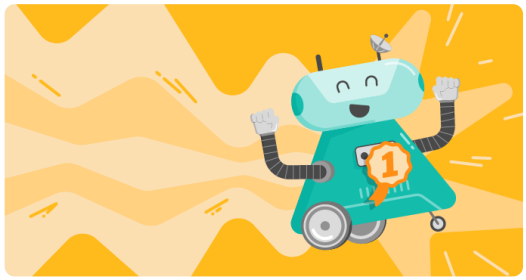
# Charting champions

Discover the power of lists in Python by creating an interactive chart of Olympic medals



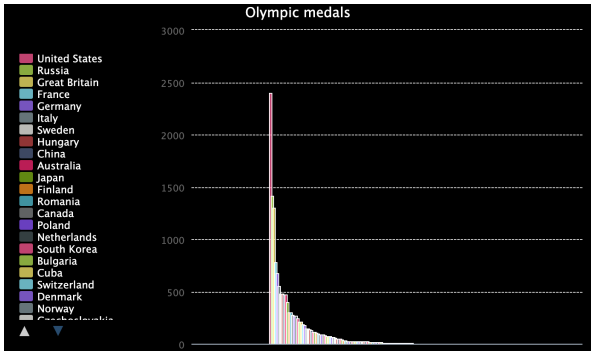
## Step 1 You will make

Discover the power of lists in Python by creating an interactive chart of Olympic medals.

> The Olympic Games began in 1896: thousands of athletes represent hundreds of nations from around the world. The modern games were inspired by ancient contests held in Olympia, Greece.

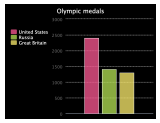
You will:

- Use lists to store related data
- Create a chart using the `pygal` library
- Load data by having your program read a file



## Step 2 Make a chart

Create a chart and some lists of data to display on it.




Open the Charting champions starter project (https://trinket.io/python/61b2224327). Trinket will open in another browser tab.

If you have a Trinket account, you can click on the Remix button to save a copy to your `My Trinkets` library.

If you are not using Trinket in your browser, you will need to download the project files and you may need to install `pygal` before you can import it.

### Offline project files

Download the project files, unzip them, and store them on your computer. For this step you will need `starter.py`, the other files will be used later in the project.

### Installing pygal

On Windows

In the Command Prompt type the following and press the **Enter** key:

```
pip install pygal
```

Wait for the installation to complete and then continue with the project.

On a Mac

In the Terminal type the following and press the **Enter** key:

```
pip3 install pygal
```

Wait for the installation to complete and then continue with the project.

On Linux, including Raspberry Pi OS

In the Terminal type the following and press the **Enter** key:

```
pip install pygal
```

Wait for the installation to complete and then continue with the project.

The starter project already has some code to import the `pygal` library, which you will use to draw your chart.

main.py

```
from pygal import bar
```

Make a chart

Find the `# Create a chart` comment and add code below it to make a bar chart called `chart`. Then give your chart a title.

main.py

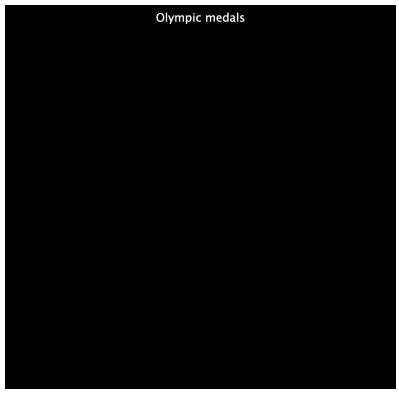
```
# Create a chart
chart = Bar()
chart.title = 'Olympic medals'
```

Call `chart.render()` to display the chart.

main.py

```
# Display the chart
chart.render()
```

Test: Run your code to see the chart. It will be empty because it doesn't have data yet.




Debug: If you see an error about `Bar()` or `chart.render()` being `not defined`:

- If the error is for `Bar()`, make sure it has an uppercase B at the start, and brackets at the end
- If the error is for `chart.render()`, check that it has the `.` between `chart` and `render`, as well as the brackets at the end

Debug: If you are not using Trinket, and the graph hasn't appeared when you run your code, replace `chart.render()` with `chart.render_in_browser()`.

Add some data

Python can store related data as a list. You can create lists by using square brackets `[]`. Items in a list are separated with commas.

Create three lists of data to show on your chart.

Each list will store a nation's name and the number of medals won by that nation.

main.py

```
# Add data to the chart
us = ['United States', 2399]
ru = ['Russia', 1413]
gb = ['Great Britain', 1304]
```

When you store something in a list, it gets an index. An index is a number that tells you an item's position in a list. List indexes start from `0`, instead of `1`.

You can get an item from a list by its index. For example, `my_list[3]` will get the fourth item in `my_list`, because indexes start at `0`.

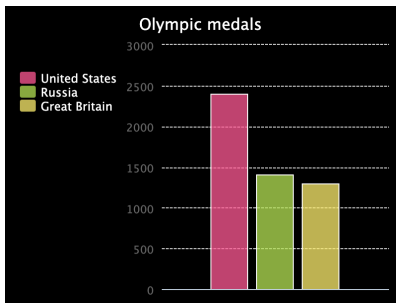
Use the indexes of your lists and `chart.add()` to display your data. The nation's name at item 0 will be used as a category label for the chart and the amount of medals at item 1 will determine the height of the bar.

main.py

```
gb = ['Great Britain', 1304]

chart.add(us[0], us[1])
chart.add(ru[0], ru[1])
chart.add(gb[0], gb[1])
```

Test: Run your code to see the chart.




Debug: If you see a message about an `IndexError`, your code is trying to get a value from a list index that doesn't exist (e.g. `us[2]`). To fix this:

- Check each of your `chart.add` lines to be sure you are only using `0` and `1` as indexes.
- Check the lines where you created your lists. Make sure each list has two items, separated by a comma.

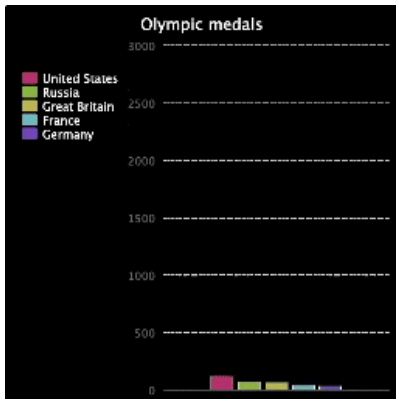
Now load two more teams by adding new lists and `chart.add()` calls.

main.py

```
# Add data to the chart
us = ['United States', 2399]
ru = ['Russia', 1413]
gb = ['Great Britain', 1304]
fr = ['France', 780]
de = ['Germany', 671]

chart.add(us[0], us[1])
chart.add(ru[0], ru[1])
chart.add(gb[0], gb[1])
chart.add(fr[0], fr[1])
chart.add(de[0], de[1])
```

Test: Run your code to see the updated chart. Try clicking on the United States' name. Then watch the scale of the chart change.



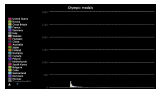

Debug: If you see a message about an `IndexError`, your code is trying to get a value from a list index that doesn't exist (e.g. `fr[2]`). To fix this:

- Check each of your `chart.add` lines to be sure you are only using `0` and `1` as indexes.
- Check the lines where you created your lists. Make sure each list has two items, separated by a comma.

Save your project

# Step 3 Load data from a file

The chart looks good! But, almost 150 nations have competed in the Olympics. To chart them, you're going to load their data from a file. It will save a lot of typing!



Computers and data You're just starting to learn how to get your computer work with data. Computers can do amazing things with the right data. And they can read more data in minutes than a human could in years. Python is one of the best programming languages there is for data. Python is what the YouTube algorithm, that picks the videos to show you, is made with.

Open the second starter project (https://trinket.io/python/b8e0125fe2). Trinket will open in another browser tab.

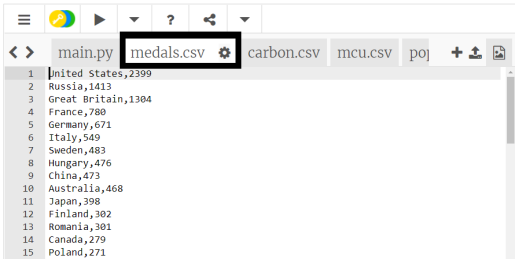
If you have a Trinket account, you can click on the Remix button to save a copy to your `My Trinkets` library.

Working offline

You will need `starter2.py` and the `.csv` files for this step.

There are several `.csv` files included in this starter project that contain the data you need for your charts.

Open `medals.csv` and look at the data in it. See how each line has a team name and the number of medals they have won, separated by a comma.




CSV files are Comma-Separated Values files. They contain data in rows and columns, like a table. Each line is a row, with commas separating that row's values into columns.

```
United States,2399
Russia,1413
Great Britain,1304
France,780
Germany,671
```

You'll need to turn each line of `medals.csv` into a text string and a number in Python, like in the lists you made.

Click on the `main.py` tab and add code to load the file into a variable by using `with open() as`. Then use a `for` loop to `print` each line from the variable.

The `for` loop will let you repeat code. So you will load hundreds of teams to your chart with just a few lines of code!

### Read a file with Python

To read a text file in Python you must `open` the file and then `read` its contents.

When opening the file, use `with` and `as`. This makes sure that, when your indented code has run, the file will automatically close. Closing files you don't need saves memory in your computer.

```
with open(filename) as f:
```

Where `filename` is the name of the file you are opening, e.g. `'info.txt'`.

Once you load your file, you can turn the whole file into a text string. Or, you can use a `for` loop to go through the file line-by-line.

#### Turn the whole file into a text string

The file is loaded into the `f` variable, but not as a text string that Python can work with. To get the file as text, you need to use `read()`.

```
with open(filename) as f:
  file_text = f.read()
  # Do something with the text
```

#### Loop through the file, line by line

The file is loaded into the `f` variable. You can use a `for` loop to run the same code on each line of the file.

```python
with open(filename) as f:
for file_line in f:
# Do something with the line
```

main.py

```
# Add data to the chart
with open('medals.csv') as f:
  for line in f:
    print(line)
```

Test: Run your code and look at the text it prints out.

Notice that each line has two values, separated by commas.

```
Togo,1

Tonga,1

United Arab Emirates,1

Virgin Islands,1

Liechtenstein,0
```

Debug: If the code doesn't work, make sure you have indented it under the `with` line, like in the example above.

Each string that your loop prints is made up of two pieces separated by a comma. Your `chart.add()` function needs each of those pieces as separate inputs.

The `split()` function breaks a string into a list, just like the lists you made earlier. The `split(',')` function makes a new list item every time it sees a comma.

Put a `#` in front of the code that prints `line`. This will turn that code into a comment, so Python will ignore it.

Use the `split()` method to break up each sting at a `,` and then store the first and second pieces in a new list. Then print those lists out.

main.py

```
9  with open('medals.csv') as f:
10   for line in f:
11     #print(line)
12     pieces = line.split(',') # Breaks the string into a list
13     print(pieces) # Print each list
```

Tip: `split()` can split a string into a list around any text you want. You can split on punctuation, a letter, or even whole words.

Test: Run your code and look at the text it prints out. Each line should be a list with two items. You may notice that the second item has `\n` at the end. `\n` is usually invisible. It tells the computer it has reached the end of the line in a file.

```
['Tonga', '1\n']
['United Arab Emirates', '1\n']
['Virgin Islands', '1\n']
['Liechtenstein', '0\n']
```

Debug: If your `pieces` are printing out as lists with only one item then check that you have `','` in the `()` of `line.split()`.

Debug: If you see a message about `split` being 'not defined', check that you have included `line.` before it.

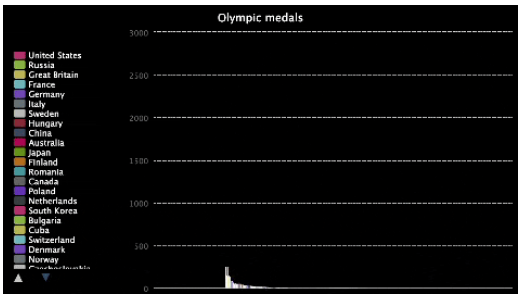
Load your data into the chart as part of your `for` loop. `team` is a string so can be used as a label on the chart. `medal` is currently a string, but needs to be converted to a number. You can use the `int()` function to cast a string to a number.

main.py

```
with open('medals.csv') as f:
  for line in f:
    #print(line)
    pieces = line.split(',')
    #print(pieces)
    team = pieces[0]
    medals = pieces[1]
    chart.add(team, int(medals))  # Make medals a number
```

Tip: You can now use `#` to turn `print(pieces)` into a comment too.

Test: Run your code and look at the chart it creates. Try hovering over some of the bars, or clicking on the names of teams to add and remove them from the chart.




Debug: If your chart is empty, check that you have `int(medals)` in your `chart.add()`.

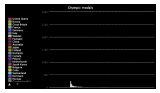
Debug: If you see a message about an `IndexError`, your code is trying to get a value from a list index that doesn't exist (e.g. `pieces[2]`). To fix this:

- Check each of your `team` and `medals` variables to be sure you are only using `0` and `1` as indexes.
- Check the printed `pieces` lists to be sure they have two items: `['Tonga', '1\n']`, not `['Tonga,1\n']`. If they don't, then check that you have `','` in the `()` of `line.split()`.
- Check you do not have a blank line at the bottom of your .csv file.

Save your project

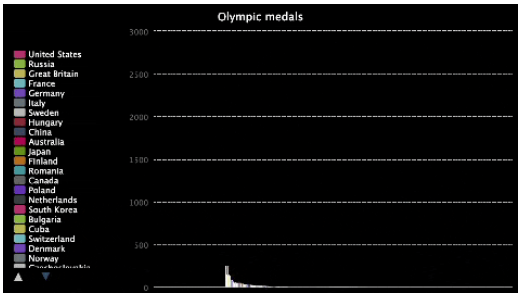
# Step 4 Investigate with data

Now your program can draw charts from files of data. You can use it on different files to compare their charts to see what you can learn.



Who has the most medals?

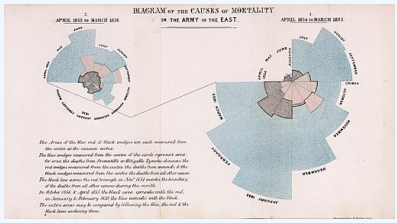
Look at the chart you've made. The taller a bar is, the more medals that team has won. Hover the mouse over some of the tallest bars and notice which teams they belong to.




Why might they have the most medals?

A good idea might be to look at both the population and wealth of teams, to see if there is any sort of pattern.

Data analysis: People have done these kinds of investigations since long before computers were invented. For example, in the 1850s, Florence Nightingale, a nurse, used charts and graphs to show the importance of disease prevention in caring for the sick.




Population sizes

A file, called `pop.csv`, with data on the populations of different countries, is part of the starter project. Because the data in `pop.csv` is also made up of a text string and a number, you can re-use your code with only small changes.

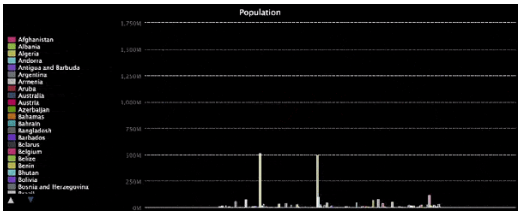
Change the chart title, the file you are opening, and the category name to draw a chart based on the population data in `pop.csv`.

main.py

```
chart.title = 'Population'

# Add data to the chart
with open('pop.csv') as f:
  for line in f:
    #print(line)
    pieces = line.split(',')
    #print(pieces)
    team = pieces[0]
    population = pieces[1]
    chart.add(team, int(population))  # Make population a number
```

Now run your program and look at the chart it draws.




Hover the mouse over the biggest bars and notice which countries they belong to. Click the names of the really big ones to remove them from the chart; that will let you get a closer look at the others. Do any of the countries with lots of people have a large number of medals?

## Wealth

A file called `gdp.csv` is part of the starter project. It has data on the annual GDP of different countries. Just like with `pop.csv`, you'll only need to make small changes to use it.

> GDP is the Gross Domestic Product. It measures the value, in money, of everything produced in an area over a given time period. It can measure how rich an area is.

Change the chart title, the file you are opening, and the category name to draw a chart based on the GDP data in `gdp.csv`.

