In [1]:

```
# type of exception: ZeroDivisionError
10 * (1/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-0b280f36835c> in <module>()
----> 1 10 * (1/0)

ZeroDivisionError: division by zero
```

In [4]:

```
# type of exception: TypeError
'2' + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-7a05eb93dd30> in <module>
      1 # type of exception: TypeError
----> 2 '2' + 2

TypeError: can only concatenate str (not "int") to str
```

In [5]:

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")

# the try statement works as follows
# first, the try clause is executed
# if no exception occurs, the except clause is skipped
# and execution of the try statement is finished
'''If an exception occurs during execution of the try clause,
the rest of the clause is skipped. Then if its type matches
the exception named after the except keyword, the except clause
is executed, and then execution continues after the
try statement.
If an exception occurs which does not match the exception
named in the except clause, it is passed on to outer
try statements. If no handler is found, it is an
unhandled exception and execution stops with a message
as shown above.'''
```

```
Please enter a number: a
Oops!  That was no valid number.  Try again...
Please enter a number: *
Oops!  That was no valid number.  Try again...
Please enter a number: 2
```

In [6]:

```
'''The raise statement allows us to force
a specified exception to occur.'''
raise NameError('HiThere')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-ee03ec41fb39> in <module>()
      1 '''The raise statement allows us to force
      2 a specified exception to occur.'''
----> 3 raise NameError('HiThere')

NameError: HiThere
```

In [7]:

```
'''If you need to determine whether
an exception was raised, but don't intend
to handle it, a simpler form of the raise statement
allows you to re-raise the exception.'''

try:
    raise NameError('HiThere')
except NameError:
    print('An exception flew by!')
    raise
```

```
An exception flew by!

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-7db7e351f35f> in <module>()
      5
      6 try:
----> 7     raise NameError('HiThere')
      8 except NameError:
      9     print('An exception flew by!')

NameError: HiThere
```

In [8]:

```
def divide(x, y):
    try:
        result = x / y
    except ZeroDivisionError:
        print("division by zero!")
    else:
        print("result is", result)
    finally:
        print("executing finally clause")

divide(2, 1)
```

```
result is 2.0
executing finally clause
```

In [9]:

```
divide(2, 0)
```

```
division by zero!
executing finally clause
```

In [10]:

```
divide("2", "1")
```

```
executing finally clause

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-3ad63cdb9b7d> in <module>()
----> 1 divide("2", "1")

<ipython-input-8-cbf715cb4bcb> in divide(x, y)
      1 def divide(x, y):
      2     try:
----> 3         result = x / y
      4     except ZeroDivisionError:
      5         print("division by zero!")

TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

In [6]:

```
'''The finally clause is executed in any event.
The TypeError raised by dividing two strings is
not handled by the except clause and therefore
re-raised after the finally clause has been executed.
In real world applications, the finally clause is useful
for releasing external resources (such as files or
network connections), regardless of whether the use
of the resource was successful.'''

2 + "2"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-0dc3f6e3cf7a> in <module>
      8 of the resource was successful.'''
      9 
---> 10 2 + "2"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [ ]: