
# Python Programming - 2301CS404

# Lab - 10

23010101161 - Smit Maru - 260

## 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 [8]:
```
try:
    a = 11
    b = 0
    c = '0'
    print(a/b)
except ZeroDivisionError:
    print("Number not Divisiable by zero")
try:
    d = int(input("Enter number"))
except ValueError:
    print("Enter integer number")
try:
    print(a+c)
except TypeError:
    print("Number is not concat with string/char")
```

```
Number not Divisiable by zero
```

```
Enter integer number
Number is not concat with string/char
```

## 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [7]:
```python
l1 = [10,20,30,40]
try:
    print(l1[4])
except IndexError:
    print("Index is out of range")
d1 = {1:'a',2:'b'}
try:
    print(d1[3])
except:
    print("Key not found in d1")
```

```
Index is out of range
Key not found in d1
```

## 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [12]:
```python
try:
    fp = open("abc.txt",'r')
    print(fp.read())
except FileNotFoundError:
    print("File not found")
try:
    import aaaa
except ModuleNotFoundError:
    print("Module not found")
```

```
File not found
Module not found
```

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

In [18]:
```python
try:
    a = 11
    b = 0
    c = '0'
    d = int(input("Enter number"))
    print(a+c)
    print(a/b)
except Exception as e:
    print(type(e).__name__+" : "+str(e))
```

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

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

In [19]:
```
try:
    print(11/0)
except ZeroDivisionError:
    print("Number can not devide by zero")
finally:
    print("Finally")
try:
    print(11/2)
except ZeroDivisionError:
    print("Number can not devide by zero")
else:
    print("Else")
```

```
11.0
Finally
5.5
Else
```

## 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 [35]:
```
try:
    s = input("Enter comma-separated numbers: ")
    a = s.split(',')
    numbers = [int(i) for i in a]
    print("Converted numbers:", numbers)
except ValueError:
    print("values could not be converted to an integer.")
```

```
Converted numbers: [1, 2, 3, 4]
```

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

In [32]:
```
class zero(Exception):
    pass
try:
    a = 1
    b = 0
    if b != 0:
        print(a/b)
    else:
```

```
        raise zero
except zero:
    print("Number not devide by zero")
```

```
Number not devide 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 [22]:
```
class ageError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    age = int(input("Enter the age"))
    if age > 18:
        print(age)
    else:
        raise ageError("Age grate then 18")
except ageError as e:
    print(e)
```

```
19
```

## 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 [25]:
```
class InvalidUsernameError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    s = input("Enter the name")
    if (len(s) >= 5) and (len(s) <= 15):
        print(s)
    else:
        raise InvalidUsernameError("Inavlid User Name")
except InvalidUsernameError as e:
    print(e)
```

```
smitmaru
```

## 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 [29]:
```
class NegativeNumberError(Exception):
    def __init(self,msg):
        msg = self.msg
try:
    a = int(input("Enter the age"))
    if a > 0:
        print(a)
    else:
        raise NegativeNumberError("a must be grate then 0")
except NegativeNumberError as e:
    print(e)
```

```
a must be grate then 0
```