try block with multiple except blocks
if try block raises multiple exceptions or errors, it is handled by using multiple except blocks

```
try:
   statement-1
   statement-2
except <error-type>:
   statement-3
except <error-type>:
   statement-4
except <error-type> as <object-name>:
    statement-4
```

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

```
int(“65”)
int(“abc”) ➔ ValueError
int(“1.5”) ➔ ValueError
```

exception TypeError
Raised when an operation or function is applied to an object of inappropriate type

```
list(range(1,11))
list({10,20,30,40})
list(“PYTHON”)
list(10) ➔ TypeError
```

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

```
Dict1={1:100,2:200,3:300}
Dict1[1] ➔ 100
```

Dict1[2] ➔ 200
Dict1[3] ➔ 300
Dict1[4] ➔ KeyError

Example:

```
def main():
    users_dict={'naresh':'nit123',
                'suresh':'s123',
                'kishore':'k123',
                'rajesh':'r123'}
    try:
        print("*****Login*****")
        uname=input("UserName :") # nit
        p=users_dict[uname]
        pwd=input("Password :") # n123
        if p==pwd:
            print(f'{uname} welcome')
        else:
            print("invalid password")
    except KeyError as k:
       print("invalid user name")


main()
```

Output:

```
*****Login*****
UserName :naresh
Password :nit123
naresh welcome
>>>

*****Login*****
UserName :nit
invalid user name
>>>
```

Example:

```
def main():
    try:
        n1=int(input("enter the value of n1"))
        n2=int(input("enter the value of n2"))
        n3=n1/n2
        print(n1,n2,n3)
    except ZeroDivisionError:
        print("cannot divide number with zero")
    except ValueError:
        print("value must be integer")

main()
```

Output:

```
enter the value of n15
enter the value of n22
5 2 2.5
>>>
========= RESTART: C:/Users/user/Desktop/python6pm/py166.py
=========
enter the value of n15
enter the value of n20
cannot divide number with zero
>>>
========= RESTART: C:/Users/user/Desktop/python6pm/py166.py
=========
enter the value of n1abc
value must be integer
>>>
```

Except block without any type is called generic except block, this block is able to handle any type of error.

Example:

```
import sys
def main():
    try:
        n1=int(input("enter the value of n1"))
        n2=int(input("enter the value of n2"))
        n3=n1/n2
        print(n1,n2,n3)
    except:
        t=sys.exc_info()
        print(t[1])


main()
```

Output:

```
enter the value of n15
enter the value of n20
division by zero
>>>
```

sys.exc_info()
This function returns a tuple of three values that give information about the exception that is currently being handled.
"sys" is a predefined module. This module consist of the functions belongs to python virtual machine.

finally block
finally block is not error handler
it is block which is used to de-allocate the resources allocated with in try block.
Finally is executed after executing try block or except block
This code which is common for try and except block is written inside finally block.

Syntax1:

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

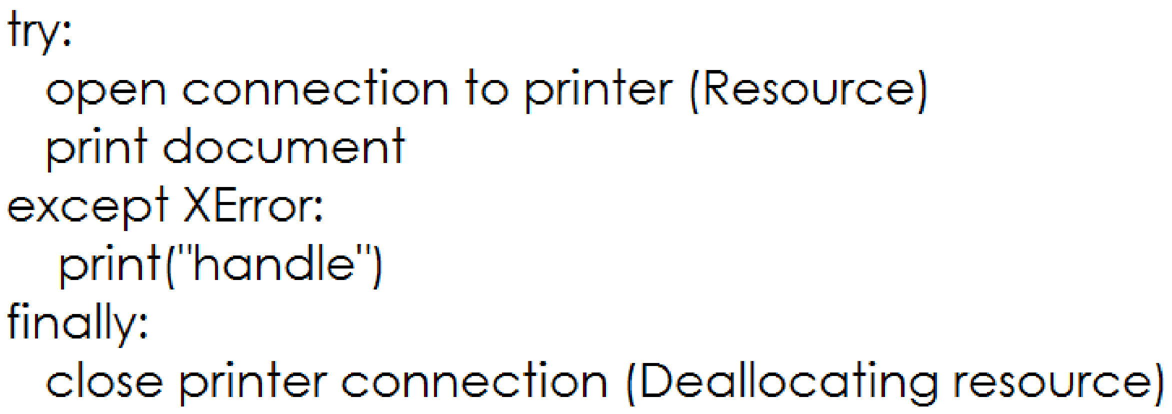
Syntax2:

```
try:
  statement-1
  statement-2
finally:
  statement-3
```

When finally block is executed?

1. After execution of try block (NO Error)

2. After execution of except block (There is error inside try block and handle by except block)
3. If there is an error inside try block not handled by except block (unhandled error), it execute finally and terminates execution of program
4. When forced exit occurs using break and return statements



```
try:
  open connection to printer (Resource)
  print document
except XError:
   print("handle")
finally:
  close printer connection (Deallocating resource)
```

Try block followed by only one finally block.

Example:

```
def main():
   try:
      print("inside try block")
      n1=int(input("first number"))
      n2=int(input("second number"))
      n3=n1/n2
      print(n3)
   except ZeroDivisionError:
      print("inside except block")
   finally:
      print("inside finally block")

   print("continue...")
main()
```

Output:

```
inside try block
first number5
```

```
second number2
2.5
inside finally block
continue...
>>>
========= RESTART: C:/Users/user/Desktop/python6pm/py167.py
=========
inside try block
first number5
second number0
inside except block
inside finally block
continue...
>>>
========= RESTART: C:/Users/user/Desktop/python6pm/py167.py
=========
inside try block
first number5
second numberabc
inside finally block
Traceback (most recent call last):
  File "C:/Users/user/Desktop/python6pm/py167.py", line 14, in <module>
    main()
  File "C:/Users/user/Desktop/python6pm/py167.py", line 5, in main
    n2=int(input("second number"))
ValueError: invalid literal for int() with base 10: 'abc'
>>>
```