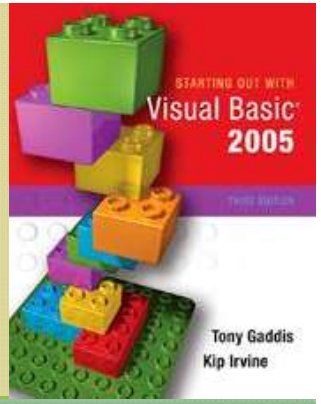
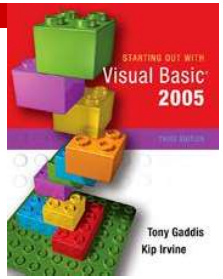


3.4 Exception Handling

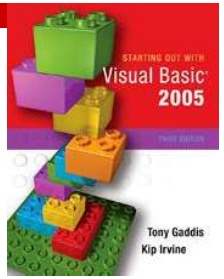
A Well-Engineered Program Should Report Errors and Try To Continue Or Explain Why It Can't Continue and Then Shut Down.

Use Exception Handling to Recover Gracefully from Errors



Runtime Errors

- We've shown two possible runtime errors
  - DivideByZeroException
  - InvalidCastException
  - There are many others
- Runtime errors occur for may reasons
- A runtime error results when:
  - Visual Basic *throws an exception*
  - And it is an *unhandled exception*
- Exception handling allows a program to fail gracefully and recover if possible



Message Boxes

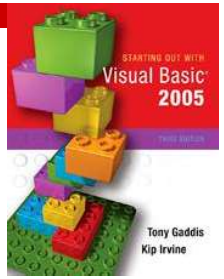
- A message box is an easy way to notify the user when an error occurs
- **`MessageBox.Show`** displays a pop-up window with a message and an *OK* button
- There are two basic formats

```
MessageBox.Show( message )

MessageBox.Show( message , caption )
```

- **`message`** appears in the body of the window
- **`caption`** appears in the title bar of the window

Message Box Example

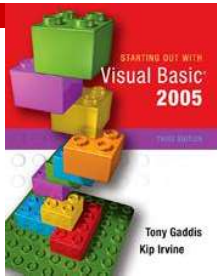


The following code displays the message box shown below

```
MessageBox.Show("Please try again, and
    enter a number", "Entry Error")
```

The capabilities of the MessageBox will be presented in more detail in Chapter 4

Handling Exceptions

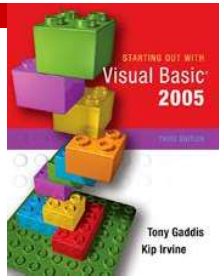


- Visual Basic provides an *exception handler*
- A simple form that ignores some options is:

```
Try
  try-block
Catch [exception-type]
  catch-block
End Try
```

- The *try-block* contains program statements that ***might*** throw an exception
- The *catch-block* contains statements to execute ***if*** an exception is thrown

Exception Handling Example

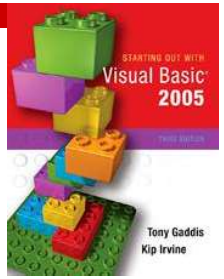


- Consider the following exception handling code

```
Try
  Dim decSalary as Decimal
  decSalary = CDec(txtSalary.Text)
  MessageBox.Show("Your salary is " _
    & decSalary & " dollars")
  Catch
    MessageBox.Show(" Please try again," _
    & "and enter a number", "Entry Error")
End Try
```

- If `CDec` throws a cast exception, the try block catches it, jumps to and executes the catch block, and displays the error message

