Try block

Try block contains the statements which have to be monitored for exception handling (OR) the statements which generate error during runtime are included inside try block.

Syntax:
```
try:
  statement-1
  statement-2
```

except block
If there is an error within try block, it is handled by except block. Except block is error handler block. Try block followed by one or more than one except block.

Syntax1: try with one except	Syntax2: try with multiple except

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

Example of try with one except block
```
n1=int(input("Enter first number "))
n2=int(input("Enter second number "))
try:
   n3=n1/n2
   print(f'division of {n1}/{n2}={n3}')
```

```
except ZeroDivisionError:
    print("cannot divide number with zero,try again")


print("continue")
```
Output
```
Enter first number 8
Enter second number 2
division of 8/2=4.0
continue

Enter first number 7
Enter second number 0
cannot divide number with zero,try again
continue
```

if try block generates one error, it is handled using one except block. If try block generates more than one error, it is handled using multiple except blocks.

Example:
```
try:
    n1=int(input("Enter first number "))
    n2=int(input("Enter second number "))
    n3=n1/n2
    print(f'division of {n1}/{n2}={n3}')
except ZeroDivisionError:
    print("cannot divide number with zero,try again")
except ValueError:
    print("input only integer type")
print("continue")
```

Output

```
Enter first number 5
Enter second number 2
division of 5/2=2.5
continue

Enter first number 7
Enter second number 0
cannot divide number with zero,try again
continue

Enter first number abc
input only integer type
continue
```

Example

```
# Login Application or Program
users={'naresh':'n123','suresh':'s321','kishore':'k567'}
try:
   user=input("UserName :")
   pwd=input("Password :")
   if users[user]==pwd:
      print(f'{user} welcome')
   else:
      print("invalid password")
except KeyError:
   print("invalid username")
```

Output

```
UserName :suresh
```

```
Password :s321
suresh welcome

UserName :ramesh
Password :r123
invalid username
```

How to handle multiple errors using one except block?

One except block handles only one type error. In order to handle multiple errors except block is defined without error type.
Except block without error type is called generic except block.

```
try:
        statement-1
        statement-2
except: □ generic except block
        statement-3
        statement-4
```

Example:

```
email_list=['naresh@nareshit.com',
          'suresh@gmail.com','kishore@gmail.com','raman@gmail.com']

try:
   index=int(input("Enter Index of Email-Id :"))
   email_id=email_list[index]
   print(f'EmailId is {email_id}')

except: # Generic except Block
   print("Invalid Index or input value must be integer")
```

Output

```
Enter Index of Email-Id :2
EmailId is kishore@gmail.com
>>>
Enter Index of Email-Id :8
Invalid Index or input value must be integer
>>>
Enter Index of Email-Id :abc
Invalid Index or input value must be integer
>>>
```

How to get information about error object handled by PVM?
sys module provides exc_info() function, which returns exception object or error object currently hold by PVM.

exc_info() function return tuple.
1. type of error
2. error object
3. traceback object

Example:

```
import sys
email_list=['naresh@nareshit.com',
          'suresh@gmail.com','kishore@gmail.com','raman@gmail.com']

try:
   index=int(input("Enter Index of Email-Id :"))
   email_id=email_list[index]
   print(f'EmailId is {email_id}')

except:
   t=sys.exc_info()
```

```
    print(t)
    print(t[1])
```

Output

```
Enter Index of Email-Id :8
(<class 'IndexError'>, IndexError('list index out of range'), <traceback
object at 0x000001F924926B00>)
list index out of range

Enter Index of Email-Id :abc
(<class 'ValueError'>, ValueError("invalid literal for int() with base 10:
'abc'"), <traceback object at 0x000002871A343480>)
invalid literal for int() with base 10: 'abc'
```

finally block

finally is not error handler block.
finally block get executed after execution of try block or except block.
finally block is used to de-allocate resources allocated within try block.

Syntax1: try-except-finally	**Syntax-2: try-finally**
try: statement-1 statement-2 except <error-type>: statement-3 finally: statement-4	try: statement-1 statement-2 finally: statement-3

Example:

```
try:
    print("inside try block")
    n1=int(input("Enter First Number "))
    n2=int(input("Enter Second Number "))
    n3=n1/n2
    print(f'division of {n1}/{n2}={n3}')

except ZeroDivisionError:
    print("inside except block")
finally:
    print("inside finally block")

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

Output:

```
inside try block
Enter First Number 4
Enter Second Number 2
division of 4/2=2.0
inside finally block
continue...

inside try block
Enter First Number 8
Enter Second Number 0
inside except block
inside finally block
continue...

inside try block
Enter First Number abc
inside finally block
Traceback (most recent call last):
  File "E:/python7amdec23/ex7.py", line 3, in <module>
    n1=int(input("Enter First Number "))
ValueError: invalid literal for int() with base 10: 'abc'
```

raise keyword