for num in range(A, B + 1):

if not is\_prime(num):

non\_primes.append(num)

return non\_primes

# Sample Input

A = 12

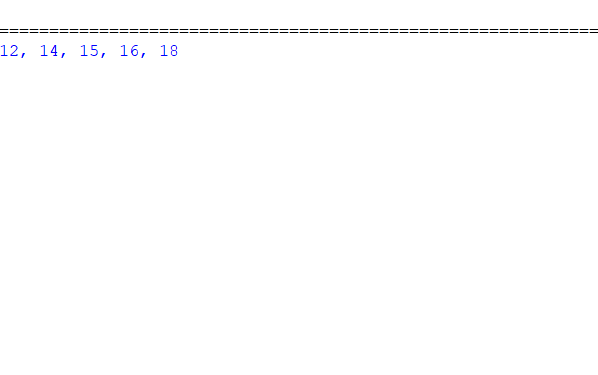
B = 19

# Sample Output

non\_primes = non\_prime\_numbers(A, B)

print(", ".join(map(str, non\_primes)))

output:



4

Print the pattern

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Code:

def print\_pattern(n):

for i in range(1, n + 1):

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

print(j, end=" ")

print()

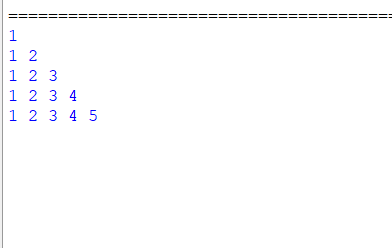
# Sample Input

n = 5

# Sample Output

print\_pattern(n)

output:



5) 5

Write a program to print the total amount available in the ATM machine with the conditions applied.

Total denominations are 2000, 500, 200, 100, get the denomination priority from the user and the total number of notes from the user to display the total available balance to the user

Sample Input:

Enter the 1st Denomination: 500 Enter the 1st Denomination number of notes: 4

Enter the 2nd Denomination: 100 Enter the 2nd Denomination number of notes: 20

Enter the 3rd Denomination: 200 Enter the 3rd Denomination number of notes: 32

Enter the 4th Denomination: 2000 Enter the 4th Denomination number of notes: 1

Sample Output:

Total Available Balance in ATM: 12400"

Code:

def calculate\_total\_balance():

total\_balance = 0

for i in range(1, 5):

denomination = int(input(f"Enter the {i}st Denomination: "))

num\_notes = int(input(f"Enter the {i}st Denomination number of notes: "))

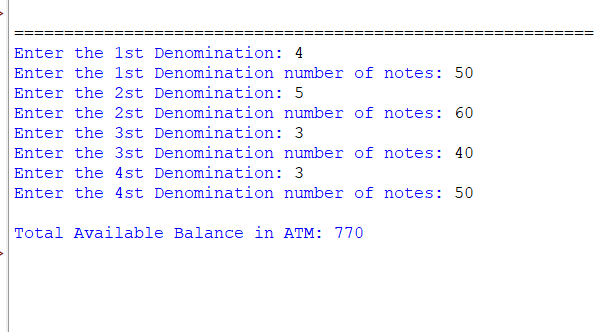
total\_balance += denomination \* num\_notes

print(f"\nTotal Available Balance in ATM: {total\_balance}")

# Call the function to execute

calculate\_total\_balance()

output:



6

Write a Python program to find the maximum of three numbers entered by the user.

Input : 3 7 2

Output : 7

Test Cases

5 8 2

10 12 16

0.1 10 0.4

a b z

code:

def find\_max\_of\_three():

# Prompt the user to enter three numbers

num1 = input("Enter the first number: ")

num2 = input("Enter the second number: ")

num3 = input("Enter the third number: ")

# Convert inputs to float

num1 = float(num1)

num2 = float(num2)

num3 = float(num3)

# Find the maximum number

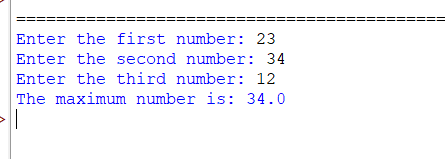
max\_num = max(num1, num2, num3)

# Print the maximum number

print(f"The maximum number is: {max\_num}")

# Call the function to execute

find\_max\_of\_three()

output:  


7

"Write a Python program to find the Nth Fibonacci number. The program should take the value of N as input where n=8

Output 0 1 1 2 3 5 8 13"

Code:

def fibonacci(n):

fib\_sequence = [0, 1]

while len(fib\_sequence) < n:

next\_number = fib\_sequence[-1] + fib\_sequence[-2]

fib\_sequence.append(next\_number)

return fib\_sequence

# Sample Input

n = 8

# Sample Output

fib\_sequence = fibonacci(n)

print(" ".join(map(str, fib\_sequence)))

output:



8

Write a Python program to create a basic calculator that can perform addition, subtraction, multiplication, and division using functions.

Code:

def add(a, b):

return a + b

def subtract(a, b):

return a - b

def multiply(a, b):

return a \* b

def divide(a, b):

if b != 0:

return a / b

else:

return "Error! Division by zero."

def calculator():

print("Select operation:")

print("1. Addition")

print("2. Subtraction")

print("3. Multiplication")

print("4. Division")

while True:

choice = input("Enter choice (1/2/3/4): ")

if choice in ['1', '2', '3', '4']:

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

if choice == '1':

print(f"{num1} + {num2} = {add(num1, num2)}")

elif choice == '2':

print(f"{num1} - {num2} = {subtract(num1, num2)}")

elif choice == '3':

print(f"{num1} \* {num2} = {multiply(num1, num2)}")

elif choice == '4':

result = divide(num1, num2)

if result == "Error! Division by zero.":

print(result)

else:

print(f"{num1} / {num2} = {result}")

# Ask if the user wants to perform another calculation

next\_calculation = input("Do you want to perform another calculation? (yes/no): ")

if next\_calculation.lower() != 'yes':

break

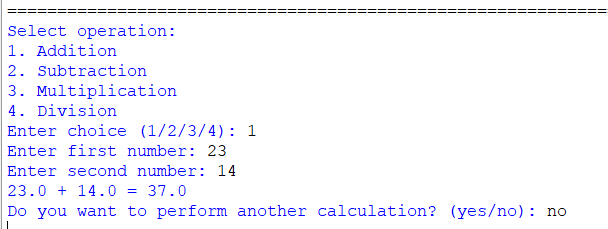
else:

print("Invalid input")

# Call the calculator function to execute

calculator()

output:



9

Sample Input:

Enter your age:

7

Sample output:

You are allowed to vote after 11 years

Test cases:

a) 25

b) Eighteen

c) 12

d) -18

e) 5. 34.5

code:

def check\_voting\_eligibility():

try:

age = float(input("Enter your age: "))

if age < 0:

print("Age cannot be negative.")

elif age < 18:

years\_left = 18 - age

print(f"You are allowed to vote after {int(years\_left)} years")

else:

print("You are eligible to vote.")

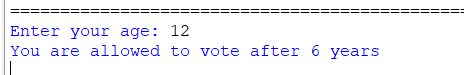
except ValueError:

print("Invalid input. Please enter a valid age.")

# Call the function to execute

check\_voting\_eligibility()

output:



10) Write a program to reverse a word using loop?(Not to use inbuilt functions)

Sample Input:

String: TEMPLE

Sample Output:

Reverse String: ELPMET

Test cases:

a) SIGN UP

b) AT-LEAST

c) 1245

d) !@#$%

e) 145\*999=144855

code:

def reverse\_string(input\_string):

reversed\_string = ""

for char in input\_string:

reversed\_string = char + reversed\_string

return reversed\_string

# Sample Input

input\_string = input("Enter a string: ")

# Sample Output

print("Reverse String:", reverse\_string(input\_string))

output:

