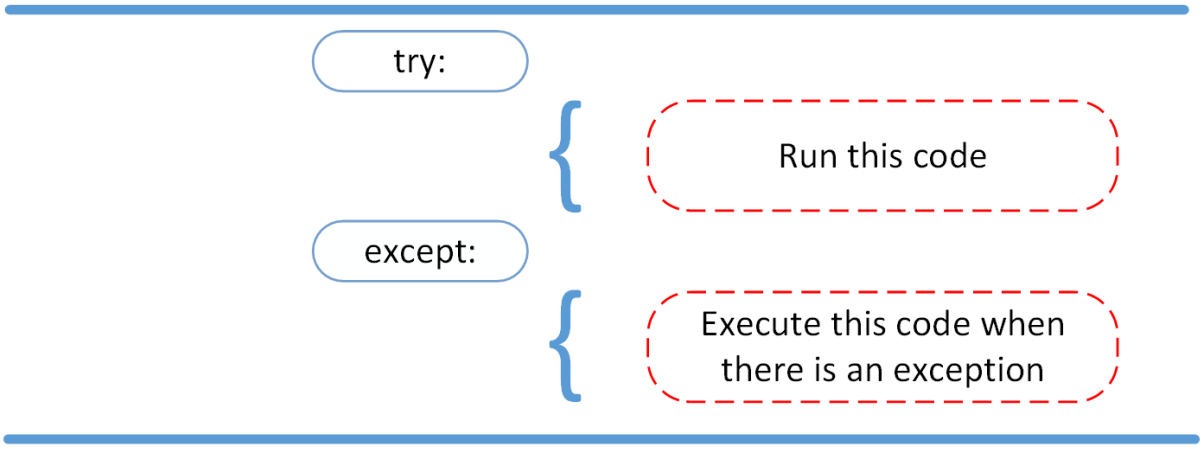
# Catching Exceptions

In Python, exceptions can be handled using try-except blocks.

- If the Python program contains suspicious code that may throw the exception, we must place that code in the try block.
- The try block must be followed by the except statement, which contains a block of code that will be executed in case there is some exception in the try block.
- We can thus choose what operations to perform once we have caught the exception.




**Syntax:**

```
try:
        # Some Code....

except:
        # optional block
        # Handling of exception (if required)
```

**Example:**

```
l = ['a', 0, 2]

for ele in l:
try:
   print("The entry is", ele)
   r = 1/int(ele)

except Exception as e: #Using Exception class
   print("Oops!", e.__class__, "occurred.")
   print("Next entry.")
   print()

print("The reciprocal of", ele, "is", r)
```

**We get the output to this code as:**

```
The entry is a
Oops! <class 'ValueError'>occurred.

The entry is 0
Oops! <class 'ZeroDivisonError'>occured.

The entry is 2 T
The reciprocal of 2 is 0.5
```

- In this program, we loop through the values of a list l.
- As previously mentioned, the portion that can cause an exception is placed inside the try block.
- If no exception occurs, the except block is skipped and normal flow continues(for last value).
- But if any exception occurs, it is caught by the except block (first and second values).
- Here, we print the name of the exception using the **exc_info()** function inside **sys** module.
- We can see that element “a” causes ValueError and 0 causes ZeroDivisionError.

Every exception in Python inherits from the base **Exception** class. Thus we can write the above code as:

```
l = ['a', 0, 2]
for ele in l:
 try:
     print("The entry is", ele)
     r = 1/int(ele)

 except Exception as e: #Using Exception class

     print("Oops!", e.__class__, "occurred.")
     print("Next entry.")
     print()

print("The reciprocal of", ele, "is", r)
```

**Output:**

```
This program has the same output as the above program.
```

**Note:** We can use more than one except in a single python program.

## Catching Specific Exceptions in Python

- In the above example, we did not mention any specific exception in the **except** clause.
- This is not a good programming practice as it will catch all exceptions and handle every case in the same way.
- We can specify which exceptions an **except** clause should catch.
- A try clause can have any number of **except** clauses to handle different exceptions, however, only one will be executed in case an exception occurs.
- You can use multiple **except** blocks for different types of exceptions.
- We can even use a tuple of values to specify multiple exceptions in an **except** clause. Here is an example to understand this better:

**Syntax:**

```
try:
        # Some Code....

except:
        # optional block
        # Handling of exception (if required)
```

**Example:**

```
try:
    a=10/0

except(ArithmeticError, IOError):
    print("Arithmetic Exception")
```

**Output:**

```
Arithmetic Exception
```