Exceptions

- Exceptions occur due to circumstances beyond programmer's control
  - Invalid input data
  - File cannot be accessed
- Even though it might be user’s fault
  - Programmer must anticipate
  - Include code to work around the occurrence

Exceptions

- Sample problem:

```
numDependents = int(input("Enter number of dependents: "))
taxCredit = 1000 * numDependents
print("Tax credit:", taxCredit)
```

  - Entry may be non numeric or just <Enter> key
  - Python gives following error message

```
ValueError: invalid literal for int() with base 10: ''
```

The *try* Statement

- Robust program explicitly handles previous exception
  - Protecting the code with a *try* statement.

```
try:
    numDependents = int(input("Enter number of dependents: "))
except:
    print("\nYou did not respond with an integer value.")
    print("We will assume your answer is zero.\n")
    numDependents = 0
taxCredit = 1000 * numDependents
print("Tax credit: ", taxCredit)
```

Common Exceptions

Exception Name	Description and Example
AttributeError	An unavailable functionality (usually a method) is requested for an object. `(2, 3, 1).sort()` or `print(x.endswith(3)) # where x = 23`
FileNotFoundError	Requested file doesn't exist or is not located where expected. `open("NonexistantFile.txt", 'r')`
ImportError	Import statement fails to find requested module. `import nonexistentModule`
IndexError	An index is out of range. `letter = "abcd"[7]`
KeyError	No such key in dictionary. `word = d['c']) # where d = {'a':"alpha", 'b':"bravo"}`
NameError	The value of a variable cannot be found. `term = word # where word was never created`
TypeError	Function or operator receives the wrong type of argument. `x = len(23)` or `x = 6 / '2'` or `x = 9 + 'W'` or `x = abs(-3,4)`
ValueError	Function or operator receives right type of argument, but inappropriate value. `x = int('a')` or `L.remove(item) # where item not in list`
ZeroDivisionError	The second number in a division or modulus operation is 0. `num = 1 / 0` or `num = 23 % 0`

The *try* Statement

- Three types of *except* clauses:

`except:`	(Its block is executed when any exception occurs.)
`except` ***`ExceptionType`*****`:`**	(Its block is executed only when the specified type of exception occurs.)
`except` ***`ExceptionType as exp`*****`:`**	(Its block is executed only when the specified type of exception occurs. Additional information about the problem is assigned to *exp*.)

The *try* Statement

- Exception handler usage with a specific type of exception:
  - Will not catch other types of exceptions!

```
try:
    numDependents = int(input("Enter number of dependents: "))
except ValueError:
    print("\nYou did not respond with an integer value.")
    print("We will assume your answer is zero.\n")
    numDependents = 0
taxCredit = 1000 * numDependents
print("Tax credit: ", taxCredit)
```

The *else* and *finally* Clauses

- *try* statement also can include a single *else* clause
  - Follows the *except* clauses
  - Executed when no exceptions occur
- *try* statement can end with a *finally* clause
  - Usually used to clean up resources such as files that were left open
- *try* statement must contain either an *except* clause or a *finally* clause.

The *else* and *finally* Clauses

```
try:
    x = 100
    y = 35
    z = x / y
except ZeroDivisionError:
    print("\nError: Zero Division!")
else:
    print("z = ", z)
    print("we get here only if there are no exceptions!")
finally:
    print("we get here anyway!")
```

```
z =  2.857142857142857
we get here only if there are no exceptions!
we get here anyway!
```

```
try:
    x = 100
    y = 35
    z = x / 0
except ZeroDivisionError:
    print("\nError: Zero Division!")
else:
    print("z = ", z)
    print("we get here only if there are no exceptions!")
finally:
    print("we get here anyway!")
```

```
Error: Zero Division!
we get here anyway!
```

Exception handling - Example

```
try:
    boxA = float(input("Input Box A value:"))
    boxB = float(input("Input Box B value:"))
    boxC = float(input("Input Box C value:"))
    boxD = float(input("Input Box D value:"))
except:
    print("Invalid Input, please enter a numeric value!")
else:
    print()
    print("%10s" % "", "%10s" % "Box A", "%10s" % "Box B", "%10s" % "Box C", "%10s" % "BoxD")
    print("%10s" % "Initial:", "%10.2f" % boxA, "%10.2f" % boxB,"%10.2f" % boxC,"%10.2f" % boxD)

    temp1 = boxA
    boxA = boxB
    temp2 = boxD
    boxD = temp1
    boxB = boxC
    boxC = temp2

    print("%10s" % "Final:", "%10.2f" % boxA, "%10.2f" % boxB,"%10.2f" % boxC,"%10.2f" % boxD)

    result = (boxC * boxD) - (boxA * boxB)
    print("\nComputed Result: ", round(result,3))
finally:
    print("Goodbye!")
```

Reporting Multiple Types of Exceptions

```
try:
    x = 100
    y = 35
    z1 = x / y
    z2 = x / 0
except ValueError:
    print("\nError: Value Error!")
except ZeroDivisionError:
    print("\nError: Zero Division Error!")
except IOError:
    print("\nError: IO Error!")
else:
    print("z1 =", z1, "z2 = ", z2)
```

Exception Message

- Get the default message for the exception

```
try:
    x = 100
    y = 35
    z1 = x / y
    z2 = x / 0
except ZeroDivisionError as details:
    print("\nError:", details)
else:
    print("z1 =", z1, "z2 = ", z2)
```

- Program output

```
Error: division by zero
```