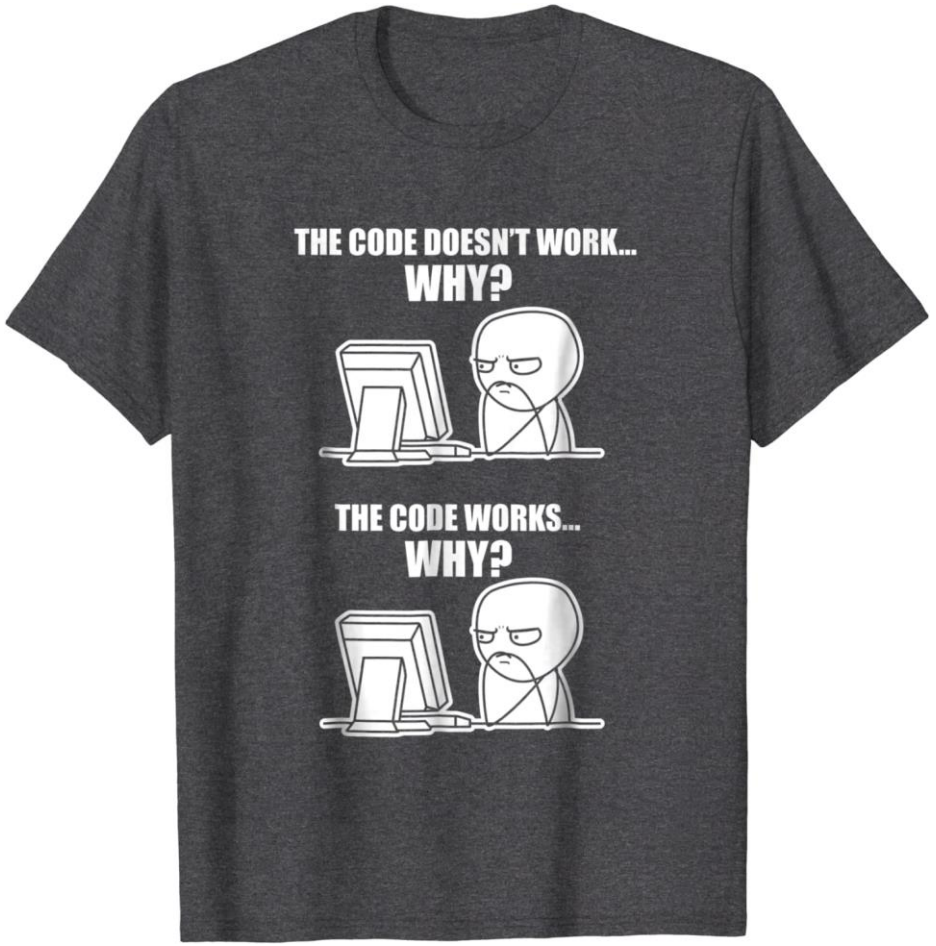
Exception handling

Introduction

- An exception is a special object that the executing program can create when it encounters an extraordinary situation.

- Such a situation almost always represents a problem, usually some sort of run-time error.



Examples

- Evaluating the expression lst[i] where lst is a list, and i ≥ len(lst).
- Attempting to convert a nonnumeric string to a number, as in int('Fred').
- Attempting to read a variable that has not been defined.
- Attempting to read data from the network when the connection is lost.

COMMON STANDARD EXCEPTIONS

Class	Meaning
AttributeError	Object does not contain the specified instance variable or method
ImportError	The import statement fails to find a specified module or name in that module
IndexError	A sequence (list, string, tuple) index is out of range
KeyError	Specified key does not appear in a dictionary
NameError	Specified local or global name does not exist
TypeError	Operation or function applied to an inappropriate type
ValueError	Operation or function applied to correct type but inappropriate value
ZeroDivisionError	Second operand of divison or modulus operation is zero

COMMON STANDARD EXCEPTIONS

```
>>> from fractions import Fractions
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'Fractions'
```

```
>>> seq = [2, 5, 11]
>>> print(seq[1])
5
>>> print(seq[3])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

COMMON STANDARD EXCEPTIONS

```
>>> d = {}
>>> d['Fred'] = 100
>>> print(d['Fred'])
100
>>> print(d['Freddie'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Freddie'
```

```
>>> print(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
```

COMMON STANDARD EXCEPTIONS

```
>>> print(int('Fred'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'Fred'
```

