```
# ### Pattern Programs: Right-Angled Triangle and Inverted Right-Angled Triangle Shapes (5
Questions)

# 1. Write a program to print a right-angled triangle of stars (*) with n rows, where n is input
by the user.
#    Example: For n = 5
#    Output:

#    *
#    **
#    * *
#    *  *
#    *****
# n=int(input("Enter no "))
# for i in range(n):
#     for j in range(i+1):
#         print("*",end="")
#     print()


# 2. Write a program to print an inverted right-angled triangle of stars (*) with n rows.
#    Example: For n = 5
#    Output:

# * * * * *
#  * * * *
#   * * *
#    * *
#     *



# 3. Write a program to print a right-angled triangle of numbers, where each row contains
increasing numbers starting from 1.
#    Example: For n = 4
#    Output:

#    1
#    12
#    123
#    1234
# for i in range(1,n+1):
#     for j in range(1,i+1):
```

```
#        print(j,end="")
#    print()
```

# 4. Write a program to print an inverted right-angled triangle of numbers in decreasing order starting from n for each row.

```
#   Example: For n = 4
#   Output:

#   4321
#   321
#   21
#   1
# n=int(input())
# for i in range(n,0,-1):
#   for j in range(i,0,-1):
#       print(j,end="")
#   print()
```

# 5. Write a program to print a right-angled triangle where the character alternates between * and # in each row.

```
#   Example: For n = 4
#   Output:

#   *
#   ##
#   ***
#   ####
# n=int(input())
# for i in range(1,n+1):
#    for j in range(1,i+1):
#        if i%2==0:
#            print('#',end="")
#        else:
#            print('*',end="")
#    print()
# 6. Write a program that accepts a list of numbers from the user and divides each number by
a user-specified divisor. Use try-except to handle division by zero.
#    Input: List = [10, 20, 30], Divisor = 0
#    Output:

#    Error: Division by zero is not allowed.
# List=list(map(int,input("Enter elements ").split(" ")))
# d=int(input("Enter Divisor "))
# l=[]

# for i in List:

#     try:
#       l.append(i//d)
#     except:
#        print('Division by zero is not allowed')
#        break
# print(l)
# output:
# Enter elements 10 20 30
# Enter Divisor 0
# Division by zero is not allowed
# Enter elements 10 20 30
# Enter Divisor 2
# [5, 10, 15]


# 7. Write a program to prompt the user for two numbers and perform division. Use try-except
to handle invalid input (non-numeric values).
#    Input: a = "abc", b = 5
#    Output:
#    Error: Please enter valid numbers.
# a=input("Enter No1 ")
# b=input("Enter no2 ")
# try:
#    d=int(a)/int(b)
#    print(d)
# except ValueError:
#    print('Please Enter Valid Number')
# output:
# Enter No1 10
# Enter no2 '10'
# Please Enter Valid Number
# Enter No1 10
# Enter no2 abc
# Please Enter Valid Number
# Enter No1 10
# Enter no2 2
# 5.0


# 8. Write a program that iterates through a list of strings and converts each to an integer. Use
try-except to skip non-numeric strings and print an error message for each.
#    Input: List = ["10", "abc", "30"]
#    Output:

#    10
#    Error: Invalid input for "abc"
#    30
# List = ["10", "abc", "30"]
# for i in List:
#    try:
#       print(int(i))
#    except ValueError:
#       print('Invalid input for ',i)
# # output:
# 10
# Invalid input for  abc
# 30


# 9. Write a program to calculate the square root of a user-provided number. Use exception
handling to manage negative inputs.
#    Input: Number = -16
#    Output:
#    Error: Cannot calculate the square root of a negative number.
# import math
# n=int(input("Enter no "))
# try:
#     print(math.sqrt(n))
# except:
#     print('Cannot calculate the square root of a negative number.')
# output:
# Enter no -10
# Cannot calculate the square root of a negative number.
# Enter no 10
# 3.1622776601683795


# 10. Write a program that asks the user to input a file name, reads the file, and prints its content. Use exception handling to handle the case when the file does not exist.
#     Input: File name = "nonexistent.txt"
#     Output:

#     Error: File not found.
# with open('sample.txt')

# 11. Write a program that accepts a list of integers and calculates their sum. If a non-integer value is encountered, skip it and display an error message.
#     Input: List = [10, "abc", 20]
#     Output:

#     Error: Invalid input for "abc"
#     Sum = 30
# List = [10, "abc", 20]
# sum=0
# for i in List:
#     try:
#         sum=sum+int(i)
#     except:
#         print("Invalid input for ",i)
# print("Sum=",sum)
# output:
# Invalid input for  abc
# Sum= 30



# 12. Write a program that continuously prompts the user for numbers and calculates the average. Allow the user to type "done" to exit and handle invalid inputs gracefully.

#     Input: 10, abc, 20, done
#     Output:

#     Error: Invalid input for "abc"
#     Average = 15.0
# print()
# v=input("Enter number/Enter done once you completed ")
# sum=0
# l=0
# while(v!='done'):
#     try:
#       sum+=int(v)
#       l+=1
#     except:
#        print("Invalid input for ",v)
#     v=input("Enter number ")
# print("Avg=",sum/l)
# output:
# Enter number/Enter done once you completed 10
# Enter number 10
# Enter number 10
# Enter number done
# Avg= 10.0



# 13. Write a program to simulate a login system where the user has 3 attempts to enter the
correct password. Use exception handling to handle invalid input types (e.g., integers instead
of strings).
#     Input: Password = 123
#     Output:

#     Error: Password must be a string.
# pwd='lava@2203'
# j=0
# for i in range(3):
#     p=input("Enter Password ")
#     try:
#        p=int(p)
#        print("password should be in form of string ")
#        j+=1
#     except:
#         if p==pwd:
#             print("logined succefully ")
#             break

#         j+=1
# if j>=3:
```

```
#     print("Password is locked")
# output:
# Enter Password 123
# password should be in form of string
# Enter Password lava@2203
# logined succefully
# Enter Password 123
# password should be in form of string
# Enter Password 123
# password should be in form of string
# Enter Password 123
# password should be in form of string
# Password is locked


# 14. Write a program to calculate the factorial of a number using a loop. Use exception
handling to catch invalid inputs (negative numbers or non-integer values).
#     Input: Number = -5
#     Output:

#     Error: Factorial is not defined for negative numbers.
# n=int(input("Enter no "))
# while(True):
#     try:
#         if n>0:
#             break
#         else:
#             n=int(input("Enter no in positive"))
#     except:
#         n=int(input("only postive integers are allowed"))
# fact=1
# for i in range(1,n+1):
#     fact*=i
# print("Factorial",fact)
# output:
# Enter no -2
# Enter no in positive5
# Factorial 120
# Enter no -2
# Enter no in positiveabc
# only postive integers are allowed5
# Factorial 120
# 15. Write a program that reads a list of integers from the user and prints the largest. Handle
cases where the list contains non-integer values or is empty.
#     Input: List = ["10", "abc", "30"]
#     Output:

#     Error: Invalid input for "abc"
#     Largest = 30
# l=list(map(str,input("Enter nos ").split(" ")))
# max=0
# for i in l:
#     try:
#         if int(i)>max:
#             max=int(i)
#     except:
#         print("Invalid input for",i)
# print("Largest element ",max)
# output:
# Enter nos 1 2 3 30 5 60 abc
# Invalid input for abc
# Largest element  60
```