Download (right-click, save target as ...) this page as a jupyterlab notebook from: Lab15

Laboratory 15: Matplotlib

Medrano, Giovanni

R11521018

ENGR 1330 Laboratory 12 - In-Lab

In [1]:
```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Matplotlib and Visual Display of Data

This lesson will introduce the `matplotlib` external module package, and examine how to construct line charts, scatter plots, bar charts, and histograms using methods in `matplotlib` and `pandas`

The theory of histograms will appear in later lessons, here we only show how to construct one using `matplotlib`

About `matplotlib`

Quoting from: https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py

`matplotlib.pyplot` is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In `matplotlib.pyplot` various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

Background

Data are not always numerical. Data can music (audio files), or places on a map (georeferenced attributes files), images (various imge files, e.g. .png, jpeg)

They can also be categorical into which you can place individuals:

- The individuals are cartons of ice-cream, and the category is the flavor in the carton
- The individuals are professional basketball players, and the category is the player's team.

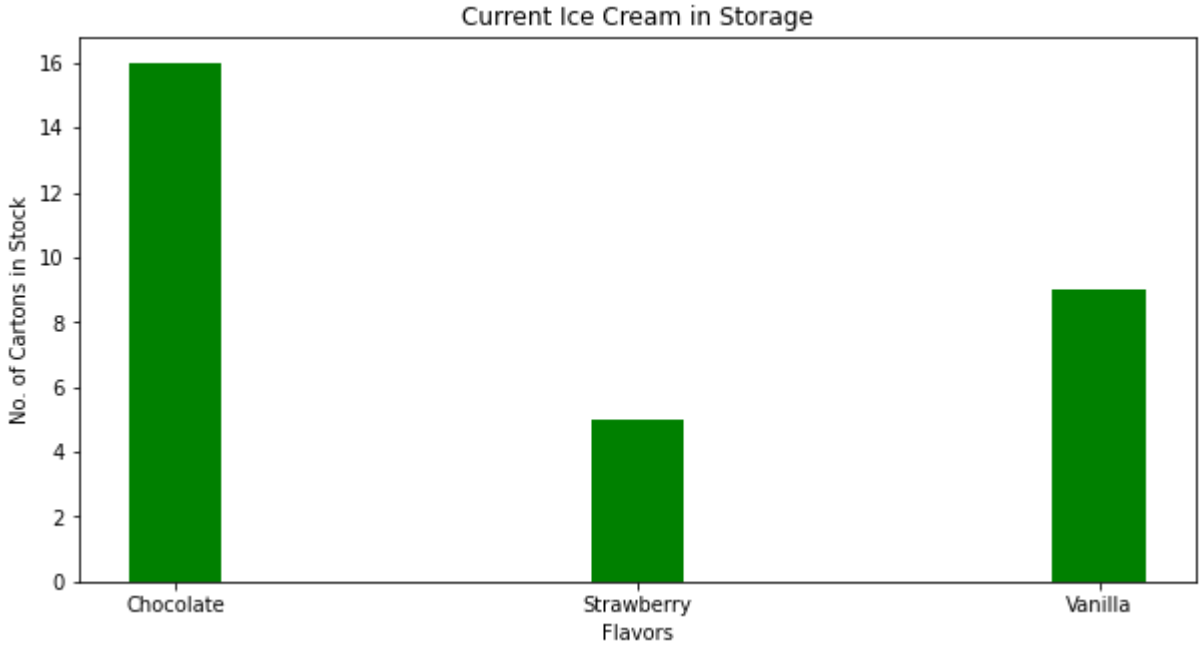
Bar Graphs

Bar charts (graphs) are good display tools to graphically represent categorical information. The bars are evenly spaced and of constant width. The height/length of each bar is proportional to the `relative frequency` of the corresponding category.

`Relative frequency` is the ratio of how many things in the category to how many things in the whole collection.

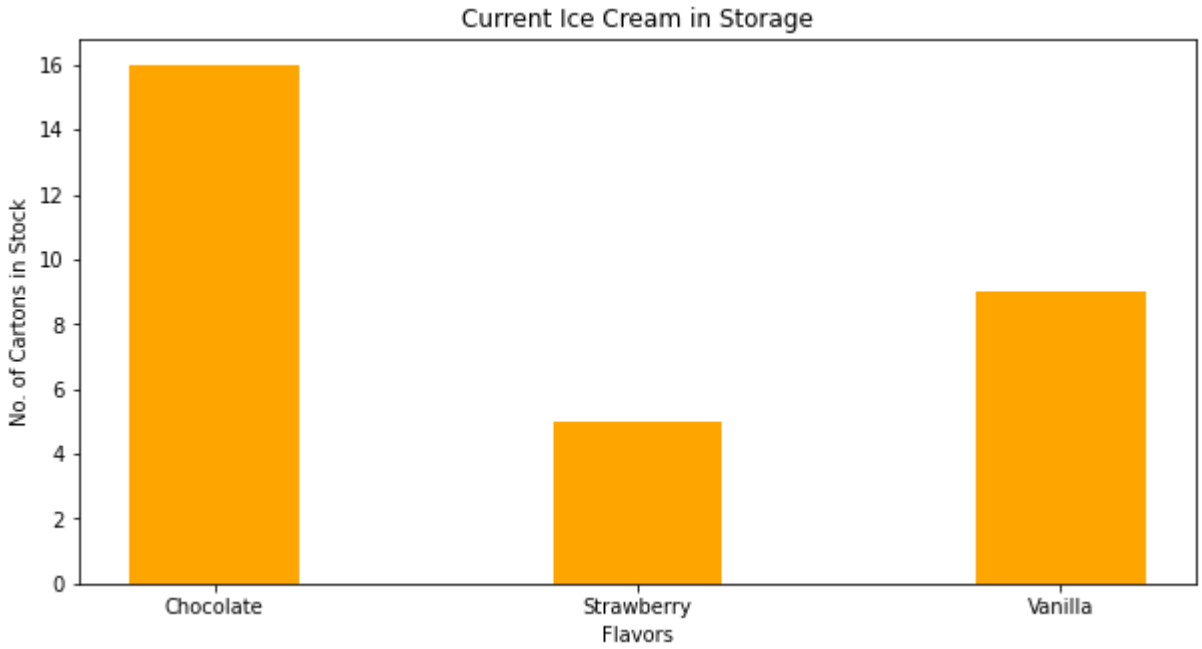
The example below uses `matplotlib` to create a box plot for the ice cream analogy, the example is adapted from an example at https://www.geeksforgeeks.org/bar-plot-in-matplotlib/

In [6]:
```
ice_cream = {'Chocolate':16, 'Strawberry':5, 'Vanilla':9} # build a data model
import matplotlib.pyplot # the python plotting library

flavors = list(ice_cream.keys()) # make a list object based on flavors
cartons = list(ice_cream.values()) # make a list object based on carton count -- assume

myfigure = matplotlib.pyplot.figure(figsize = (10,5)) # generate a object from the figu

# Built the plot
matplotlib.pyplot.bar(flavors, cartons, color ='green', width = 0.2)
matplotlib.pyplot.xlabel("Flavors")
matplotlib.pyplot.ylabel("No. of Cartons in Stock")
matplotlib.pyplot.title("Current Ice Cream in Storage")
matplotlib.pyplot.show()
```



Lets tidy up the script so it is more understandable, a small change in the import statement makes a simpler to read (for humans) script - also changed the bar colors just 'cause!

In [7]:
```
ice_cream = {'Chocolate':16, 'Strawberry':5, 'Vanilla':9} # build a data model
import matplotlib.pyplot as plt # the python plotting library

flavors = list(ice_cream.keys()) # make a list object based on flavors
cartons = list(ice_cream.values()) # make a list object based on carton count -- assume

myfigure = plt.figure(figsize = (10,5)) # generate a object from the figure class, set

# Built the plot
plt.bar(flavors, cartons, color ='orange', width = 0.4)
plt.xlabel("Flavors")
plt.ylabel("No. of Cartons in Stock")
plt.title("Current Ice Cream in Storage")
plt.show()
```



Using pandas, we can build bar charts a bit easier.

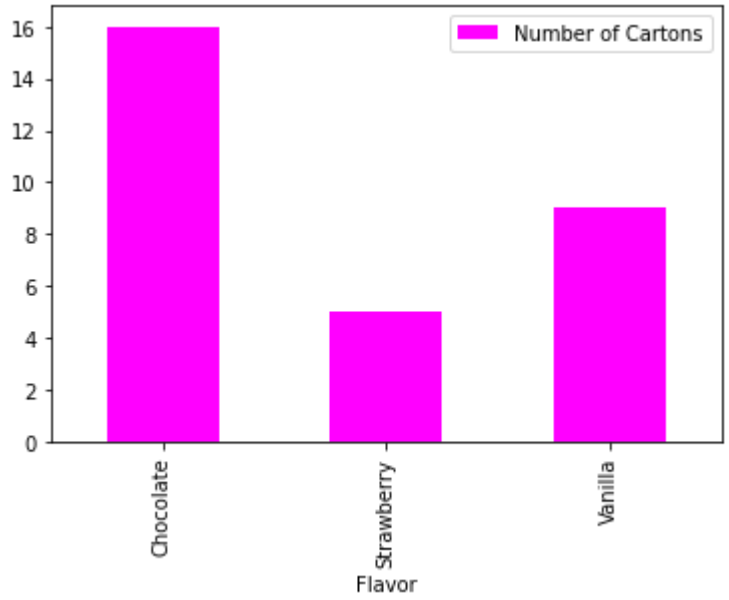
In [8]:
```
import pandas as pd

my_data = {
    "Flavor": ['Chocolate', 'Strawberry', 'Vanilla'],
    "Number of Cartons": [16, 5, 9]
          }
df = pd.DataFrame(my_data)
df.head()
```

Out[8]:

	Flavor	Number of Cartons
0	Chocolate	16
1	Strawberry	5
2	Vanilla	9

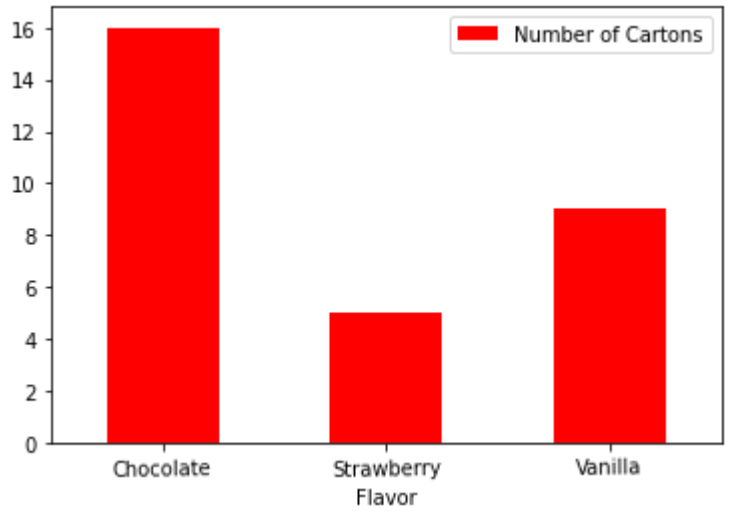
In [9]:
```
df.plot.bar(x='Flavor', y='Number of Cartons', color='magenta' )
```

Out[9]: <AxesSubplot:xlabel='Flavor'>



In [11]:
```
df.plot.bar(x='Flavor', y='Number of Cartons', color="red", rot = 1) # rotate the categ
```

Out[11]: <AxesSubplot:xlabel='Flavor'>



Example- Language Bars!

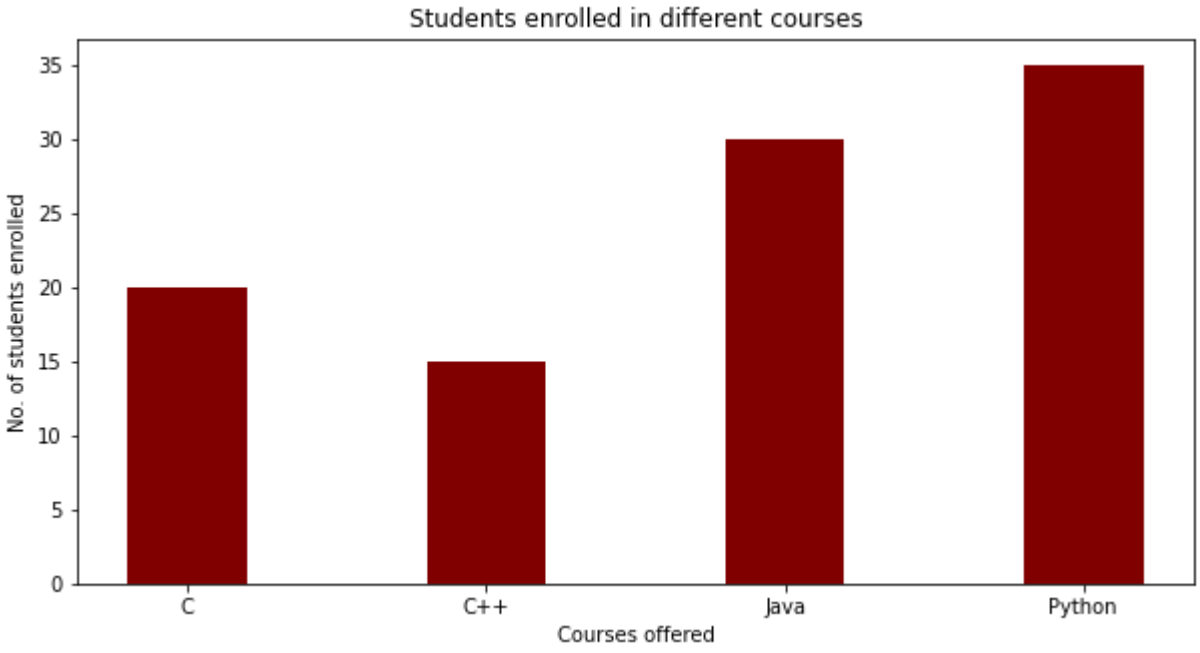
Consider the data set "data" defined as

```
data = {'C':20, 'C++':15, 'Java':30, 'Python':35}
```

which lists student count by programming language in some school.

Produce a bar chart of number of students in each language, where language is the classification, and student count is the variable.

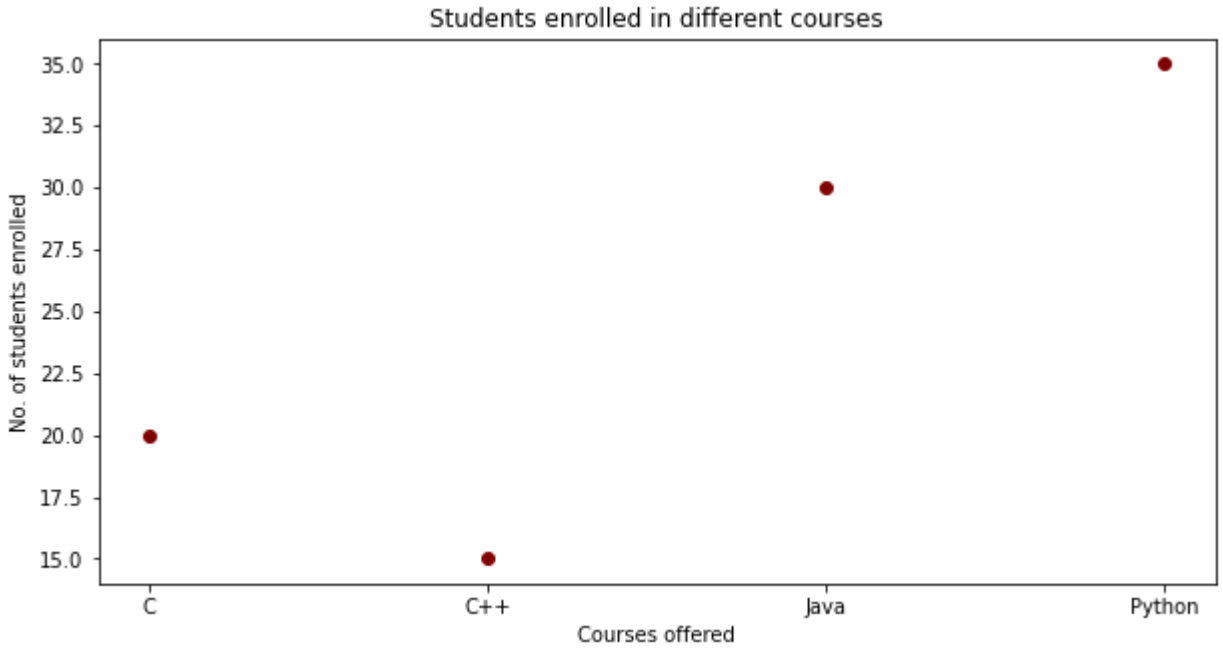
In [12]:
```
# Code and run your solution here

import numpy as np
import matplotlib.pyplot as plt


# creating the dataset
data = {'C':20, 'C++':15, 'Java':30,
        'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
plt.show()
```



Plot it as a horizontal bar chart:

In [17]:
```
# Code and run your solution here

# creating the dataset
data = {'C':20, 'C++':15, 'Java':30,
        'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.scatter(courses, values, color ='maroon')
```

```
plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
plt.show()
```



Line Charts

A line chart or line plot or line graph or curve chart is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments.

It is a basic type of chart common in many fields. It is similar to a scatter plot (below) except that the measurement points are **ordered** (typically by their x-axis value) and joined with straight line segments.

A line chart is often used to visualize a trend in data over intervals of time – a time series – thus the line is often drawn chronologically.

The x-axis spacing is sometimes tricky, hence line charts can unintentionally decieve - so be careful that it is the appropriate chart for your application.

Example- Speed vs Time

Consider the experimental data below

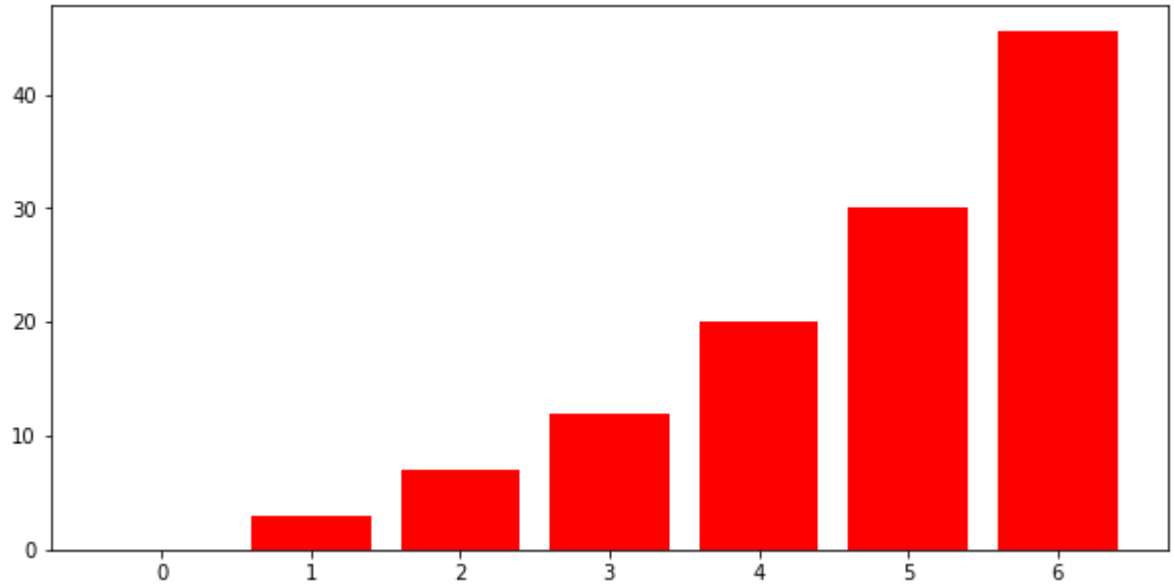
Elapsed Time (s)	Speed (m/s)
0	0
1.0	3

Elapsed Time (s)	Speed (m/s)
2.0	7
3.0	12
4.0	20
5.0	30
6.0	45.6

Show the relationship between time and speed. Is the relationship indicating acceleration? How much?

In [14]:
```
# Create two lists; time  and speed.
time = [0,1.0,2.0,3.0,4.0,5.0,6.0]
speed = [0,3,7,12,20,30,45.6]
```

In [18]:
```
# Create a line chart of speed on y axis and time on x axis
mydata = plt.figure(figsize = (10,5)) # build a square drawing canvass from figure clas
plt.bar(time, speed, color='red') # basic line plot
plt.show()
```



From examination of the plot, estimate the speed at time t = 5.0 (eyeball estimate)

Example- Add a linear fit

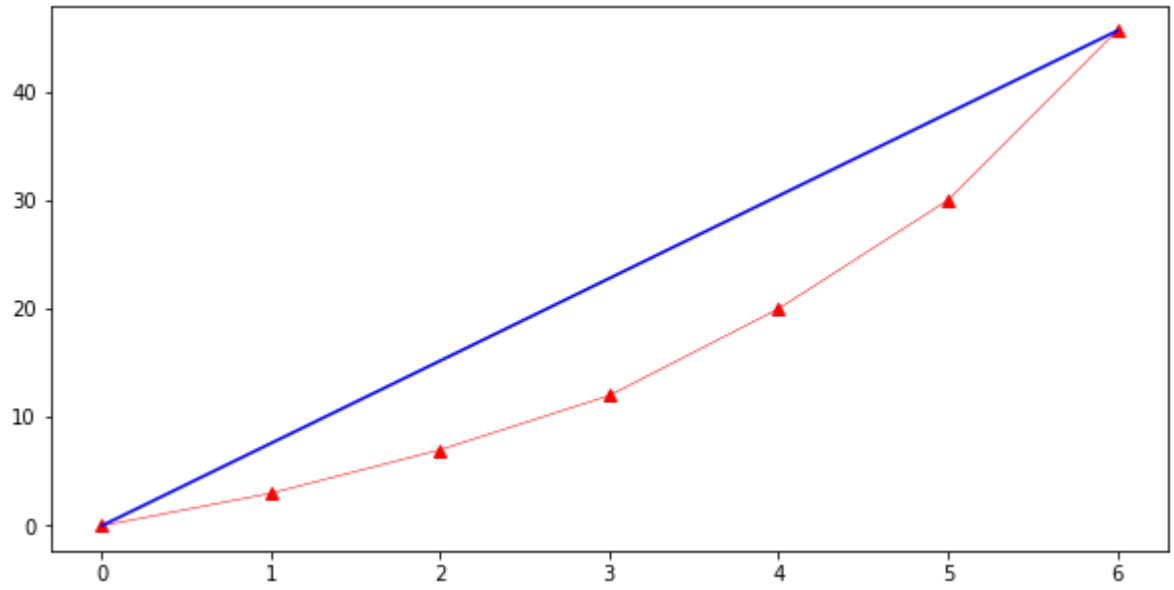
Using the same series from Exercise 1, Plot the speed vs time (speed on y-axis, time on x-axis) using a line plot. Plot a second line based on the linear model

$$y = mx + b$$

,

where $$b=0~\text{and}~m=7.6$$.

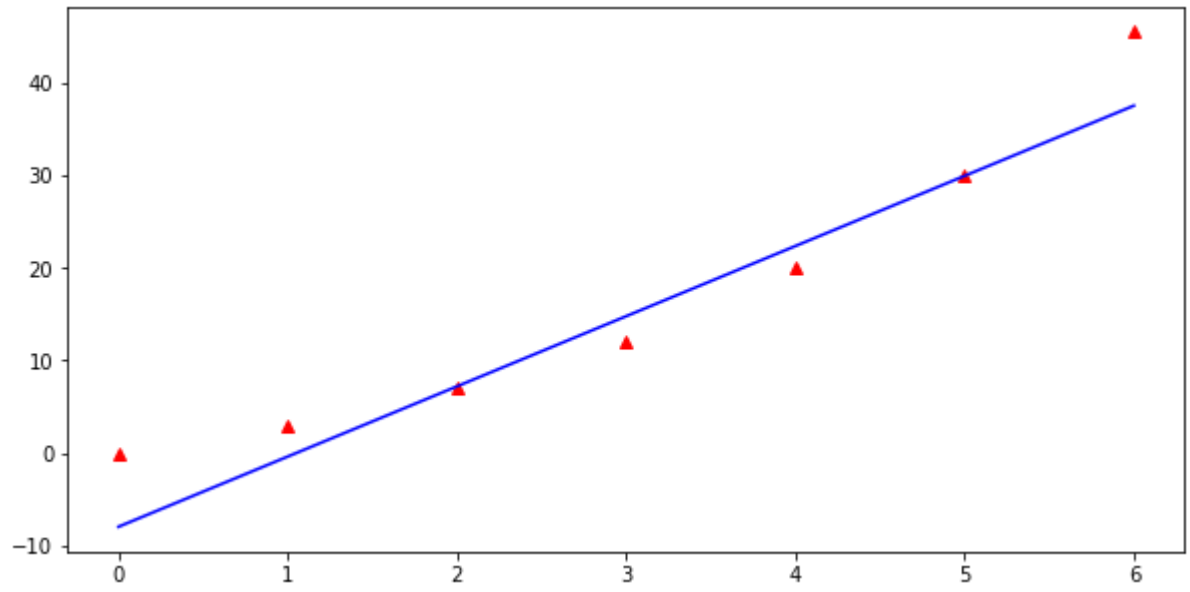
In [19]:
```
# Code and run your solution here:
def ymodel(xmodel,slope,intercept):
    ymodel = slope*xmodel+intercept
    return(ymodel)

yseries = []
slope = 7.6
intercept = 0.0

for i in range(0,len(time)):
    yseries.append(ymodel(time[i],slope,intercept))
# Create a markers only line chart
mydata = plt.figure(figsize = (10,5)) # build a square drawing canvass from figure clas
plt.plot(time, speed, c='red', marker='^',linewidth=0.5) # basic line plot
plt.plot(time, yseries, c='blue')
plt.show()
```



Example- Find a better fit

Using trial and error try to improve the 'fit' of the model, by adjusting values of $$m~\text{and}~b$$.

In [20]:
```
# Code and run your solution here:
yseries = []
slope = 7.6
intercept = -8.0

for i in range(0,len(time)):
    yseries.append(ymodel(time[i],slope,intercept))
# Create a markers only line chart
mydata = plt.figure(figsize = (10,5)) # build a square drawing canvass from figure clas
plt.plot(time, speed, c='red', marker='^',linewidth=0) # basic scatter plot
plt.plot(time, yseries, c='blue')
plt.show()
```



Scatter Plots

A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. If the points are coded (color/shape/size), one additional variable can be displayed. The data are displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.

A scatter plot can be used either when one continuous variable that is under the control of the experimenter and the other depends on it or when both continuous variables are independent. If a parameter exists that is systematically incremented and/or decremented by the other, it is called the control parameter or independent variable and is customarily plotted along the horizontal axis. The measured or dependent variable is customarily plotted along the vertical axis. If no dependent variable exists, either type of variable can be plotted on either axis and a scatter plot will illustrate only the degree of correlation (not causation) between two variables.

A scatter plot can suggest various kinds of correlations between variables with a certain confidence interval. For example, weight and height, weight would be on y axis and height would be on the x axis. Correlations may be positive (rising), negative (falling), or null (uncorrelated). If the pattern of dots slopes from lower left to upper right, it indicates a positive correlation between the variables being studied. If the pattern of dots slopes from upper left to lower right, it indicates a negative correlation.

A line of best fit (alternatively called 'trendline') can be drawn in order to study the relationship between the variables. An equation for the correlation between the variables can be determined by established best-fit procedures. For a linear correlation, the best-fit procedure is known as linear regression and is guaranteed to generate a correct solution in a finite time. No universal best-fit procedure is guaranteed to generate a solution for arbitrary relationships. A scatter plot is also very

useful when we wish to see how two comparable data sets agree and to show nonlinear relationships between variables.

Furthermore, if the data are represented by a mixture model of simple relationships, these relationships will be visually evident as superimposed patterns.

Scatter charts can be built in the form of bubble, marker, or/and line charts.

Much of the above is verbatim/adapted from: https://en.wikipedia.org/wiki/Scatter_plot

---

Example- Examine the dataset with heights of fathers, mothers and sons

In [21]:
```
########## CODE TO AUTOMATICALLY DOWNLOAD THE DATABASE #################
import requests # import needed modules to interact with the internet
# make the connection to the remote file (actually its implementing "bash curl -O http:/
remote_url = 'http://54.243.252.9/engr-1330-webroot/8-Labs/Lab15/galton_subset.csv' # a
response = requests.get(remote_url) # Gets the file contents puts into an object
output = open('galton_subset.csv', 'wb') # Prepare a destination, local
output.write(response.content) # write contents of object to named local file
output.close() # close the connection
```

In [22]:
```
import pandas as pd
```

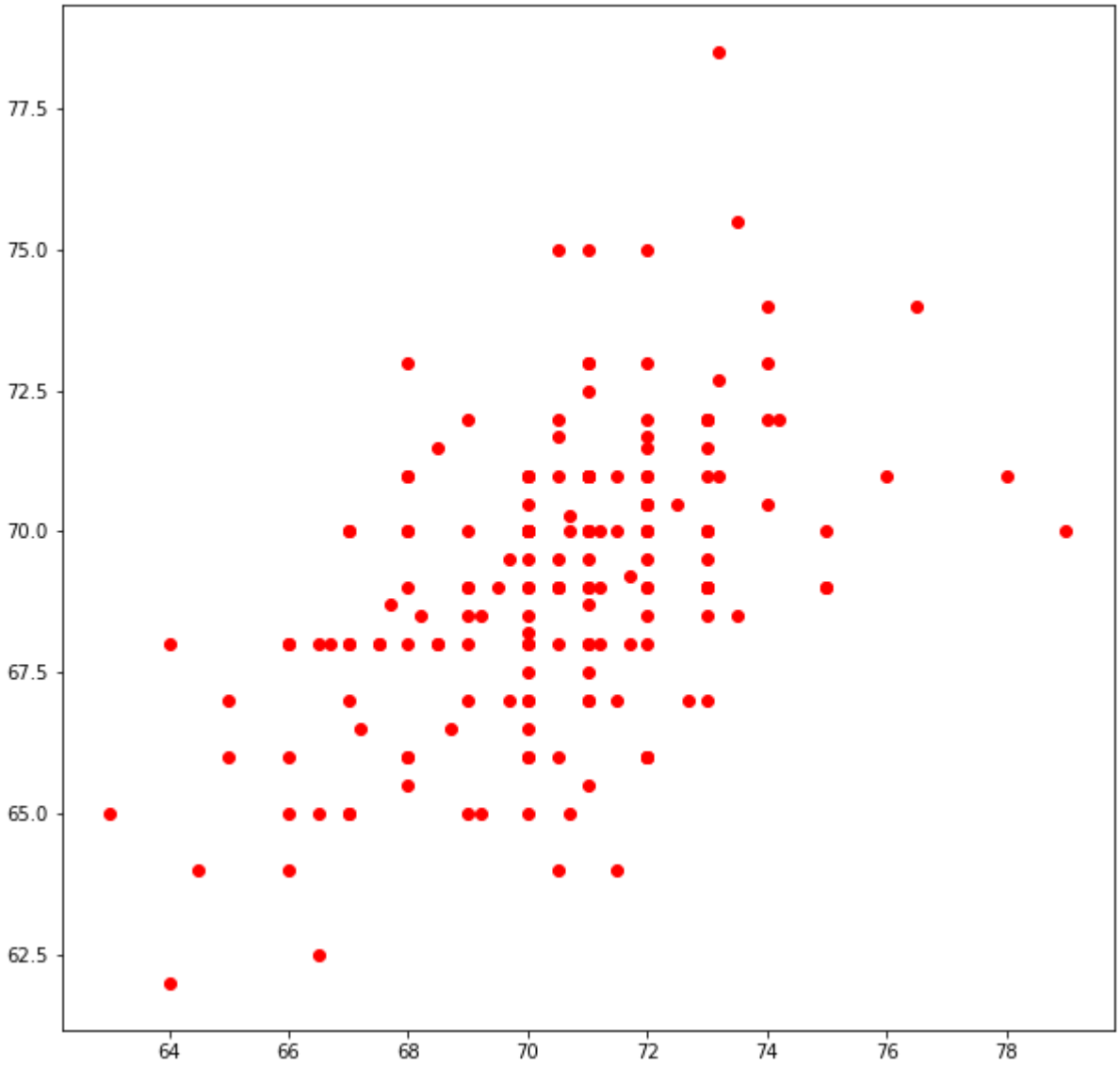
In [23]:
```
df = pd.read_csv('galton_subset.csv')
df['child']= df['son'] ; df.drop('son', axis=1, inplace = True) # rename son to child -
df.head()
```

Out[23]:

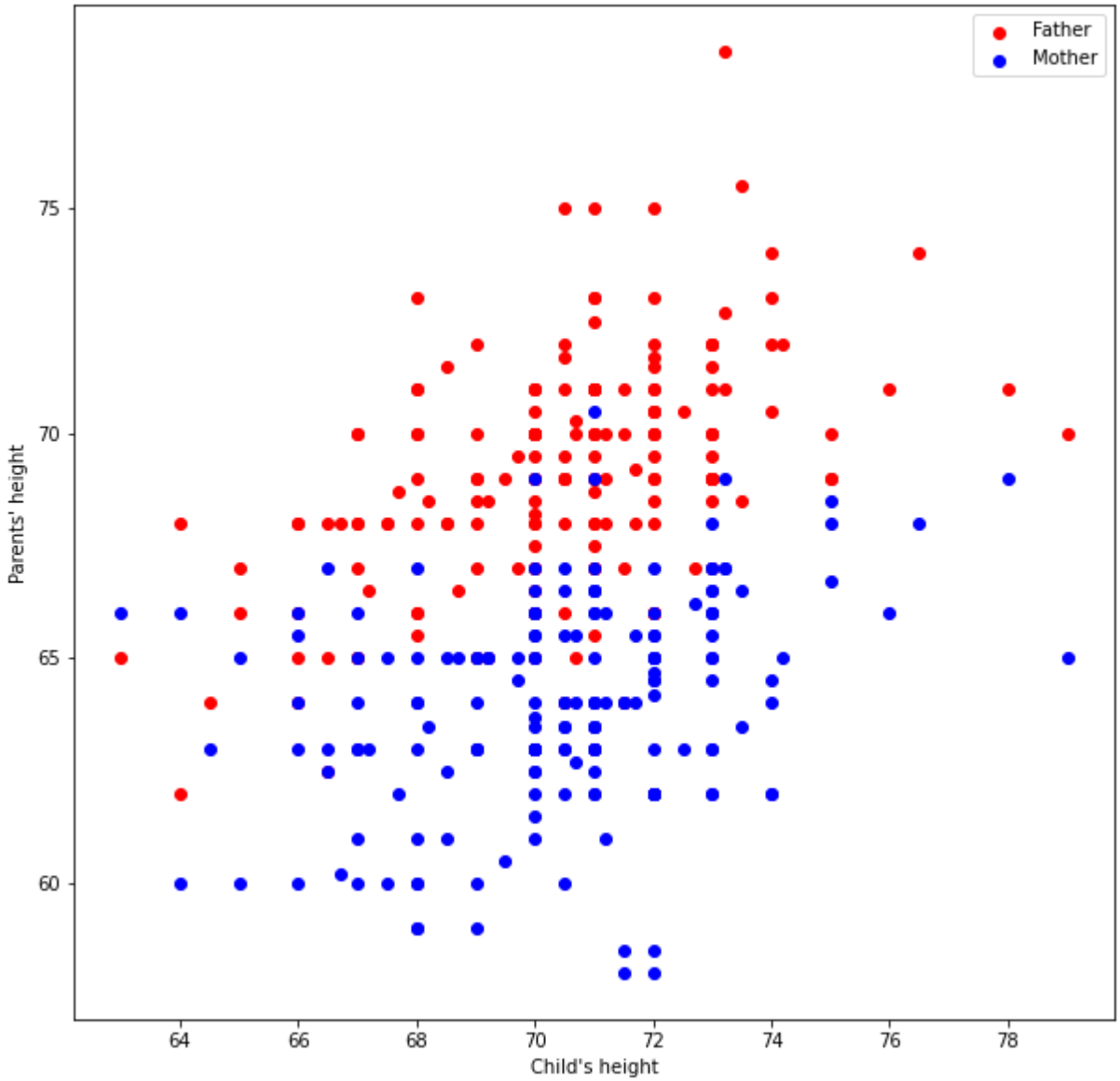
	father	mother	child
0	78.5	67.0	73.2
1	75.5	66.5	73.5
2	75.0	64.0	71.0
3	75.0	64.0	70.5
4	75.0	58.5	72.0

In [24]:
```
# build some lists
dad = df['father'] ; mom = df['mother'] ; son = df['child']
```

In [25]:
```
myfamily = plt.figure(figsize = (10, 10)) # build a square drawing canvass from figure
plt.scatter(son, dad, c='red') # basic scatter plot
plt.show()
```

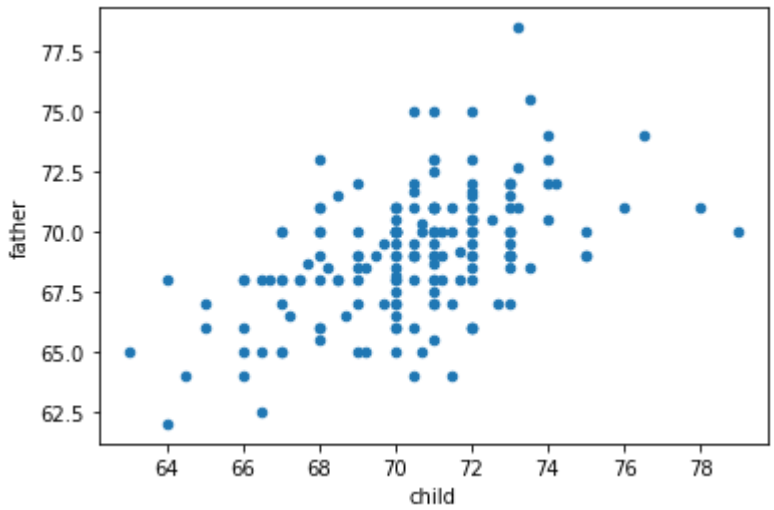


In [27]:
```
# Looks lousy, needs some labels
myfamily = plt.figure(figsize = (10, 10)) # build a square drawing canvass from figure
plt.scatter(son, dad, c='red' , label='Father') # one plot series
plt.scatter(son, mom, c='blue', label='Mother') # two plot series
plt.xlabel("Child's height")
plt.ylabel("Parents' height")
plt.legend()
plt.show() # render the two plots
```



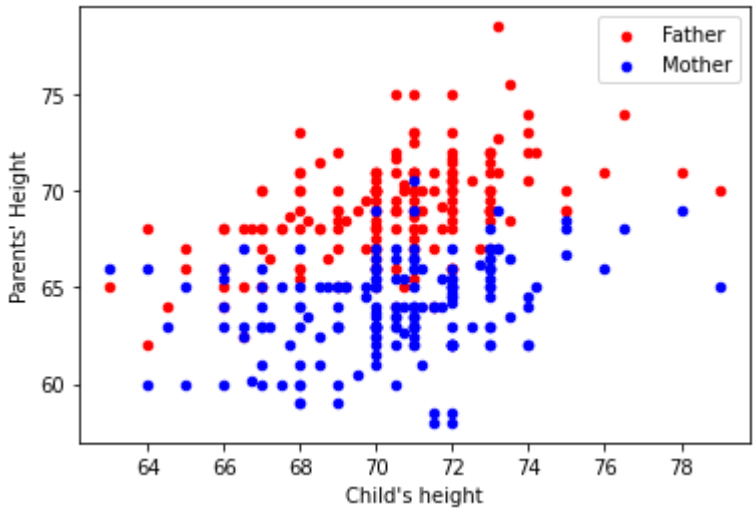
In [28]:
```
# Repeat in pandas - The dataframe already is built
df.plot.scatter(x="child", y="father")
```

Out[28]: <AxesSubplot:xlabel='child', ylabel='father'>



In [29]:
```
ax = df.plot.scatter(x="child", y="father", c="red", label='Father')
df.plot.scatter(x="child", y="mother", c="blue", label='Mother', ax=ax)

ax.set_xlabel("Child's height")
ax.set_ylabel("Parents' Height")
```

Out[29]:
```
Text(0, 0.5, "Parents' Height")
```



Histograms

Quoting from https://en.wikipedia.org/wiki/Histogram

"A histogram is an approximate representation of the distribution of numerical data. It was first introduced by Karl Pearson.[1] To construct a histogram, the first step is to "bin" (or "bucket") the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but not required to be) of equal size.

If the bins are of equal size, a rectangle is erected over the bin with height proportional to the frequency—the number of cases in each bin. A histogram may also be normalized to display "relative" frequencies. It then shows the proportion of cases that fall into each of several categories, with the sum of the heights equaling 1.

However, bins need not be of equal width; in that case, the erected rectangle is defined to have its area proportional to the frequency of cases in the bin. The vertical axis is then not the frequency but frequency density—the number of cases per unit of the variable on the horizontal axis. Examples of variable bin width are displayed on Census bureau data below.

As the adjacent bins leave no gaps, the rectangles of a histogram touch each other to indicate that the original variable is continuous.

Histograms give a rough sense of the density of the underlying distribution of the data, and often for density estimation: estimating the probability density function of the underlying variable. The

total area of a histogram used for probability density is always normalized to 1. If the length of the intervals on the x-axis are all 1, then a histogram is identical to a relative frequency plot.

A histogram can be thought of as a simplistic kernel density estimation, which uses a kernel to smooth frequencies over the bins. This yields a smoother probability density function, which will in general more accurately reflect distribution of the underlying variable. The density estimate could be plotted as an alternative to the histogram, and is usually drawn as a curve rather than a set of boxes. Histograms are nevertheless preferred in applications, when their statistical properties need to be modeled. The correlated variation of a kernel density estimate is very difficult to describe mathematically, while it is simple for a histogram where each bin varies independently.

An alternative to kernel density estimation is the average shifted histogram, which is fast to compute and gives a smooth curve estimate of the density without using kernels.

The histogram is one of the seven basic tools of quality control.

Histograms are sometimes confused with bar charts. A histogram is used for continuous data, where the bins represent ranges of data, while a bar chart is a plot of categorical variables. Some authors recommend that bar charts have gaps between the rectangles to clarify the distinction."

Example- Explore the "top_movies" dataset and draw histograms for Gross and Year.

In [30]:
```
######### CODE TO AUTOMATICALLY DOWNLOAD THE DATABASE #################
import requests # import needed modules to interact with the internet
# make the connection to the remote file (actually its implementing "bash curl -O http:/
remote_url = 'http://54.243.252.9/engr-1330-webroot/4-Databases/top_movies.csv' # a csv
response = requests.get(remote_url) # Gets the file contents puts into an object
output = open('top_movies.csv', 'wb') # Prepare a destination, local
output.write(response.content) # write contents of object to named local file
output.close() # close the connection
```

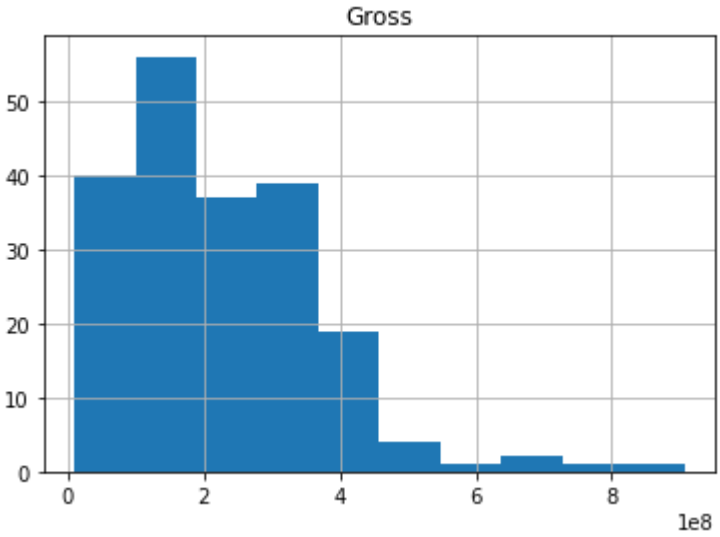
In [31]:
```
import pandas as pd

df = pd.read_csv('top_movies.csv')
df.head()
```

Out[31]:

	Title	Studio	Gross	Gross (Adjusted)	Year
0	Star Wars: The Force Awakens	Buena Vista (Disney)	906723418	906723400	2015
1	Avatar	Fox	760507625	846120800	2009
2	Titanic	Paramount	658672302	1178627900	1997
3	Jurassic World	Universal	652270625	687728000	2015
4	Marvel's The Avengers	Buena Vista (Disney)	623357910	668866600	2012

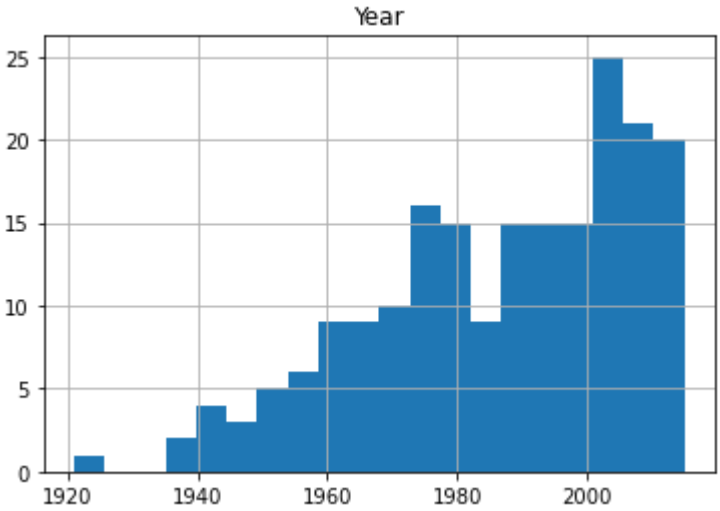
In [32]:
```
df[["Gross"]].hist()
```

Out[32]:
```
array([[<AxesSubplot:title={'center':'Gross'}>]], dtype=object)
```



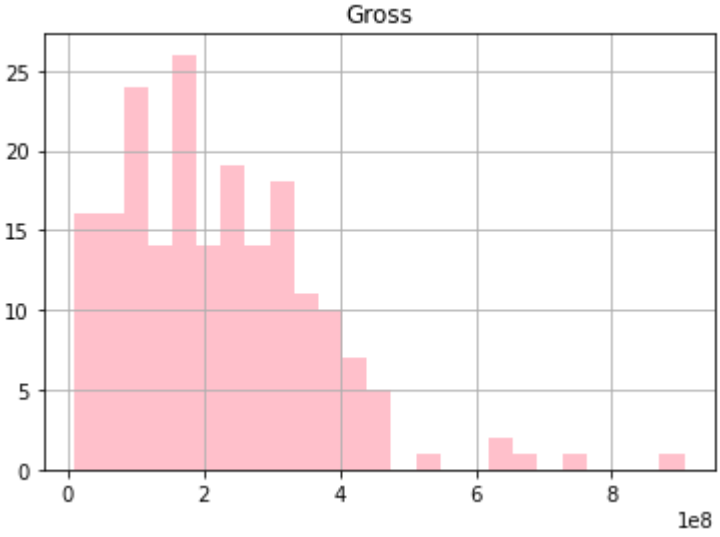
In [35]:
```
df[["Year"]].hist(bins=20)
```

Out[35]:
```
array([[<AxesSubplot:title={'center':'Year'}>]], dtype=object)
```

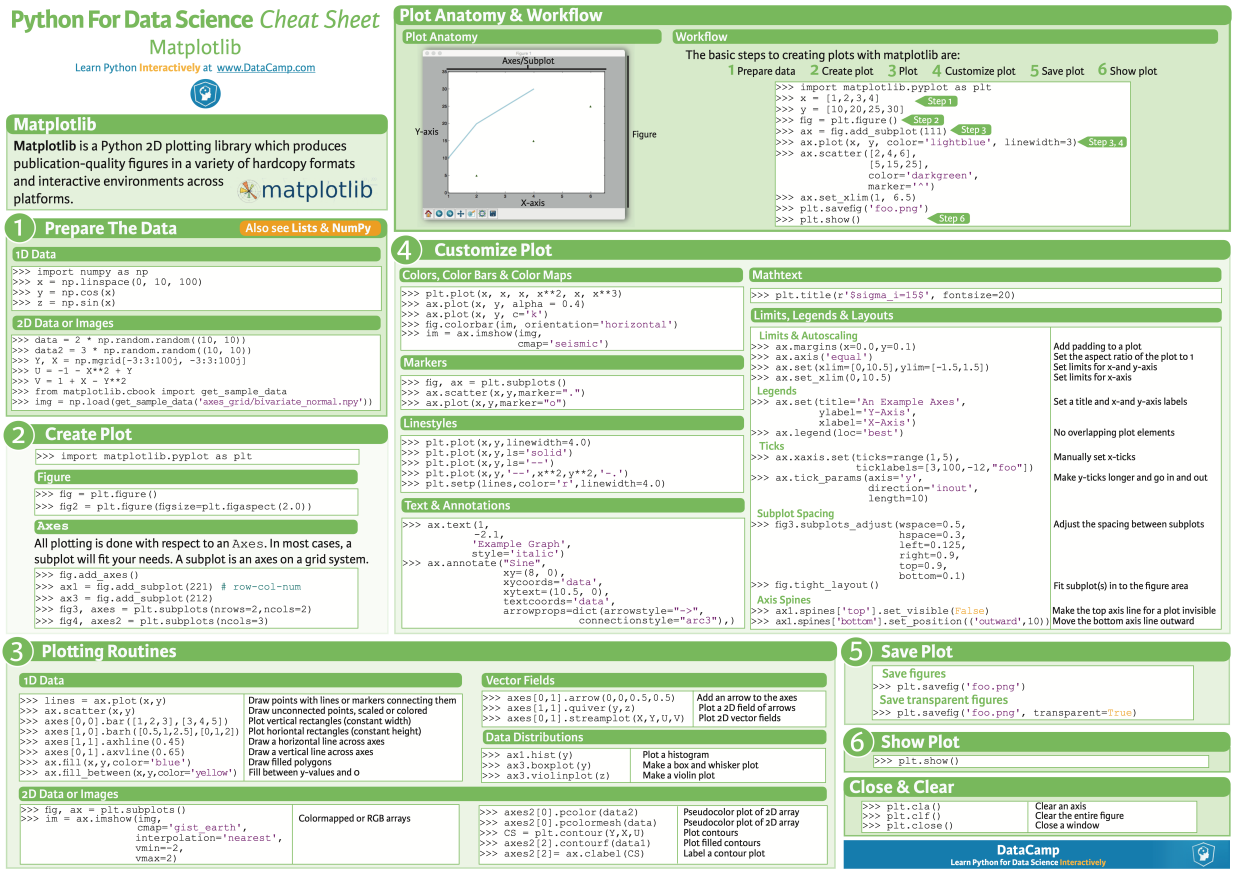


In [37]:
```
df[["Gross"]].hist(bins=25, color= 'pink')
```

Out[37]:
```
array([[<AxesSubplot:title={'center':'Gross'}>]], dtype=object)
```



This is a Matplotlib Cheat Sheet



Here are some of the resources used for creating this notebook:

- **"Discrete distribution as horizontal bar chart"** available at *https://matplotlib.org/stable/gallery/lines_bars_and_markers/horizontal_barchart_distribution.html
- **"Bar Plot in Matplotlib"** available at *https://www.geeksforgeeks.org/bar-plot-in-matplotlib/

Here are some great reads on this topic:

- **"Python | Introduction to Matplotlib"** available at *https://www.geeksforgeeks.org/python-introduction-matplotlib/
- **"Visualization with Matplotlib"** available at *https://jakevdp.github.io/PythonDataScienceHandbook/04.00-introduction-to-matplotlib.html
- **"Introduction to Matplotlib — Data Visualization in Python"** by **Ehi Aigiomawu** available at *https://heartbeat.fritz.ai/introduction-to-matplotlib-data-visualization-in-python-d9143287ae39
- **"Python Plotting With Matplotlib (Guide)"** by **Brad Solomon** available at *https://realpython.com/python-matplotlib-guide/

Here are some great videos on these topics:

- **"Matplotlib Tutorial (Part 1): Creating and Customizing Our First Plots"** by **Corey Schafer** available at *https://www.youtube.com/watch?v=UO98lJQ3QGI
- **"Intro to Data Analysis / Visualization with Python, Matplotlib and Pandas | Matplotlib Tutorial"** by **CS Dojo** available at *https://www.youtube.com/watch?v=a9UrKTVEeZA
- **"Intro to Data Visualization in Python with Matplotlib! (line graph, bar chart, title, labels, size)"** by **Keith Galli** available at *https://www.youtube.com/watch?v=DAQNHzOcO5A

Exercise: Bins, Bins, Bins!

Selecting the number of bins is an important decision when working with histograms. Are there any rules or recommendations for choosing the number or width of bins? What happens if we use too many or too few bins?

* Make sure to cite any resources that you may use.

it is ideal to have an appropriate amount of bins that is proportional to the amount of data you have in order to properly display it as clearly as possible.

In [ ]: