Oide

Tacú leis an bhFoghlaim Ghairmiúil i measc Ceannairí Scoile agus Múinteoirí

Supporting the Professional Learning of School Leaders and Teachers

Computer Science

Professional Learning Booklet

2025-2026

Session 1

Breakout Activities

The focus of these activities is to become familiar with creating charts with Pygal and rendering the output as SVG files so they can be displayed in a web browser.

The octocat icon is the GitHub integration symbol. This symbol appears beside sample code. When you click on the octocat you will be directed to the source code on GitHub. Participants are recommended to copy the code from GitHub for use in breakout activities.

Activity 1.1

The program shows the code for a bar chart of the most popular boys name in 2024 in Ireland.

```
import pygal

boys_chart = pygal.Bar(
    title="Most Popular Boys Names in Ireland - 2024",
    x_title="Name",
)

boys_names = ["Jack", "Noah", "Rían", "Cillian", "James"]
boys_counts = [490, 486, 432, 352, 336]

boys_chart.x_labels = boys_names
boys_chart.add("Boys", boys_counts)
boys_chart.render_to_file("ireland_boys_names_2024.svg")
```

1. Copy the code and run the program. Display the bar chart in your browser.
2. Add a label called *Frequency* to the y-axis.
3. Using the following code, extend the existing program to create a bar chart of the most popular girls name in 2024.

```
girls_names = ["Sophie", "Éabha", "Grace", "Emily", "Fiadh"]
girls_counts = [294, 293, 291, 290, 286]
```

Activity 1.2

The following Python code shows the mean monthly temperatures in Boyle and Roscommon Town. Key in the code or download it from GitHub and complete the following activities.

```
import pygal
#Temperature lists
boyle_temps = [4.7, 4.8, 6.0, 8.1, 10.9, 13.3, 14.7, 14.3, 12.8]
roscommon_temps = [5.2, 5.3, 6.4, 8.5, 11.3, 13.8, 15.2, 14.8, 13.3]
```

1. Create a Pygal line chart object and give it the title *Mean Monthly Temperatures: Boyle v Roscommon Town.*

2. Add x-axis labels for the nine months from January – September.

3. Using the code provided, add the list **`boyle_temps`** as a data series, and **`roscommon_temps`** as a data series.

4. Save the chart (render the chart) to a file called county_roscommon_temps.svg

5. Add a third data series to the line chart for the code below.

```
castlerea_temps = [4.9, 5.0, 6.2, 8.2, 11.0, 13.5, 14.8, 14.4,
12.9]
```

6. Modify the legend to show the three location names, and include a different colour or style for Castlerea.

7. Modify the program to include data for the three towns from October to December shown below, so each data set shows the mean temperatures for 12 months. Make any other appropriate modifications to the chart.

```
boyle_oct_dec = [9.9, 6.8, 5.0]   #Temperature data Oct-Dec
roscommon_oct_dec = [10.5, 7.4, 5.6]
castlerea_oct_dec = [10.1, 7.0, 5.2]
```

Activity 1.3

The following program shows the gold, silver and bronze medals won by Ireland at the Summer Olympics from 1992-2024.

```
years =
["1992","1996","2000","2004","2008","2012","2016","2020","2024"]
gold = [1, 3, 0, 0, 0, 1, 0, 2, 4]
silver = [1, 0, 1, 0, 1, 1, 2, 0, 0]
bronze = [0, 1, 0, 0, 2, 3, 0, 2, 3]
```

1. Create a stacked bar chart using Pygal with the title *Ireland Olympic Medals (1992-2024)*.

2. Add an x-axis label *Olympic Year* and a y-axis label *Number of Medals*.

3. Add the gold list from the code above as a data series with the label Gold. Add the silver and bronze lists as data series with suitable labels.

4. Render the chart as an SVG file called Ireland_olympic_medals.svg

5. Use the following code to change the colours of the medals on the bars to the traditional medal colours.

```
import pygal
from pygal.style import Style

#Create a style with custom colours
custom_style = Style(
    colors=["#FFD700", "#C0C0C0", "#CD7F32"]
#Gold, Silver, Bronze
)
```

Session 2

Breakout Activities

Activity 2.1

The following code shows the sales of the most popular tractor brands in Ireland in 2023.

```
import pygal
bar_chart = pygal.Bar(
    title="Most Popular Tractor Brands in Ireland 2023"
)
bar_chart.x_labels = [
    "John Deere", "Massey Ferguson", "New Holland", "Others",
    "Case-IH", "Valtra", "Claas", "Kubota", "Fendt", "Deutz"
]

bar_chart.add("Units Sold", [595, 374, 334, 273, 194, 115, 94,
89, 81, 43])
bar_chart.render_to_file("ireland_tractor_brands.svg")
```

1. Add an x-axis label *Tractor brand* and a y-axis label *Units sold*.

2. Attributes of the bar chart are set when creating the Bar object as shown below. If there is more than one attribute, they must be separated by a comma. Set the width of the bars to 75 as shown below and render the bar chart.

```
bar_chart = pygal.Bar(width = 75,
    title="Most Popular Tractor Brands in Ireland 2023"
)
```

3. Remove the width setting and set the height of the bars to 150 and render the chart.

4. Margins add space around the chart. Add an attribute to the Bar object **margin=50** and render the chart. Modify the margin value to **150** and render the chart.

5. The margin attribute sets all margins to an equal size. Use the attributes **margin_top, margin_bottom, margin_left** and **margin_right** to set the values of the margins to 30, 50, 70, and 15 respectively.

6. Adjust the widths of the individual bars using the bar_width attribute by setting the **bar_width = 25**. Adjust the spacing between the bars by setting **bar_spacing = 15**.

7. The legend defaults to the left on Pygal charts. Adjust the position of the legend using the **legend_at_bottom** attribute. As this is a Boolean attribute, set the value to **True**.

Activity 2.2

The following program shows the code to generate and embed sparklines for the population of four countries from 1800-2000 in a webpage.

```
import pygal

population_data = {
    "Ireland": [5.1, 6.9, 4.5, 2.9, 3.8],
    "UK": [16.0, 27.5, 38.0, 50.0, 59.1],
    "France": [29.0, 36.0, 41.0, 42.5, 59.3],
    "Germany": [24.0, 35.0, 56.0, 69.0, 82.0]
}

years = ["1800", "1850", "1900", "1950", "2000"]

#Create bar chart
bar_chart = pygal.Bar(show_legend=True, x_title="Year",
y_title="Population (millions)")
for country, data in population_data.items():
    bar_chart.add(country, data)

#Generate table of population data
html_table = bar_chart.render_table(style=True)

#Generate sparklines
spark_texts = {}
for country, data in population_data.items():
    chart = pygal.Line()
    chart.add(country, data)
    spark_texts[country] = chart.render_sparktext()

#Generate HTML page
html_content = f"""
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Population Bar Chart, Table & Sparklines</title>
    <style>
        body {{ font-family: Arial, sans-serif; }}
        table {{ border-collapse: collapse; width: 70%; margin-
bottom: 30px; }}
        th, td {{ border: 1px solid #ccc; padding: 6px; text-
align: center; }}
        th {{ background-color: #eee; }}
    </style>
</head>
```

```
#continued from previous page
<body>
    <h1>Population of Ireland, UK, France, Germany (1800-
2000)</h1>

    <h2>1. Bar Chart</h2>
    {bar_chart.render(is_unicode=True)}

    <h2>2. Data Table for Population of each Country</h2>
    {html_table}

    <h2>3. Sparklines</h2>
    <table>
        <thead>
            <tr><th>Country</th><th>Sparkline</th></tr>
        </thead>
        <tbody>
"""

for country, spark in spark_texts.items():
    html_content += f"
<tr><td>{country}</td><td>{spark}</td></tr>\n"

html_content += """
        </tbody>
    </table>
</body>
</html>
"""
#Save as HTML file
with open("population_combined.html", "w", encoding="utf-8") as
f:
    f.write(html_content)


print("HTML page generated: population combined.html")
```

1. Copy the code and run the program. Open the HTML file that is generated to display a bar chart, a table of population data and sparklines for each country's population.

2. Modify the code to add labels to the x-axis to display the years under each group of columns.

3. Using the data below, modify the original program to display the monthly sunshine hours at four meteorological stations in Ireland to include the following:

a. Display the data as a line chart
b. Add labels to the x-axis and y-axis
c. Add x labels to show the months on the x-axis
d. Set a margin of 60 around all sides of the chart
e. Place the legend at the bottom of the chart
f. Create a table to show the sunshine data from each station
g. Create a set of sparklines for each station
h. Output all data in the form of a webpage

```
#The code displays the average monthly hours of sunshine
#from June to November 2025 in four meteorological
#weather stations in Ireland.
#Data from CSO.ie

months = [
    "June 2025",
    "July 2025",
    "August 2025",
    "September 2025",
    "October 2025",
    "November 2025"
]

shannon = [153.0, 155.1, 145.6, 150.5, 67.5, 79.3]
dublin = [157.7, 158.5, 167.2, 130.6, 69.4, 91.7]
cork = [135.8, 181.4, 138.1, 160.8, 72.3, 78.6]
casement = [181.1, 155.2, 190.6, 138.3, 67.7, 84.3]
```

Session 3

Breakout Activities

Activity 3.1

You have been asked to create a website that visualises Eurovision statistics using charts and tables.

1. Using the following code create a webpage which will display various charts showing some of the interesting Eurovision statistics from the history of the competition. An interactive chart can be embedded in a webpage using the **<embed>** tag shown below. The name of your SVG file should replace the sample.svg file shown in bold. Your SVG files must be saved in the same location as your webpage to be embedded in the webpage.

```
<!DOCTYPE html>
<html>
  <head>
    <!-- ... -->
  </head>
  <body>
    <figure>
      <embed type="image/svg+xml" src="sample.svg" />
    </figure>
  </body>
</html>
```

1. As part of the webpage you have been asked to create a bar chart showing the total wins by country, for the data set shown on the following page. Include all appropriate axes and data labels for the chart.

2. Create line chart showing all the countries which have finished in last place the greatest number of times, including axes and data labels.

3. Use the data for the countries which have placed last in the competition the greatest number of times to create a doughnut (pie) chart. To create a doughnut chart from a pie chart, set the **inner_radius** attribute of the Pie object. The value of the inner radius must be between 0 and 1, where 0 is a full pie chart and 1 is empty space. Typical values are between 0.2 and 0.6.

```
# Total wins by country
countries = ["Ireland", "Sweden", "France", "Luxembourg",
"Netherlands", "United Kingdom", "Israel"]
total_wins = [7, 7, 5, 5, 5, 5, 4]

# Nul points received
nul_countries = ["Austria", "Norway", "Finland", "Spain",
"Switzerland", "Germany"]
nul_points = [4, 4, 3, 3, 3, 3]

# Last-place finishes
last_place_countries = ["Norway", "Finland", "Germany",
"Austria"]
last_place_finishes = [12, 9, 9, 7]
```

4. Using the data provided on the following page, create a webpage which displays the data for Ireland and Sweden's placing at each Eurovision competition since 1965. A value of -1 in the dataset is where the country did not qualify/participate in that year.

5. Create a sparkline for each country to show the trend in their participation. Make a copy of each data set and modify it so the years when each country came first will be a peak in the sparkline.

```
# Years
years = list(range(1965, 2026))

# Ireland positions
ireland_positions = [
    6, 7, 8, 4, 10, 1, 12, 11, 9, 15,
    13, 10, -1, 3, 5, 1, 17, 2, 2, 2,
    8, -1, 1, 4, 3, 1, 17, 1, 1, 1,
    16, 15, 15, 16, -1, 23, 13, -1, 20, -1,
    -1, -1, 24, -1, 23, 9, 23, -1, -1, -1,
    -1, -1, -1, -1, -1, 8, -1, -1, -1, 6,
    -1
]

# Sweden positions
sweden_positions = [
    -1, 2, 8, 7, 5, 3, 5, 2, 8, 1,
    5, 10, 4, 12, 9, 7, 8, 3, 1, 11,
    2, 15, 13, 7, 5, 7, 1, 4, 3, 4,
    8, 19, 5, 6, 5, 18, 8, 21, 3, 11,
    3, 14, 3, 10, 5, 14, 5, -1, 14, 15,
    1, 6, -1, 14, 15, 1, 6, 4, 4, 7,
    4
]
```