Data Viz 1

Content

- Intro to Matplotlib & Seaborn
- Tencent Use Case
- Anatomy of Matplotlib
- Components of a Matplotlib plot
- Univariate Data Visualization
  - Categorical
    - Bar chart
    - Countplot
    - Pie chart
  - Continous
    - Histogram
    - KDE plot
    - Box and Whiskers plot

Plots Presentation:

https://docs.google.com/presentation/d/1DkLTjTe6YmGbDHtr4v9Jso553DlCuP3cfSnwvUN1ı
usp=sharing

Importing Matplotlib & Seaborn

In case of `matplotlib`,

- We don't need to import the entire library but just its sub-module `pyplot`.
- We'll use the alias name `plt`.

What is `pyplot`?

- `pyplot` is a **sub-module for visualization** in `matplotlib`.
- Think of it as a **high-level API** which **makes plotting an easy task**.
- Data Scientists stick to using `pyplot` only unless they want to create **something totally new.

For `seaborn`,

- We will be importing the whole seaborn library as alias `sns`.

What is seaborn?

Seaborn is another visualization library which uses Matplotlib in the backend for plotting.

What is the major difference then between both matplotlib and seaborn?

- `Seaborn` is built on the top of `Pandas` and `Matplotlib`.
- Seaborn uses **fascinating themes** and **reduces number of code lines** by doing a lot of work in the backend.
- While matplotlib is used to **plot basic plots and add more functionlaity** on top of that

As we proceed through the lecture, we will see the difference between both the libraries.

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Before we dive into learning these libraries, lets answer some general questions.

Why do even we need to visualize data?

- **Exploratory** - I can't see certain patterns just by crunching numbers (avg, rates, %ages).
- **Explanatory** - I have the numbers crunches and insights ready, but I'd like a visual art for storytelling.

What do we already know?

Data

- Rows: Samples, Data-points, Records
- Columns: Features, Variables

At the fundamental level, we have two types of data:

- Numerical/Continous
- Categorical

Categorical can be further divided into:

- **Ordinal:** Categorical data with an order (e.g. low, medium, high)
- **Non-ordinal/nominal:** Categorical data without any order (e.g. Male/Female)

Video Games Analysis

You are a Data Scientist at "Tencent Games".

You need to analyze what kind of games the company should create in order to perform better in the market.

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
data = pd.read_csv('final.csv')
data.head()
```

Out[3]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
0	2061	1942	NES	1985.0	Shooter	Capcom	4.569217	3.033887	3.439
1	9137	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442	3.033
2	14279	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339	1.493
3	8359	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	3.228
4	7109	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	1.440

Notice that,

- columns like `Platform`, `Genre` are Categorical
- columns like `NA_Sales`, `Global_Sales`, `Rank` are Continuous

Furthermore,

- `Platform` is of nominal type (no proper order between the categories)
- `Year` is of ordinal type (an order exists between the categories)

Introduction to Matplotlib

How to create a basic plot using `plt`?

Let's say we want to draw a curve passing through 3 points:

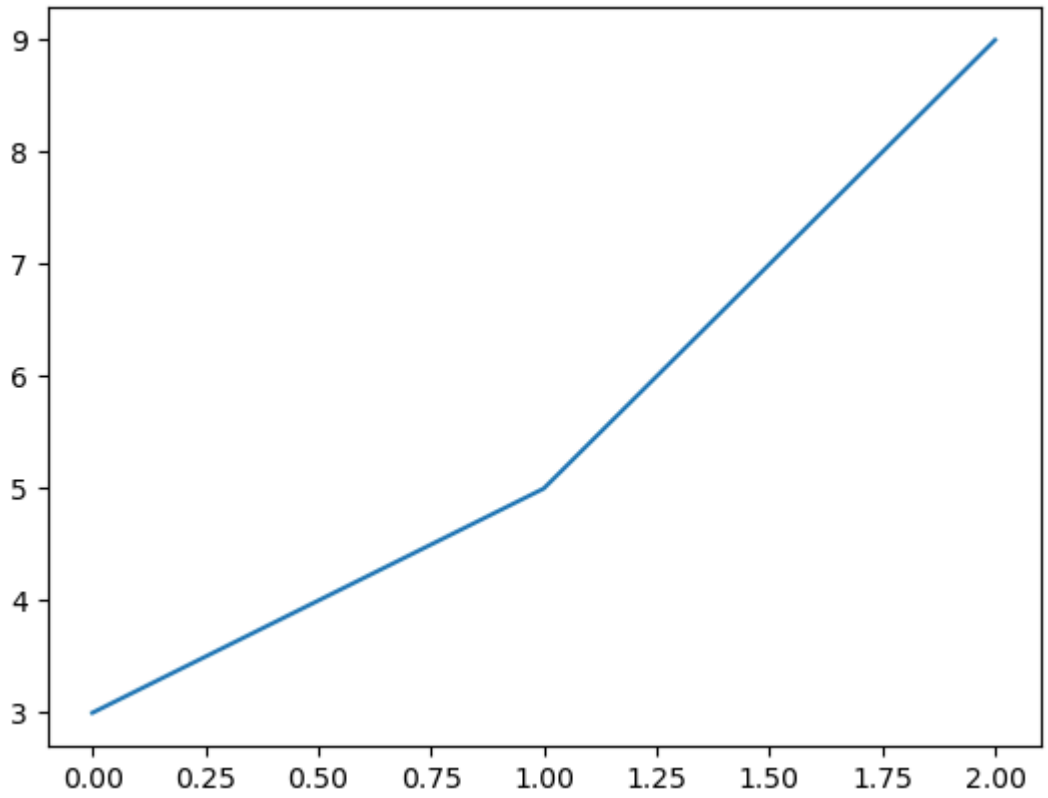
- (0, 3)
- (1, 5)
- (2, 9)

How can we draw a curve using `matplotlib`?

- by using the `plt.plot()` function

In [4]:
```
x_val = [0, 1, 2]
y_val = [3, 5, 9]
plt.plot(x_val, y_val)
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x159767d10>]
```

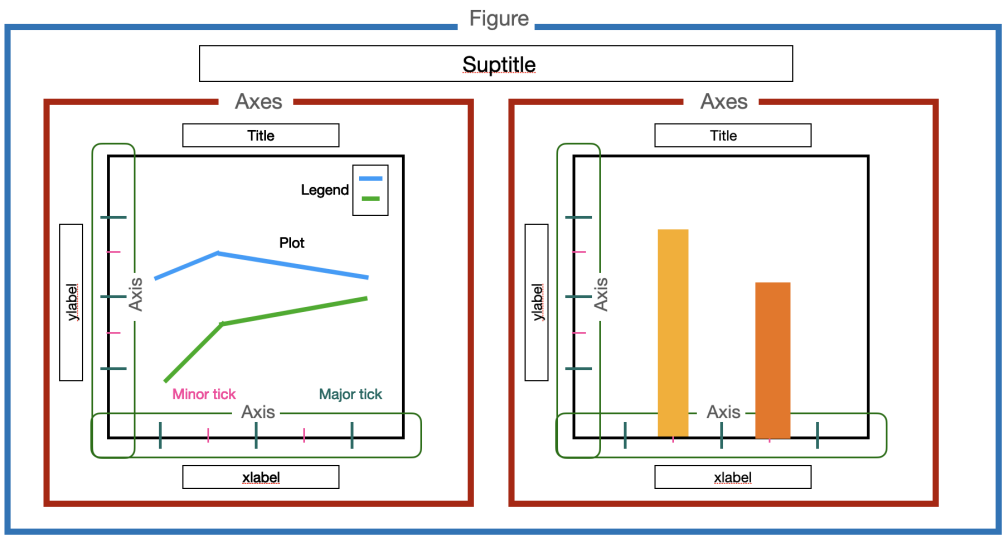


What can we observe from this plot?

- `plt.plot()` automatically decided the scale of the plot.
- It also prints the **type of object** i.e. `matplotlib.lines.Line2D`

While this command decided a lot of things for you, you can customise each of these by understanding the **components of a matplotlib plot**.

Anatomy of Matplotlib



Woah! There is a lot of information in this image. Let's understand them one at a time.

Components of a Matplotlib plot

- Figure: The **overall window** or page that everything is drawn on.
  - You can create multiple independent Figures in Jupyter.
  - If you run the code in terminal, separate windows will pop-up.
- Axes: You can add multiple **Axes** to the Figure, which represents a plot.
- **Axis**: Simply the `x-axis` and `y-axis`
- **Axes**: It is the **area** on which the **data is plotted** with functions such as `plot()`.
  - **x-label**: Name of x-axis
  - **y-label**: Name of y-axis
- **Major ticks**:
  - Subdivides the axis into major units.
  - They appear by default during plotting.
- **Minor ticks**:
  - Subdivides the major tick units.
  - They are by default hidden and can be toggled on.
- **Title**: Title of each plot **(Axes)**
- **Subtitle**: The common title of all the plots.
- **Legend**:
  - Describes the elements in the plot.
  - Blue and Green curves in this case.

These are the major components of a matplotlib plot.

How to choose the right plot?

Firstly, it depends on the what is your question of interest.

When the question is clear

- How many variables are involved?
- Whether the variable(s) are numerical or categorical?

How many variables are involved?

- 1 Variable $\rightarrow$ Univariate Analysis
- 2 Variables $\rightarrow$ Bivariate Analysis
- 3+ Variables $\rightarrow$ Multivariate Analysis

What are the possible cases?

Univariate

- Numerical

- Categorical

Bivariate

- Numerical-Numerical
- Numerical-Categorical
- Categorical-Categorical

Multivariate

Let's start with these and then we can generalize.

- Numerical-Numerical-Categorical
- Categorical-Categorical-Numerical
- Categorical-Categorical-Categorical
- Numerical-Numerical-Numerical

Univariate Data Visualization - Categorical Data

What kind of questions we may want to ask for a categorical variable?

- What is the Distribution/Frequency of the data across different categories?
- What proportion does a particular category constitutes?

...and so on

Let's take the categorical column "Genre".

How can we find the top-5 genres?

In [5]:
```
cat_counts = data['Genre'].value_counts()
cat_counts
```

Out[5]:
```
Action          3316
Sports          2400
Misc            1739
Role-Playing    1488
Shooter         1310
Adventure       1286
Racing          1249
Platform         886
Simulation       867
Fighting         848
Strategy         681
Puzzle           582
Name: Genre, dtype: int64
```

What kind of plot can we use to visualize this information?

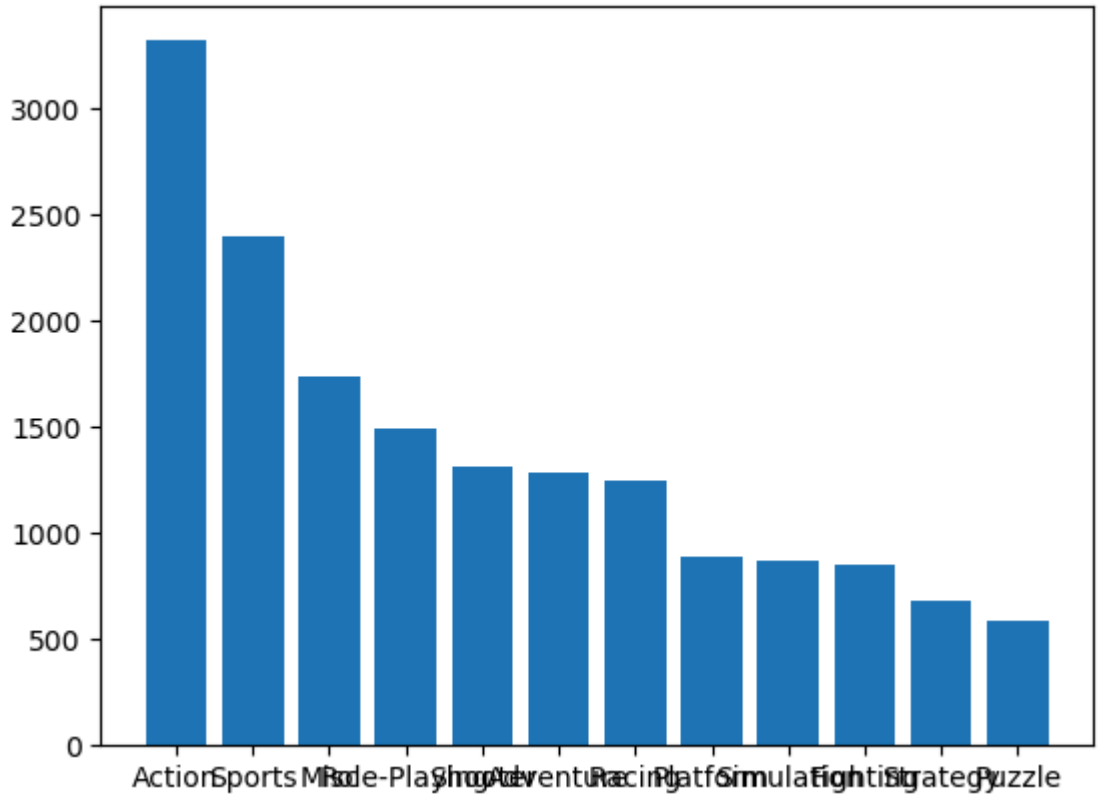
- We can perhaps plot categories on X-axis and their corresponding frequencies on Y-axis.
- This is called a `Bar Chart` or a `Count Plot`.

Bar Chart

- We can draw a bar plot using `plt.bar()`.
- The data is binned here into categories.

In [6]:
```
x_bar=cat_counts.index
y_bar=cat_counts
plt.bar(x_bar,y_bar)
```

Out[6]:
```
<BarContainer object of 12 artists>
```



The names seem to be overlapping.

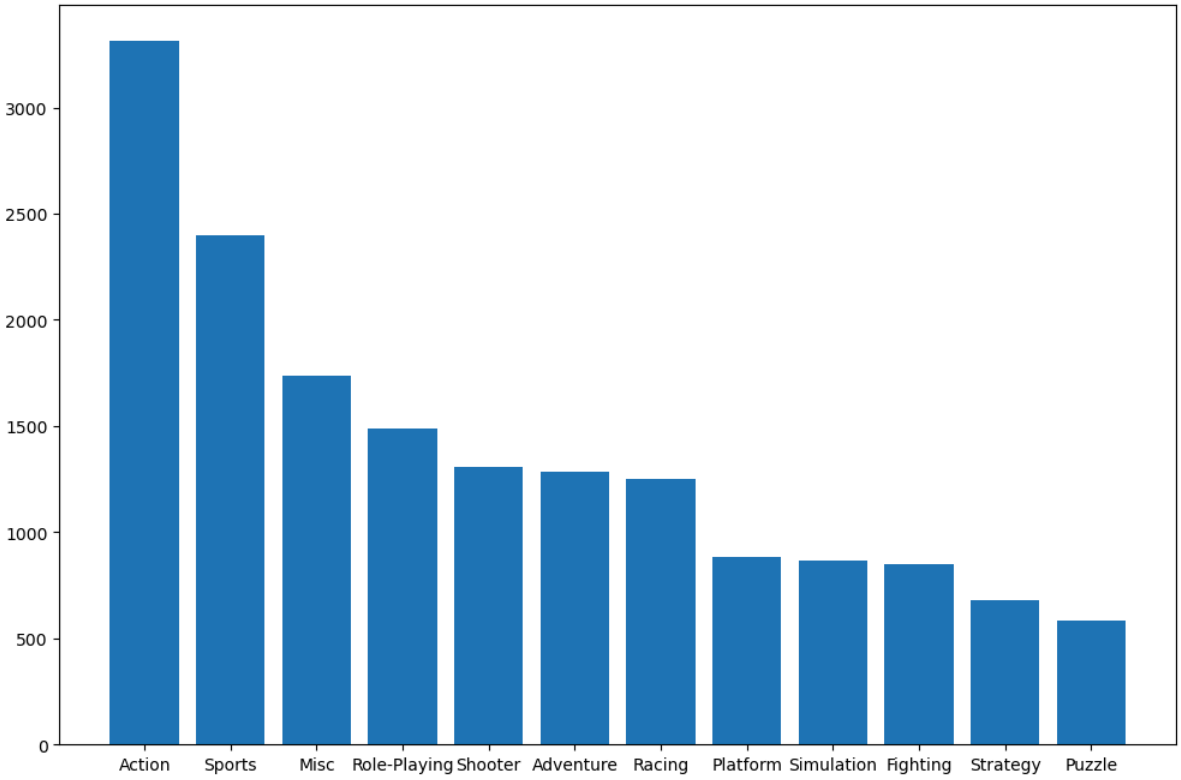
How can we handle overlapping labels?

1. Decrease the font size (not preferred)
2. Increase the figure size
3. Rotate the labels

How can we change the plot size?

In [8]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
```

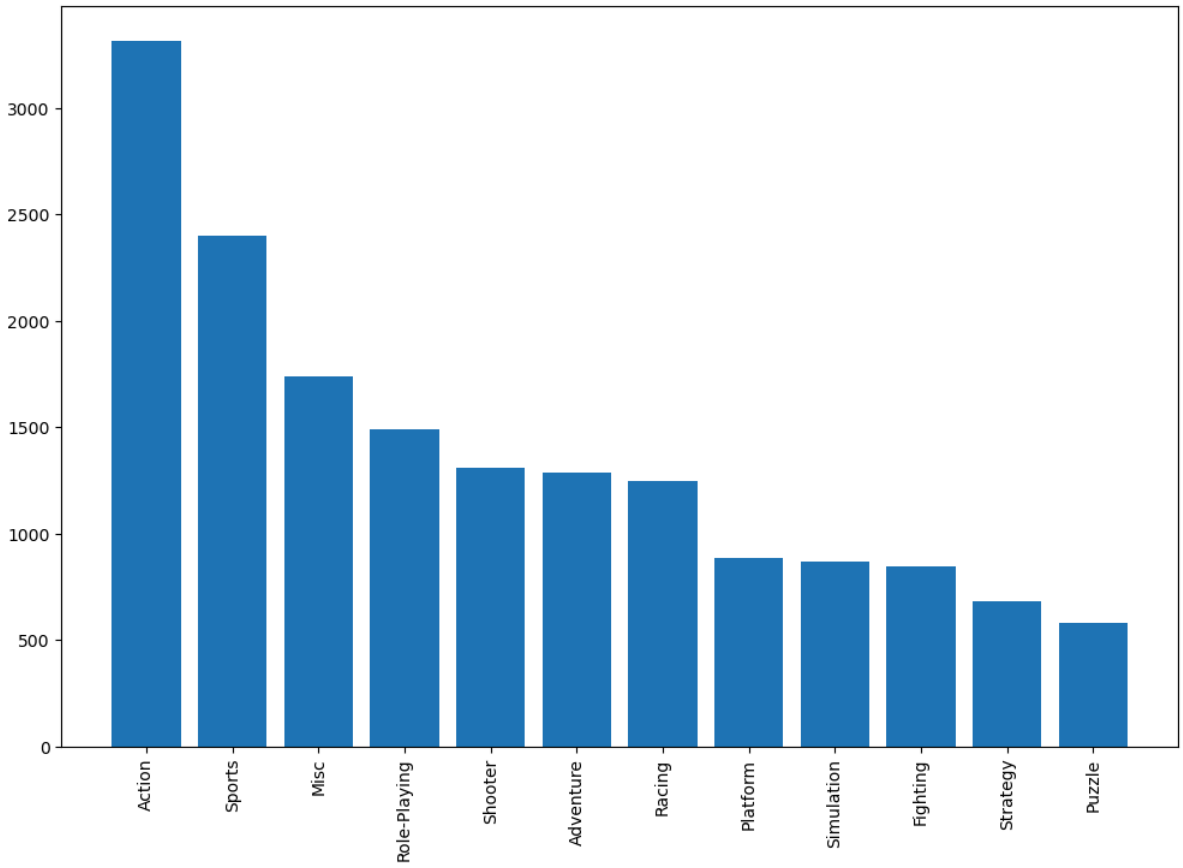
Out[8]:
```
<BarContainer object of 12 artists>
```



How can we rotate the tick labels, also increase the fontsize of the same?

In [9]:
```
plt.figure(figsize=(12,8))
plt.bar(x_bar,y_bar)
plt.xticks(rotation=90, fontsize=10)
```

Out[9]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```

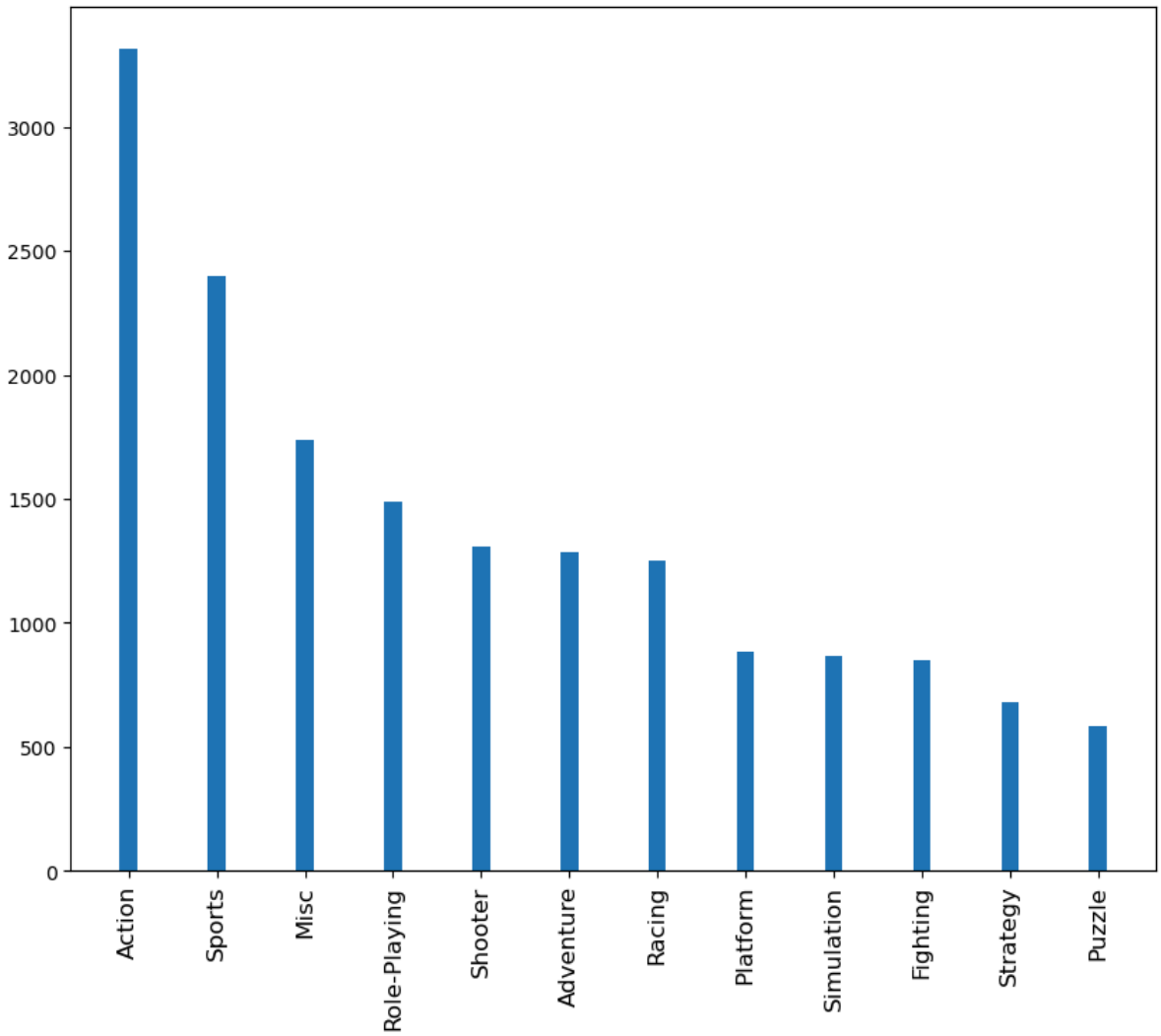


If you notice, the width of each bar is **1**.

Can we also change the width of these bars?

In [10]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2)
plt.xticks(rotation=90, fontsize=12)
```

Out[10]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```

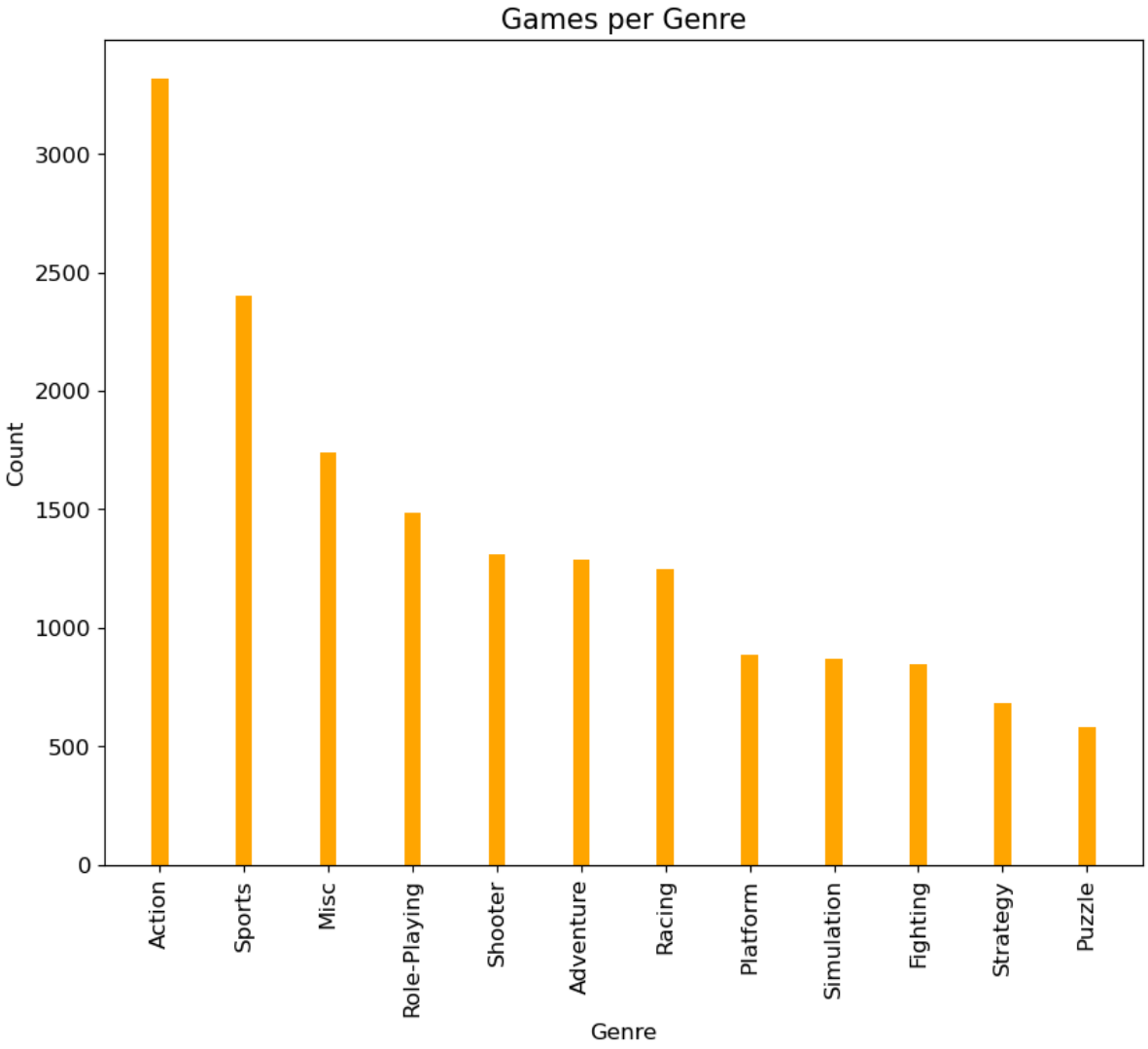


What about any adding some styling to the bars?

- We can **change the colour of bars**
- We can add a **title to the axes**
- We can also **add x and y labels**

In [11]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
```

Out[11]:
```
(array([   0.,  500., 1000., 1500., 2000., 2500., 3000., 3500.]),
 [Text(0, 0.0, '0'),
  Text(0, 500.0, '500'),
  Text(0, 1000.0, '1000'),
  Text(0, 1500.0, '1500'),
  Text(0, 2000.0, '2000'),
  Text(0, 2500.0, '2500'),
  Text(0, 3000.0, '3000'),
  Text(0, 3500.0, '3500')])
```



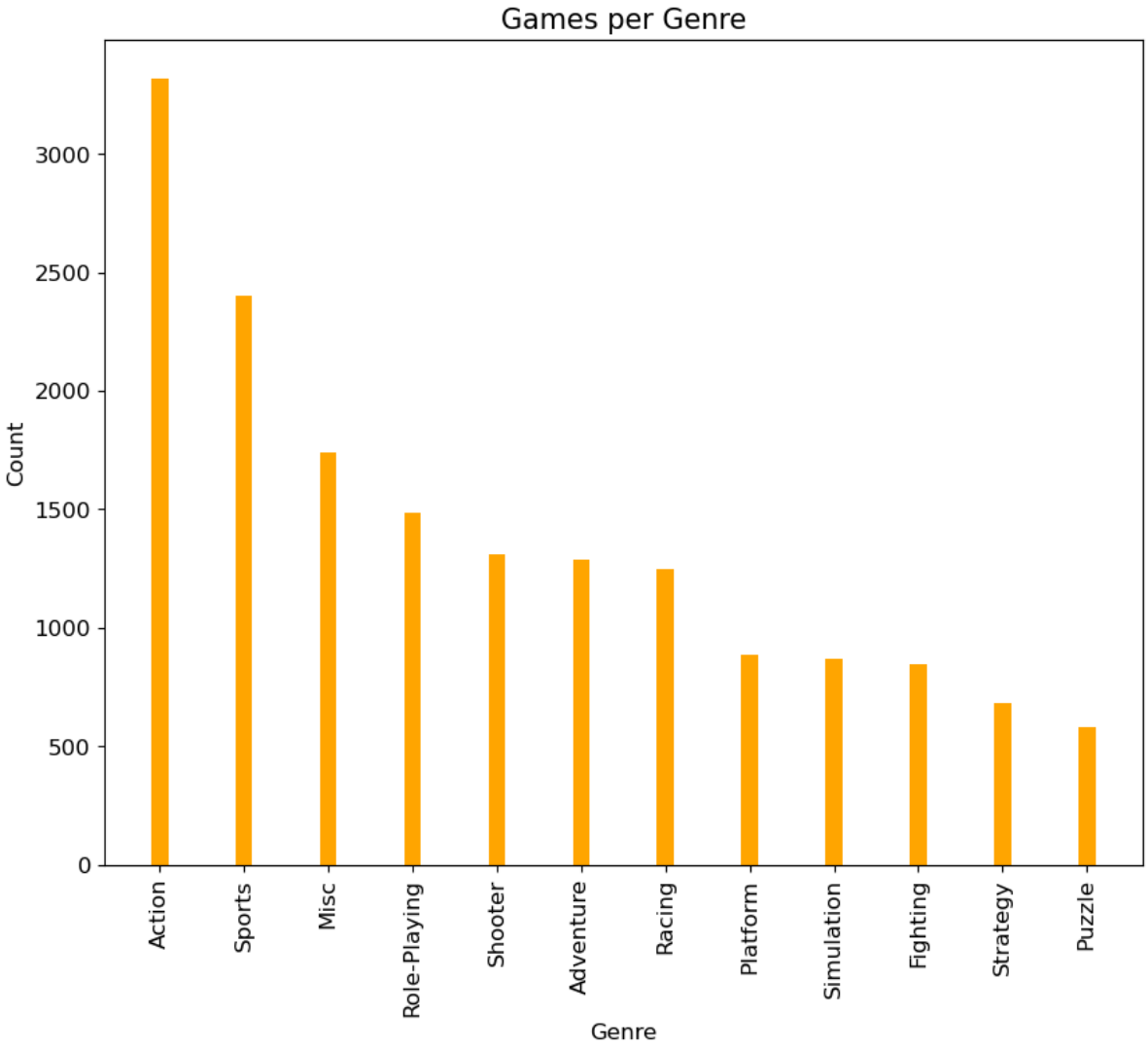
If you notice, there's some text which is always printed before the plots.

This contains the data information of the plot.

How can we remove the text printed before the plot?

- by using `plt.show()` at the end

In [12]:
```
plt.figure(figsize=(10,8))
plt.bar(x_bar,y_bar,width=0.2,color='orange')
plt.title('Games per Genre',fontsize=15)
plt.xlabel('Genre',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation = 90, fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



How can we draw a bar chart in Seaborn?

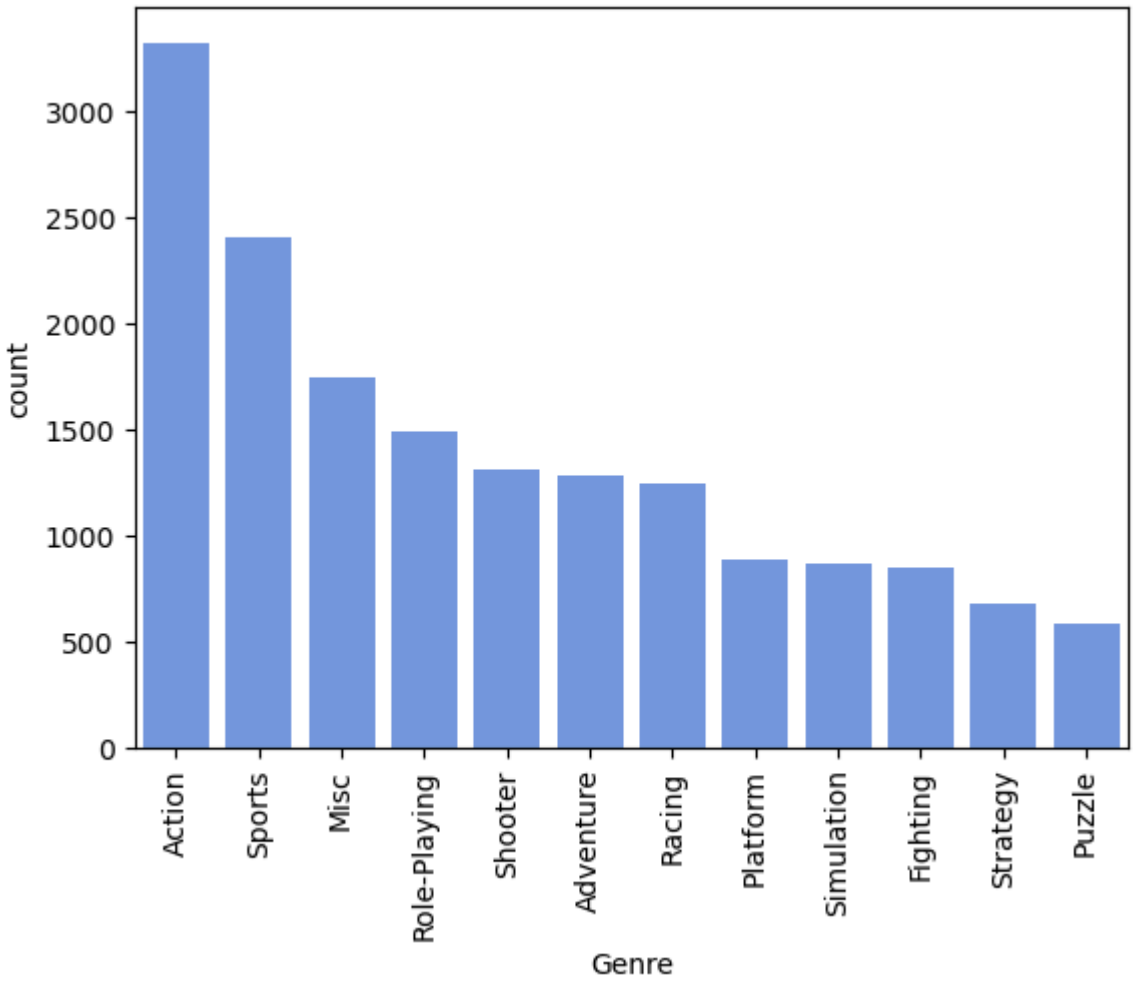
- In Seaborn, the same plot is called a **countplot**.
- It automatically does the counting of frequencies for you.

Why not just call it a barplot?

There is another function in Seaborn called a **barplot** which has some other purpose. We'll discuss this later.

In [13]:
```
sns.countplot(x='Genre', data=data, order=data['Genre'].value_counts().index
plt.xticks(rotation=90)
```

Out[13]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'Action'),
  Text(1, 0, 'Sports'),
  Text(2, 0, 'Misc'),
  Text(3, 0, 'Role-Playing'),
  Text(4, 0, 'Shooter'),
  Text(5, 0, 'Adventure'),
  Text(6, 0, 'Racing'),
  Text(7, 0, 'Platform'),
  Text(8, 0, 'Simulation'),
  Text(9, 0, 'Fighting'),
  Text(10, 0, 'Strategy'),
  Text(11, 0, 'Puzzle')])
```



The top 5 genres are action, sports, misc, role-playing, and shooter.

Pie Chart

What if instead of actual frequencues, we want see the proportion of the categories?

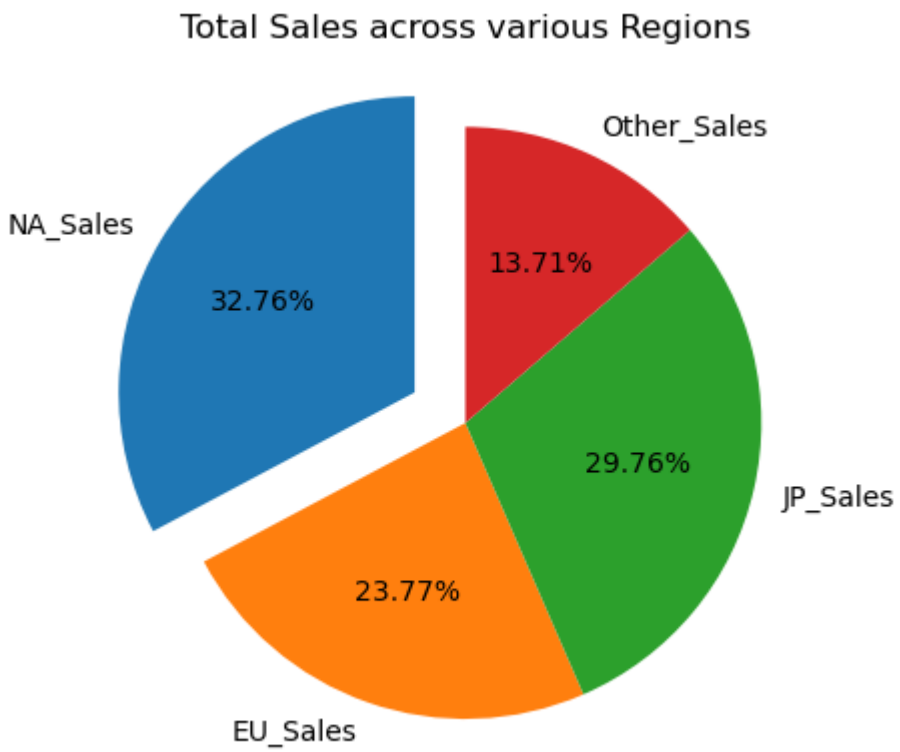
Say, we want to compare the distrubution/proportion of sales across different regions?

Which plot can we use for this? A pie-chart!

In [15]:
```
sales_data = data[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]
region_sales = sales_data.T.sum(axis='columns')

plt.pie(region_sales,
        labels=region_sales.index,
        startangle=90,
        explode=(0.2,0,0,0),
        autopct = '%.2f%%') # label the wedges with their numeric value

plt.title('Total Sales across various Regions')
plt.show()
```



In [17]:
```
region_sales = sales_data.T.sum(axis='columns')
region_sales
```

Out[17]:
```
NA_Sales       45831.525845
EU_Sales       33251.970702
JP_Sales       41624.625635
Other_Sales    19180.256828
dtype: float64
```

Univariate Data Visualisation - Numerical Data

What kind of questions we may have regarding a numerical variable?

- How is the data distributed?
- Is the data skewed? Are there any outliers?
- How much percentage of data is below/above a certain number?
- Statistics like - Min, Max, Mean, Median, etc.

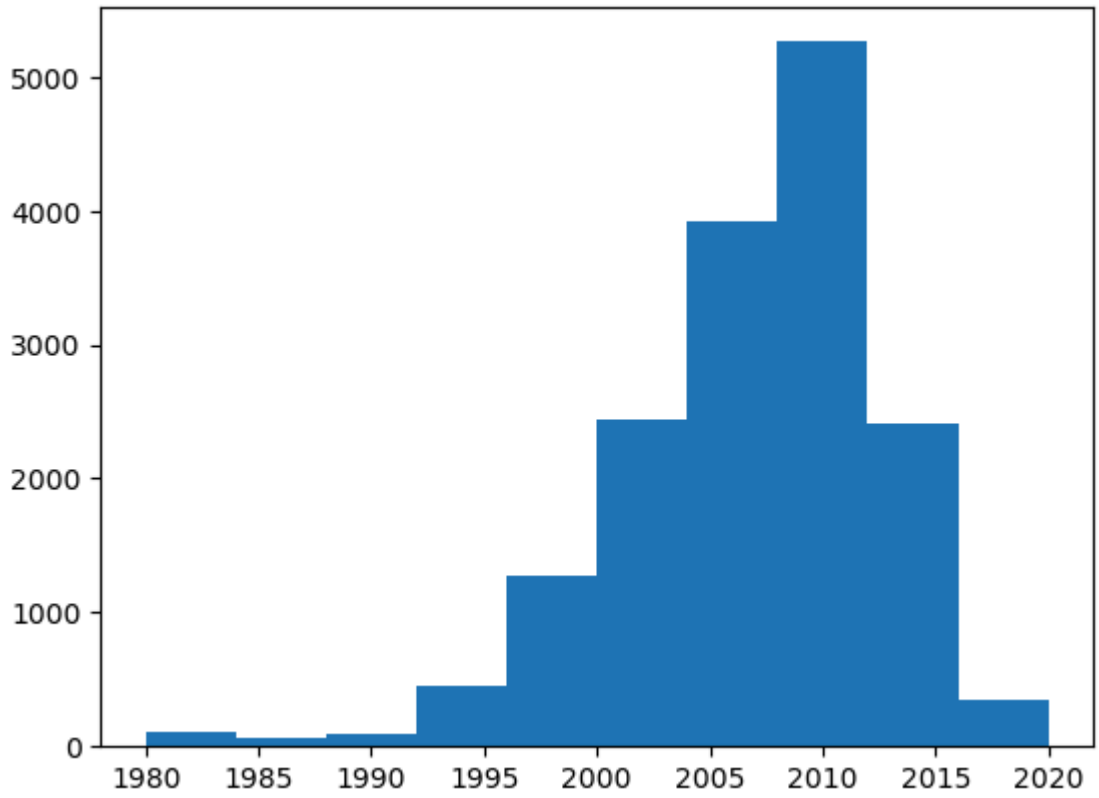
Now say you want to find the distribution of games released every year.

Unlike barplot, to see the distribution here we will have to `bin` the data.

How can we understand popularity of video games year by year?

Histogram

In [18]:
```
plt.hist(data['Year'])
plt.show()
```



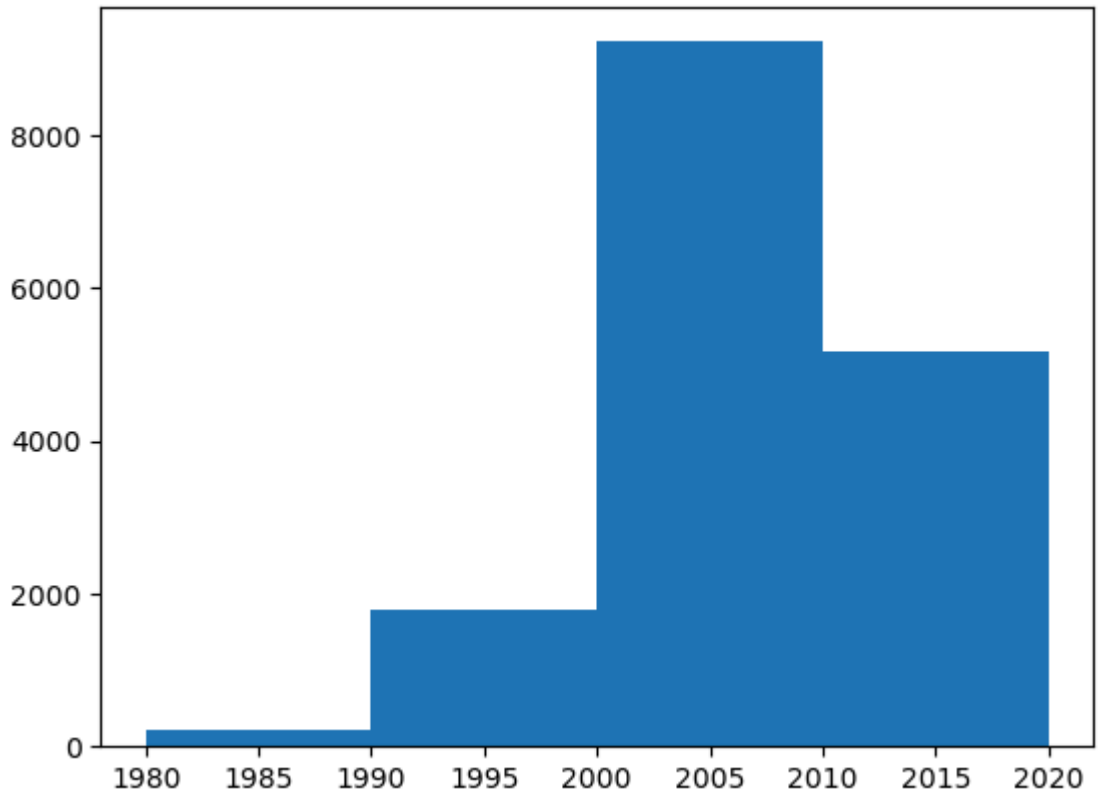
- The curve is left skewed, with a lot more games being published in 2005-2015.
- This shows that games started becoming highly popular in the last 1-2 decades.
- Maybe could point to increased usage of internet worldwide.

If you notice, histograms are basically frequency charts.

We can also vary the number of bins. **The default number of bins is 10**

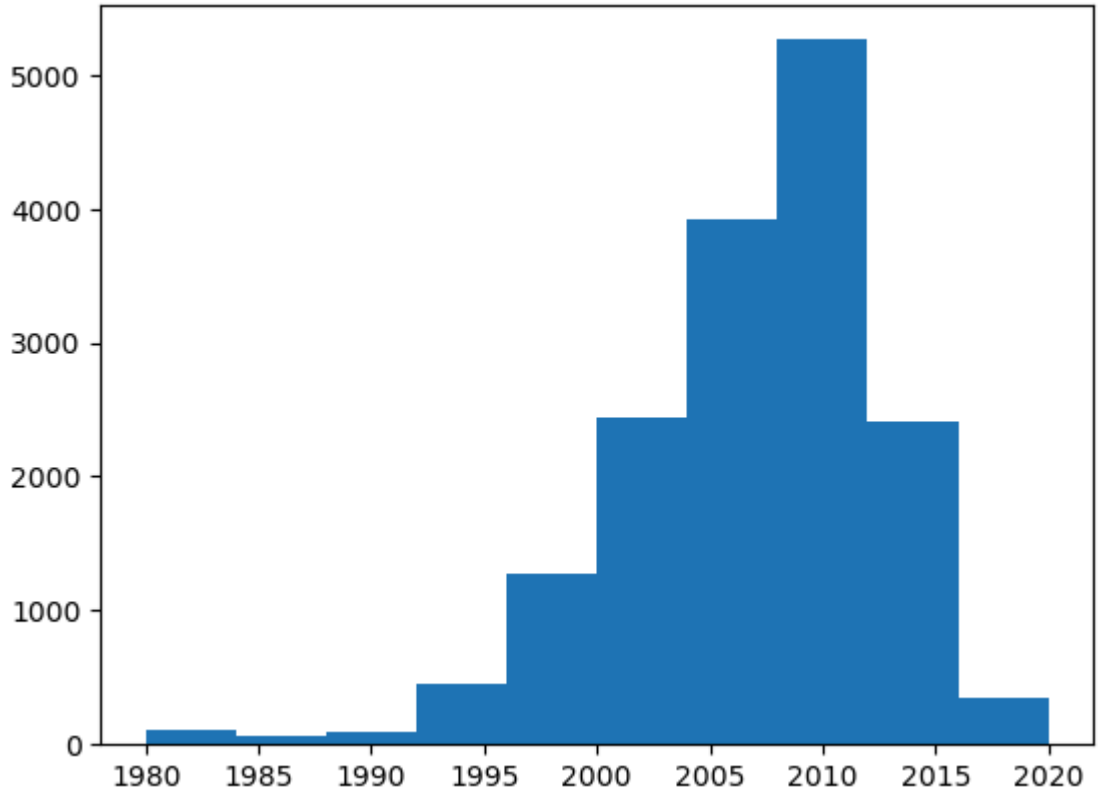
If we want to see this data per decade, we would need 40 years in 4 bins.

In [19]:
```
plt.hist(data['Year'], bins=4)
plt.show()
```



We can also get the data of each bin, such as range of the boundaries, values, etc.

In [20]:
```
count, bins, _ = plt.hist(data['Year'])
```



In [21]:
```
count
```

Out[21]:
```
array([ 112.,   70.,   92.,  449., 1274., 2440., 3921., 5262., 2406.,
        355.])
```

In [22]:
```
bins
```

Out[22]:
```
array([1980., 1984., 1988., 1992., 1996., 2000., 2004., 2008., 2012.,
       2016., 2020.])
```

What do these `count` and `bins` mean?

- **bins** provides bin edges
- **counts** provides it corresponding counts

What is the length of `count`?

- 10

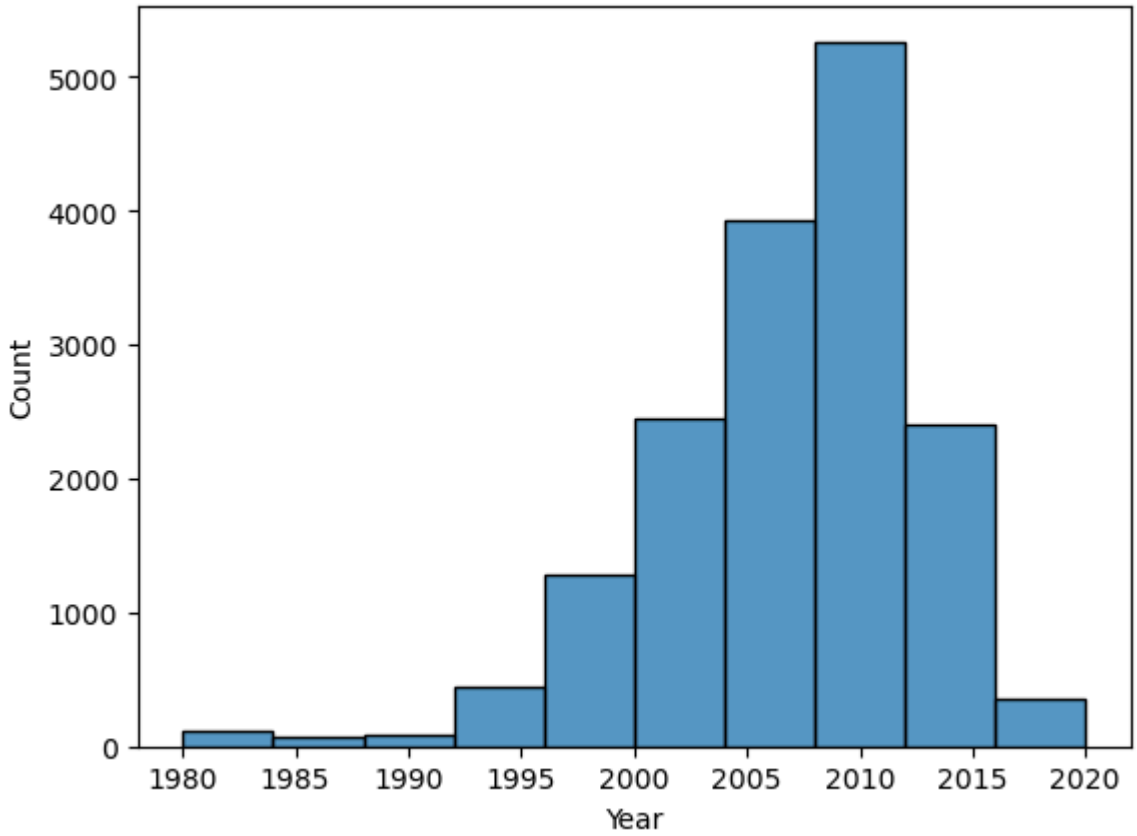
What shoud be the length of `bins`?

- 10 + 1 = 11

How can we plot a histogram in Seaborn?

In [23]:
```
sns.histplot(data['Year'], bins=10)
```

Out[23]:
```
<Axes: xlabel='Year', ylabel='Count'>
```



Notice that,

- The boundaries are more defined than matplotlib's plotting.
- The x and y axis are labelled automatically.

Kernel Density Estimate (KDE) Plot

- A KDE plot, similar to histrogram, is a method for visualizing the distributions.
- But instead of bars, KDE represents data using a **continuous probability density curve**.

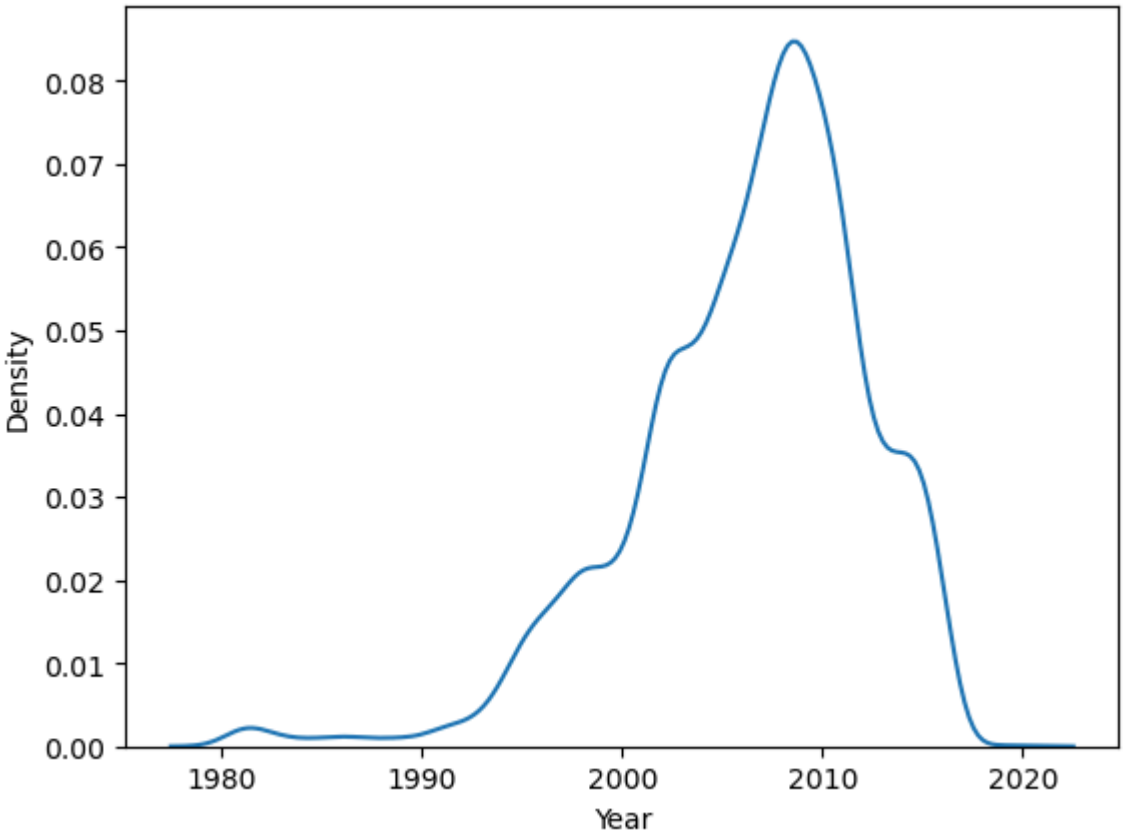
Why do we even need KDE plots?

- Compared to histogram, KDE produces a plot which is **less cluttered** and **more interpretable**.
- Think of it as a **smoothened version** of a histogram.

\ Let's plot KDE using `seaborn`'s `kdeplot`.

In [24]:
```
sns.kdeplot(data['Year'])
```

Out[24]:
```
<Axes: xlabel='Year', ylabel='Density'>
```



Can you notice the difference between KDE plot and histogram?

The Y-axis has **probability density estimation** instead of count.

You can read more about this on:

- https://en.wikipedia.org/wiki/Kernel_density_estimation
- https://www.youtube.com/watch?v=DCgPRaIDYXA

Boxplot

What if we want to find the aggregates like median, min, max and percentiles of the data.

Say I want the typical earnings of a game when it is published.

What kind of plot can we use here? Boxplot

What exactly is a Box plot?

- A box plot or **box and whiskers plot** shows the **distribution of quantitative data**.
- It facilitates comparisons between
  - attributes
  - across levels

of a categorical attribute.

- The **box** shows the **quartiles** of the dataset.
- The **whiskers** show the **rest of the distribution**.
- Except for points that are determined to be **"outliers"** using a method that is a function of the **inter-quartile range**.

Let's go through the terminology one-by-one.

Box plots show the five-number summary of data:

1. Minimum score
2. First (lower) quartile
3. Median
4. Third (upper) quartile
5. Maximum score

1. Minimum Score

- It is the **lowest value**, excluding outliers.
- It is shown at the **end of bottom whisker**.

2. Lower Quartile

- **25% of values** fall below the lower quartile value.
- It is also known as the **first quartile**.

3. Median

- Median marks the **mid-point of the data**.
- It is shown by the **line that divides the box into two parts**.
- **Half the scores are greater than or equal to this value and half are less**.
- It is sometimes known as the **second quartile**.

4. Upper Quartile

- **75% of values** fall below the upper quartile value.
- It is also known as the **third quartile**.

Maximum Score

- It is the **highest value**, excluding outliers.
- It is shown at the **end of upper whisker**.

Whiskers

- The upper and lower whiskers represent **values outside the middle 50%**.

- That is, the **lower 25% of values** and the **upper 25% of values**.

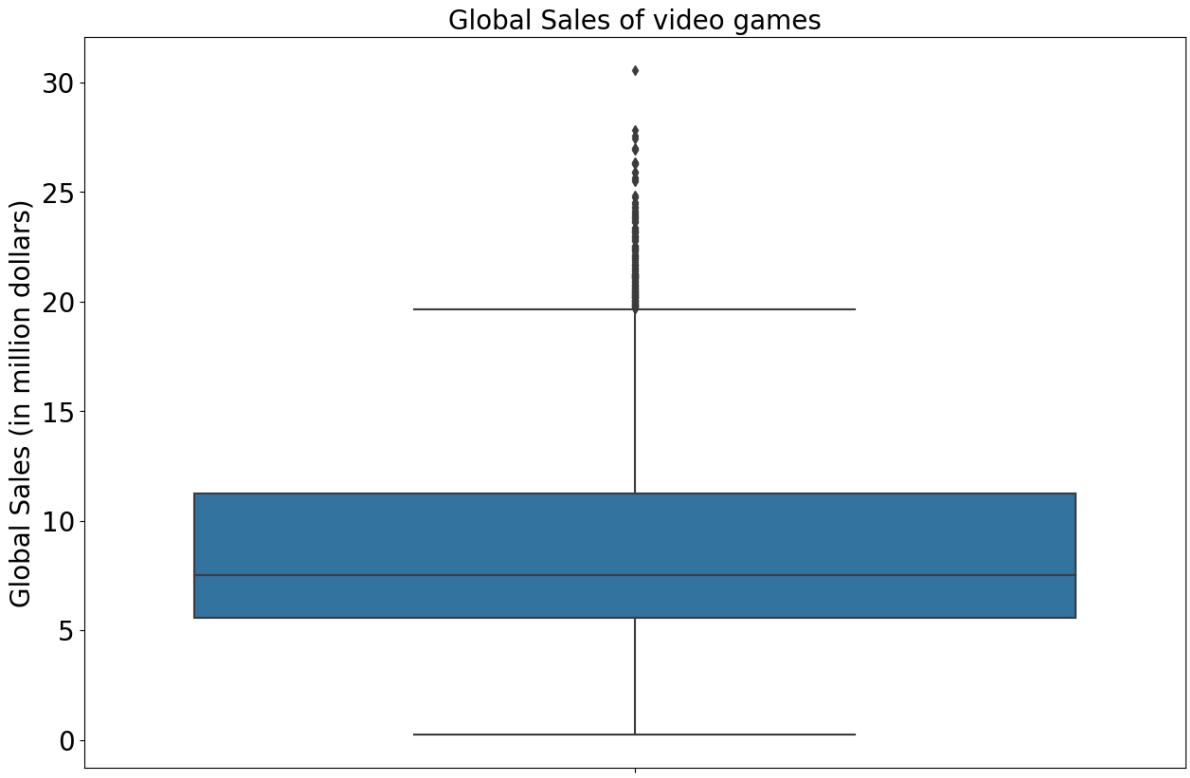
Interquartile Range (or IQR)

- This is the box plot showing the **middle 50% of scores**.
- It is the **range between the 25th and 75th percentile**.

\ Let's plot a box plot to find the average typical earnings for a game.

In [25]:
```
plt.figure(figsize=(15,10))
sns.boxplot(y = data["Global_Sales"])
plt.yticks(fontsize=20)
plt.ylabel('Global Sales (in million dollars)', fontsize=20)
plt.title('Global Sales of video games', fontsize=20)
```

Out[25]:
```
Text(0.5, 1.0, 'Global Sales of video games')
```



What can we infer from this?

The 5 point estimates here are:

- Minimum, excluding outliers: 0
- Maximum, excluding outliers: 20 (in million dollars)
- 25th Quantile: 6 million
- Median: around 7 million
- 75th Quantile: 12 million

Note:

- The outliers always will appear either below the minimum or above the maximum.
- There are quite a few outliers above 20 million dollars, represented by black colored circles.

Question-1

Q. For analyzing marks given by an edtech company, we want to find the range in which the most number of students have scored.

Which will be the best suited plot for this?

a. Pie Chart

b. Histogram

c. Countplot

d. Line Plot

Answer: Histogram

Explanation:

A histogram provides a visual representation of the distribution of numerical data, divided into intervals called bins. It allows you to see the frequency or count of data points within each bin, making it ideal for identifying the range in which the most number of students have scored. By observing the peak(s) in the histogram, you can easily determine the range(s) with the highest frequency of scores.

Question-2

Q. The telecom company "Airtel" wants to find the count customers across different payment modes opted by the customer.

Which will be the best suited plot for this?

a. Pie Chart

b. Countplot

c. Line Plot

d. Boxplot

Answer: Count Plot

Explanation:

A countplot is specifically designed for visualizing the count of observations in each category of a categorical variable. In this case, the payment modes are categorical variables, and the countplot will provide a clear representation of the number of customers using each payment mode. This makes it ideal for comparing the distribution of customers across different payment modes.

Question-3

Q. In the state “Haryana”, we want to find the proportion of people who smoke.

Which will be the best suited plot for this?

a. Pie Chart

b. Bar Chart

c. Countplot

d. Boxplot

Answer: Pie Chart

Explanation:

A pie chart is an effective way to represent proportions or percentages within a whole. In this case, the proportion of people who smoke can be represented as a fraction of the entire population of "Haryana". Each section of the pie chart would represent a different category, such as smokers and non-smokers, allowing for a clear visual comparison of the proportion of smokers in the population.