Key Resources

ZETCODE.COM
Step by step instructions on building some WinForms in Iron Python
http://zetcode.com/tutorials/ironpythontutorial/introduction/

VOIDSPACE.ORG
Similar to ZetCode, with a few more methods and controls covered

MICROSOFT WINFORMS DOCUMENTATION
The full documentation of WinForms. Examples of all available methods etc. Written in C#
https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.form?view=netframework-4.8

DYNAMO FORUM
Its not a popular topic, but there are a lot of people on the forum including myself and the Data-Shapes team that can help
https://forum.dynamobim.com/search?q=winform
Dock & Anchor
IronPython & Windows Forms, Part VII

```python
import clr
clr.AddReference('System.Windows.Forms')

from System.Windows.Forms import Application, Form, Button, DockStyle

class MainForm(Form):

    def __init__(self):
        for i in range(1, 6):
            btn = Button()
            btn.Text = "Button %s" % i
            btn.Dock = DockStyle.Top
            self.Controls.Add(btn)

Application.EnableVisualStyles()
form = MainForm()
Application.Run(form)
```
Key Resources

ZETCODE.COM
Step by step instructions on building some WinForms in Iron Python
http://zetcode.com/tutorials/ironpythontutorial/introduction/

VOIDSPACE.ORG
Similar to Zetcode, with a few more methods and controls covered

MICROSOFT WINFORMS DOCUMENTATION
The full documentation of WinForms. Examples of all available methods etc. Written in C#
https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.form?view=netframework-4.8

DYNAMO FORUM
Its not a popular topic, but there are a lot of people on the forum including myself and the Data-Shapes team that can help
https://forum.dynamobim.com/search?q=winform
KEY TERMINOLOGY

CLASS & OBJECT

A blueprint created by a programmer for an object. This defines a set of attributes that will characterize any object that is instantiated from this class.

```python
# In winforms, any window or a dialog is a Form.
class DropDownForm(Form):
    def __init__(self):
        # the __init__ method inside a class is its constructor
        self.Text = "AU London"  # text that appears in the GUI titlebar
        self.Icon = Icon.FromHandle(icon.GetHicon())  # takes a bitmap image and converts

ddForm = DropDownForm()
```

An Object is simply an instance of a class

```python
ComboBox = ComboBox()  # drop down control form
```
FUNCTION & DEFINITION

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

A definition is simple a user defined function.

def okButtonPressed(self, sender, args):
    self.close()  # Trigger to close the GUI when button is pressed
    self.runNextOutput = True  # If the ok button is pressed set runNextOutput as True

btnOk.Anchor = (AnchorStyles.Bottom | AnchorStyles.Right)
btnOk.Click += self.okButtonPressed  # Register the event on the button press to trigger the def
Key Terminology

**METHOD & PROPERTY**

Similar to a function, it is called by its name, but it is implicitly associated with an object/class.

https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.controlcollection.addrange?view=netframework-

Properties: controllable attributes of an object
Dynamo Setup

#INPUTS HERE:

run = IN[0]
message = IN[1]
listInput = tuple(IN[2])
linkaddress = IN[3]
logoFile = IN[4]
icon = IN[5]
A Boiler plate is essentially code that can be reused without context in new applications

- Not all references are required to build a standalone GUI
  - I keep them incase I will need them further down the line.
  - Then delete unused when imports when it is finished
2: Hello World GUI

• 1: Add the WinForms and drawing assemblies
• 2: import the individual references we need now
• 3: For the main form create a class called “DropDownForm” this will use the Form class to create our GUI
• 4: __init__ is a constructor in Python classes. It's called to create the GUI object and initiate the forms attributes
• 5: self represents the instance of the class. By using the "self" keyword we can access the attributes and methods of the class such as .Text which sets the title of the GUI
• 6: Create an instance of our DropDownForm() class
• 7: Run the application
Tip 1

To find a list of all properties and methods of the Form Class check the below documentation

https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.form?view=netframework-4.8

Tip 2

If you use a control and get the above error it will be because you have not imported the control from Windows.Forms
3: Basic Form Controls

• 1: GUI require widgets/controls: ways of receiving inputs (buttons etc.) When we use any control it must be imported.

• 2: Get a bitmap image from at the input path and use it as the GUI icon

• 3: These methods ensure the GUI comes to the front your screen and is scaled normally

• 4: Get the size of the user's screen and make the GUI ¼ dimension of the screen.

• 5: FormBorderStyle = FormBorderStyle.FixedDialog stops the form from being scaled in size by a user. Disable this for testing incase controls disappear

• 6: Create 2 output values to store the values of the item selected and run control
Setting out the GUI Controls

self.Height = screenSize.Height / 4
uiHeight = self.DisplayRectangle.Height

self.Width = screenSize.Width / 4
uiWidth = self.DisplayRectangle.Width

Spacing = 10
4: Adding Simple Controls

- **1**: The label control allows text to be added to the GUI.
- **2**: Point object is required in order to location a control (X,Y)
- **3**: Size the label using uiWidth + uiHeight. This ensures is scales will the GUI for different type screens.
- **4**: self.Controls.Add() Adds the control to the GUI
- **5**: Get the ratio of logo height to width. Value must be a float as an int will round to the nearest 1.
- **6**: SizeMode scale the image to fit the input size.
- **7**: AnchorStyles lock the control to a given corner(s) this is needed if the user can change the size of the GUI
Setting out the GUI Controls

```
userMessage
```

```
screenSize.ButWidth / 4
uiHeight = self.DisplayRectangle.Height

Spacing = 10

self.Width = screenSize.Width / 4
uiWidth = self.DisplayRectangle.Width
```
5: ComboBox Control

- 1: ComboBox is a drop down and text input control
- 2: Add a list of items for the drop down using .AddRange()
- 3: ComboBoxStyle.DropDownList remove the ability for a user to input a text value into the control
- 4: SelectedIndexChanged+= registers the event handler of the user selecting an item from the control to then run the def dropDownOutput
- 5: self = instance of the form class (dropDownOutput) sender = the control object sending that raised the event Args = the argument/event from the sender assign the select item to userOutput variable
- 6: assign the output of ddForm.userOutput to results and output it
6: Button Controls

- 1: Create a button control called btnOk
- 2: when the button is clicked register the event to run the def okButtonPressed
- 3: when the button is pressed close the form and set the runNextOutput variable btnOk = True, btnCancel = False
- 4: default values created act as output control. So if a user selects an item from the dropdown but does not press the next button the runNextOutput variable will return False.
- 5: User must select an item and click the btnOk to output the select item and True.
6: Finishing Touches

- 1: Create a linkLabel control: A button which activates a link
- 2: link is assigned to helpLink.Tag
- 3: when the link is pressed the event is logged to run the def openLink
- 4: Create a Panel. It’s meant for grouping but can be used for colour
- 5: Sets its colour using the FromArgb method and set the border style to fixed 3D
- 6: If the link is a weblink you need webbrowser.Open()
if it’s a pdf etc you need System.Diagnostics.Process……

Don’t forget the additional import references!