**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"**

**Solution:**

def isValid(s: str) -> bool:

# Dictionary to hold the mappings of closing to opening brackets

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

# Stack to keep track of opening brackets

stack = []

# Iterate over each character in the input string

for char in s:

# If the character is a closing bracket

if char in bracket\_map:

# Pop the top element from the stack if it's not empty, otherwise assign a dummy value

top\_element = stack.pop() if stack else '#'

# Check if the popped element is the matching opening bracket

if bracket\_map[char] != top\_element:

return False

else:

# If it's an opening bracket, push it onto the stack

stack.append(char)

# If the stack is empty at the end, all brackets were properly closed

return not stack

print("puneeth")

# Test cases

print(isValid("()")) # True

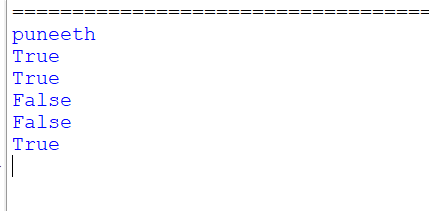
print(isValid("()[]{}")) # True

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

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

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

Output:



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**

Solution:

def is\_leap\_year(year):

# Check if the year is a leap year

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

return True

return False

def find\_anniversary(year):

if is\_leap\_year(year):

print("Leap year")

next\_anniversary = year + 1

print(f"The next anniversary is: {next\_anniversary}")

else:

print("Not a leap year")

previous\_anniversary = year - 1

print(f"The previous anniversary is: {previous\_anniversary}")

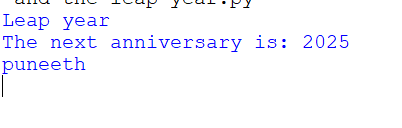
# Sample input

anniversary\_year = 2024

find\_anniversary(anniversary\_year)

print("puneeth")

output:



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

Solution:

def is\_prime(n):

# Check if n is less than 2

if n < 2:

return False

# Check if n is 2 or 3

if n in (2, 3):

return True

# Eliminate even numbers and numbers divisible by 3

if n % 2 == 0 or n % 3 == 0:

return False

# Check for factors up to the square root of n

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def print\_non\_primes(A, B):

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

if not is\_prime(num):

print(num)

# Define the range

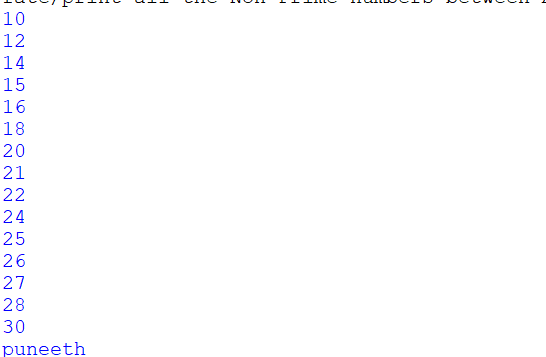
A = 10

B = 30

print\_non\_primes(A, B)

print("puneeth")

output:



4 **Print the pattern**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

**Solution:**

def print\_pattern(n):

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

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

print(j, end=" ")

print()

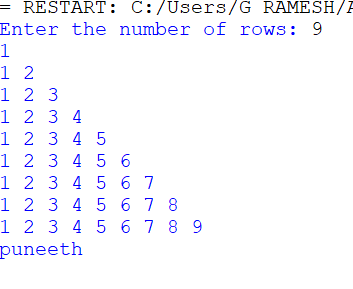
# Get the number of rows for the pattern from the user

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

print\_pattern(n)

print("puneeth")

output:



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**

Solution:

def calculate\_total\_amount(denominations, notes):

total\_amount = 0

for denomination, count in zip(denominations, notes):

total\_amount += denomination \* count

return total\_amount

def main():

# Available denominations

denominations = [2000, 500, 200, 100]

print("puneeth")

# Get denomination priority from the user

print("Enter the priority of denominations (separated by space, e.g., 2000 500 200 100):")

user\_input = input().split()

user\_priority = [int(denomination) for denomination in user\_input]

# Get the number of notes for each denomination from the user

notes = []

for denomination in user\_priority:

count = int(input(f"Enter the number of {denomination} notes: "))

notes.append(count)

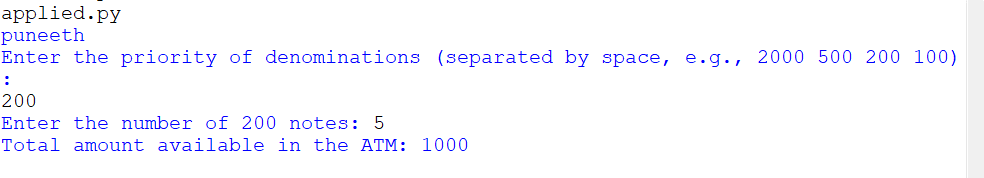
# Calculate and print the total amount

total\_amount = calculate\_total\_amount(user\_priority, notes)

print(f"Total amount available in the ATM: {total\_amount}")

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

main()

Output: 

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

Solution:

def find\_maximum(a,b, c):

# Using Python's built-in max function to find the maximum

return max(a, b, c)

def main():

print("puneeth")

# Get input from the user

try:

a = float(input("Enter the first number: "))

b = float(input("Enter the second number: "))

c = float(input("Enter the third number: "))

# Find the maximum number among the three

maximum\_number = find\_maximum(a, b, c)

# Print the result

print(f"The maximum of the three numbers {a}, {b}, {c} is: {maximum\_number}")

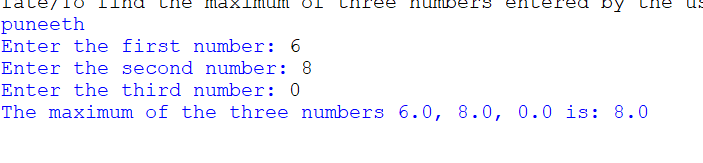
except ValueError:

print("Invalid input. Please enter numeric values.")

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

main()

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**

Solution:

print("puneeth")

def fibonacci(n):

# Check if n is less than or equal to 0

if n <= 0:

return "Invalid input. Please enter a positive integer greater than zero."

# Initialize the first two Fibonacci numbers

fib\_0, fib\_1 = 0, 1

# Calculate Fibonacci numbers iteratively

for \_ in range(2, n + 1):

fib\_next = fib\_0 + fib\_1

fib\_0 = fib\_1

fib\_1 = fib\_next

return fib\_1

def main():

# Get input from the user

try:

n = int(input("Enter the value of N to find the Nth Fibonacci number: "))

# Call the fibonacci function to find the Nth Fibonacci number

result = fibonacci(n)

# Print the result

print(f"The {n}th Fibonacci number is: {result}")

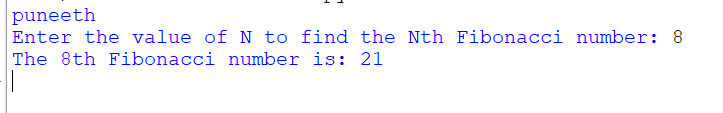
except ValueError:

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

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

main()

output:



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

Solution

print("puneeth")

# Function to perform addition

def add(x, y):

return x + y

# Function to perform subtraction

def subtract(x, y):

return x - y

# Function to perform multiplication

def multiply(x, y):

return x \* y

# Function to perform division

def divide(x, y):

# Check if the denominator is zero

if y == 0:

return "Error: Division by zero!"

else:

return x / y

def main():

print("Welcome to the Basic Calculator!")

print("Operations available:")

print("1. Addition (+)")

print("2. Subtraction (-)")

print("3. Multiplication (\*)")

print("4. Division (/)")

while True:

# Take input from the user

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

# Check if choice is one of the four options

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':

print(f"{num1} / {num2} = {divide(num1, num2)}")

# Ask user if they want to perform another calculation

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

if another\_calculation != 'yes':

break

else:

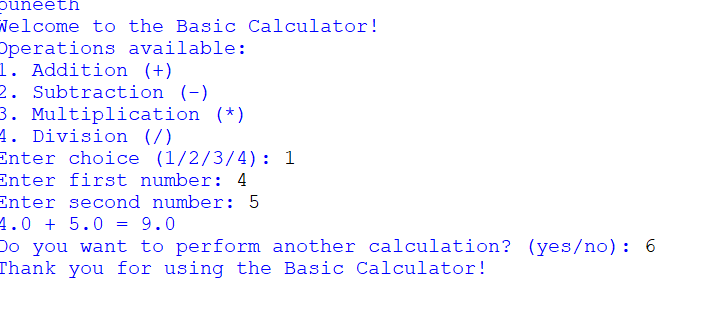
print("Invalid input. Please enter a valid choice (1/2/3/4).")

print("Thank you for using the Basic Calculator!")

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

main()

OUTPUT:



9.

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

**Sample Input:**

SOLUTION:

print("puneeth")

def reverse\_word(word):

reversed\_word = ""

# Iterate through the word from the last character to the first

for i in range(len(word) - 1, -1, -1):

reversed\_word += word[i]

return reversed\_word

# Get input from the user

word = input("Enter a word to reverse: ")

# Call the function to reverse the word

reversed\_word = reverse\_word(word)

# Print the reversed word

print(f"The reversed word is: {reversed\_word}")

output:

