# ANSWERS

# PYTHON ASSIGNMENT – 1

1. **A python program to understand the compile time error.**

**Ans -**

# Before the program can be run, the source code must be compiled

# into the machine code. If the conversion can not performed,

# Python will inform you that your application can not be run before the error is fixed.

x = int(input('Enter a number : '))

# Gives compile time error

if x%2 == 0

    print("You have entered an even number.")

else:

    print ("You have entered an odd number.")

1. **A python program to demonstrate compile-time error.**

**Ans -**

class CompileDemo:

    x = 100

    y = 155

    # Gives compile time error

    # Comma missed - invalid syntax error

    print(x y)

CompileDemo()

1. **A python program to understand runtime errors.**

**Ans -**

# Errors that occur after the code has been compiled and the program is running, This is called runtime error.

# An example of an runtime error is the division by zero.

dividend = float(input("Enter a dividend : "))

divisor = float(input("Enter a divisor: "))

quotient = dividend/divisor

# If divisor is 0(zero) then it throws runtime error

print ("Quotient = ", quotient)

1. **A python program to demonstrate runtime error.**

**Ans -**

# Way 1 :

try:

    dividend = int(input("Please enter the dividend : "))

    divisor = int(input("Please enter the divisor : "))

    print("%d / %d = %f" % (dividend, divisor, dividend/divisor))

except ValueError:

    print("The divisor and dividend have to be numbers!")

except ZeroDivisionError:

    print("The dividend may not be zero!")

# Way 2 :

try:

    dividend = int(input("Please enter the dividend : "))

except ValueError:

    print("The dividend has to be a number!")

try:

    divisor = int(input("Please enter the divisor: "))

except ValueError:

    print("The divisor has to be a number!")

try:

    print("%d / %d = %f" % (dividend, divisor, dividend/divisor))

except ZeroDivisionError:

    print("The dividend may not be zero!")

1. **A python program to increment the salary of an employee by 15%.**

**Ans -**

oldSalaryPerMonth = int(input("Enter your old salary per month :"))

hike = 15

# Finding the incremented salary of employee by 15%

presentSalaryPerMonth = oldSalaryPerMonth + (oldSalaryPerMonth \* hike / 100)

# Incremented present salary by 15%

print("After hike your present salary per month is : ", presentSalaryPerMonth)

1. **A python program to understand the effect of an exception.**

**Ans -**

try:

    number = int(input("Please enter the number : "))

# except block handles the any exception error

except:

        print ("The input has to be a number.")

else:

    print ("Success, no error!")

1. **A python program to handle the ZeroDivisionError exception.**

**Ans -**

try:

    dividend = int(input("Please enter the dividend : "))

    divisor = int(input("Please enter the divisor : "))

    print("%d / %d = %f" % (dividend, divisor, dividend/divisor))

# If divisor = 0 throws ZeroDivisionError

except ZeroDivisionError:

    print("The dividend may not be zero!")

1. **A python program to handle syntax error given by eval() function.**

**Ans -**

try:

    eval("1 + 2) + 3")

# Handling syntax error given by eval() function

except SyntaxError:

    print("Handle the eval() function error.")

1. **A python program to handle IOError produced by open() function.**

**Ans -**

try:

    with open('file1.txt') as file:

        read\_data = file.read()

# Handling IOError produced by open() function

except IOError:

    print('Could not open file.log')

1. **A python program to handle multiple exceptions.**

**Ans -**

# Way 1 :

try:

    a=10/0

# Multiple exceptions handled by except block

except (ArithmeticError,ValueError,IndentationError) as e:

    print(e)

else:

    print("Successfully Done")

# Way 2 :

try:

    d = 8

    d = d + '5'

# Multiple exceptions handled by except block

except(TypeError, SyntaxError)as e:

    print(e)

1. **A python program to understand the usage of try with finally blocks.**

**Ans -**

# No exception Exception raised in try block

try:

    # Raises divide by zero exception.

    k = 5 // 0

    print(k)

# Handles ZeroDivision exception

except ZeroDivisionError:

    print("Can't divide by zero")

# finally block is always executed regardless of exception generation.

finally:

    print('This is always executed')

1. **A python program using the assert statement and catching AssertionError.**

**Ans -**

# Way 1 :

# Initializing number

a = 4

b = 0

# Using assert to check for 0

print("The value of a / b is : ")

assert b != 0, "Divide by 0 error"

print(a / b)

# Way 2 :

# Handling it manually

try:

    x = 1

    y = 0

    assert y != 0, "Invalid Operation"

    print(x / y)

# Error\_message provided by the user gets printed

except AssertionError as msg:

    print(msg)

1. **A python program to use the assert statement with a message.**

**Ans -**

def avg(marks):

    # Assert statement with message

    assert len(marks) != 0, "List is empty."

    return sum(marks)/len(marks)

mark1 = []

print("Average of mark1:", avg(mark1))

mark2 = [55,88,78,90,79]

print("Average of mark2 : ", avg(mark2))

1. **A python program to create our own exception and raise it when needed.**

**Ans -**

# Define Python user-defined exceptions

class Error(Exception):

   """Base class for other exceptions"""

   pass

class ValueTooSmallError(Error):

   """Raised when the input value is too small"""

   pass

class ValueTooLargeError(Error):

   """Raised when the input value is too large"""

   pass

# user guesses a number until he/she gets it right

# you need to guess this number

number = 10

while True:

    try:

        i\_num = int(input("Enter a number : "))

        if i\_num < number:

           raise ValueTooSmallError

        elif i\_num > number:

           raise ValueTooLargeError

        else:

            print("Congratulations! You guessed it correctly.")

        break

    # Raise error when entered number is smaller than given number

    except ValueTooSmallError:

        print("This value is too small, try again!")

        break

    # Raise error when entered number is grater than given number

    except ValueTooLargeError:

        print("This value is too large, try again!")

        break

1. **A python program that creates a log file with errors and critical messages.**

**Ans -**

1. **A python program to store the messages released by any exception into a log file.**

**Ans -**