Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 10

223 | Vishal Baraiya | 23010101014

Exception Handling

01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError

Note: handle them using separate except blocks and also using single except block too.

In [4]:
```
try:
    a = int(input("Enter the Number : "))
    b = int(input("Enter the Number : ")) # Value Error a = "str"
    c = a / b
    print (f"{a}/{b} = {c}") # Zero Division Error
    print('a'+1) # Type Error
except ZeroDivisionError :
    print("Zero Division Error")
except ValueError :
    print("Value Error")
except TypeError :
    print("Type Error")
```

```
12/3 = 4.0
Type Error
```

02) WAP to handle following exceptions:

1. IndexError

2. KeyError

In [14]:
```
try:
    # a = [1, 2, 3]
    # print(a[5])  # IndexError

    d = {'a':1,'b':2,'c':3}
    print(d[2])  # KeyError

except IndexError as ie:
    print(type(ie).__name__,":",ie)
    print("Index Error is Occured.")
except KeyError as ke:
    print(type(ke).__name__,":",ke)
    print("Key Error is Occured.")
```

```
1
KeyError : 2
Key Error is Occured.
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [17]:
```
try:
    import index

    # with open('demo.txt') as fp:
    #     print(fp.read())
except FileNotFoundError as e:
    print(type(e).__name__,e)
except ModuleNotFoundError as e:
    print(type(e).__name__,e)
```

```
ModuleNotFoundError No module named 'index'
```

04) WAP that catches all type of exceptions in a single except block.

In [20]:
```
try :
    print(1/0)
except Exception as e:
    print(type(e).__name__,e)
```

```
ZeroDivisionError division by zero
```

05) WAP to demonstrate else and finally block.

In [23]:
```
try:
    a = int(input("Enter the Number : "))
    b = int(input("Enter the Number : "))
    ans = a / b
    print(f"{a} / {b} = {ans}")
except Exception as e:
    print(type(e).__name__,e)
else:
    print("Else block Executed!") # It will Execute when Error Will not Occured
finally:
    print("Finally block Executed!") # It will Execute Every Time
```

```
10 / 2 = 5.0
Else block Executed!
Finally block Executed!
```

06) Create a short program that prompts the user for a list of grades separated by commas.

Split the string into individual grades and use a list comprehension to convert each string to an integer.

You should use a try statement to inform the user when the values they entered cannot be converted.

In [29]:
```
try:
    s = input("Enter the Grades using ( , ) Seprated : ")
    l = s.split(',')
    l = [int(i) for i in l]
    print(l)
except Exception as e:
    print(type(e).__name__,e)
```

```
ValueError invalid literal for int() with base 10: 'rt'
```

07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

In [36]:
```
def divide(a,b):
    try:
        return a/b
    except Exception as e:
        print(type(e).__name__,e)

divide(12,0)
```

```
ZeroDivisionError division by zero
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

In [47]:
```
try:
    age = int(input("Enter the Age : "))
    if (age < 18):
        raise ValueError("This is Value Error Genrated By me.")
    else:
        print(f"Age = {age}")

except ValueError as e:
    print(type(e).__name__,e)
```

```
Age = 18
```

09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

In [58]:
```
class InvalidUsernameError(Exception) :
    def __init__(self,msg):
        self.msg = msg
try:
    name = input("Enter the User Name : ")
    if (len(name) >= 5) and (len(name) <= 15):
        print(f"UserName : {name}")
    else:
        raise InvalidUsernameError("Username must be between 5 and 15 characters
except Exception as e:
    print(type(e).__name__,e)
```

```
InvalidUsernameError Username must be between 5 and 15 characters long
```

10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

In [65]:
```
import math
class NegativeNumberError(Exception) :
    def __init__(self,msg):
        self.msg = msg
try:
    n = int(input("Enter the Number : "))
    if (n < 0):
        raise NegativeNumberError("Cannot calculate the square root of a negativ
    else:
        print(f"sqrt Of {n} = {math.sqrt(n)}")
except Exception as e:
    print(type(e).__name__,e)
```

```
NegativeNumberError Cannot calculate the square root of a negative number.
```

In [ ]: