Darshan UNIVERSITY

योग: कर्मसु कौशलम्

# Python Programming - 2301CS404

## Lab - 12

Roll No : 352

Name : Smit Gohel

# 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 1st number: '))
    b = int(input('Enter 2nd number: '))
    name = input('Enter name: ')
    print(a / b)
    print(a + name)

except ZeroDivisionError as err:
    print(err)
except ValueError as err:
    print(err)
except TypeError as err:
    print(err)
```

```
2.5
unsupported operand type(s) for +: 'int' and 'str'
```

## 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [8]:
```
try:
    d = {1:'a', 2:'b'}
    key = int(input('Enter key: '))
    d[key]

    l = [1, 2, 3]
    idx = int(input('Enter index: '))
    print(l[idx])

except IndexError as err:
    print(err)

except KeyError as err:
    print(err, type(err).__name__)
```

```
5 KeyError
```

## 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [24]:
```
try:
    file = input('Enter file name: ')
    fp = open(file+".txt", 'r')
    fp.read()

    import smit

except FileNotFoundError as err:
    print(err)
except ModuleNotFoundError as err:
    print(err)

finally:
    fp.close()
```

```
No module named 'smit'
```

## 04) WAP that handles all type of exceptions in a single except block in standard error message format.

In [25]:
```
try:
    a = int(input('Enter 1st number: '))
    b = int(input('Enter 2nd number: '))
    name = input('Enter name: ')
    print(a / b)
    print(a + name)

    d = {1:'a', 2:'b'}
    key = int(input('Enter key: '))
    d[key]

    l = [1, 2, 3]
```

```
    idx = int(input('Enter index: '))
    print(l[idx])

    file = input('Enter file name: ')
    fp = open(file+".txt", 'r')

except Exception as err:
    print(err)

finally:
    fp.close()
```

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

## 05) WAP to demonstrate else and finally block.

In [29]:
```
try:
    a = int(input('Enter 1st number: '))
    b = int(input('Enter 2nd number: '))

except Exception as err:
    print(err)

else:
    print(a / b)

finally:
    print("Exit")
```

```
0.5
Exit
```

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

In [35]:
```
def divide(a,b):
    try:
        res = a / b

    except Exception as err:
        print(err)

    else:
        print(res)

    finally:
        print("Exit")

divide(8,8)
```

```
1.0
Exit
```

## 07) WAP to accept item prices and calculate total. Raise ValueError if price is negative, otherwise print the total bill amount.

In [38]:
```
try:
    price = int(input('Enter Price: '))
    total = int(input('Enter Total Items: '))

    if price < 0:
        raise ValueError('Price must be Postive.')

except ValueError as err:
    print(err)

else:
    total_bill = price * total
    print(f'Total Bill: {total_bill}')

finally:
    print('Exit')
```

```
Price must be Postive.
Exit
```

## 08) 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 [85]:
```
try:
    user = input('Enter Grades: ').split()

    for i in user:
        i = int(i)

except Exception as err:
    print(err)
```

## 09) Accept 5 subject marks (0-100). Use assert for validation.

### Sample Input (Error Case)

Enter marks: 110

### Sample Output (Error Case)

AssertionError: Marks should be between 0 and 100.

### Sample Input (Normal Case)

Valid marks entered

### Sample Output (Normal Case)

Average Marks displayed

In [54]:
```
try:
    li = []
    for i in range(1,6):
        li.append(int(input(f'Enter marks of subject {i}: ')))
        assert li[i-1] < 100 and li[i-1] > 0, 'Error: Marks range (0-100)'

except AssertionError as err:
    print(err)
```

```
Error: Marks range (0-100)
```

## 10) WAP that gets password from user. Password must have at least 8 characters and one digit.

## Raise WeakPasswordError if invalid, print the messsage "Correct" otherwise.

In [62]:
```
class WeakPasswordError(Exception):
    pass

try:
    password = input('Enter Password: ')

    if len(password) < 8:
        raise WeakPasswordError('Weak Password')
    elif not password.isalnum():
        raise WeakPasswordError('At least one digit')
    else:
        print('Correct')
except WeakPasswordError as err:
    print(err)
```

```
Correct
```

## 11) 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 [67]:
```
import math
# from math import *
class NegativeNumberError(Exception):
    pass

try:
    number = int(input('Enter number: '))
    if number < 0:
        raise NegativeNumberError('Cannot calculate the square root of a negativ

    res = math.sqrt(number)
    print(f'Square root: {res}')
```

```
except NegativeNumberError as err:
    print(err)
```

```
Square root: 2.0
```