EXCEPTION HANDLING			
Sr.No.	**Syntax/command**	**Output**	**Remarks**

1	a = int(input("enter first number"))
	b = int(input("enter second number"))
	try:
	  c = a/b
	  print(c)
	except:
	  print("b should not be zero")
		enter first number10
		enter second number0
		b should not be zero
			It will give except-ion case output and will not give an error for the same.

2	a = int(input("enter first number"))
	b = int(input("enter second number"))
	try:
	  c = a/b
	  print(c)
	except ZeroDivisionError :
	  print("b should not be zero")
		enter first number10
		enter second number0
		b should not be zero
			You can also mention the type of error after the except keyword. Parent keyword exception can also be used to address all types of errors.

3	try:
	  c=a/b
	  print(c)
	  d=a+b
	  print(d)
	
	except ZeroDivisionError :
	  print("b should not be zero")
	except ValueError:
	  print('please input number only')
			You can use multiple exception blocks for single try block

4	while True: print("sunny")		It will print the word infinitly until and unless you give break statement.

5	while True:
	  name = input("enter your name: ")
	  if name == "sunny":
	  print("sunny")
	  break
			Until the input given does not match the name specified, it will pop up again and again. Once matched the name will be printed just once as break is mentioned.

6	while True:
	  try:
	  a=int(input("enter first number"))
	  b=int(input('enter second number'))
	  d=a/b
	  print(d)
	  except ValueError :
	  print("there should not be string")
	  break
	  except ZeroDivisionError:
	  print("please enter non-zero denominator")
	  break
			The complete expcetion handling example

7	try:
	  a = int(input("enter a number"))
	  print(a)
	except:
	  print('string not allowed')
			To allow input of only numbers and not givie error in other cases.

8	while True:
	try:
	a=int(input("first number"))
	b=int(input('second number'))
	c=a/b
	print("div:" , c)
	break
	except:
	print("b should not be zero")
		first number10
		second number0
		b should not be zero
		first number10
		second number1
		div: 10.0
			The while true loop will run again and again unless you don't give proper values for input command. Also the loop will break only when correct input and output is given.

9	while True:
	try:
	a=int(input("first number"))
	b=int(input('second number'))
	c=a/b
	print("div:" , c)
	break
	except ZeroDivisionError as e:
	print€
		first number10
		second number0
		division by zero
			To use in-built comments for error as your own comments.

10	import sys
	while True:
	try:
	a=int(input("first number"))
	b=int(input('second number'))
	c=a/b
	print("div:" , c)
	break
	except:
	print(sys.exc_info())
		first number10
		second number0
		(<class 'ZeroDivisionError'>, ZeroDivisionError('division by zero'), <traceback object at 0x0000022A4AE41140>)
			It will define in detail the exact error that is in the code. No need to mention anything else in the exception.

11	while True:
	try:
	a=int(input("first number"))
	b=int(input('second number'))
	c=a/b
	print("div:" , c)
	break
	except:
	a,b,c=sys.exc_info()
	print("Exception class" , a)
	print("Exception message" , b)
	print("Line number", c.tb_lineno)
		first number10
		second number0
		Exception class <class 'ZeroDivisionError'>
		Exception message division by zero
		Line number 6
			In order to show in a proper format the error msg from exc_info()

12	import traceback
	while True:
	try:
	a=int(input("first number"))
	b=int(input('second number'))
	c=a/b
	print("div:" , c)
	break
	except:
	print(traceback.format_exc())
			You can also use this in place of sys.exc_info() and both the commands are used for debugging the entire project

13	while True:
	  try:
	  a=int(input("first number"))
	  b=int(input('second number'))
	  if a<0 or b<0:
	  raise Exception("neg number not allowed")
	  c=a/b
	  print("div:" , c)
	  break
	  except ValueError:
	  print("please enter int only")
	  except ZeroDivisionError:
	  print("please enter non-zero int")
	  except Exception as e:
	  print(e)
			Raise exception is used to provide exception at a specific line of code and can also be used in try block.

14	class NegativeNumberException(Exception): pass		

15	while True:
	  try:
	  a=int(input("first number"))
	  b=int(input('second number'))
	  if a<0 or b<0:
	  raise NegativeNumberException("neg number not allowed")
	  c=a/b
	  print("div:" , c)
	  break
	  except ValueError:
	  print("please enter int only")
	  except ZeroDivisionError:
	  print("please enter non-zero int")
	  except NegativeNumberException as e:
	  print(e)
			When you want to use custom exception in place of pre-defined once

16	while True:
	  try:
	  a=int(input("first number"))
	  b=int(input('second number'))
	  c=a/b
	  print(c)
	  except:
	  print("non-zero denominator")
	  finally:
	  print("hello")
	  print("python")
			The finally exception command is used to keep important code execution in last part and to keep the data safe. Case 1: We have created one database but we forgot to close the connection. Case 2: On cloud service we forgot to terminate.

17	while True:
	  try:
	  a=int(input("first number"))
	  b=int(input('second number'))
	  c=a/b
	  print(c)
	  except:
	  print("non-zero denominator")
	  else:
	  print("hello")
			Else exception shall run on correct input output combination or else exception command will run.

18	def askint1():
	  while True:
	  try:
	  val = int(input("please enter int"))
	  break
	  except:
	  print("looks like you have nor entered a int")
	  continue
			While true will run the code in continuous loop unless you enter a integer.

19	def sqrt(x):
	if not isinstance(x, (int, float)):
	raise TypeError(x must be numeric)
	elif x < 0:
	raise ValueError(x cannot be negative
			Checking the type of an object can be performed at run-time using the built-in function, **isinstance**. In simplest form, isinstance(obj, cls) returns True if object, obj, is an instance of class, cls,

20	age = −1 # an initially invalid choice
	while age <= 0:
	try:
	age = int(input(Enter your age in years:))
	if age <= 0:
	print(Your age must be positive)
	except (ValueError, EOFError):
	print(Invalid response
			We use the tuple, (ValueError, EOFError), to designate the types of errors that we wish to catch with the except-clause.

21	except (ValueError, EOFError): pass		If we preferred to have the while loop continue without printing the Invalid response message. It quietly catches the exception, thereby allowing the surrounding while loop to continue.
22	factors(100)		Generates the series 1,100,2,50,4,25,5,20,10 but not generated in increasing order
23	divmod(a, b),		Returns the pair of values (a // b, a % b)
24	quotient, remainder = divmod(a, b)		
25	for x, y in [(7, 2), (5, 8), (6, 4)]:		there will be three iterations of the loop. During the first pass, x=7 and y=2, and so on.

26	def fibonacci( ):
	a, b = 0, 1
	while True:
	yield a
	a, b = b, a+b
			Fibonacci series

27	import math a = math.sqrt(3) print(a)		When you just import modules

28	from math import sqrt a = sqrt(3) print(a)		When you import modules with defination

29	class Circle(object):
	
	# Constructor
	def __init__(self, radius=3, color='blue'):
	self.radius = radius
	self.color = color
	
	# Method
	def add_radius(self, r):
	self.radius = self.radius + r
	return(self.radius)
	
	# Method
	def drawCircle(self):
	plt.gca().add_patch(plt.Circle((0, 0), radius=self.radius, fc=self.color))
	plt.axis('scaled')
	plt.show()
			Defining a class and creating an instance of the class

30	RedCircle = Circle(10, 'red')		
31	dir(RedCircle)		Give data attributes of the object
32	RedCircle.radius		Return radius
33	RedCircle.color		Return colour
34	RedCircle.drawCircle()		Show a circle as defined