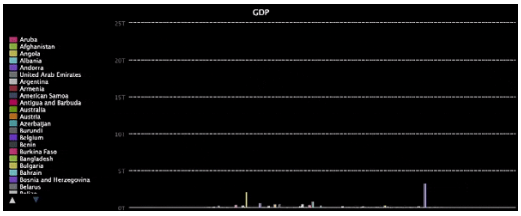
The `gdp.csv` file stores the GDP as decimal numbers. Update the type from `int` to `float` so that the numbers are in the correct format.

main.py

```
chart.title = 'GDP'

# Add data to the chart
with open('gdp.csv') as f:
  for line in f:
    #print(line)
    pieces = line.split(',')
    #print(pieces)
    team = pieces[0]
    gdp = pieces[1]
    chart.add(team, float(gdp))  # Make GDP a number
```

Now run your program and look at the chart it draws.




Hover the mouse over the biggest bars and notice which countries they belong to. Click the names of the really big ones to remove them from the chart; that will let you take a closer look at the others. Did any of the richest countries' teams have very large numbers of medals?

What did you find?

What did you discover by using your program to look at this data?

- There are some signs that the number of people a team has to choose from helps it earn medals.
- But population doesn't explain how countries like France have so many medals. Or why India doesn't have as many medals as China or the USA.
- Money seems to explain more. Most of the countries that have lots of medals have high GDPs too.
- Neither of them explains everything. There are teams that don't follow this pattern.

### Jamaica does better than bigger and richer countries

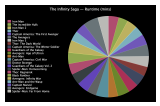
This isn't a problem in a maths book. So the answer isn't simple. For example, look at Jamaica. Jamaica doesn't have a large population, or a large GDP.

Mexico, Ireland, Portugal, Saudi Arabia, and Singapore all have more people and more money. They even have more money per person. But Jamaica has won more medals than any of them!

So there's more to what it takes to win Olympic medals than just people and money. What else might it be? What other ideas could you test, and what kind of data would you need to do so?

## Upgrade your project

In this step, change how your chart looks, or what data it uses.



### Use a pie chart

Try using a pie chart for a different look, or to show how something is divided.

To create a pie chart instead of a bar chart, change the import from `pygal` to `Pie` instead of `Bar`. Do the same for the function you call to create `chart`.

### Use a different set of data

You can load and chart any data that's in a `.csv` file with the program you've written.

Choose: Pick a different datafile for your project. There are two available:

- `mcu.csv` is the runtime and gross income from the Marvel Cinematic Universe films
- `carbon.csv` is the total (thousands of tons) and per-person (tons) carbon dioxide emissions of different countries and regions

Update the code that reads from `medals.csv` to read from your new file.

These files have more than one column of numbers. Use indexes on the `tally` list to choose which to add to your chart.

The carbon dioxide data uses numbers with decimals. To convert them from text strings, you'll need to use `float()` instead of `int()`.

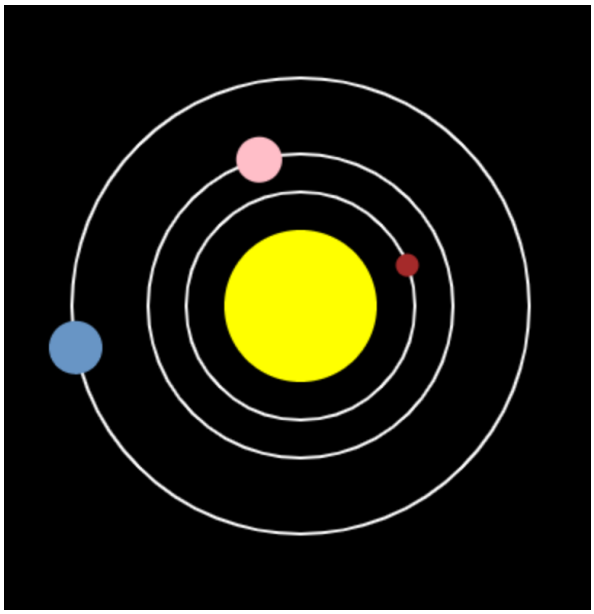
### Completed project

You can view the completed project here (https://trinket.io/python/1f312ddc4c).

Save your project

What next?

If you are following the More Python (https://projects.raspberrypi.org/en/pathways/more-python) path, you can move on to the Solar system (https://projects.raspberrypi.org/en/projects/solar-system-simulator/) project. In that project, you will make a model of the solar system to teach people about the planets.



If you want to have more fun exploring Python, then you could try out any of these projects (https://projects.raspberrypi.org/en/projects?software%5B%5D=python).

---