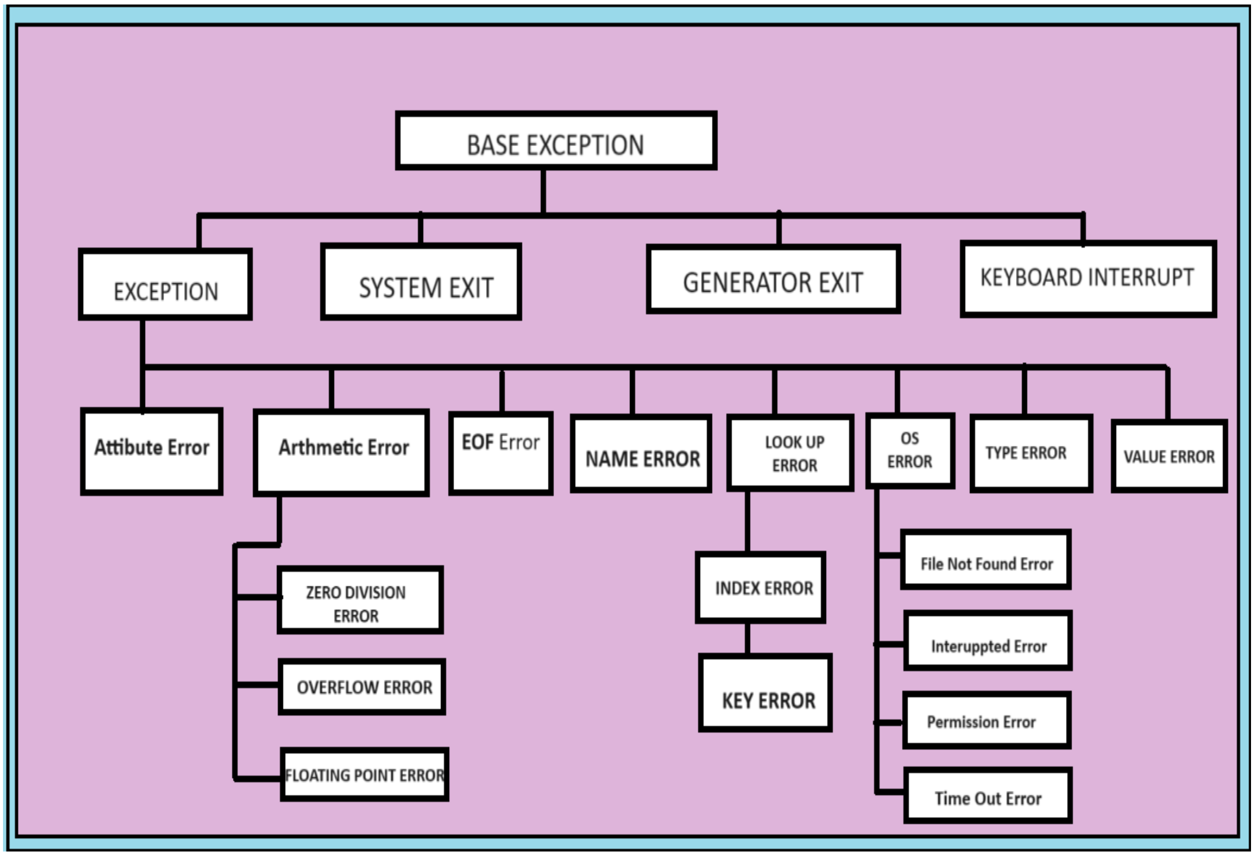
Error Handling

An Exception is an error that happens during the execution of a program. When that error occurs, Python generates an exception that can be handled, which prevents your program from crashing.

Why do we use Exceptions?
Exceptions are convenient for handling errors and special conditions in a program. When you think that you have a code that can produce an error then you can use exception handling.



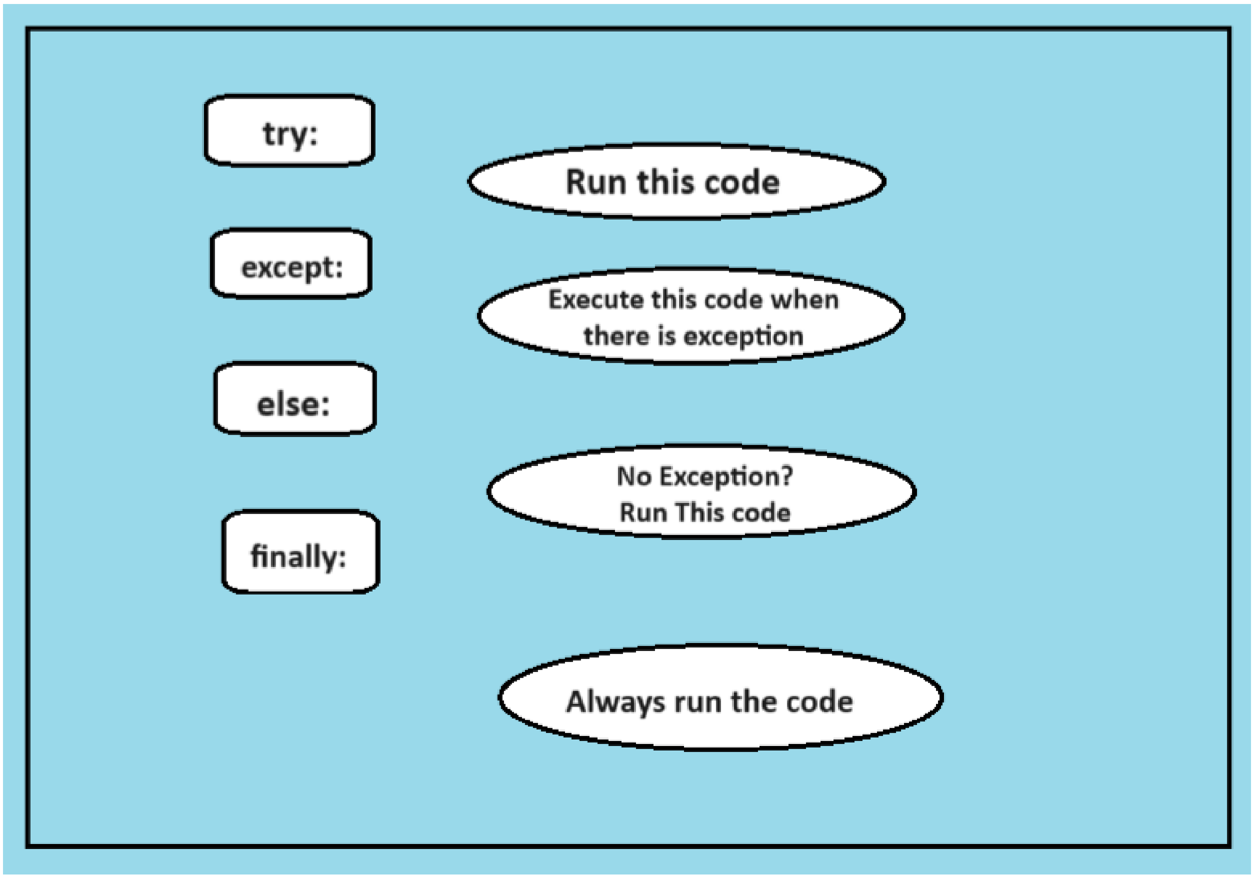
Compile-Time Errors:
Errors that occur when you violate the rules of writing syntax are known as compile-time errors. This compiler error indicates something that must be fixed before the code can be compiled. All these errors are detected by the compiler and thus are known as compile-time errors.

Examples:

Missing Symbols like: {},[],()
Printing the value of variables without declaring it
Missing colon(Terminator)

Run-Time Errors:
Errors that occur during the program execution after a successful compilation are called run-time errors. One of the most common run-time errors is division by zero also known as Division error. These types of errors are hard to find as the compiler doesn't point to the line at which the error occurs.

```
try:
    print(a)
except:
    print("Error occurred")
else:
    print("Error did not occur")
finally:
  print("Always")
```


try:
Run this code
except:
Execute this code when there is exception
else:
No Exception? Run This code
finally:
Always run the code

Multiple Exception can be handled by using nested try blocks:

```
try:
   print('a'+10)
except:
   print("Outer")
   try:
      print(a)
   except:
      print("inner")
```

Different Types of Errors:

Attribute Error: Raised by syntax Obj. foo has no member named foo
EOF Error: Raised if "end of file" reached for console or file input
IOError: Raised upon failure of I/O Operation(e.g., Opening file)
Index Error: Raised if index to sequence is out of bounds
Key Error: Raised if the non-existent key is requested for set or dictionary
KeyBoardInterrupt: Raised if the user types ctrl-C while the program is executing
Name Error: Raised If a nonexistent identifier is used
StopIteration: Raised by next(iterator) if no element; see Section 1.8
TypeError: Raised when the wrong type of parameter is sent to a function
ValueError: Raised when parameter is sent to a function
ZeroDivisionError: Raised when any division Operation is used with 0 as the divisor.