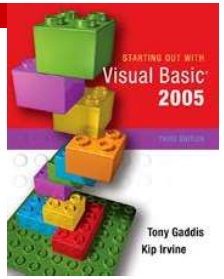
More Exception Handling Features



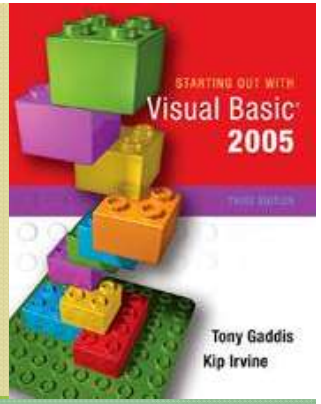
- Can catch specific types of messages
- Can capture and show the exception message issued by Visual Basic

```
Try
  Dim decAnnualSalary as Decimal
  Dim intPayPeriods as Integer
  Dim decSalary as Decimal
  decAnnualSalary = CDec(txtAnnualSalary.Text)
  intPayPeriods = CInt(txtPayPeriods.Text)
  decSalary.Text = decAnnualSalary / intPayPeriods
  lblSalary.Text = decSalary.ToString()
  Catch ex as InvalidCastException
    MessageBox.Show(ex.Message, "Entry Error")
  Catch ex as DivideByZeroException
    MessageBox.Show("Zero Value Not Allowed " _
      & " for Pay Periods")
End Try
```

Exception Handling Exercise



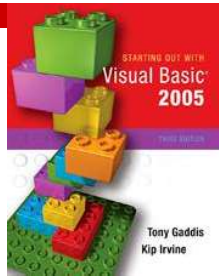
- Tutorial 3-8 provides an opportunity to work with exception handling concepts



3.5 Formatting Numbers for Output

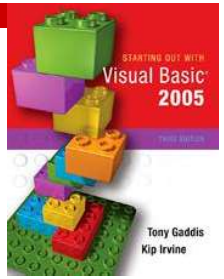
Numbers May Be Formatted in Various Ways for Output

Number Formats



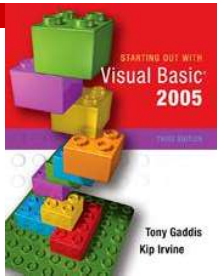
- **`FormatNumber`** - includes commas and the specified number of decimal places
- **`FormatCurrency`** – formats as currency with dollar sign or other currency symbol
- **`FormatPercent`** – displays a number as a percent
- **`FormatDate`** – formats a number as a date, time, or both
- The computer's regional settings determine some format items such as currency symbol

FormatNumber Function



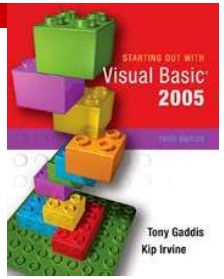
- Used to format a number for display
- **`FormatNumber(expression [, DecimalPoints])`**
- Expression is evaluated and output as a number with commas and a decimal point
- Optional second argument gives the number of decimal places to display
- If not specified, decimal positions default to 2
- Decimal positions not shown are rounded
- **`FormatNumber(3921.387)`** returns “3,921.39”
- **`FormatNumber(.75)`** returns “0.75”

FormatCurrency Function



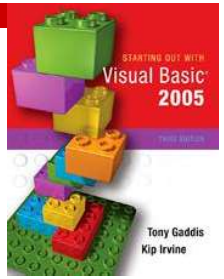
- Used to format a number for display as a currency figure such as dollars and cents
- **`FormatCurrency(expression [, DecimalPoints])`**
- Expression is evaluated and returned with commas, decimal point, and currency symbol
- As with **`FormatNumber`**, an optional second argument specifies the decimal places
- Decimal positions default to 2 and are rounded
- **`FormatCurrency(3921.387)`** returns "$3,921.39"
- **`FormatCurrency(.87)`** returns "$0.87"

FormatPercent Function



- Used to format a number for display as a percent
- **`FormatPercent(expression [, DecimalPoints])`**
- Multiplies expression by 100 and adds the % sign
- As with **`FormatNumber`**, an optional second argument specifies the decimal places
- Decimal positions default to 2 and are rounded
- **`FormatPercent(.78466)`** returns “78.47%”
- **`FormatPercent(8.2, 0)`** returns “820%”

FormatDateTime Function



- Displays a date in various formats
- **`FormatDateTime(expression [, Format])`**
- Expression must evaluate to a Date data type
- Optional second argument specifies the desired format, e.g.
  - DateFormat.GeneralDate – “4/7/2006 3:22:18 PM”
  - DateFormat.LongDate – “Friday, April 7, 2006”
  - DateFormat.ShortDate – “4/7/2006”
  - DateFormat.LongTime – “03:22:18 PM”
  - Dateformat.ShortTime – “15:22”
- Tutorial 3-9 provides an opportunity to work with number formatting concepts