```
>>> print(1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

Handling Exceptions

```
#  Get two integers from the user
print('Please enter two numbers to divide.')
num1 = int(input('Please enter the dividend: '))
num2 = int(input('Please enter the divisor: '))
print('{0} divided by {1} = {2}'.format(num1, num2, num1/num2))
```

```
Please enter two numbers to divide.
Please enter the dividend: 4
Please enter the divisor: 0
Traceback (most recent call last):
  File "dividenumbers.py", line 5, in <module>
    print('{0} divided by {1} = {2}'.format(num1, num2, num1/num2))
ZeroDivisionError: division by zero
```

Handling Exceptions

We can defend against the ZeroDivisionError exception with a conditional statement,

```
#  Get two integers from the user
print('Please enter two numbers to divide.')
num1 = int(input('Please enter the dividend: '))
num2 = int(input('Please enter the divisor: '))
if num2 != 0:
    print('{0} divided by {1} = {2}'.format(num1, num2, num1/num2))
else:
    print('Cannot divide by zero')
```

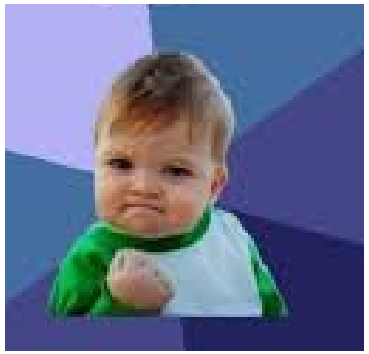
Look before you leap

LBYL

Handling Exceptions

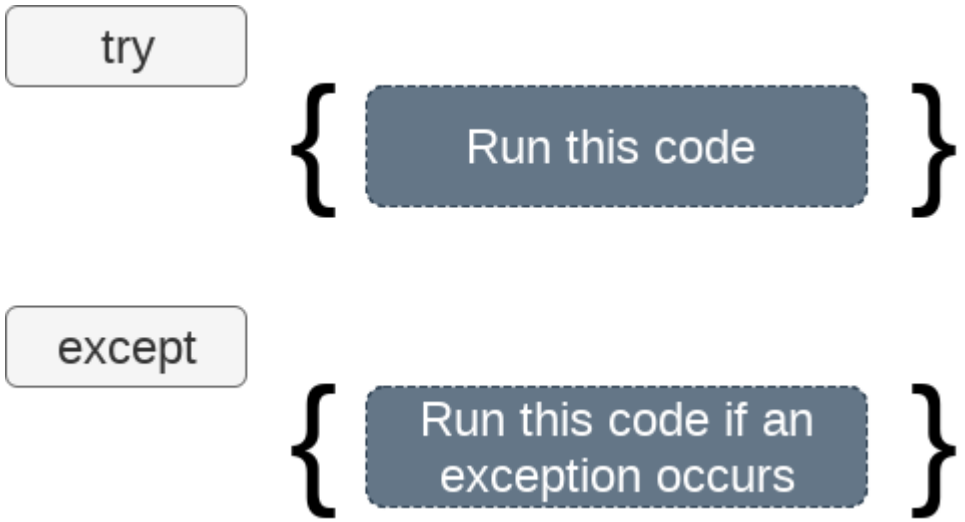
```
val = int(input("Please enter a small positive integer: "))
print('You entered', val)
```

```
Please enter a small positive integer: 5
You entered 5
```



```
Please enter a small positive integer: five
Traceback (most recent call last):
  File "enterinteger.py", line 1, in <module>
    val = int(input("Please enter a small positive integer: "))
ValueError: invalid literal for int() with base 10: 'five'
```

