Predefined Exception classes

ValueError

Raised when an operation or function receives an argument that has the right type but an inappropriate value

ZeroDivisionError

Raised when the second argument of a division or modulo operation is zero.

Try block with multiple except blocks

If try block raises more than one exception, it is handled by using multiple except blocks.

```
try:
  statement-1
  statement-2

except <error-type>:
    statement-1
except <error-type>:
   statement-2
```

Example

```
while True:
   try:
      a=int(input("Enter first integer "))
      b=int(input("Enter second integer "))
      c=a/b
      print(f"division of {a}/{b} is {c}")
      break
   except ValueError:
      print("input must be integer type")
   except ZeroDivisionError:
      print("cannot divide number with zero")
```

Output

```
Enter first integer 4
Enter second integer abc
input must be integer type
Enter first integer 5
Enter second integer 2
division of 5/2 is 2.5

Enter first integer 5
Enter second integer 0
cannot divide number with zero
Enter first integer 6
Enter second integer 3
division of 6/3 is 2.0
```

KeyError

Raised when a mapping (dictionary) key is not found in the set of existing keys.

Example:

```
stud_data={101:[40,50],
          102:[60,70],
          103:[90,80],
          104:[60,50]}

while True:
    print("Student Result Processing")
    try:
        rollno=int(input("Rollno "))
        marks=stud_data[rollno]
        total=sum(marks)
        avg=total/2
        result="pass" if marks[0]>=40 and marks[1]>=40 else "fail"
        print(rollno,marks,total,avg,result,sep="\n")
        break
    except KeyError:
        print("Invalid Rollno")
```

```
    except ValueError:
        print("Rollno must be integer")
```

Output

```
Student Result Processing
Rollno 109
Invalid Rollno
Student Result Processing
Rollno abc
Rollno must be integer
Student Result Processing
Rollno 102
102
[60, 70]
130
65.0
Pass
```

try block with one except block to handle multiple types of errors

try block with generic except block is able to handle multiple types of errors. An except block without type is called generic except block.

Syntax:

```
try:
  statement-1
  statement-2
except:   → generic except block
   statement-3
```

Example:

```
list1=[10,20,30,40,50,60,70,80,90,100]

try:
    index=int(input("Enter Index to Read Value "))
    value=list1[index]
```

```
    print(f'Value at this {index} is {value}')
except:
    print("input must be integer or index not within range")
```

Output

```
Enter Index to Read Value 20
input must be integer or index not within range

Enter Index to Read Value abc
input must be integer or index not within range
```

except block with multiple error types

except block followed by one or more than one error type is able handle multiple types of errors.

Syntax:

```
try:
   statement-1
   statement-2
except (error-type,error-type,..):
    statement-3
```

Example:

```
while True:
    try:
        a=int(input("Enter first integer "))
        b=int(input("Enter second integer "))
        c=a/b
        print(f'Division of {a}/{b}={c}')
        break
    except (ValueError,ZeroDivisionError):
        print("input value must integer and second value should not be
zero")
```

Output

```
Enter first integer 6
Enter second integer 0
```

input value must integer and second value should not be zero
Enter first integer abc
input value must integer and second value should not be zero
Enter first integer 6
Enter second integer 2
Division of 6/2=3.0

finally

finally is not error handler block.
finally block is executed after execution of try block or except block.

What is use of finally block?
Finally block is used to de-allocate resources allocated within try block.

Resource allocation
Example: open connection to database, open file, establishing connection to printer,….

Syntax 1:

```
try:
  statement-1
  statement-2
except <error-type>:
  statement-3
except <error-type>:
  statement-4
finally:
  statement-5
```

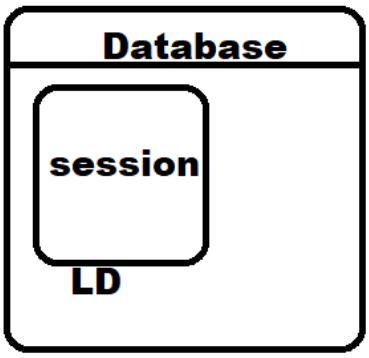
Syntax 2:

```
try:
  statement-1
  statement-2
finally:
```

statement-3

Note: try block followed by one finally block.

```
try:
   open connection to database
   send SQL statements
except <type>:
   handle error
finally:
   close database connection
```



Finally block is executed,

1. After execution of try block
2. If there is an error inside try block and handled by except block
3. If there is error inside try block and not handled by except block (after executing finally block terminates execution of program)
4. If forced exit occurs using break and return statement

Example

```
try:
   print("inside try block")
   a=int(input("enter first integer value"))
   b=int(input("enter second integer value "))
   c=a/b
   print(f'the division of {a}/{b} is {c}')
except ValueError:
   print("inside except block")
finally:
   print("inside finally block")
```

Output

```
inside try block
enter first integer value5
enter second integer value 2
the division of 5/2 is 2.5
inside finally block

inside try block
enter first integer value5
enter second integer value abc
inside except block
inside finally block

inside try block
enter first integer value6
enter second integer value 0
inside finally block
Traceback (most recent call last):
  File "E:/python5pmjun/extest7.py", line 5, in <module>
    c=a/b
ZeroDivisionError: division by zero
```

Example: