Exception Handling

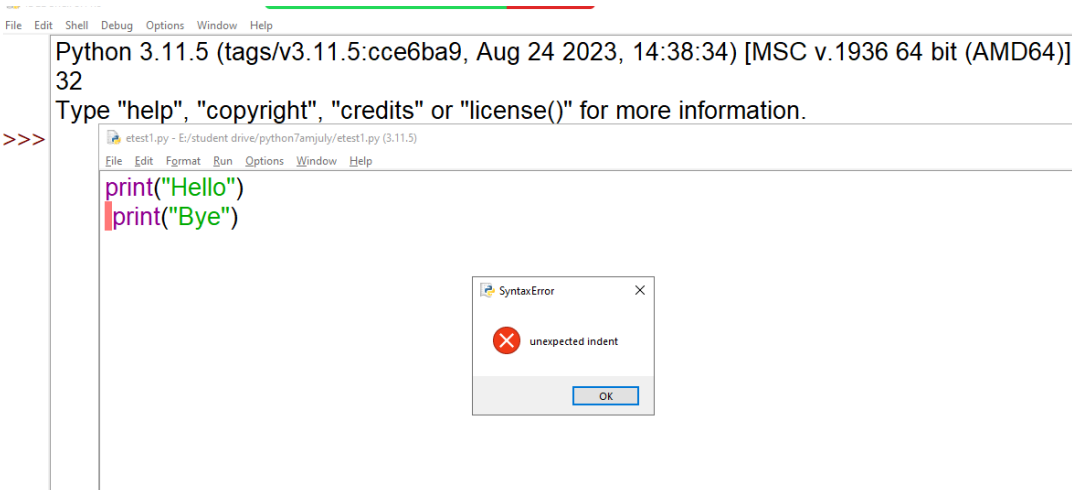
Types of Errors

While development of application or project, programmer come across 3 types of errors.

1. Compile time errors
2. Logical errors
3. Runtime errors

Compile time errors

All syntax errors are called compile time errors. If there is syntax error within program, program is not executed.
Syntax errors have to be rectified by programmer in order to execute program.
All compile time errors are syntax errors.



Logical Error

Logic is process of solving given problem. If there is an error in writing logic in order to solve given problem, program display logical error.
If there is a logical error within program, program display wrong result.
Logical error must be rectified by programmer in order to display correct result.

Example:
Write a program to find input number is even or odd

```
num=int(input("Enter any number "))
if num%2==0:
    print(f'{num} is odd')
else:
    print(f'{num} is even')
```

Output:
Enter any number 6
6 is odd

Runtime error
The error which occurs during execution of program is called runtime error. Runtime error occurs because of wrong input given by end user. When there is a runtime error within program, program execution is terminated.

Exception is an error which occurs during runtime.
Exception is a runtime error.
Exception handling is a process of handling errors occurs during runtime.

Need of exception handling
1. To avoid abnormal termination of program
2. To convert predefined error messages to user defined error messages
3. To separate business logic and error logic

Keywords used to handle exceptions
1. try
2. except
3. finally
4. raise

try keyword or try block

try block consists of the statements which has to monitored for exception handling or error handling (OR) try block contains the statements which generate errors during runtime.

Syntax:
try:

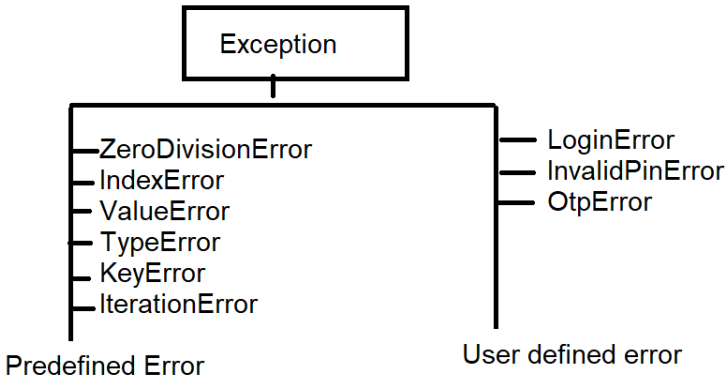
```
  statement-1
  statement-2
```

except block

except block is error handler block.
If there is an error inside try block, that error is handled by except block.
Try block followed by one or more than one except block.

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

Every error is one type or class. All predefined error types are inherited from Exception class. Predefined error types are used by python library functions.



Generating error is nothing but creating error object and giving that error object to PVM (Python Virtual Machine). PVM is having default error handler, if there is no error handler within program, that error is handle by default handler provided by PVM. Default error handler display error message and terminates execution of program. To avoid abnormal termination of program, program required user defined error handler.

Example:

```
# Write a program to divide two numbers

num1=int(input("Enter First Number "))
num2=int(input("Enter Second Number "))
try:
    num3=num1/num2
    print(f'{num1}/{num2}={num3}')
except ZeroDivisionError:
    print("cannot divide number with zero")
```

Output:

```
Enter First Number 5
Enter Second Number 2
5/2=2.5

Enter First Number 5
Enter Second Number 0
cannot divide number with zero
```

try block with multiple except blocks

if try block generate more than one exception or error, it is handled by writing multiple except blocks.

Example:

```
# Write a program to divide two numbers

try:
    num1=int(input("Enter First Number "))
    num2=int(input("Enter Second Number "))
    num3=num1/num2
    print(f'{num1}/{num2}={num3}')
except ZeroDivisionError:
    print("cannot divide number with zero")
except ValueError:
    print("value must be integer type")
```

Output:

```
Enter First Number 4
Enter Second Number abc
value must be integer type
```

Enter First Number abc
value must be integer type

Example:

```
# Write a program to divide two numbers

try:
    num1=int(input("Enter First Number "))
    num2=int(input("Enter Second Number "))
    num3=num1/num2
    print(f'{num1}/{num2}={num3}')
except ZeroDivisionError as a:
    print("cannot divide number with zero")
    print(a,type(a))
except ValueError as b:
    print("value must be integer type")
    print(b,type(b))
```

Output:

Enter First Number 5
Enter Second Number a
value must be integer type
invalid literal for int() with base 10: 'a' <class 'ValueError'>

except block without exception type or error type is called generic except block.