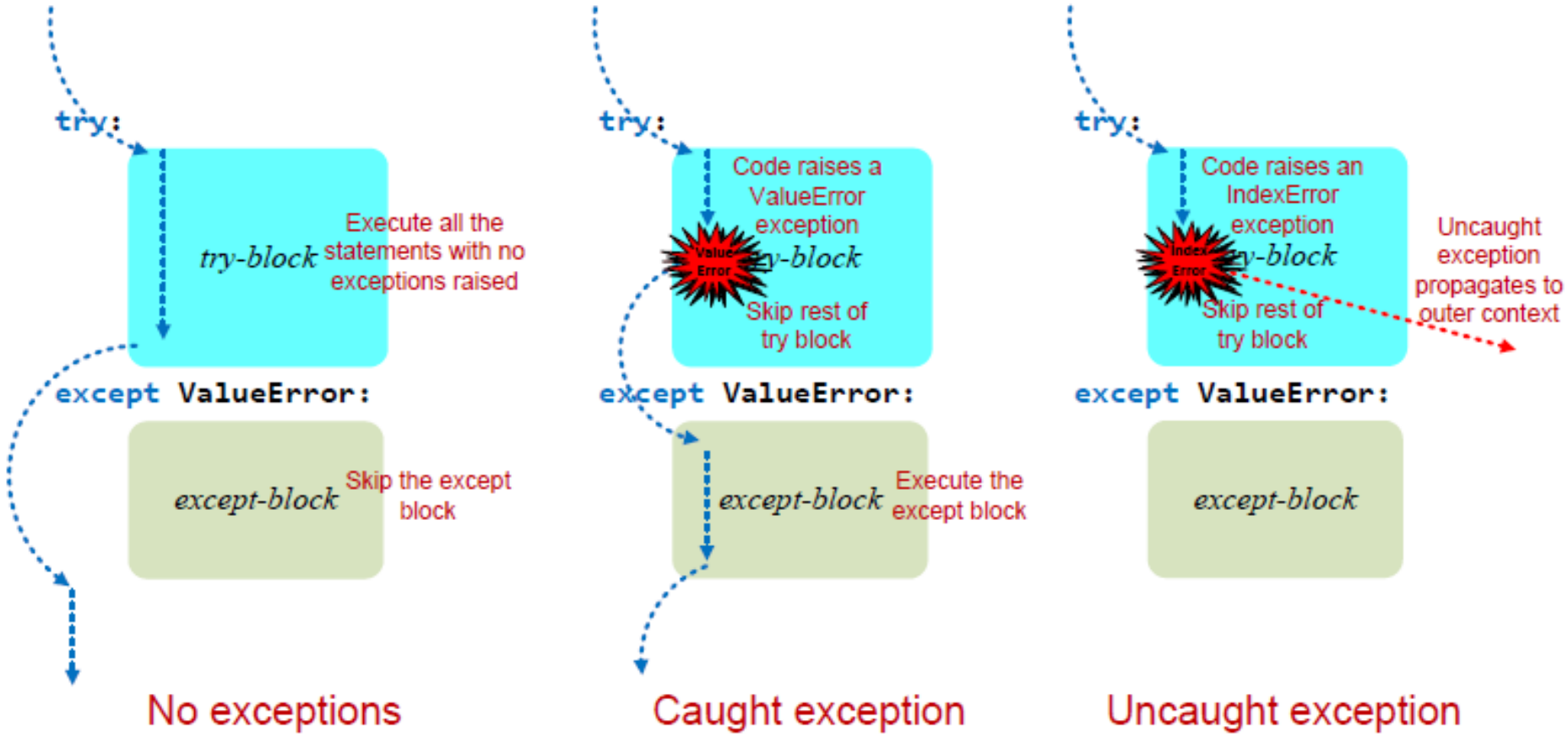
Handling Exceptions



Example

```
try:
    val = int(input("Please enter a small positive integer: "))
    print('You entered', val)
except ValueError:
    print('Input not accepted')
```

Program execution flows through a try/except



Handling Multiple Exceptions

```
import random

for i in range(10):    # Loop 10 times
    print('Beginning of loop iteration', i)
    try:
        r = random.randint(1, 3)   # r is pseudorandomly 1, 2, or 3
        if r == 1:
            print(int('Fred'))  # Try to convert a non-integer
        elif r == 2:
            [][2] = 5   # Try to assign to a nonexistent index of the empty list
        else:
            print(3/0)  # Try to divide by zero
    except ValueError:
        print('Cannot convert integer')
    except IndexError:
        print('List index is out of range')
    except ZeroDivisionError:
        print('Division by zero not allowed')

    print('End of loop iteration', i)
```

```
Beginning of loop iteration 0
List index is out of range
End of loop iteration 0
Beginning of loop iteration 1
Division by zero not allowed
End of loop iteration 1
Beginning of loop iteration 2
Cannot convert integer
End of loop iteration 2
Beginning of loop iteration 3
List index is out of range
End of loop iteration 3
Beginning of loop iteration 4
Cannot convert integer
End of loop iteration 4
Beginning of loop iteration 5
List index is out of range
End of loop iteration 5
Beginning of loop iteration 6
```

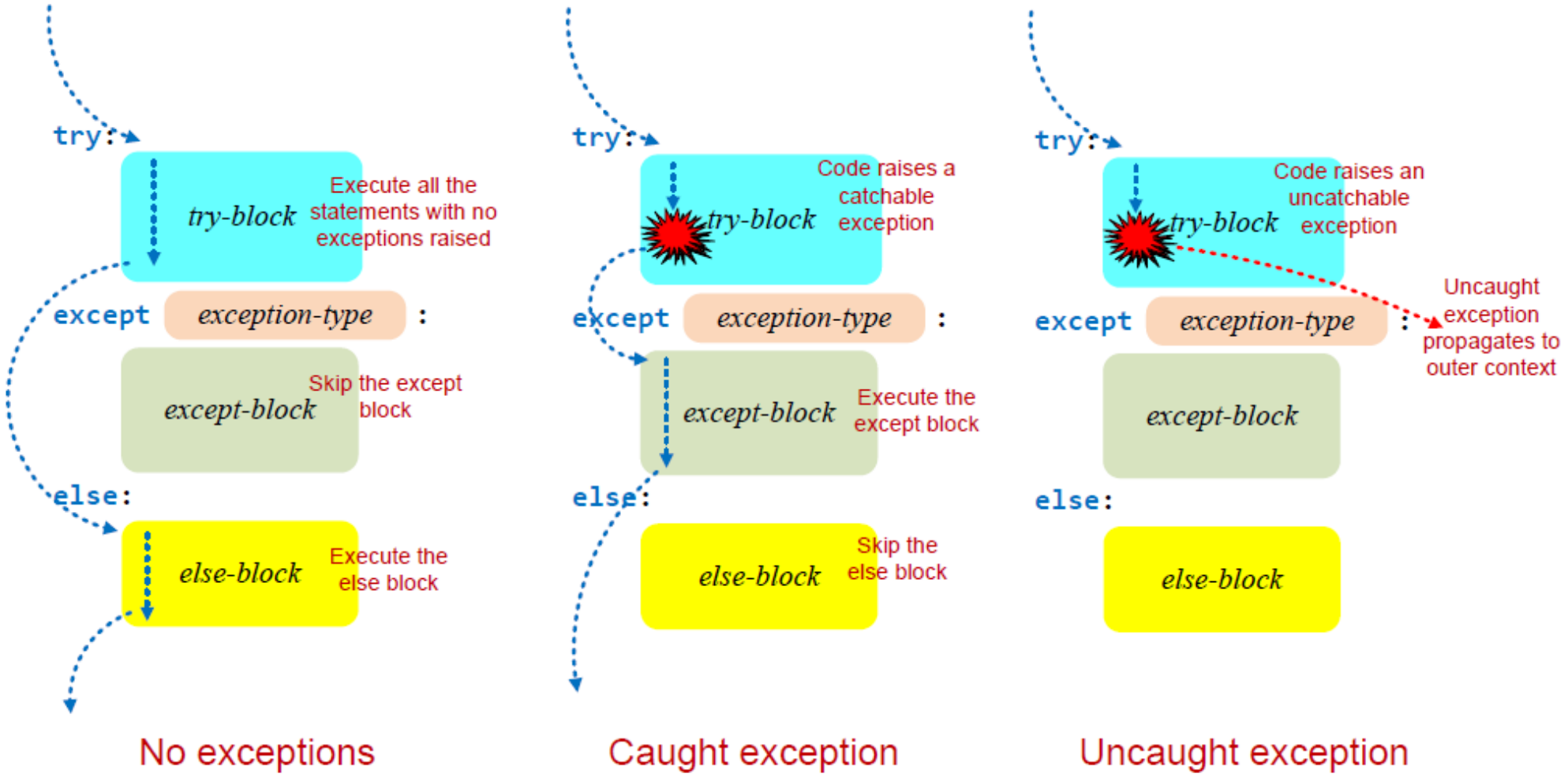
Catching Exception Objects

```
import random

for i in range(10):    # Loop 10 times
    print('Beginning of loop iteration', i)
    try:
        r = random.randint(1, 4)   # r is pseudorandomly 1, 2, 3, or 4
        if r == 1:
            print(int('Fred'))  # Try to convert a non-integer
        elif r == 2:
            [][2] = 5   # Try to assign to a nonexistent index of the empty list
        elif r == 3:
            print({}[1])  # Try to use a nonexistent key to get an item from a dictionary
        else:
            print(3/0)  # Try to divide by zero
    except ValueError as e:
        print('Problem with value     ==>', type(e), e)
    except IndexError as e:
        print('Problem with list      ==>', type(e), e)
    except ZeroDivisionError as e:
        print('Problem with division  ==>', type(e), e)
    except Exception as e:
        print('Problem with something ==>', type(e), e)

    print('End of loop iteration', i)
```

