(PRACTICAL)
COMPUTATIONAL PHYSICS

Physics 551
Lecture 14

NOTATION

Extra Reading

Optional Exercise

Recommended

- This lecture slides for this course will attempt to use a uniform notation throughout. A normal paragraph looks like this.

☞ *Italicized paragraphs with pen bullets will indicate definitions, with the defined word or phrase shown in* ***SMALL-CAPS****.*

✏ Pencil bullets will indicate the introduction of **new notation**.

☞ Pointing hand bullets indicate important points that might otherwise be overlooked.

ANNOUNCEMENTS

ANNOUNCEMENTS

- To **clone** this week's **LabVIEW demonstration materials** please open the ***Git Shell*** utility and invoke

```
$ git clone https://github.com/hughdickinson/CompPhysL14Labview.git
  C:/Users/ComputationalPhysics/Documents/labview/lecture14
```

☞ You can also find this command on the Blackboard Learn website.

ANNOUNCEMENTS

- **Project Deadline:**
  - **Midnight May 4th 2015** (although you may want to finish earlier if you have exams).
- **Project Submission:**
  - Create a **new directory** for your project called *`FinalProject`* in your private homework repositories on **GitHub**.
  - Add **all** of the elements of your project (including the manual) to this directory.
  - I will clone the contents of this directory **immediately** after the submission deadline.

ANNOUNCEMENTS

- **Project Requirements:**
  - **Functional** executable programs for C++ or LabVIEW.
  - **Functional** Python Scripts and Modules.
  - **Fully documented** and **annotated** code (Doxygen-compatible for C++).
  - A **user manual** including instructions for program **invocation** and **usage** and a **simple tutorial** section.

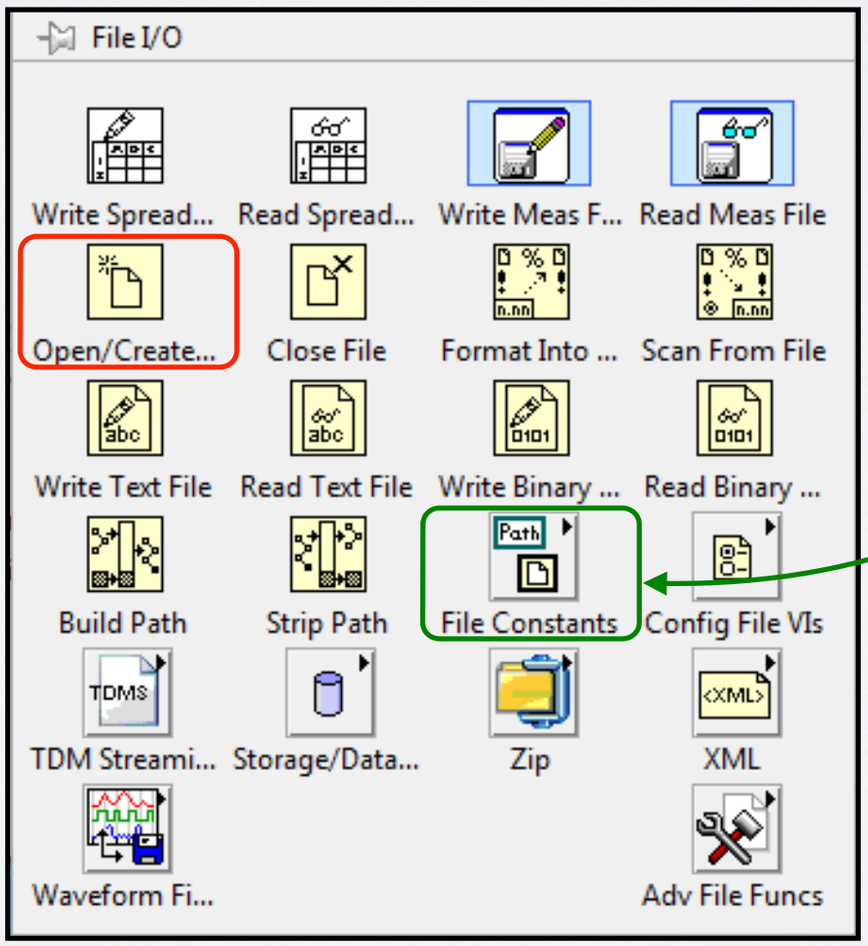
ANNOUNCEMENTS

- **Project Tips:**
  - Document code **as you write it**.
  - It is always easier to write documentation when the while the program logic is clear in your mind!
  - Use **git** to **regularly commit** your code as you develop.
  - Protect your work by regularly pushing committed changes to GitHub.
  - You may wish to back up your work in other ways.

SPECIALIZED LABVIEW DATA TYPES

PATH DATA

- LabVIEW provides a **generic** *Open/Create/Replace File* VI to **open** files that contain data in an **arbitrary** format.



PATH DATA

- The *OCR File* VI accepts an optional input connection that has the special **path** data type.

☞If an explicit path is not connected to the appropriate input then the *OCR File* VI will present a **file browser dialog** to the user.

- The *OCR File* VI returns a **File Reference** that can be connected to other *LabVIEW* file I/O VIs.

☞*LabVIEW* defines several *constant path values* that can be used to **assemble** file paths programmatically.

DEMONSTRATION

Opening Files and **Assembling Paths** Using **Predefined Constants**

WAVEFORM DATA

- Many LabVIEW applications acquire data from instruments at a **constant rate**.

☞ *In LabVIEW such data are typically stored as* ***WAVEFORM*** *data types.*

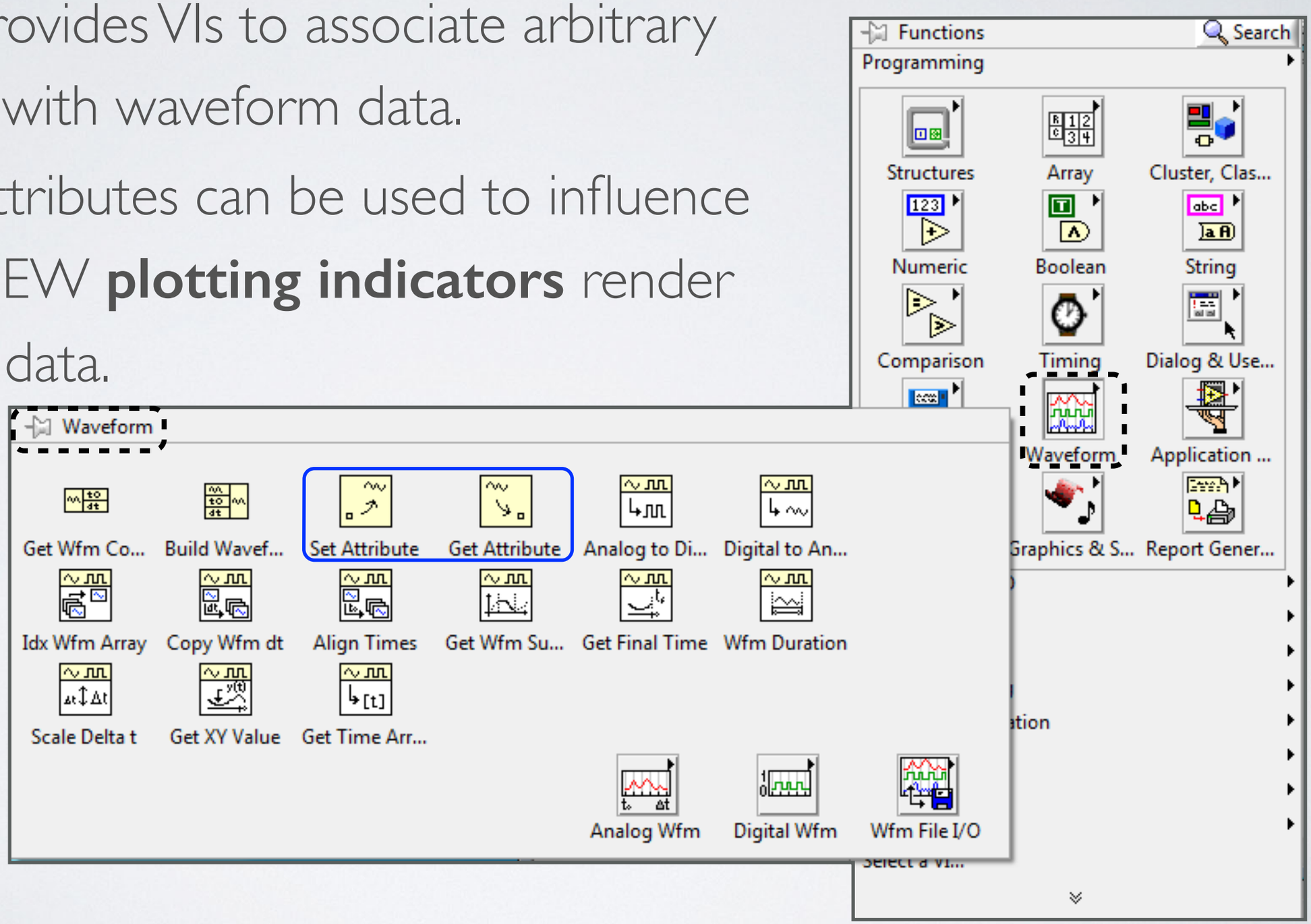
☞ Waveforms are aggregate data types that comprise **at least three** distinct components.

- An **array of sample values**.
- A value corresponding to the **time of the first sample**.
- A value specifying the **time between consecutive samples**.

WAVEFORM DATA

- Tabview provides VIs to associate arbitrary ***attributes*** with waveform data.

☞ **Specific** attributes can be used to influence how LabVIEW **plotting indicators** render ***waveform*** data.



DEMONSTRATION

Working with **Waveform Data**

DYNAMIC DATA

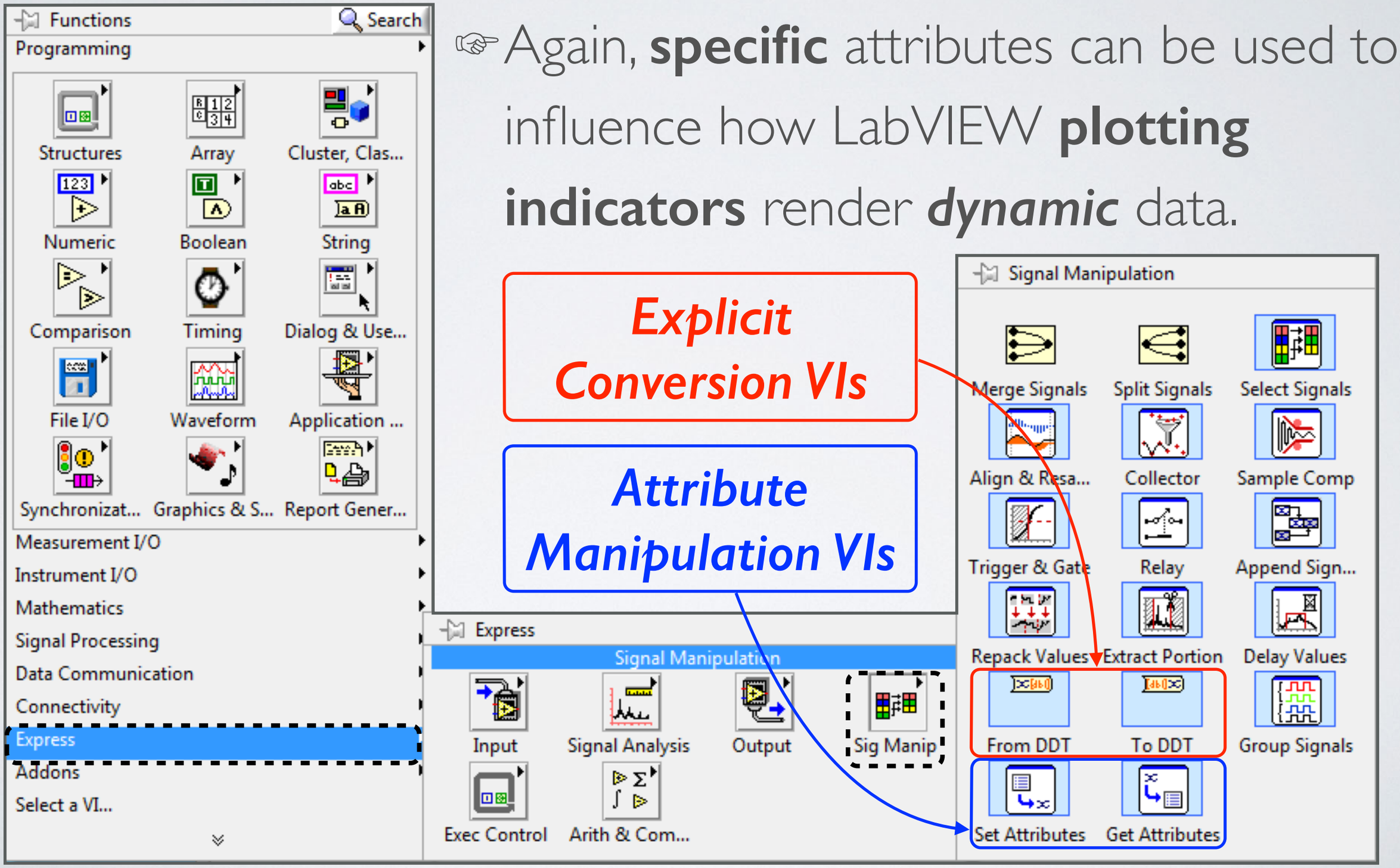
☞ *The LabVIEW package provides several, highlight configurable* ***EXPRESS VIS*** *that typically output* ***DYNAMIC TYPE DATA****.*

☞ LabVIEW provides VIs to **explicitly convert** dynamic data to fundamental data types.

☞ Many of the **plotting indicators** provided by LabVIEW accept dynamic data as an input - no explicit conversion is required.

- Dynamic data types can introduce **computational overhead**. and are typically **only** be used in conjunction with **Express VIs**.
- Like ***Waveform*** data, dynamic data types can also have ***arbitrary attributes*** associated with them.

DYNAMIC DATA



☞ Again, **specific** attributes can be used to influence how LabVIEW **plotting indicators** render ***dynamic*** data.

DEMONSTRATION

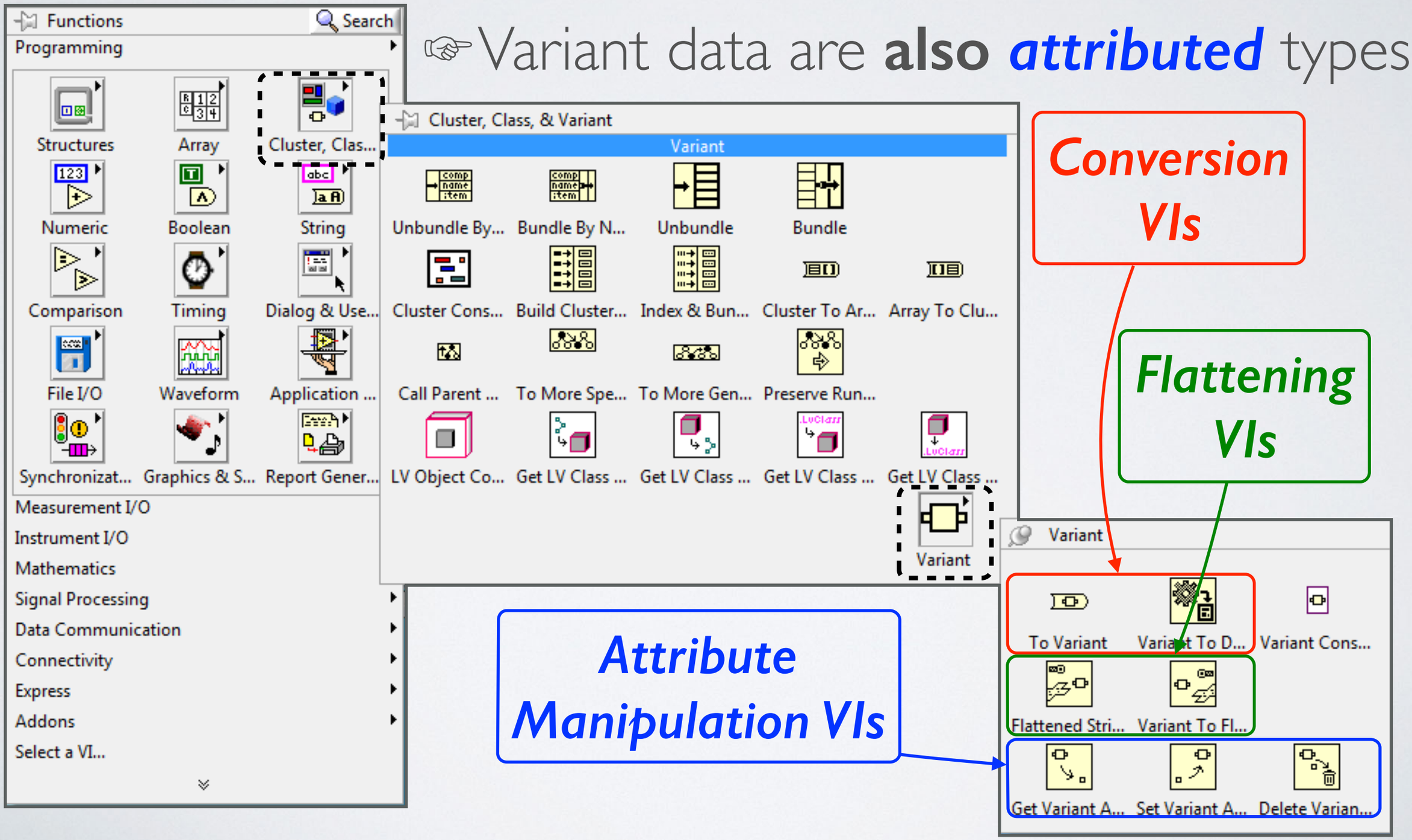
Working with **Dynamic Data**

VARIANT DATA

- ☞ *The **VARIANT** data type is designed to introduce limited type-flexibility with LabVIEW's strict typing model.*
- Variant data stores a **value** and **metadata** specifying how the value should be interpreted.
- ☞ VIs exist to **safely convert** variant data to arbitrary fundamental types. Impossible conversions produce **errors**, not improper values.
- "Flattening" VIs exist that permit the data and metadata that comprise a variant to be examined.

VARIANT DATA

☞Variant data are **also *attributed*** types.



DEMONSTRATION

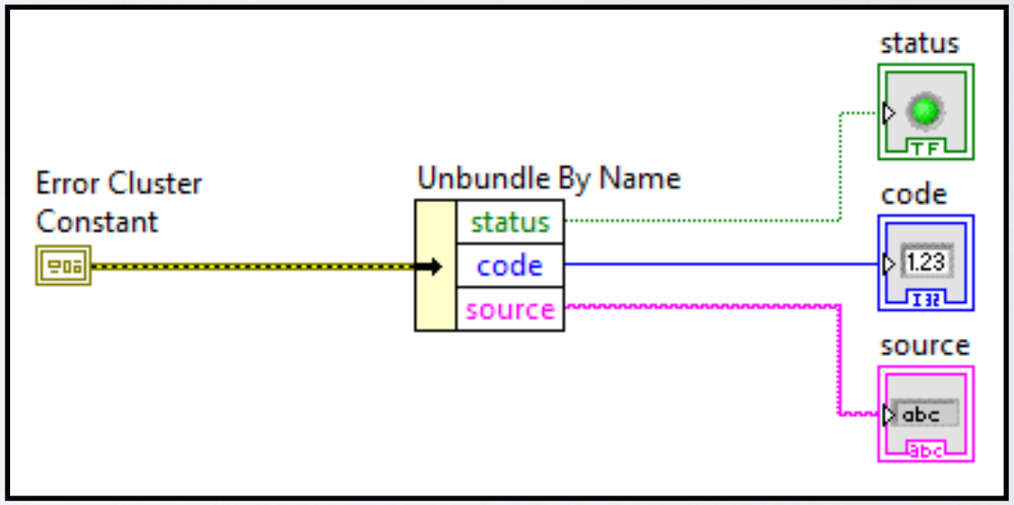
Working with **Variant Data**

ERROR CLUSTERS

- **Error handling** in LabVIEW involves **special** instances of the **cluster** data types.

  ➪ *LabVIEW* ***Error Clusters*** *comprise three* ***specific*** *elements*

  - A **boolean** value with identifier ***status***.
  - A **32 bit integer** value with identifier ***code***.
  - A **string** value with identifier ***source***.



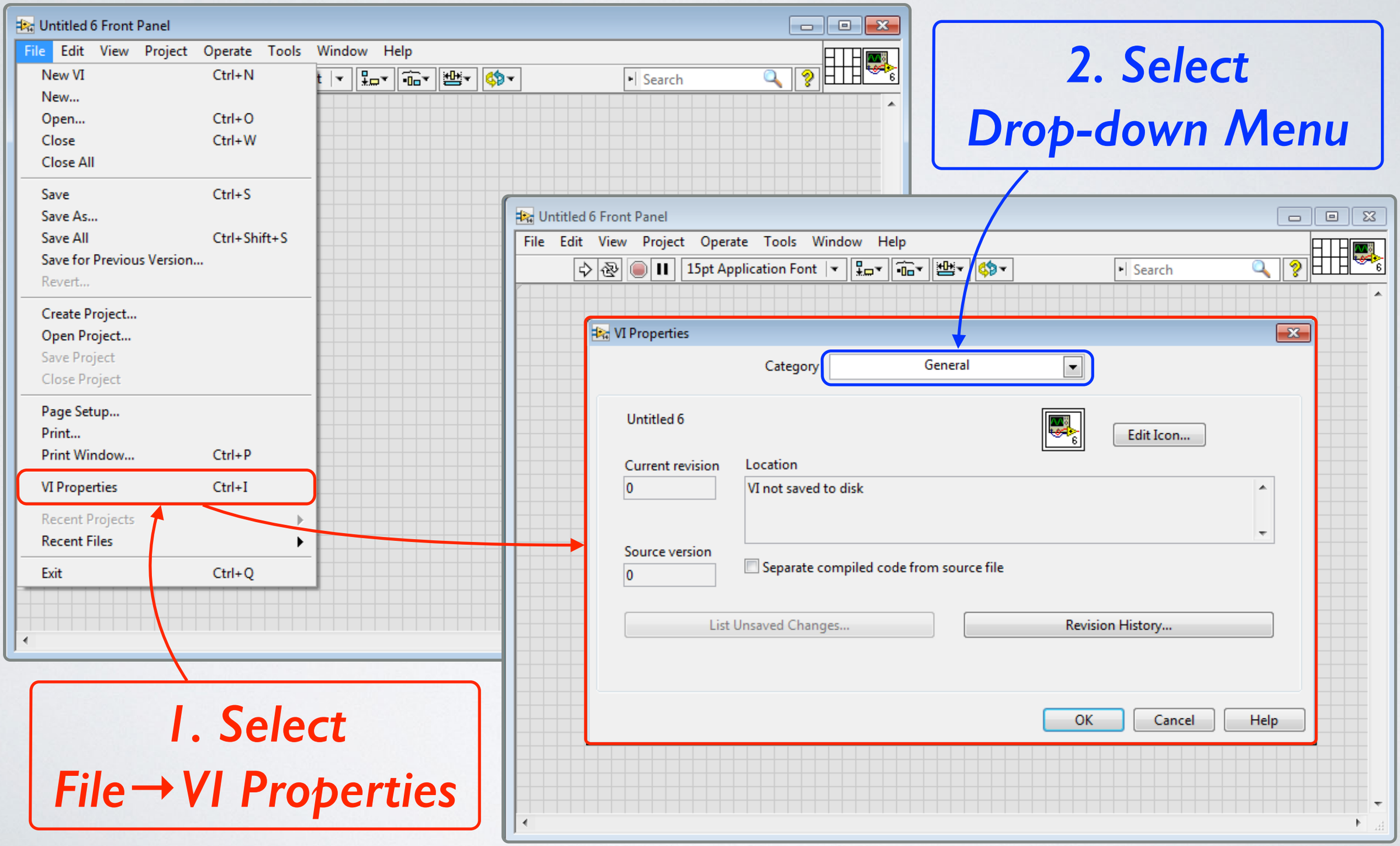
HANDLING ERRORS IN LABVIEW

LABVIEW ERROR HANDLING

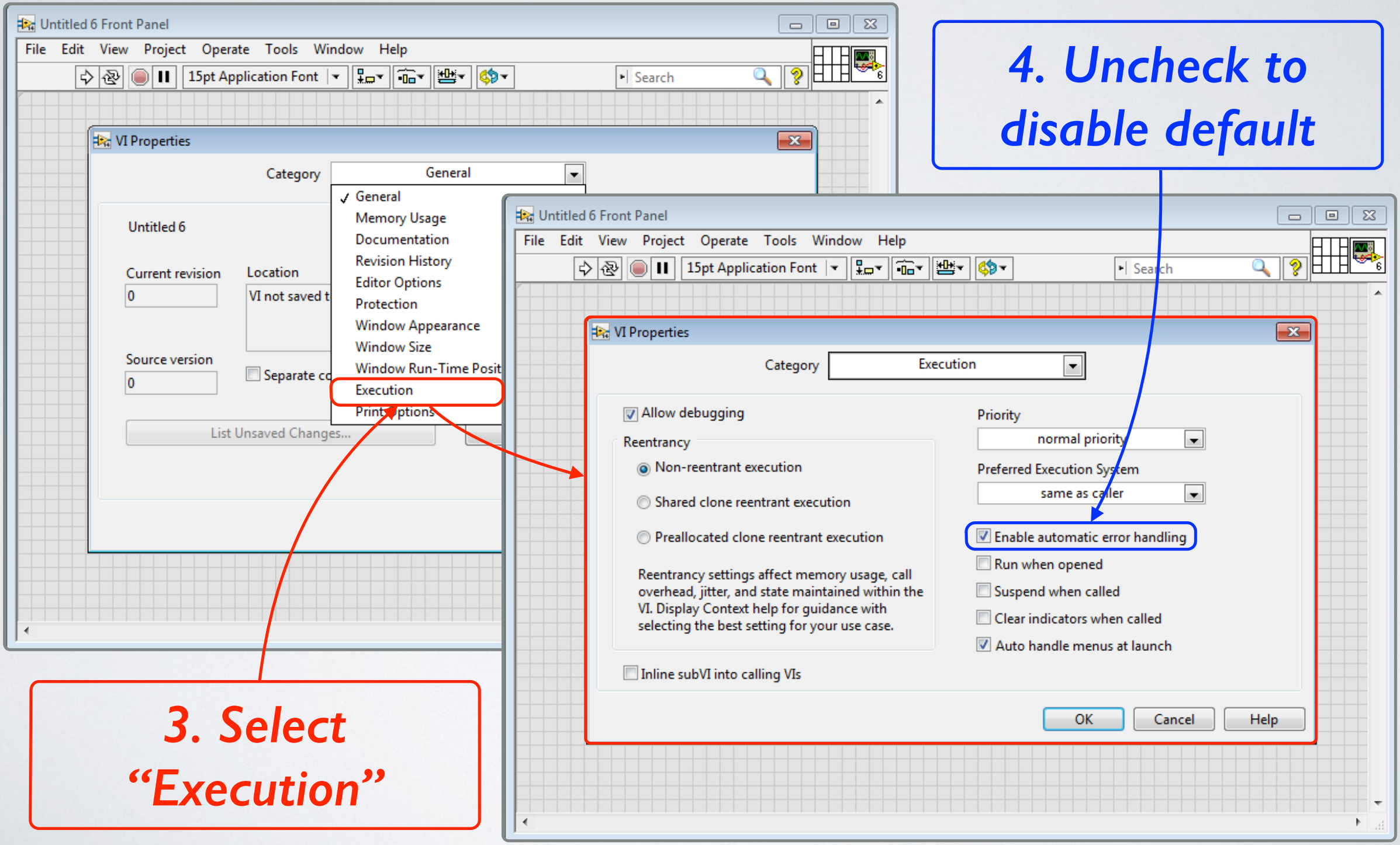
- If any of the sub-VIs or functions that populate the block diagram generate an error, the **default** behaviour of LabVIEW entails **three** operations.
  - Suspension of program execution.
  - Highlighting the point in the program where the error occurred.
  - Displaying an error dialog box.

☞ This default behaviour can be **disabled** via the LabVIEW user interface.

LABVIEW ERROR HANDLING



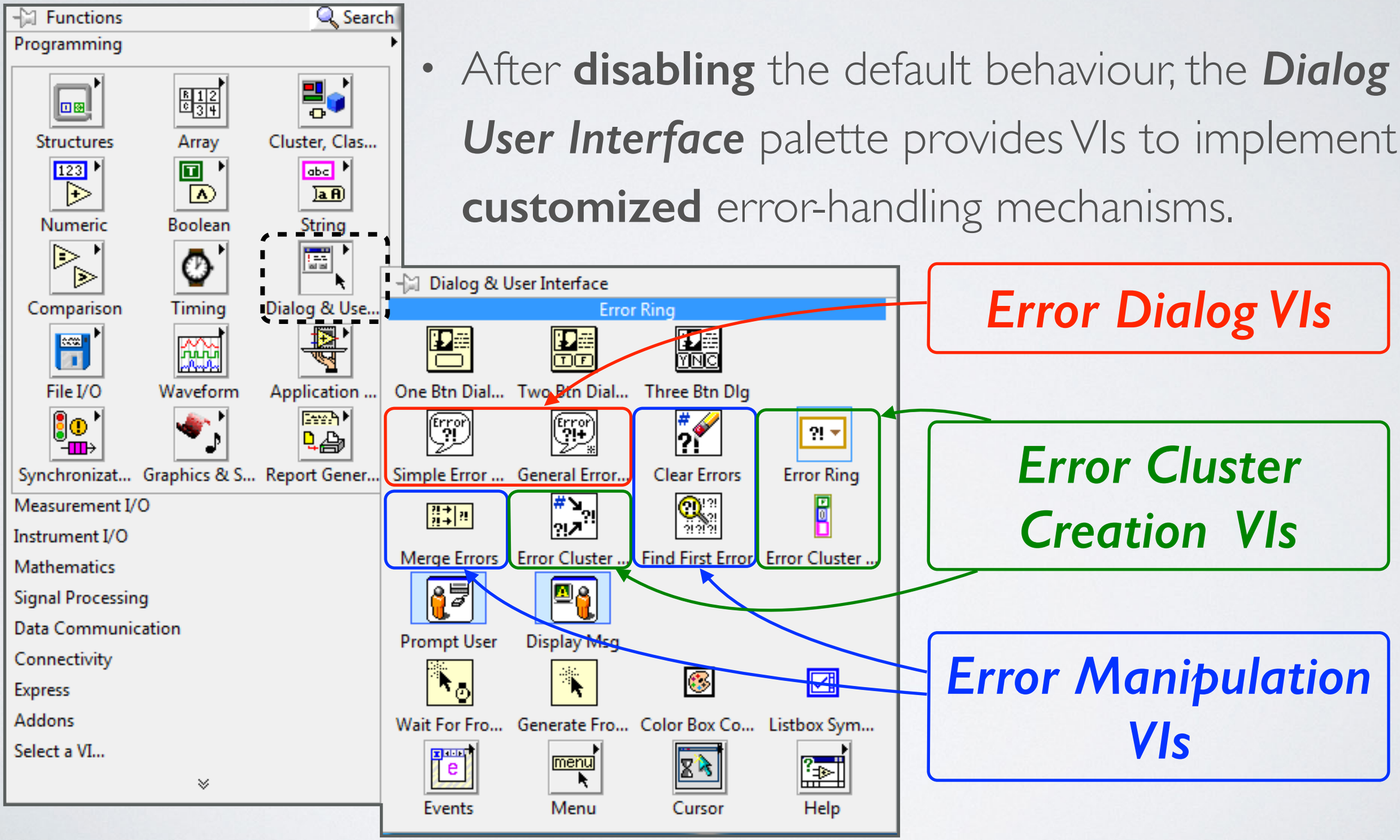
LABVIEW ERROR HANDLING



4. Uncheck to disable default

3. Select "Execution"

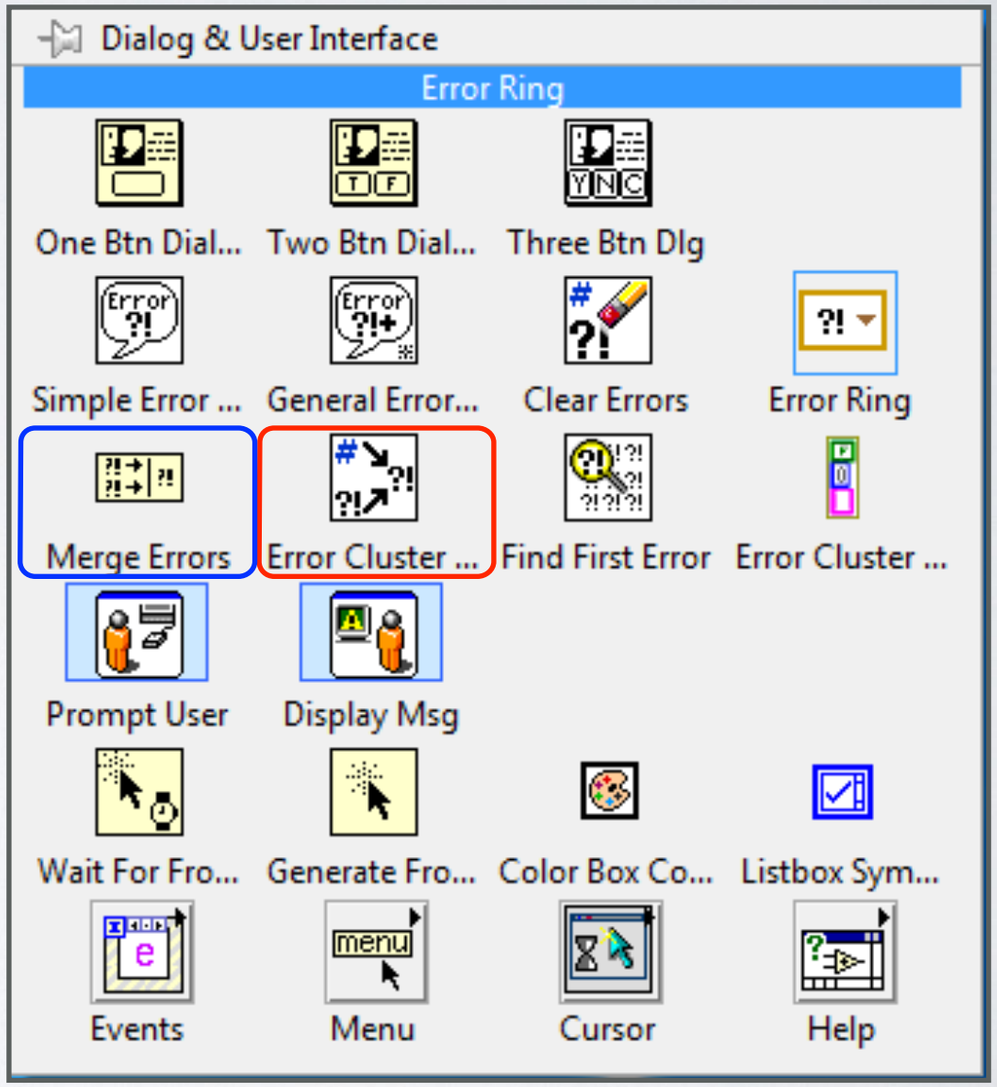
LABVIEW ERROR HANDLING



- After **disabling** the default behaviour, the ***Dialog & User Interface*** palette provides VIs to implement **customized** error-handling mechanisms.

LABVIEW ERROR HANDLING

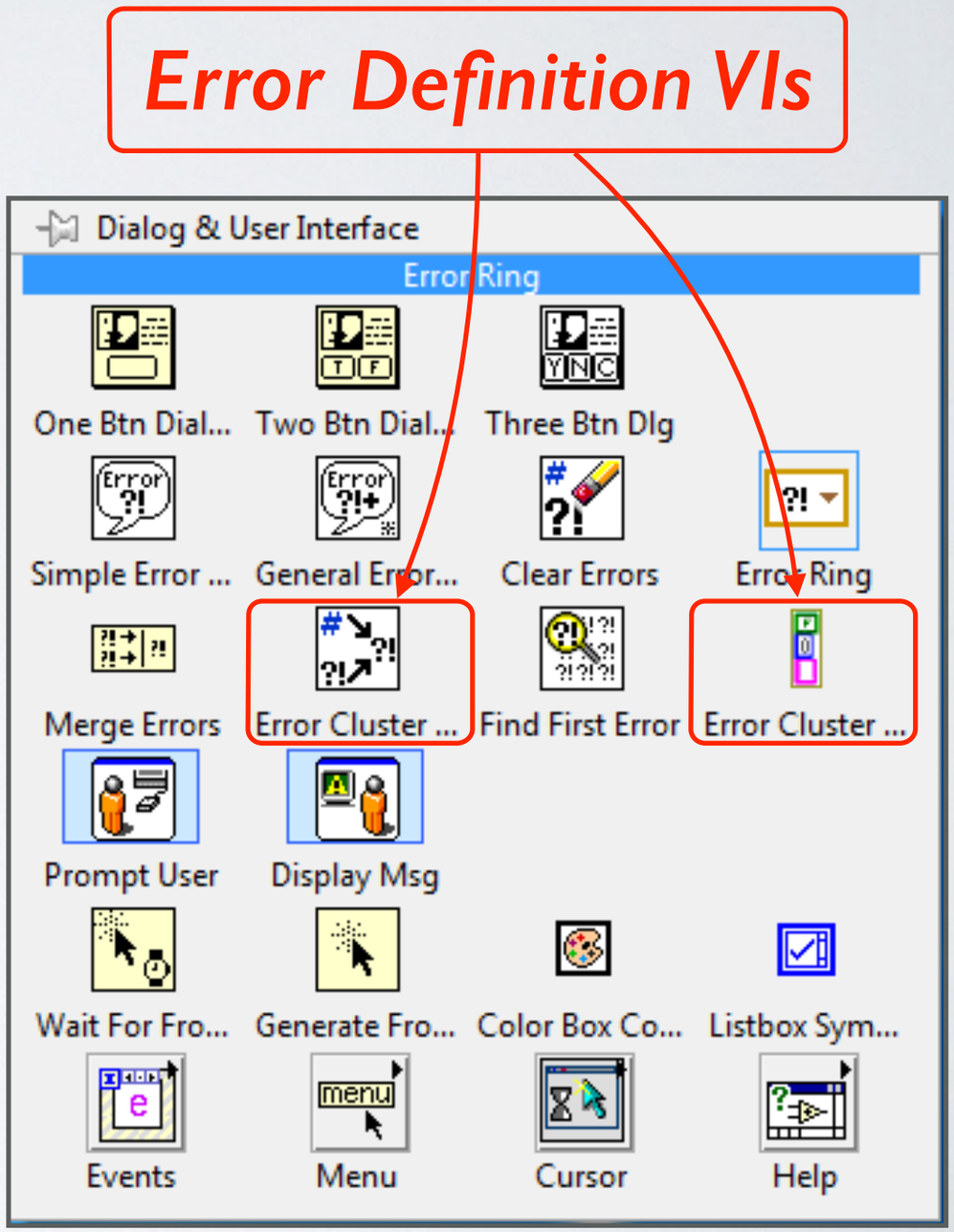
- A **subset** of LabVIEW VIs report errors using bare **numeric codes** instead of Error Clusters.
- The *Error Cluster From Error Code* VI builds **Error Clusters** from **Error Codes**.
- The error code is used to populate the ***code*** element of the **Error Cluster**.
- Multiple Error Clusters can be **compounded** using the extensible *Merge Errors* function.



LABVIEW ERROR HANDLING

- Error Clusters can be **defined** using context-specific ***integer*** Error Codes.
- LabVIEW **reserves** specific integer ranges for internal use.
- There are **three allowed ranges** for developer-defined Error Codes

-8999	→	-8000
8000	→	8999
500000	→	599999



DEMONSTRATION

Error Handling in LabVIEW

FILE INPUT AND OUTPUT

FILE INPUT AND OUTPUT

- ☞ LabVIEW provides **several** VIs that are tailored to **read** and **write** data in numerous **proprietary** and **generic** formats.
- All of these VIs are accessible from the ***File I/O*** palette of the **block diagram** context menu.
- VIs exist to handle **arbitrary textual** and **binary** data as well as formatted textual data including **XML** and **spreadsheets**.
- ☞ **Dedicated** VIs are provided for reading and writing ***Waveform*** data.
- ☞ Support is also provided for **file compression** using the ***Zip*** protocol.

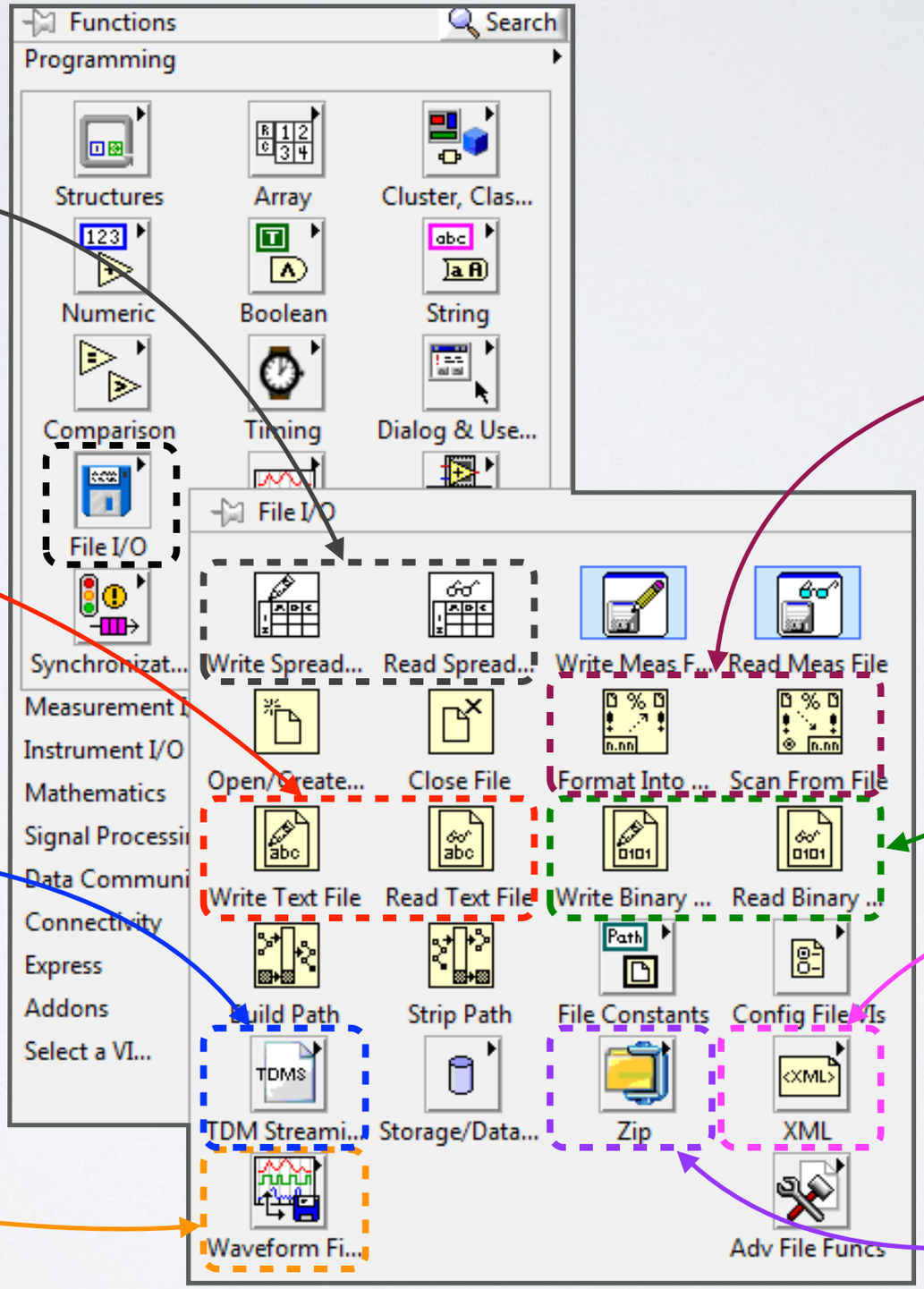
FILE INPUT AND OUTPUT

Spreadsheet
File Handling

Generic Text
File Handling

Proprietary
LabVIEW
TDMS

Waveform-Specific
File Handling



Formatted Text
File Handling

Generic Binary
File Handling

XML
File Handling

Zip
File Handling

DEMONSTRATION

File Input and **Output** in LabVIEW

PLOTTING DATA

PLOTTING USING WAVEFORM CHARTS

☞ *LabVIEW provides **CHART** indicators to display data **as it is accumulated**.*

- ***Chart**s plot and maintain values in a **fixed-length** buffer that stores a **limited** number of the most recently provided data.*
- *The data are managed in a **first-in, first-out** (FIFO) **queue**. When the data buffer is full, the **oldest data are discarded first**.*
- ***Chart**s are useful for **monitoring** continuous data acquisition.*

PLOTTING USING WAVEFORM CHARTS

- Several **numeric** types and various **layouts** of data can be connected to a ***Chart*** input.

☞ The **layout** of the data defines how the **values** should be **interpreted** and **plotted**.

- The **simplest** possible way to provide data to a ***Chart*** is to input a **single** numeric datum **whenever an update is required**.
- Alternatively, multiple values can be supplied in a single update by connecting an **array** of numeric types to the ***Chart*** input.

PLOTTING USING WAVEFORM CHARTS

- **Waveform** data can also be connected to LabVIEW ***Chart***-type indicators.

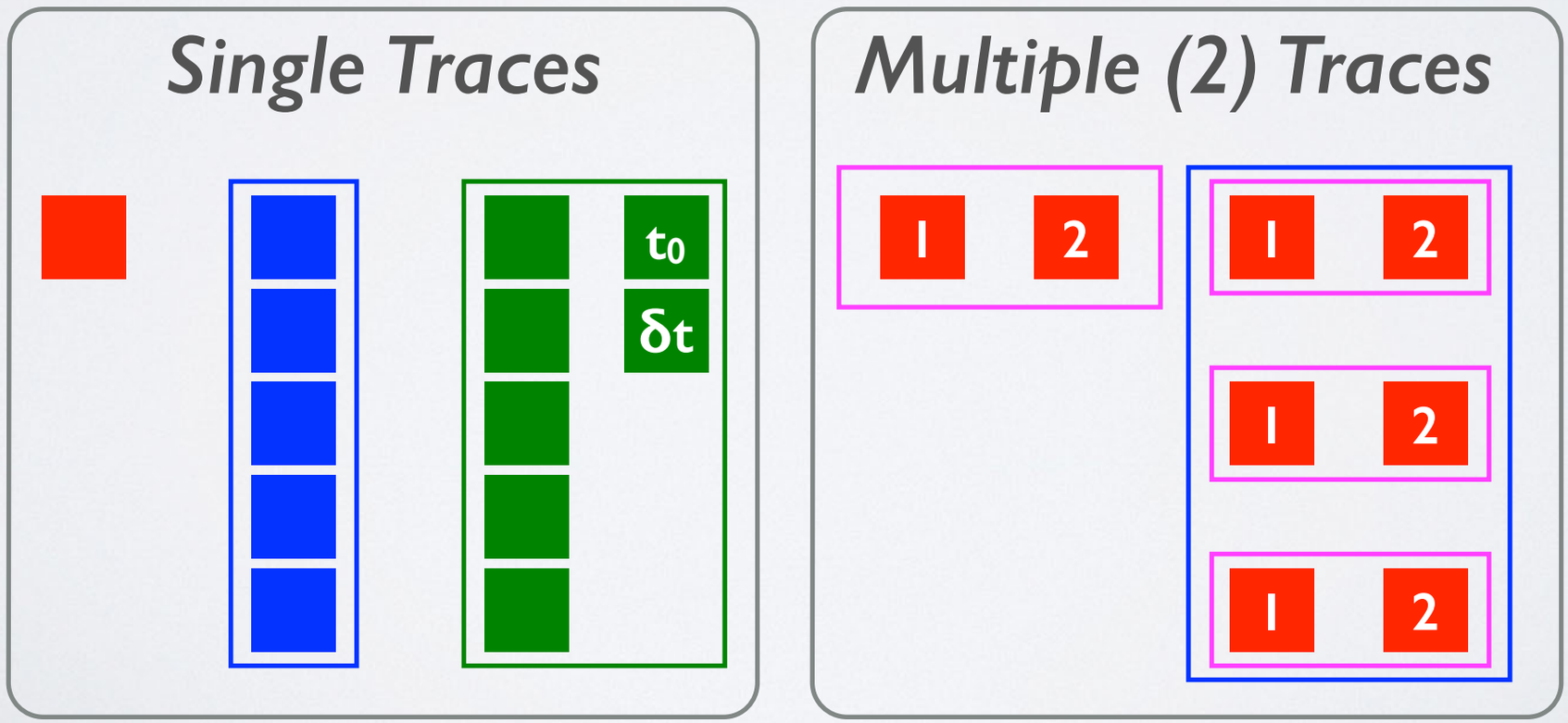
☞The **start timestamp** and **sampling interval** provided by the waveform data type are used to properly calibrate the ***Chart***'s x-axis.

☞If **successive** waveforms are not strictly **temporally adjacent** then only the most recent waveform is plotted.

- Otherwise, successive wavelengths are **appended** to the plot.

PLOTTING USING WAVEFORM CHARTS

- **Single numeric data**, **arrays** and **waveform data** are rendered as **single** traces by LabVIEW charts.
- LabVIEW **clusters** can be used to provide data to charts that is rendered as **multiple** traces.



DEMONSTRATION

Plotting using **Charts** in LabVIEW

PLOTTING USING WAVEFORM GRAPHS

- *LabVIEW provides **Graph** indicators to display data **after** it is accumulated.*
- ***Graph**s do **not** maintain a history of previous updates. Only the **most recent** update is displayed.*
- ***Graph**s are useful for **plotting** summaries of accumulated data.*
- ***Graph**s allow the **x-coordinates** of plotted data to be explicitly specified.*
- LabVIEW provides several different types of ***Graph*** indicator.

PLOTTING USING WAVEFORM GRAPHS

☞ *LabVIEW provides* ***WAVEFORM GRAPH*** *indicators to display* ***evenly sampled*** *data e.g. a snapshot from a* ***Waveform Chart****.*

- ***Waveform Graph*** indicators only plot single-valued functions of the x-coordinate i.e. $y = f(x)$.

☞ ***Waveform Graph*** indicators do not require **explicit specification of x-coordinate values.** They will be inferred from the **types** and **layout** of the data that are supplied.

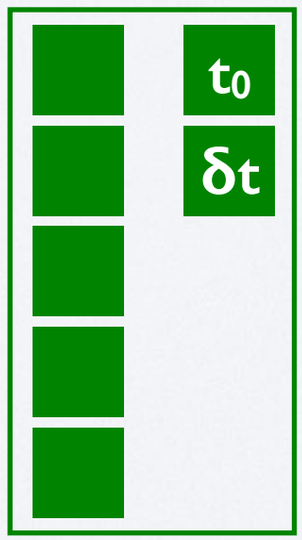
☞ ***Waveform Graph****s* also use data layout to determine whether the data values should be rendered as **single** or **multiple** traces.

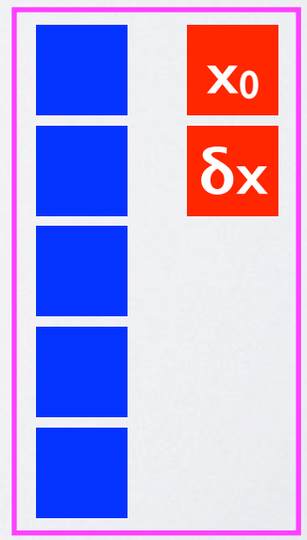
PLOTTING USING WAVEFORM GRAPHS

- The illustrated data layouts (involving combinations of **single data**, **arrays**, **clusters** and **waveforms**) can be used to plot **single** traces, **without** explicitly specifying the x-coordinates.

X-values set to **array index**

X-values computed using **start values** and **sampling intervals**

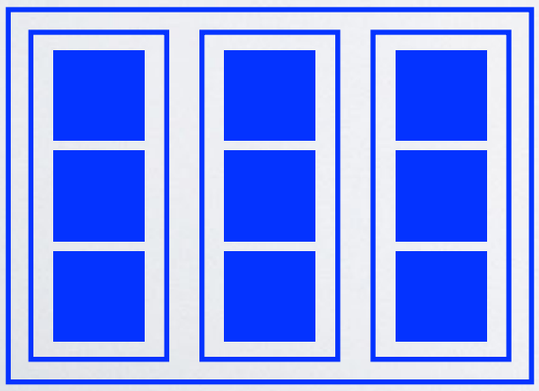




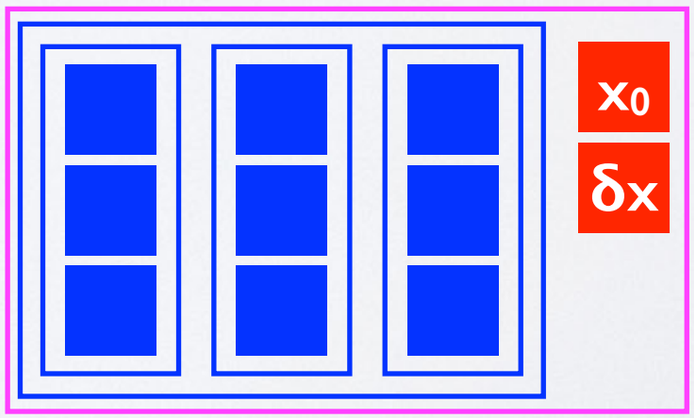
PLOTTING USING WAVEFORM GRAPHS

- The illustrated data layouts (involving combinations of **single data**, **arrays**, **clusters** and **waveforms**) can be used to plot **multiple** traces that share **identical** x-coordinates.

X-values set to **innermost** array index



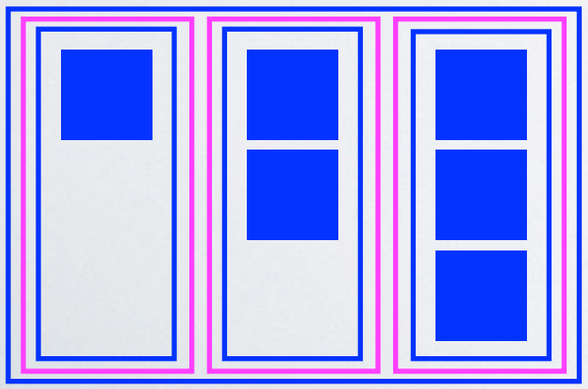
X-values computed using start value ($\mathbf{x_0}$) and sampling interval ($\mathbf{\delta x}$).
All traces share the **same** ($\mathbf{x_0}$, $\mathbf{\delta x}$).



PLOTTING USING WAVEFORM GRAPHS

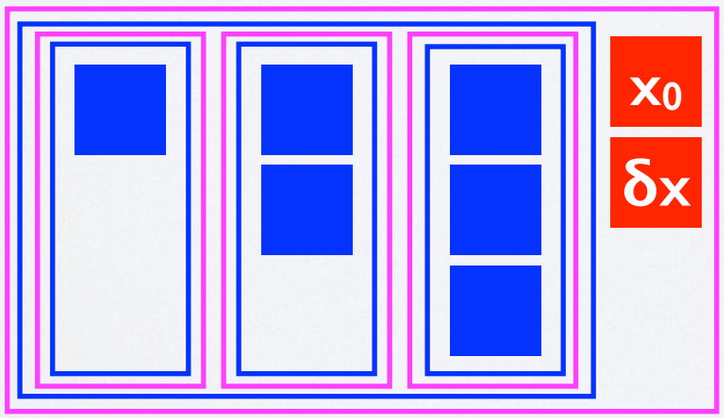
- The illustrated data layouts (involving combinations of **single data**, **arrays**, **clusters** and **waveforms**) can be used to plot **multiple** traces that have **different** x-coordinates or different numbers of points.

X-values set to **innermost** array index

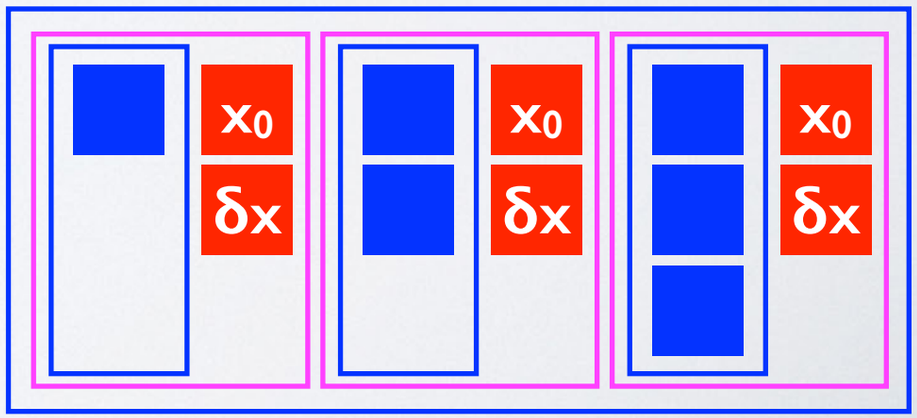


X-values computed using start value ($\mathbf{x_0}$) and sampling interval ($\mathbf{\delta x}$).

Same Sampling



Different Sampling



DEMONSTRATION

Plotting using **Waveform Graphs** in LabVIEW

PLOTTING USING INTENSITY CHARTS AND GRAPHS

- *LabVIEW provides **INTENSITY CHART** indicators to display **three-dimensional** data on a **two-dimensional** Cartesian plane **as they are accumulated**.*

- *LabVIEW provides **INTENSITY GRAPH** indicators to display **three-dimensional** data on a **two-dimensional** Cartesian plane **after they are accumulated**.*

- ***Intensity Charts*** and ***Intensity Graphs*** accept a **two-dimensional array** of numeric data as input.

PLOTTING USING INTENSITY CHARTS

- The **indices** of the input data array are interpreted as **x** and **y** coordinates, implying **even sampling** in those dimensions.
  - ☞By **default**, the **column** indices of the input array are interpreted as **y**-coordinates and the **row** indices are interpreted as **x**-coordinates.
- The **values** of the array elements are mapped to a color value and used to render a **colored region** at the corresponding x and y coordinates.

PLOTTING USING INTENSITY CHARTS

- ☞ ***Intensity Charts*** behave like ***Waveform Charts*** and maintain a **limited** history of recently provided data.
- Each **new** three-dimensional data set is rendered at the **rightmost** edge of the plot and earlier data are shifted to the left.
- ***Intensity Graphs*** handle input data in a similar manner to ***Waveform Graphs.***
- ☞ If new data are supplied to an ***Intensity Graph***, previously supplied data are **discarded**.

DEMONSTRATION

Plotting using **Intensity Charts** and **Intensity Graphs** in LabVIEW

PLOTTING USING XY GRAPHS

☞ *LabVIEW provides **XY GRAPHS** to render **arbitrarily sampled two-dimensional** data sets in a **Cartesian plane**.*

☞ XY Graphs **require explicit** specification of the **x** and **y** coordinates for each datum that they render.

☞ Several **numeric** types and various **layouts** of data can be connected to an XY Graphs as input.

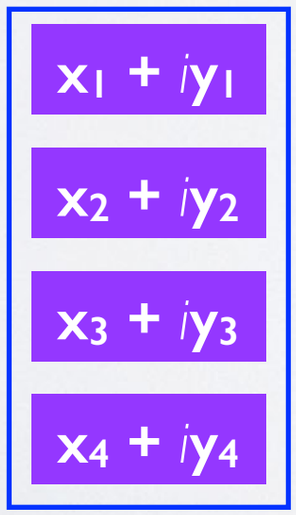
☞ The **layout** of the data defines how the **values** should be **interpreted** and **plotted**.

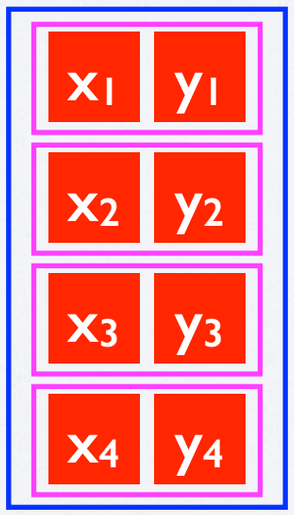
PLOTTING USING XY GRAPHS

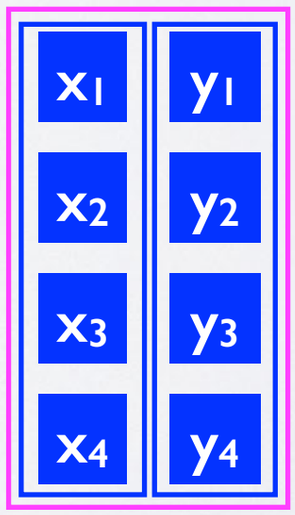
- *LabVIEW provides **XY GRAPHS** to render **arbitrarily sampled two-dimensional** data sets in a **Cartesian plane**.*
- XY Graphs **require explicit** specification of the **x** and **y** coordinates for each datum that they render.
- Several **numeric** types and various **layouts** of data can be connected to an XY Graphs as input.
- The **layout** of the data defines how the **values** should be **interpreted** and whether they should be **plotted** as single or multiple data sets.

PLOTTING USING XY GRAPHS

- The illustrated data layouts (involving combinations of **single numeric data**, **complex valued data**, **arrays** and **clusters**) can be supplied to **XY Graphs** to plot **single** datasets.
- When supplying data as **complex values**, the **real** part is interpreted as the **x**-coordinate and the **imaginary** part is interpreted as the **y**-coordinate.

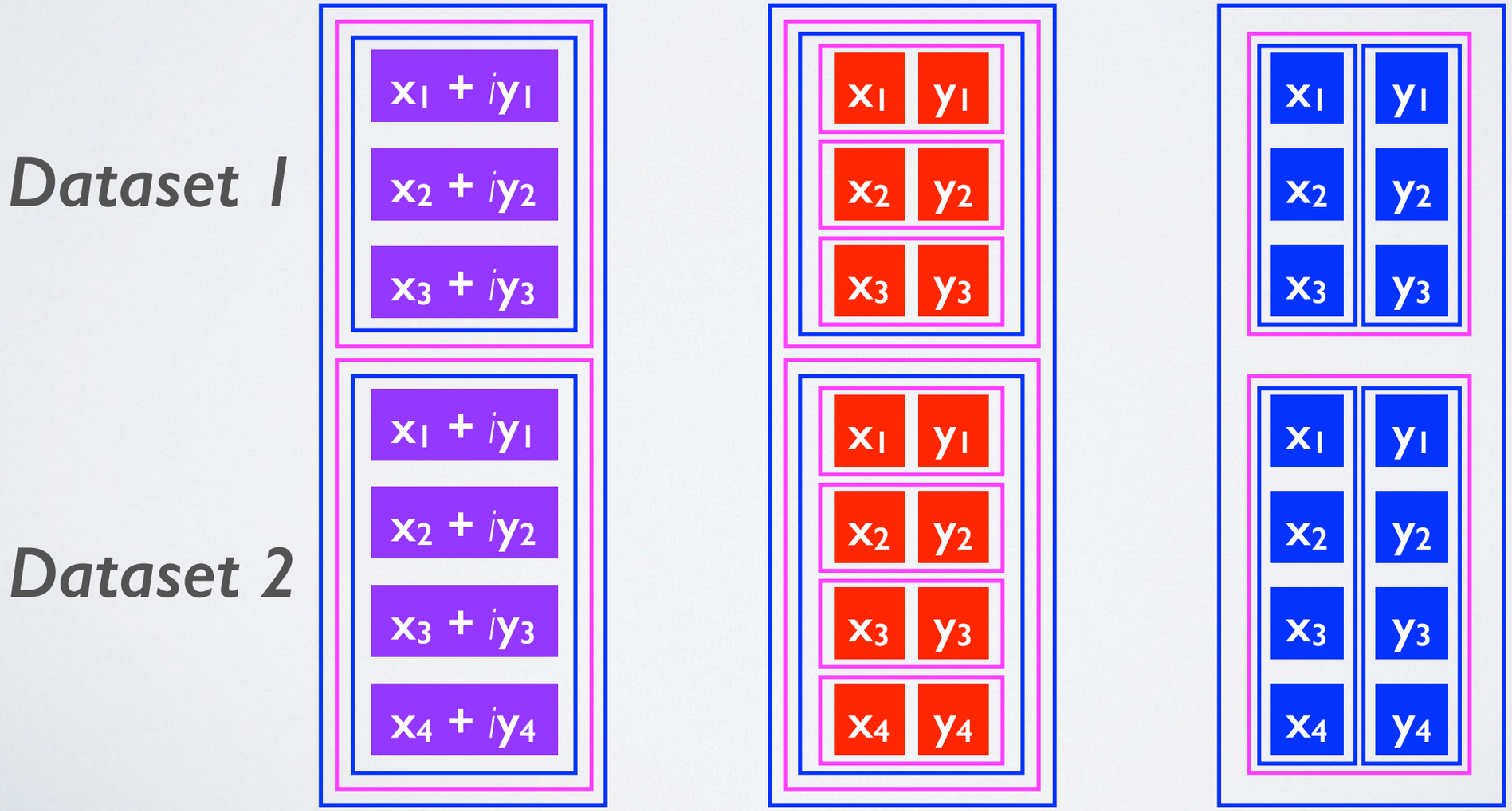






PLOTTING USING XY GRAPHS

- The illustrated data layouts (involving combinations of **single numeric data**, **complex valued data**, **arrays** and **clusters**) can be supplied to **XY Graphs** to plot **multiple** datasets.



DEMONSTRATION

Plotting using **XY Graphs** in LabVIEW

PLOTTING USING 2D AND 3D GRAPHS

☞ *To handle alternative rendering options for multidimensional data, LabVIEW defines* ***abstract 2D GRAPH*** *and* ***3D GRAPH*** *concepts.*

- **Concrete implementations** of the 2D Graph and 3D Graph concepts render 2-dimensional and 3-dimensional data in **specific** ways.

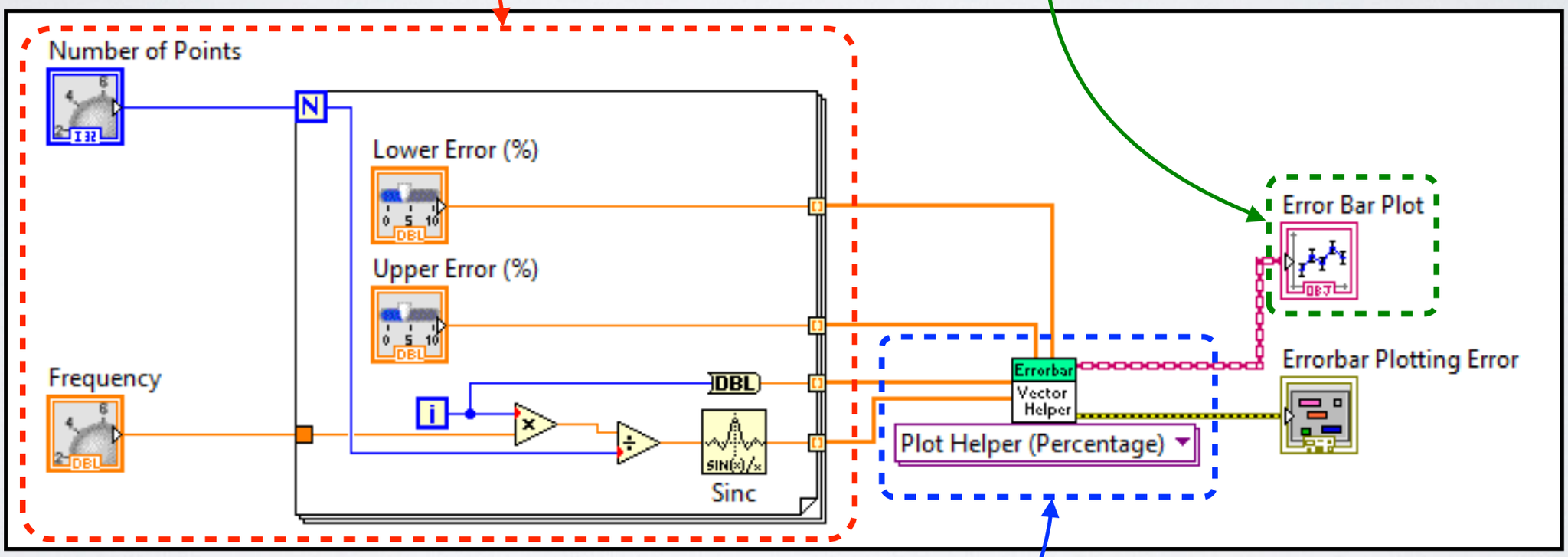
☞ *Each concrete implementation provides a* ***HELPER*** *VIs that* ***format input data*** *appropriately.*

☞ LabVIEW implements **graphs with error bars** as concrete implementations of the 2D Graph concept.

PLOTTING USING 2D AND 3D GRAPHS

Data Generation

Concrete 2D Graph Implementation Plots Data



Helper VI Formats Data

DEMONSTRATION

Plotting using **2D** and **3D Graphs** in LabVIEW

RECOMMENDED READING

Learn LabVIEW Online Video Tutorials

(http://www.ni.com/academic/students/learn-labview/)

LabVIEW Basics Online Reference

(http://www.ni.com/getting-started/labview-basics/)

LabVIEW Manual: Types of Graphs and Charts

(http://zone.ni.com/reference/en-XX/help/371361L-01/lvconcepts/types_of_graphs_and_charts/)

LabVIEW Manual: Handling Errors

(http://zone.ni.com/reference/en-XX/help/371361L-01/lvconcepts/error_checking_and_error_handling/)

LabVIEW Manual: Using the Dynamic Data Type

(http://zone.ni.com/reference/en-XX/help/371361L-01/lvconcepts/dynamic_data_type/)

LECTURE 14 HOMEWORK

Review the ***Recommended Reading*** items listed on the previous slide.

Continue to refine your **final project proposal** and begin working on your **final project** once it is approved.