t94xudnx3

March 12, 2025

Python Programming - 2301CS404

<h1>23010101202 | Khushi Patel | 31-01-25

Lab - 10

1 Exception Handling

1.0.1 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.

[24]:
```
try:
    a=int(input("enter number 1"))
    b=input("enter number 2")
    print(a+b)
except ZeroDivisionError:
    print("cannot divide by zero")
except ValueError:
    print("value error")
except TypeError:
    print("Type error")
```

```
enter number 1 1
enter number 2 0
```

```
Type error
```

[ ]:
```

```

1.0.2 02) WAP to handle following exceptions:

1. IndexError
2. KeyError

[30]:
```
# KEY ERROR
# try:
```

```
#     d={1:'a',2:'b',3:'c'}
#     index=int(input("enter the key"))
#     print(d[index])
# except KeyError:
#     print("Key not found")


# INDEX ERROR
try:
    l=[1,2,3,4,5,6,7,8,9]
    index=int(input("enter the index"))
    print(l[index])
except IndexError:
    print("index out of range")
```

```
enter the index 15
```

```
index out of range
```

1.0.3 03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

[36]:
```
# FILE NOT FOUND ERROR
# try:
#     fp=open('file10.txt','r')
#     print(fp.read())
#     fp.close()
# except FileNotFoundError:
#     print("File not found")

# MODULE NOT FOUND ERROR
try:
    import maths
except:
    print("module not found")
```

```
module not found
```

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

[53]:
```
try:
    # import maths
    a=10
    b=0
    c='a'
    print(a/b)
```

```
    # print(a+c)
except Exception as err :
    print(type(err).__name__)
```

```
ZeroDivisionError
```

1.0.5 05) WAP to demonstrate else and finally block.

[56]:
```
try:
    fp=open('file3.txt','r')
except FileNotFoundError:
    print("File not found")
else:
    print(fp.read())
    fp.close()
finally:
    print("BYEE")
```

```
File not found
BYEE
```

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

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

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

[61]:
```
try:
    grade=input("enter grades seprated by commas");
    a=grade.split(',')
    g_list=[int(i) for i in a]
    print(g_list)
except Exception as err:
    print(' values cannot be converted')
```

```
enter grades seprated by commas a,b,c,d,f
```

```
cannot be converted
```

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

[64]:
```
def divide(a,b):
    try:
        print(a/b)
    except ZeroDivisionError:
        print("cannot divide by zero")
```

```
divide(10,0)
```

```
cannot divide by zero
```

1.0.10 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.

[68]:
```
try:
    age=int(input("Enter age"))
    if(age>18):
        print(age)
    else:
        raise ValueError()
except Exception as err:
    print("Enter valid age")
```

```
Enter age 25
```

```
25
```

1.0.11 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.

[70]:
```
class InvalidUsernameError(Exception):
    def __init(self,msg):
        self.msg=msg
try:
    username=input("Enter username")
    if(len(username)>15 or len(username)<5):
        raise InvalidUsernameError()
    else:
        print(username)
except InvalidUsernameError:
    print("Username must be between 5 and 15 characters long");
```

```
Enter username khu
```

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

1.0.12 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.

[74]:
```
class NegativeNumberError(Exception):
    def __init(self,msg):
        self.msg=msg
try:
    num=int(input("Enter Number"))
    if(num<0):
        raise NegativeNumberError()
    else:
        print(num**0.5)
except NegativeNumberError:
    print("Cannot calculate the square root of a negative number");
```

```
Enter Number -5
```

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

[ ]:
```

```