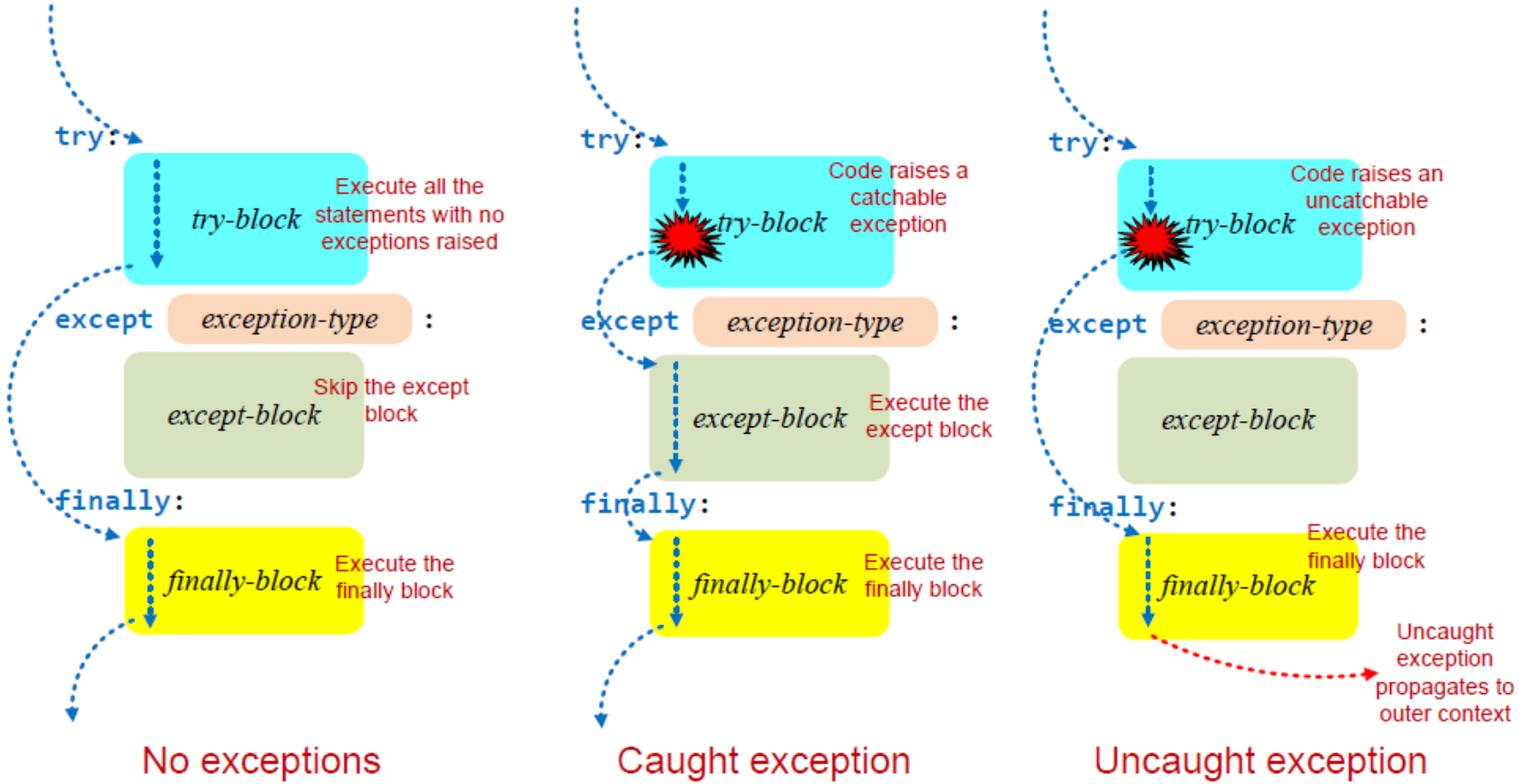
Catching Exception Objects

```
Beginning of loop iteration 0
Problem with division  ==> <class 'ZeroDivisionError'> division by zero
End of loop iteration 0
Beginning of loop iteration 1
Problem with list      ==> <class 'IndexError'> list assignment index out of range
End of loop iteration 1
Beginning of loop iteration 2
Problem with division  ==> <class 'ZeroDivisionError'> division by zero
End of loop iteration 2
Beginning of loop iteration 3
Problem with division  ==> <class 'ZeroDivisionError'> division by zero
End of loop iteration 3
Beginning of loop iteration 4
Problem with division  ==> <class 'ZeroDivisionError'> division by zero
End of loop iteration 4
Beginning of loop iteration 5
Problem with list      ==> <class 'IndexError'> list assignment index out of range
End of loop iteration 5
Beginning of loop iteration 6
```

The try Statement's Optional else Block



finally block



finally block: example

```
#  Sum the values in a text file containing integers
try:
    f = open('mydata.dat')
except OSError:
    print('Could not open file')
else:
    sum = 0
    try:
        for line in f:
            sum += int(line)
    except Exception as er:
        print(er)  # Show the problem
    finally:
        f.close()  # Close the file
    print('sum =', sum)
```