HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

Business Process Automation with VBA and Python

Mr. Eddie Chow / 7 June 2025

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

Table Of Contents

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

- What is Process Automation, RPA, IPA?
- Overview of the technological building blocks related to process automation
- Contemporary tools for process automation
- Challenges and opportunities of business process automation
- Business implications of process automation

Intended Learning Outcomes

1. describe the contemporary trends of process automation and explain the opportunities and challenges of business process automation;
2. outline key steps in process automation project management and illustrate the significance of each step;
3. apply computational tools to implement process automation;
4. discuss the development of process automation and practical cases for business.

Agenda

1. How to Manage Your Budget with a Simple Python Script
2. Financial Forecasting with Machine Learning using Python
3. Introduction to Project Management in Business Process Automation
4. Methodologies in Project Management
5. Key Steps in Project Management
6. Understanding Business Process Automation
7. Integration of Project Management and Business Process Automation
8. Organizational Issues in Business Process Automation Projects 7 Project Scope and Requirements Gathering
9. Risk Management in Automation Projects
10. Change Management Best Practices
11. Performance Measurement and Key Performance Indicators (KPIs)
12. Case Studies of Successful Automation Projects
13. Conclusion and Future Directions

Data Visualizations

1. describe the contemporary trends of process automation and explain the opportunities and challenges of business process automation;
2. outline key steps in process automation project management and illustrate the significance of each step;
3. apply computational tools to implement process automation;
4. discuss the development of process automation and practical cases for business.

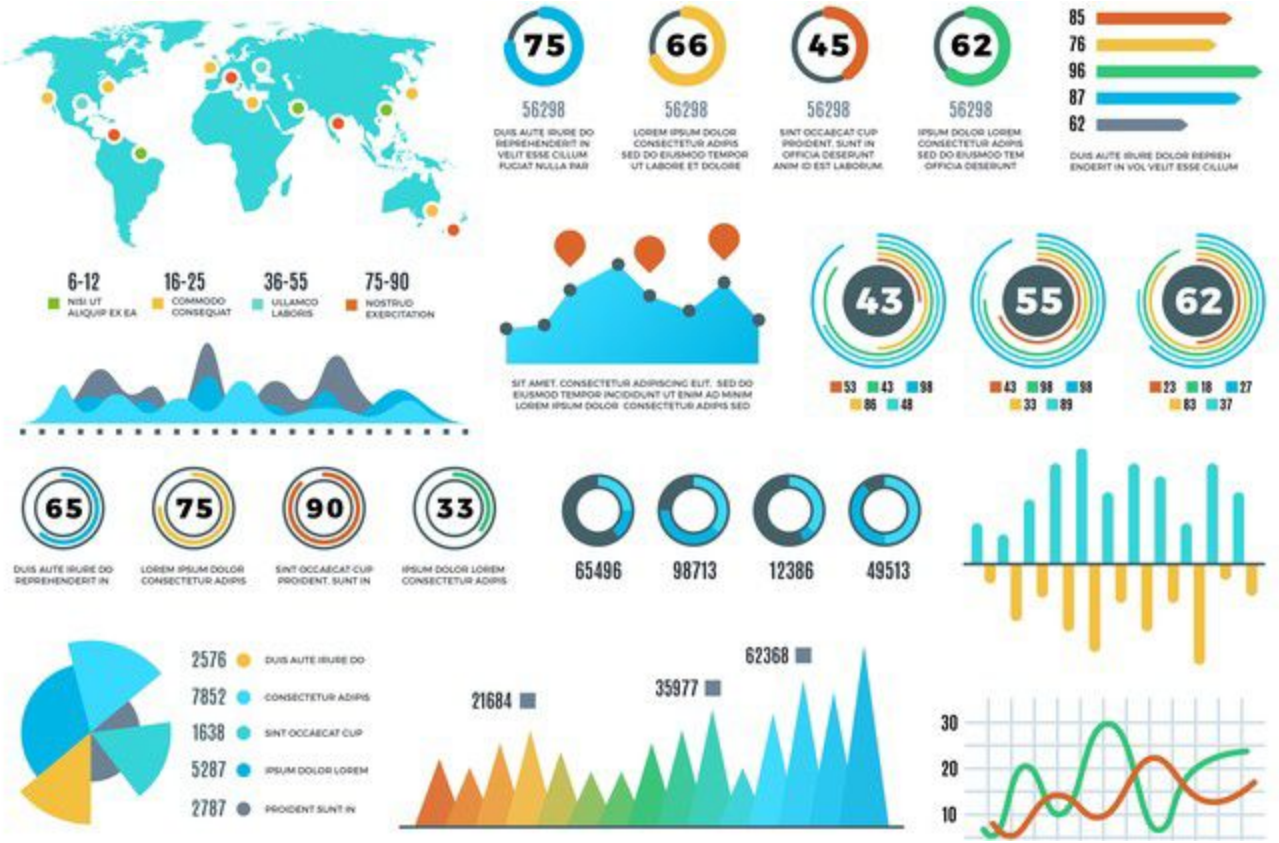
Practice 1 : Display the predicted housing price per square

```
#add necessary library
import joblib

#Define function to get user input
def get_user_input():
    """Get input data from the user."""
    # Modify prompts based on the model's expected input
    district = float(input("Enter District Number : "))
    schoolnet_no = float(input("Enter School Net Number: "))
     floor = float(input("Enter Floor: "))  # Adjust based on model input size

    return [[district, schoolnet_no, floor]]
    return pd.DataFrame([[district, schoolnet_no, floor]], columns=['District', 'School_Netnumber',
     'Floor'])

#Define function for loading random forest model
def load_model(model_path):
     """Load the Joblib model from the specified path."""
    return joblib.load(model_path)
```

Practice 1 : Display the predicted housing price per square

```
#Define function to predict housing price per square based on 3 data input
def predict(model, input_data):
    """Make a prediction using the model."""
    return model.predict(input_data)

#Main Program to display the predicted housing price per square
#Step 1 : Get Input Value to predict
input_data = get_user_input()
#Step 2 : Load the Housing Valuation Model
model_path = 'housing_valuation_model.joblib'  # Update this path
model = load_model(model_path)
#Step 3 : Predict and Show the result
result = predict(model, input_data)
print(f"Predicted Hosing Price Per Square is : HKD ${round(result[0],2)}")
```

Data Visualizations



Data Visualizations

- Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from.
- The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets.
- The term is often used interchangeably with others, including information graphics, information visualization and statistical graphics.
- Data visualization is one of the steps of the data science process, which states that after data has been collected, processed and modeled, it must be visualized for conclusions to be made.
- Data visualization is also an element of the broader data presentation architecture (DPA) discipline, which aims to identify, locate, manipulate, format and deliver data in the most efficient way possible.

Benefits of Data Visualizations

- The ability to absorb information quickly, improve insights and make faster decisions;
- An increased understanding of the next steps that must be taken to improve the organization;
- An improved ability to maintain the audience's interest with information they can understand;
- An easy distribution of information that increases the opportunity to share insights with everyone involved;
- Eliminate the need for data scientists since data is more accessible and understandable; and
- An increased ability to act on findings quickly and, therefore, achieve success with greater speed and less mistakes.

Data Visualization Roles - Change over time

Line chart +Comparisons

Most common chart type for showing change over time. A point is plotted for each time period from left to right; each point's vertical position indicates the feature's value. Points are connected by line segments to emphasize progression across time.

Sparkline +Comparisons

A miniature line chart with little to no labeling, designed to be placed alongside text or in tables. Provides a high-level overview without attracting too much attention. Can also be seen in a sparkbar form, or miniature bar chart (see below).

Data Visualization Roles - Change over time

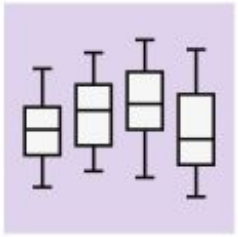
Connected scatter plot ◆ *+Relationships*

Shows change over time across two numeric variables (see scatter plot in *Relationships*). Line segments still connect points across time, but they may not consistently go from left to right like in a line chart.

Bar chart ● *+Distributions* *+Comparisons*

Each time period is associated with a bar; each bar's value is represented in its height above (or below) a zero-baseline. Works best when there aren't too many time periods to show.

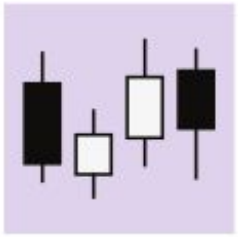
Data Visualization Roles - Change over time



Box plot +Distributions +Comparisons

Each time period is associated with a box and whiskers; each set of box and whiskers shows the range of the most common data values. Best when there are multiple recordings for each time period and a distribution of values needs to be plotted.

Tracking change over time is of key interest in the financial domain. One specialist chart developed for this field includes the following:



Candlestick chart

Looks like a box plot, but each box and whiskers encodes different statistics. The box ends indicate opening and closing prices, while color indicates the direction of change.

Data Visualization Roles - Part-to-whole composition

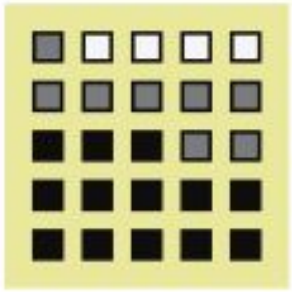
Pie chart

The whole is represented by a filled circle. Parts are proportional slices from that circle, one for each categorical group. Best with five or fewer slices with distinct proportions.

Doughnut chart

A pie chart with a hole in the center. This central area can be used to show a relevant single numeric value. Sometimes used as an aesthetic alternative to a standard progress bar (see stacked bar chart below).

Data Visualization Roles - Part-to-whole composition



Waffle chart / grid plot

Squares laid out in a (typically) 10 x 10 grid; each square represents one percent of the whole. Squares are colored based on categorical group size.

Stacked bar chart

A bar chart (see *Change over time* or *Distributions*) where each bar has been divided into multiple sub-bars to show a part-to-whole breakdown. A single stacked bar can be used as an alternative to the pie or doughnut chart; people tend to make more precise judgments of length over area or angle.

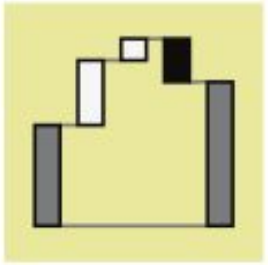
Data Visualization Roles - Part-to-whole composition

Stacked area chart ●

A line chart (see *Change over time*) where shaded regions are added under the line to divide the total into sub-group values.

Stream graph ◆

Modified version of the stacked area chart where areas are stacked around a central axis. Highlights relative changes instead of exact values.



Waterfall chart ◆

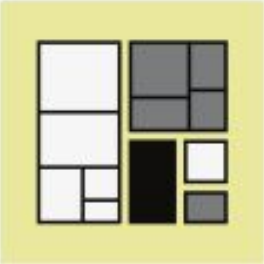
Augments a change over time with a part-to-whole decomposition. Bars on the ends depict values at two time points, and lengths of intermediate floating bars' show the decomposition of the change between points.

Data Visualization Roles - Part-to-whole composition

Certain part-to-whole compositions follow a hierarchical form. In these cases, each part can be divided into finer parts on lower levels. Here are a couple of more specialized chart types for visualizing this type of data:

Mosaic plot / Marimekko chart ■

Can be thought of as a stacked bar divided on both axes. A box is divided on one axis based on one categorical variable, then each sub-box is divided in the other axis based on a second categorical variable.



Treemap ◆

Can be thought of as a more generalized Marimekko plot. Sub-boxes do not need to have a consistent cut direction at a particular hierarchy level, and there can be more than two levels of hierarchy.

Data Visualization Roles - Flows and processes

Funnel chart ■

Seen in business contexts, showing how people encounter a product and eventually become users or customers. One bar is plotted for each stage, whose lengths reflect the number of users. Connecting regions emphasize connections in stages and give the chart type's namesake shape.

Parallel sets chart ◆

Multiple part-to-whole divisions on different dimensions are depicted as parallel stacked bars. Connecting regions show how different subgroups relate to one another between dimensions.

Data Visualization Roles - Flows and processes

Sankey diagram ◆

The width of the colored region shows the relative volume at each part of a process. Allows for multiple sources of inputs and outputs to be visualized.

Gantt chart ■

Used for project scheduling, breaking them down into individual tasks. Each task is associated with a bar, providing a timeline for when each task should begin and end.

Data Visualization Roles - Geographical data

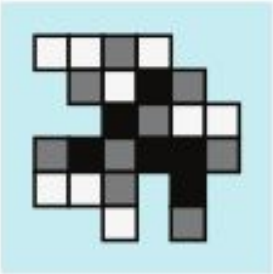
Scatter map

Scatter plot built on top of a geographical map, using geographic coordinates as point positions.

Bubble map

Bubble chart built on top of a geographic map, where point size is an indicator of value. Can also be used to group together points in a scatter map if they are too dense.

Data Visualization Roles - Geographical data



2-d histogram ●

Heatmaps can be built on top of geographic areas. Sometimes seen with a hexagon-shaped grid rather than a rectangular grid. May distort the geography on its edges.

Isopleth / contour map ◆

2-d density curve built on top of a geographic map.

Connection map ◆

Network information and flows built on top of a geographic map.

Data Visualization Roles - Geographical data

Choropleth ●

Similar to a heatmap, but colors are assigned to geopolitical regions rather than an arbitrary grid. Values are often in the form of rates or ratios to avoid distortion due to population density.

Cartogram ◆

Geopolitical regions sized by value. This necessarily requires distortion in shapes and topology.

Data Visualization Tools

- Tableau
- Infogram
- ChartBlocks
- D3.js
- Google Charts
- Fusion Charts
- Chart.js

Visualization using Programming

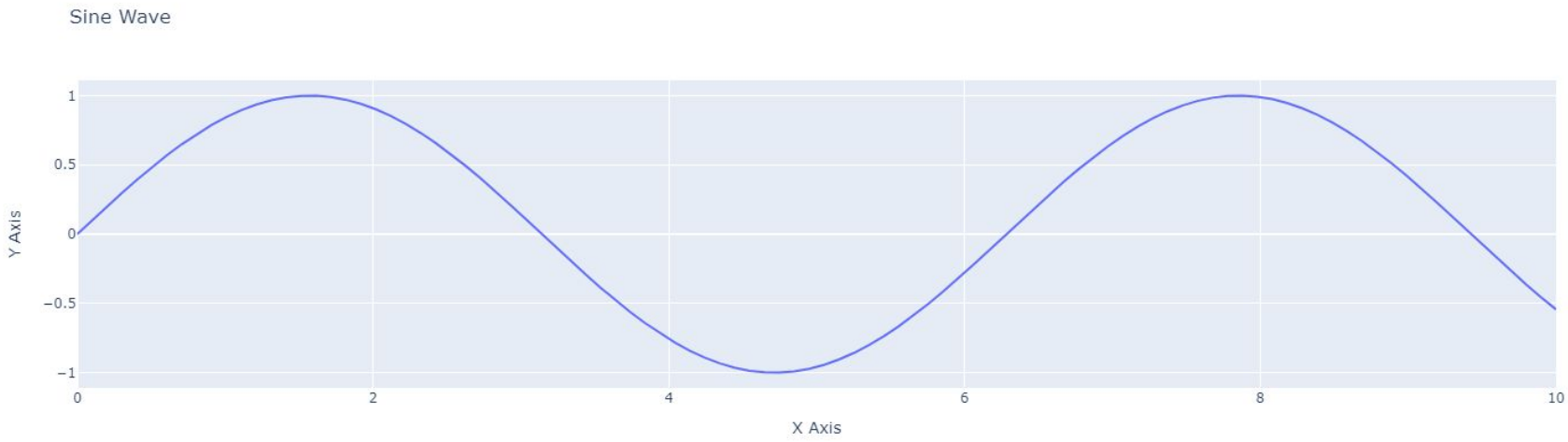
- Python
  - matplotlib
  - seaborn
  - plotly
  - pylab
- R
  - graphics
  - ggplot2

Exercise 1

Step 5

```
pip install plotly numpy
    #using plotly library and display lines chart
    import numpy as np
    import plotly.graph_objects as go

    # Generate sample data
    x = np.linspace(0, 10, 100)  # 100 points
        from 0 to 10
    y = np.sin(x)                # y is the sine of x
```

```
# Create a plotly figure
fig = go.Figure()

# Add a line plot
fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Sine
Wave'))

# Customize layout
fig.update_layout(title='Sine Wave',
                  xaxis_title='X Axis',
                  yaxis_title='Y Axis')

# Show the plot
fig.show()
```

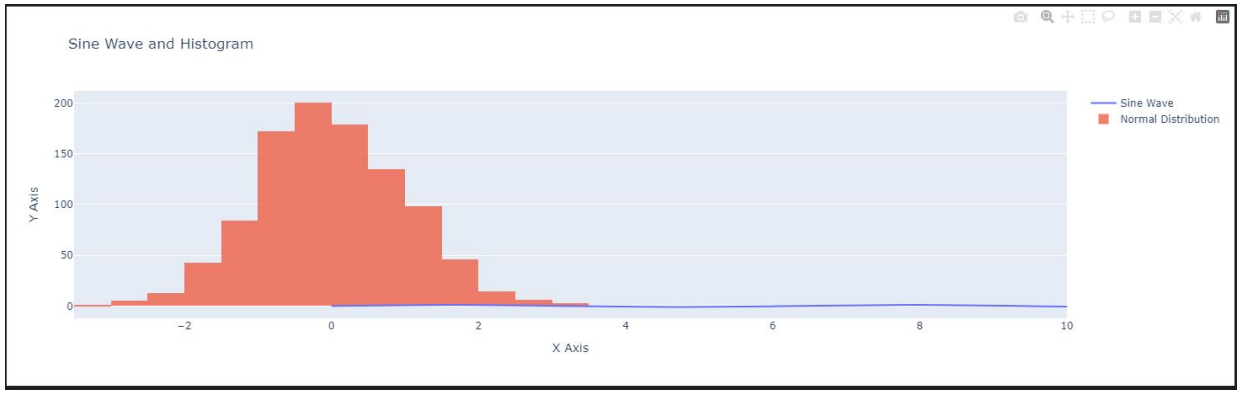


Exercise 1

Step 6

```
#using plotly library and display histgram chart
import numpy as np
import plotly.graph_objects as go

    # Generate sample data for the sine wave
x = np.linspace(0, 10, 100)  # 100 points from 0 to
    10
y = np.sin(x)                # y is the sine of x

# Generate sample data for the histogram
data = np.random.normal(loc=0, scale=1,
    size=1000)  # 1000 random values from a
    normal distribution

# Create a plotly figure
fig = go.Figure()
```

```
# Add a line plot for the sine wave
fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Sine
Wave'))

# Add a histogram
fig.add_trace(go.Histogram(x=data, name='Normal
Distribution', opacity=0.75, nbinsx=30))

# Customize layout
fig.update_layout(title='Sine Wave and Histogram',
                  xaxis_title='X Axis',
                  yaxis_title='Y Axis',
                  barmode='overlay')  # Overlay histograms

# Show the plot
fig.show()
```

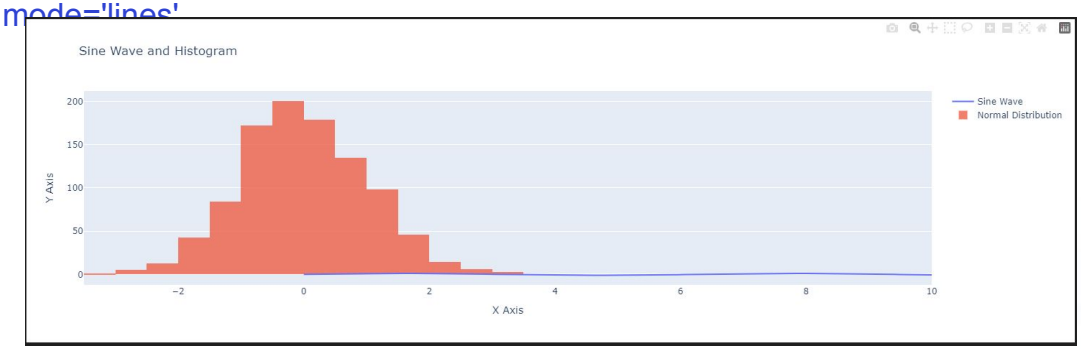


Exercise 1

Step 7

```
#using plotly, display histgram chart and download chart
import numpy as np
import plotly.graph_objects as go

# Generate sample data for the sine wave
x = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y = np.sin(x)                # y is the sine of x

# Generate sample data for the histogram
data = np.random.normal(loc=0, scale=1, size=1000)  #
    1000
random values from a normal distribution

# Create a plotly figure
fig = go.Figure()

# Add a line plot for the sine wave
fig.add_trace(go.Scatter(x=x, y=y, mode='lines'
    name='Sine Wave'))
```

```
# Add a histogram
fig.add_trace(go.Histogram(x=data, name='Normal
Distribution', opacity=0.75, nbinsx=30))

# Customize layout
fig.update_layout(title='Sine Wave and Histogram',
                  xaxis_title='X Axis',
                  yaxis_title='Y Axis',
                  barmode='overlay')  # Overlay histograms

# Show the plot
fig.show()
# Show the plot
fig.show()

# Save the figure as a PNG filepio.write_image(fig,
'sine_wave_histogram.png')
```

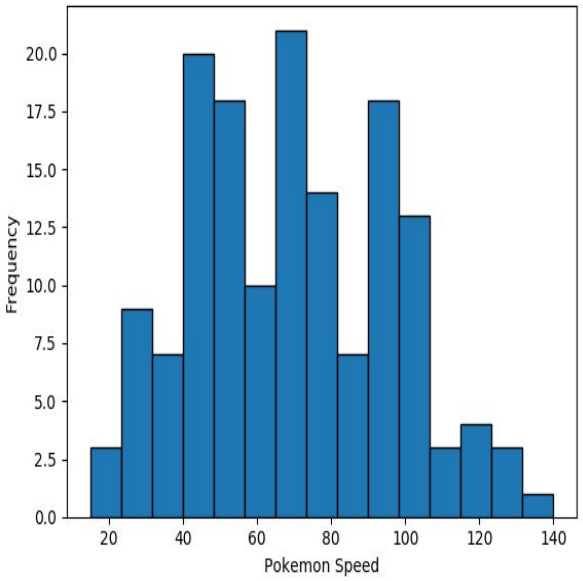


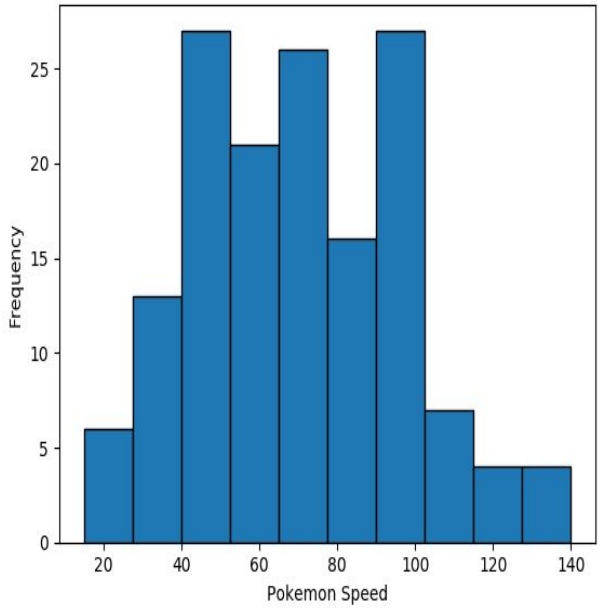
Bins

You may have noticed the two histograms we've seen so far look different, despite using the **exact** same data.

This is because they have different bin values.

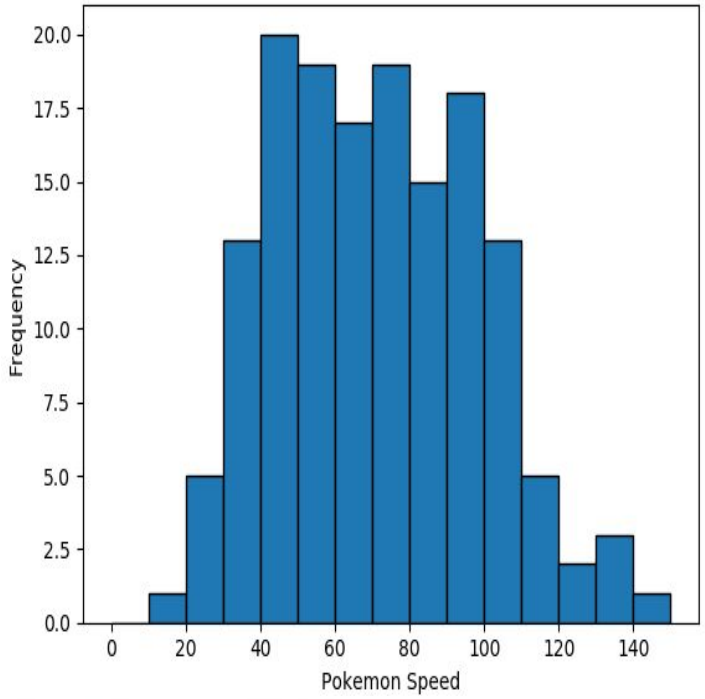
The left graph used the default bins generated by `plt.hist()`, while the one on the right used bins that I specified.





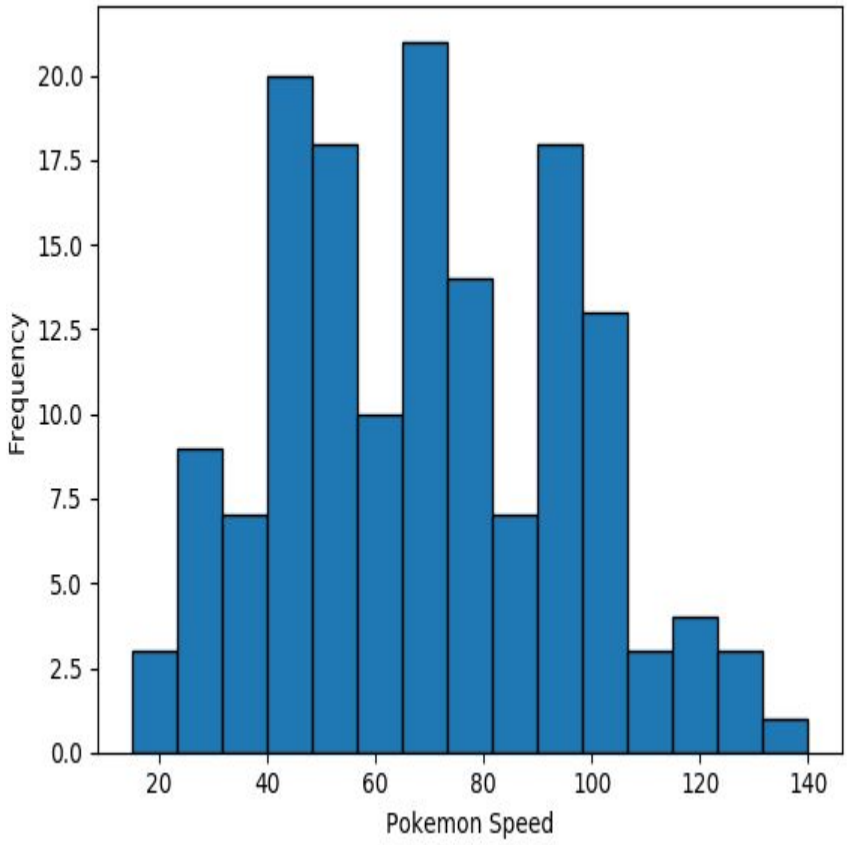
There are a couple of ways to manipulate bins in matplotlib.
Here, I specified where the edges of the bars of the histogram are; the bin edges.

```
bin_edges = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150]
g = plt.hist(df1['Speed'], histtype='bar', ec='black', bins=bin_edges)
g = plt.xlabel('Pokemon Speed')
g = plt.ylabel('Frequency')
plt.show()
```



You could also specify the number of bins, and Matplotlib will automatically generate a number of evenly spaced bins.

```
g = plt.hist(df1['Speed'], histtype='bar', ec='black', bins=15)
g = plt.xlabel('Pokemon Speed')
g = plt.ylabel('Frequency')
plt.show()
```

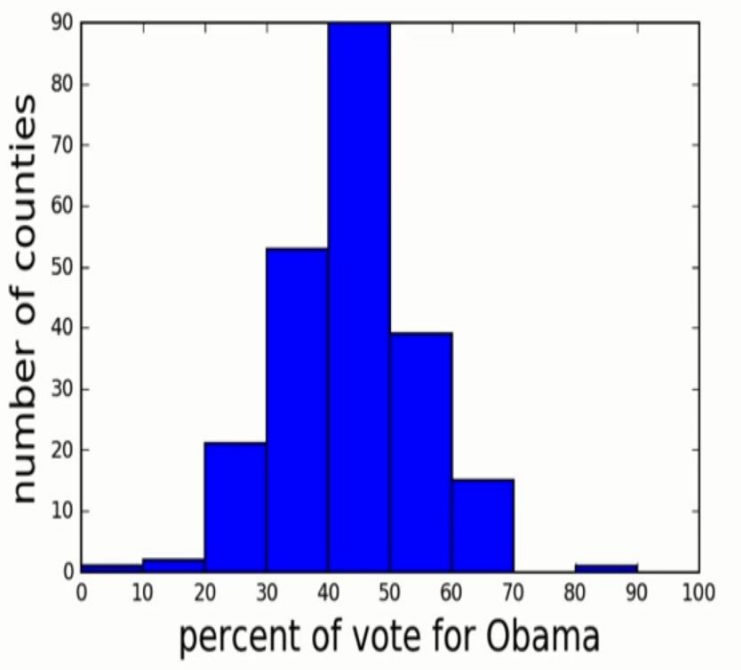


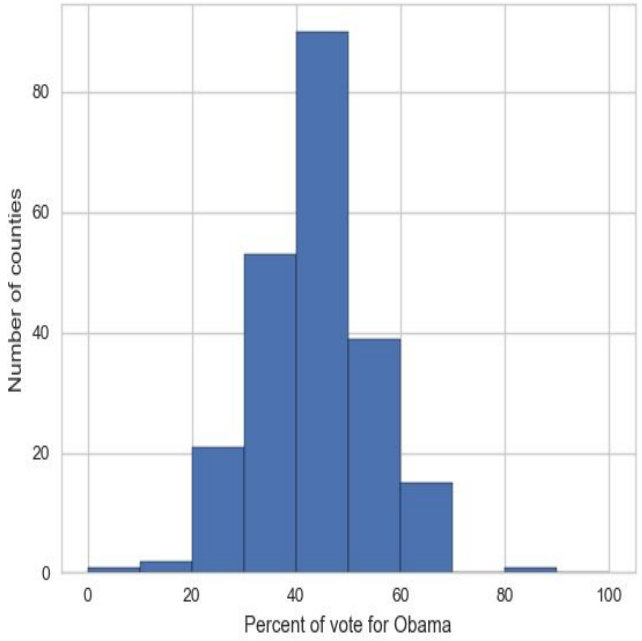
Seaborn

Matplotlib is a powerful, but sometimes unwieldy, Python library.

Seaborn provides a high-level interface to Matplotlib and makes it easier to produce graphs like the one on the right.

Some IDEs incorporate elements of this “under the hood” nowadays.





Benefits of Seaborn

Seaborn offers:

- Using default themes that are aesthetically pleasing.
- Setting custom colour palettes.
- Making attractive statistical plots.
- Easily and flexibly displaying distributions.
- Visualising information from matrices and DataFrames.

The last three points have led to Seaborn becoming the exploratory data analysis tool of choice for many Python users.

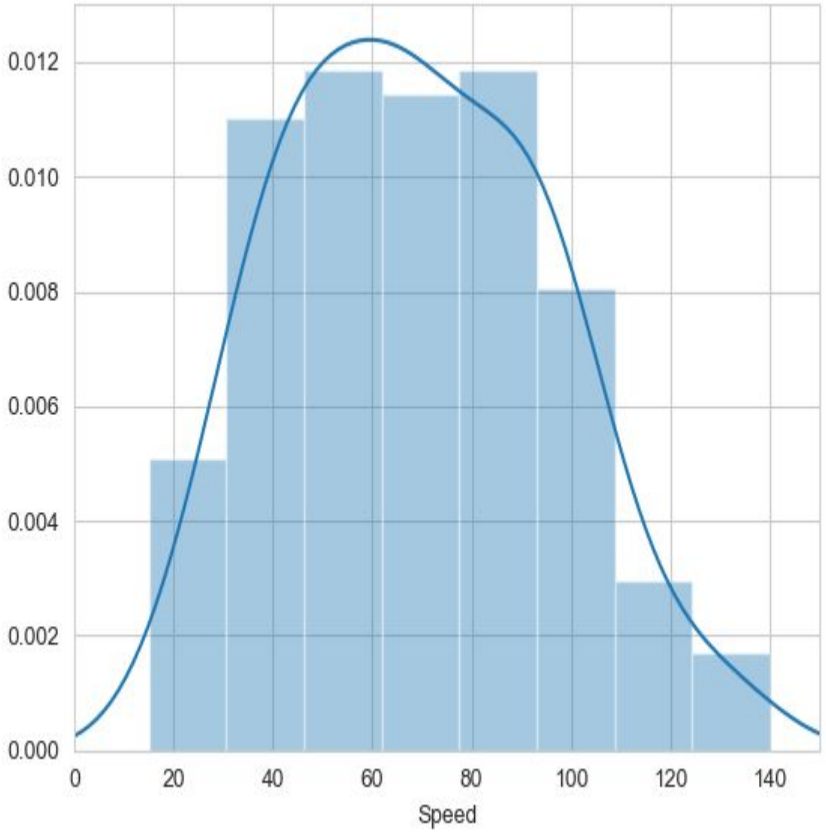
Plotting with Seaborn

One of Seaborn's greatest strengths is its diversity of plotting functions.
Most plots can be created with one line of code.
For example….

Histograms

Allow you to plot the distributions of numeric variables.

```
sns.set_style()
sns.distplot(df1.Speed)
```



Other types of graphs: Creating a scatter plot

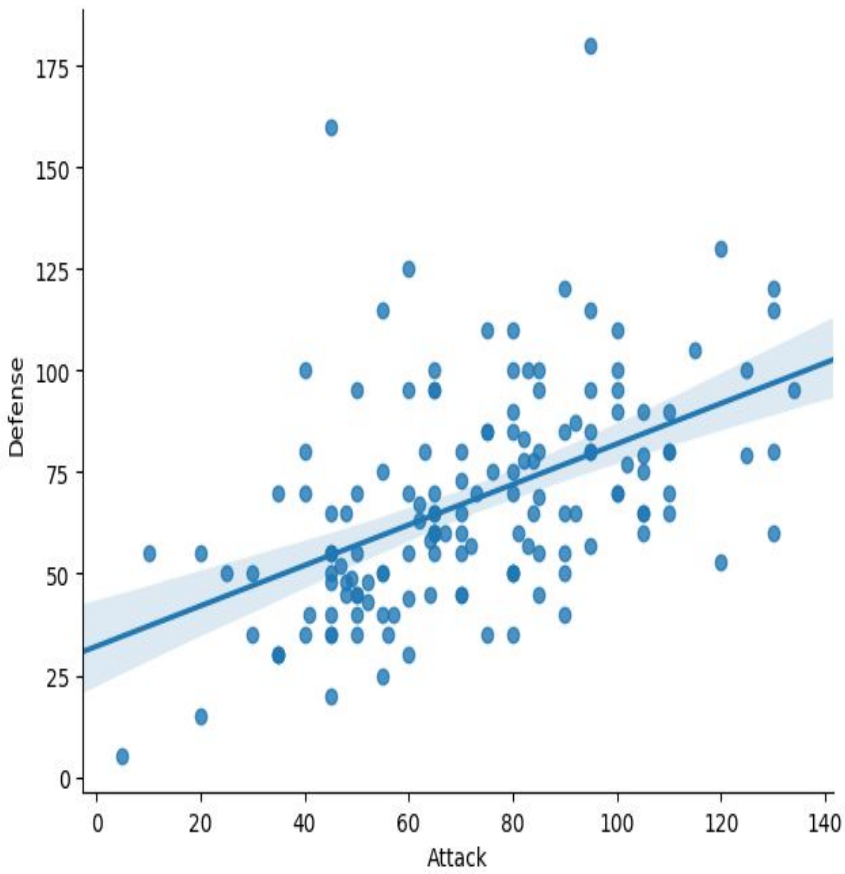
Name of variable we want on the x-axis

Name of our dataframe fed to the "data=" command

```
sns.lmplot(x='Attack', y='Defense', data=df1)
```

Seaborn "linear model plot" function for creating a scatter graph

Name of variable we want on the y-axis



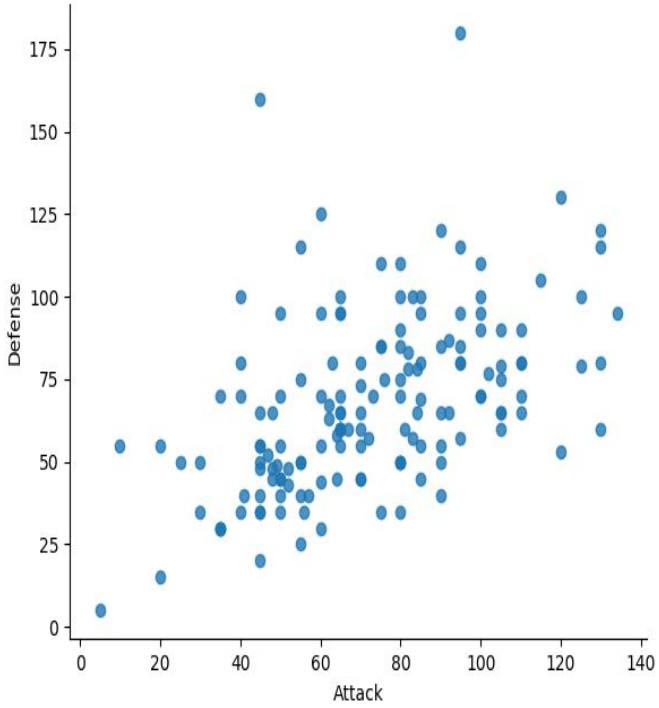
Seaborn doesn't have a dedicated scatter plot function.

We used Seaborn's function for fitting and plotting a regression line; hence lmplot()

However, Seaborn makes it easy to alter plots.

To remove the regression line, we use the fit_reg=False command

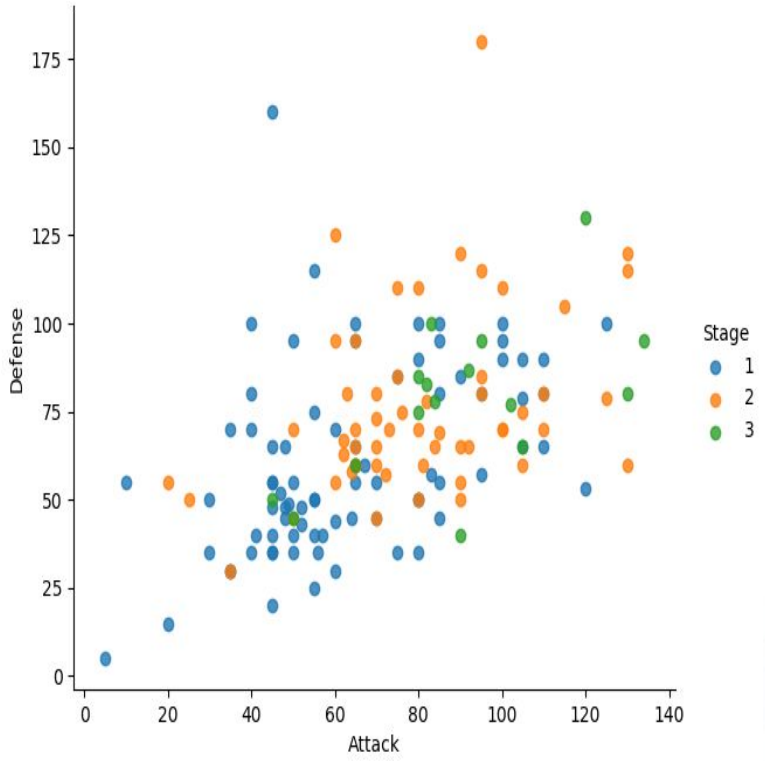
```
sns.lmplot(x='Attack', y='Defense', data=df1, fit_reg=False)
```



The hue function

Another useful function in Seaborn is the hue function, which enables us to use a variable to colour code our data points.

```
sns.lmplot(x='Attack', y='Defense', data=df1,
           fit_reg=False,
           hue='Stage')
```



Factor plots

Make it easy to separate plots by categorical classes.

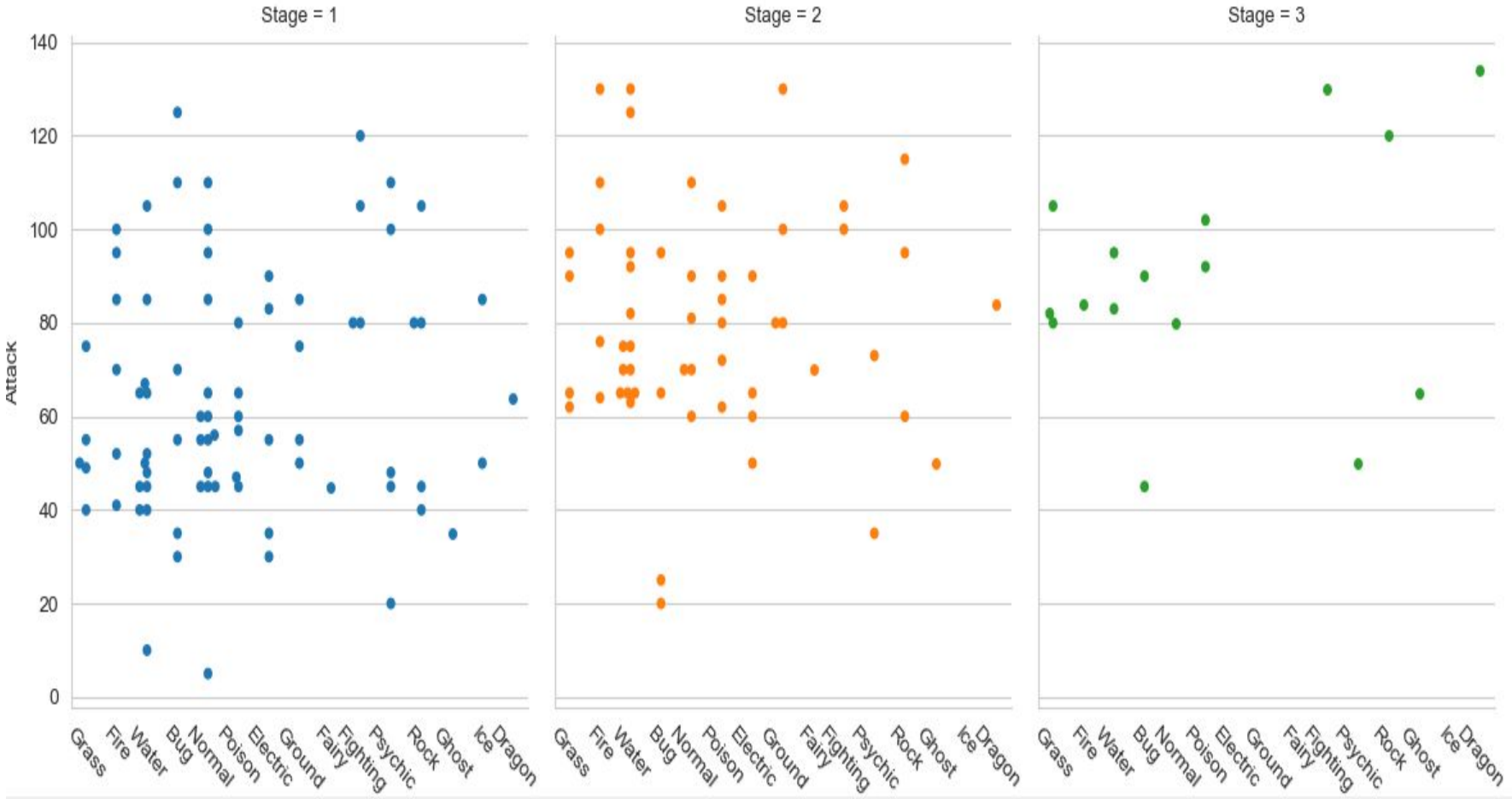
```
g = sns.factorplot(x='Type 1',
                   y='Attack',
                   data=df1,
                   hue='Stage',
                   col='Stage',
                   kind='swarm')
g.set_xticklabels(rotation=-45)
```

Colour by stage.

Separate by stage.

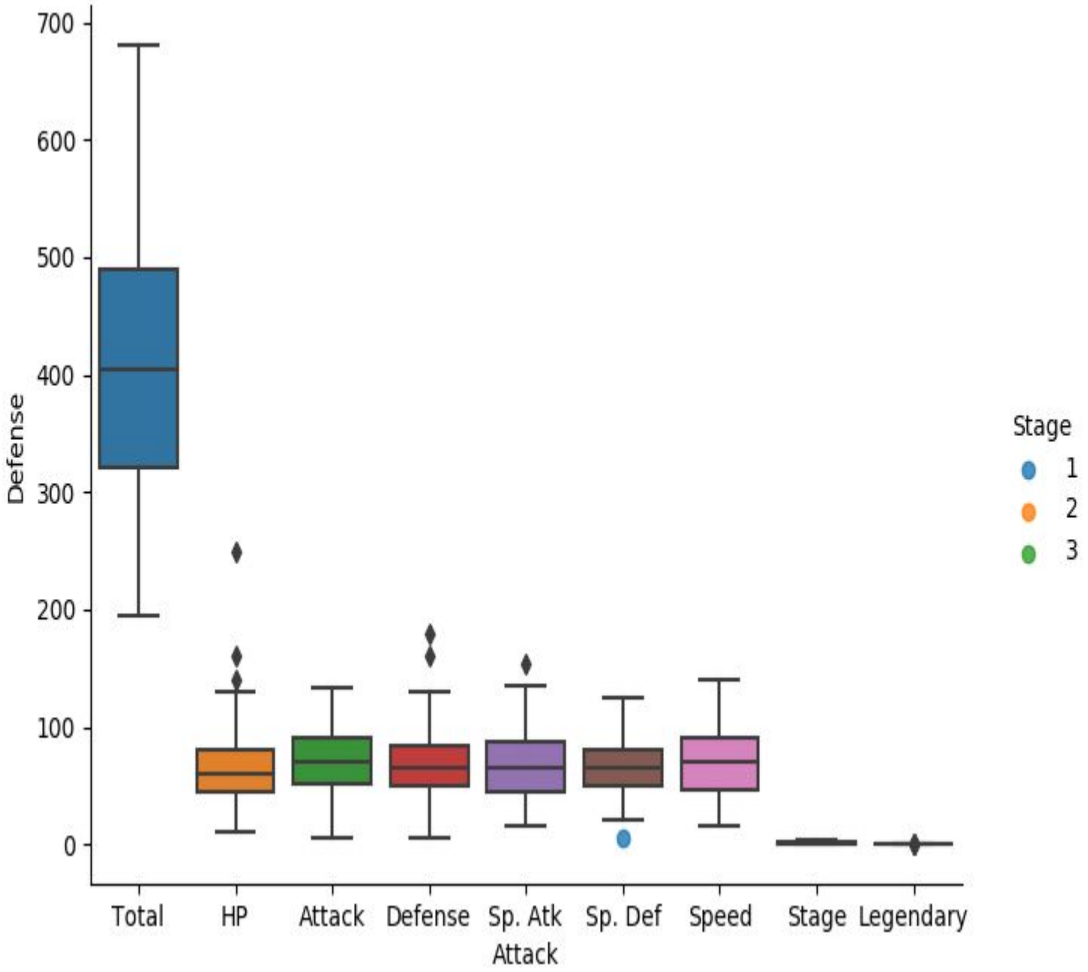
Generate using a swarmplot.

Rotate axis on x-ticks by 45 degrees.


Stage = 1
Stage = 2
Stage = 3
Attack
140
120
100
80
60
40
20
0
Grass
Fire
Water
Bug
Normal
Poison
Electric
Ground
Fairy
Fighting
Psychic
Rock
Ghost
Ice
Dragon

A box plot

```
sns.boxplot(data=df1)
```

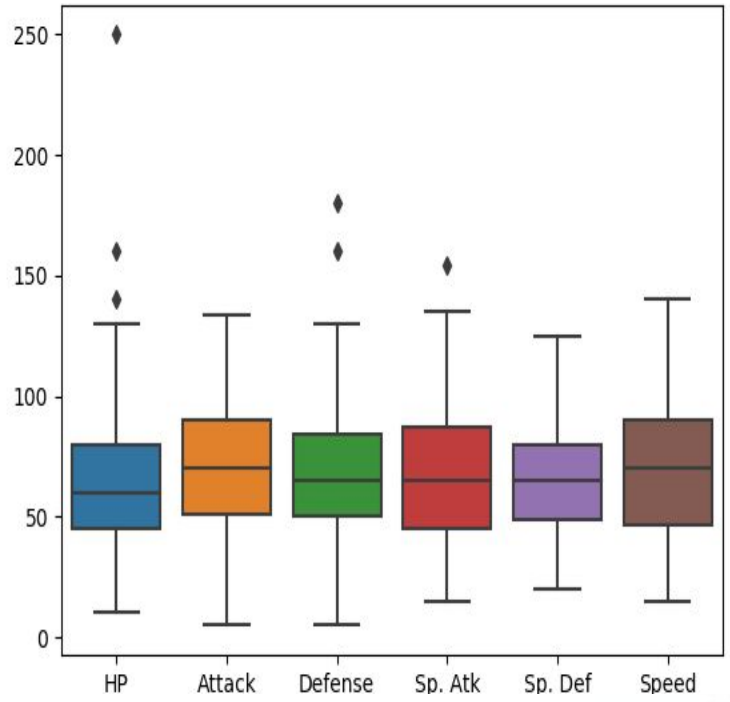


The total, stage, and legendary entries are not combat stats so we should remove them.

Pandas makes this easy to do, we just create a new dataframe

We just use Pandas' .drop() function to create a dataframe that doesn't include the variables we don't want.

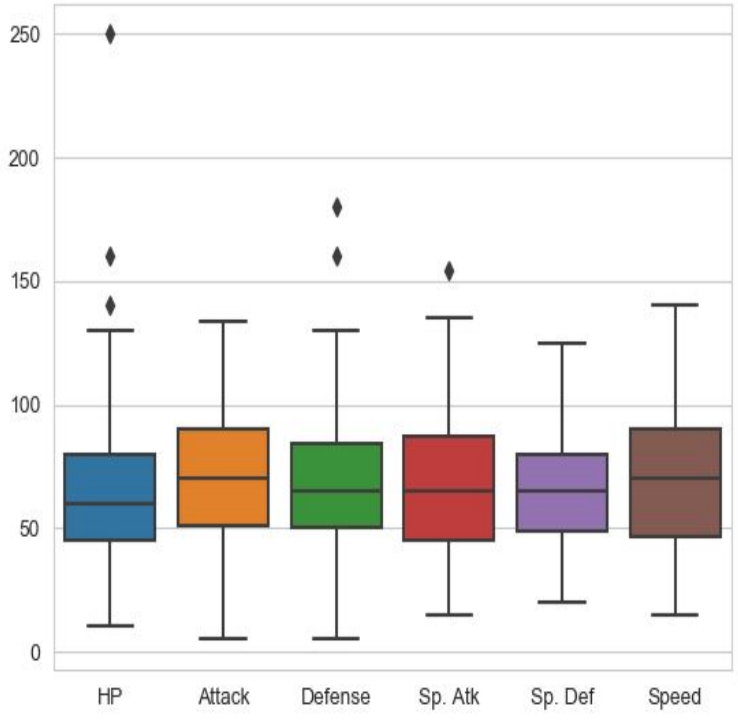
```
stats_df = df1.drop(['Total', 'Stage', 'Legendary'], axis=1)
sns.boxplot(data=stats_df)
```



Seaborn's theme

Seaborn has a number of themes you can use to alter the appearance of plots. For example, we can use "whitegrid" to add grid lines to our boxplot.

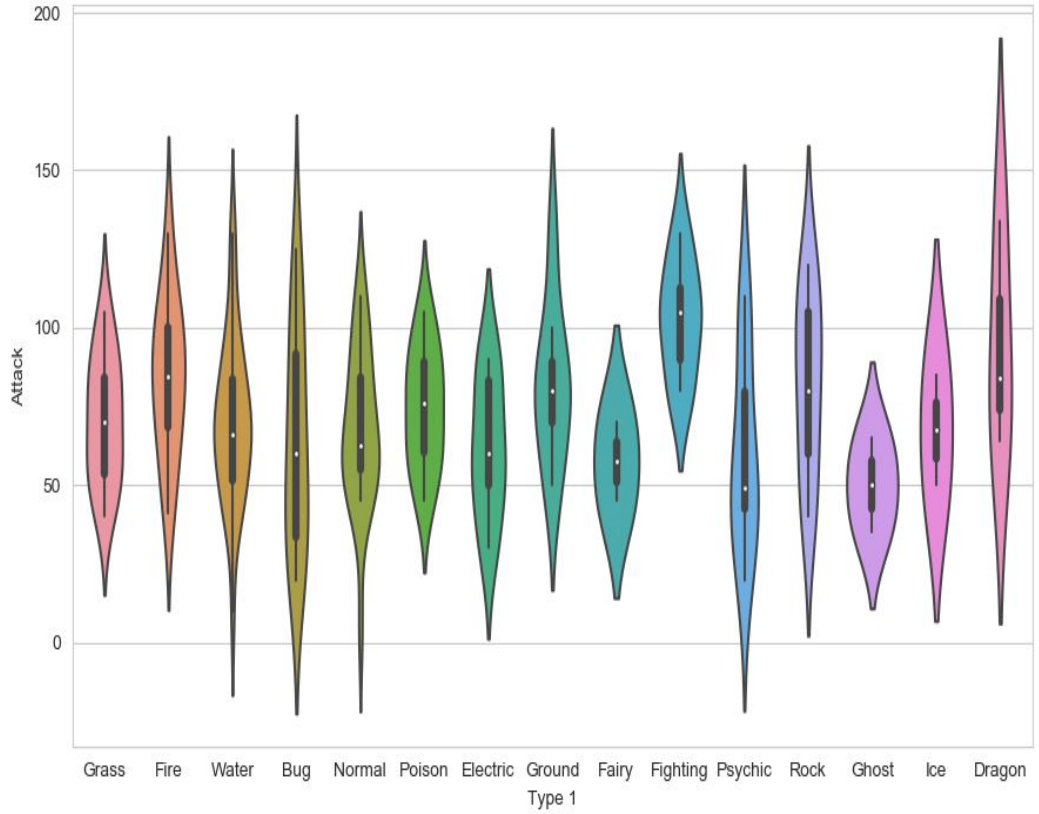
```
stats_df = df1.drop(['Total', 'Stage', 'Legendary'], axis=1)
sns.set_style('whitegrid')
sns.boxplot(data=stats_df)
```



Violin plots

Violin plots are useful alternatives to box plots.
They show the distribution of a variable through the thickness of the violin.
Here, we visualise the distribution of attack by Pokémon's primary type:

```
sns.violinplot(x='Type 1', y='Attack', data=df1)
```



- Dragon types tend to have higher Attack stats than Ghost types, but they also have greater variance. But there is something not right here….
- The colours!

Seaborn's colour palettes

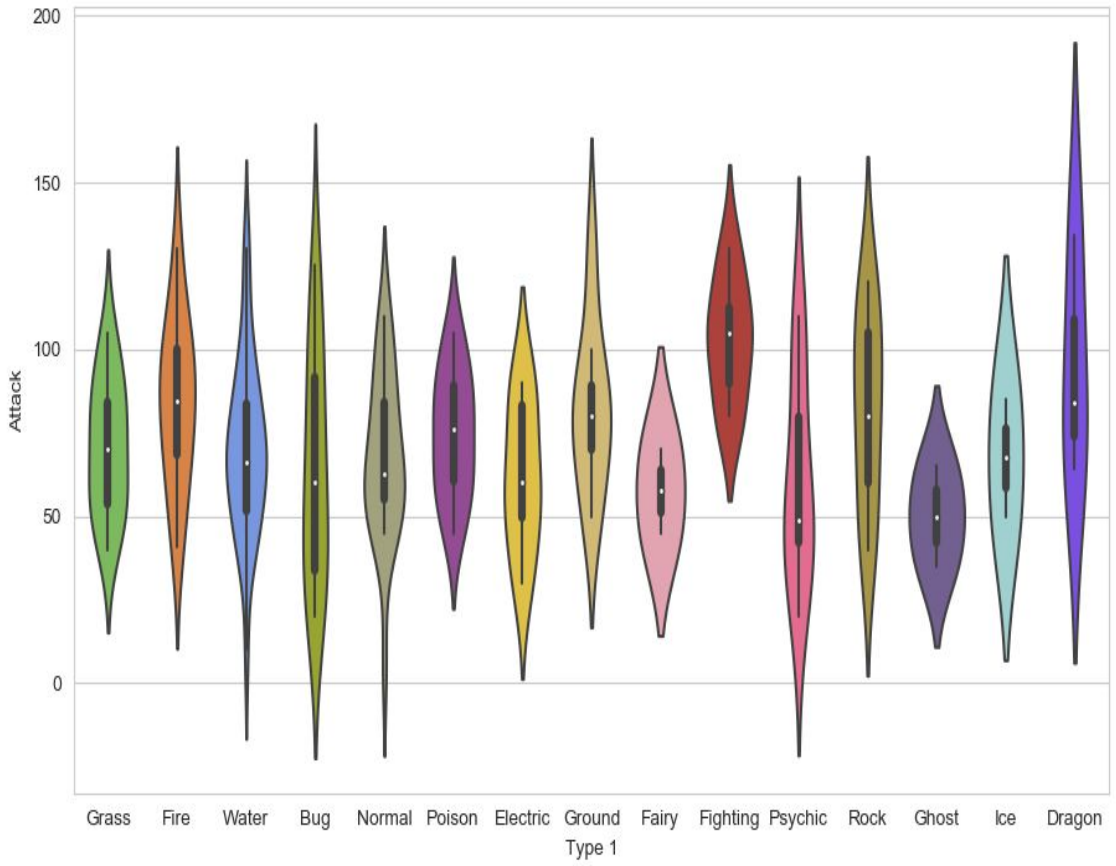
Seaborn allows us to easily set custom colour palettes by providing it with an ordered list of colour hex values.

We first create our colours list.

```
type_colors = ['#78C850',  # Grass
               '#F08030',  # Fire
               '#6890F0',  # Water
               '#A8B820',  # Bug
               '#A8A878',  # Normal
               '#A040A0',  # Poison
               '#F8D030',  # Electric
               '#E0C068',  # Ground
               '#EE99AC',  # Fairy
               '#C03028',  # Fighting
               '#F85888',  # Psychic
               '#B8A038',  # Rock
               '#705898',  # Ghost
               '#98D8D8',  # Ice
               '#7038F8',  # Dragon
              ]
```

Then we just use the palette= function and feed in our colours list.

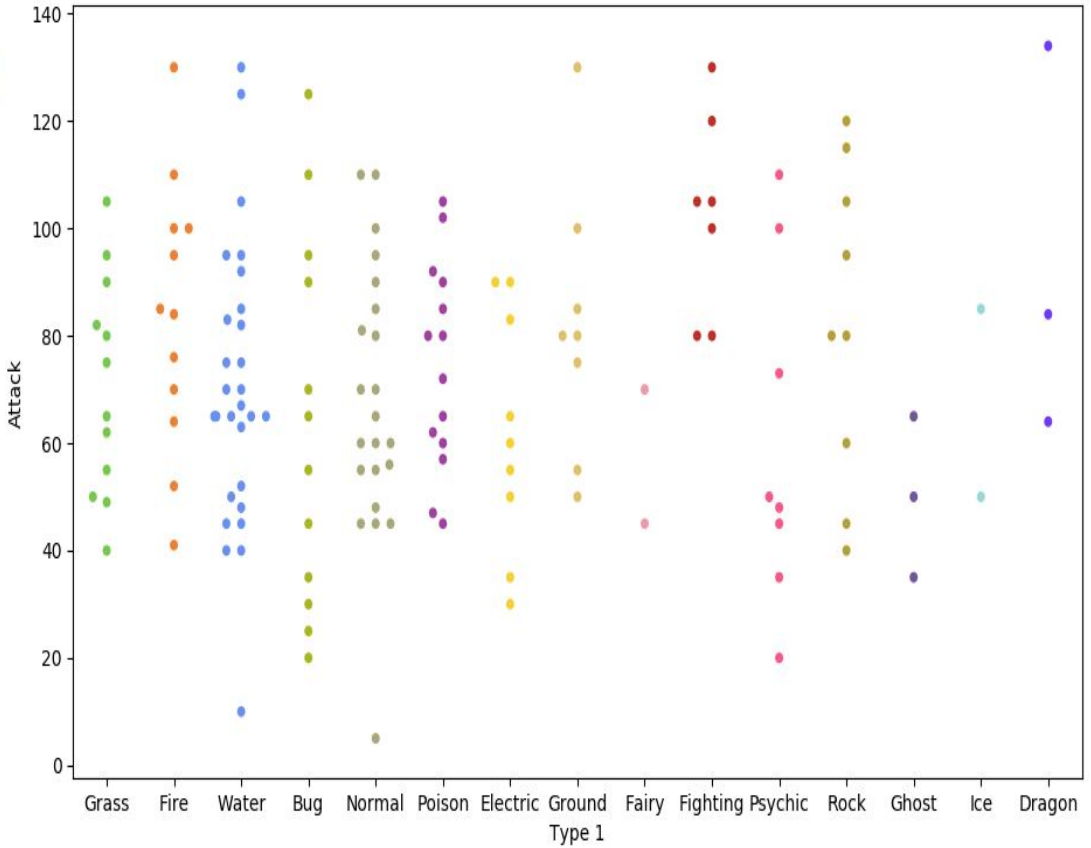
```
sns.violinplot(x='Type 1', y='Attack', data=df1,
               palette=type_colors)
```



Because of the limited number of observations, we could also use a swarm plot.

Here, each data point is an observation, but data points are grouped together by the variable listed on the x-axis.

```
sns.swarmplot(x='Type 1', y='Attack',
              data=df1,
              palette=type_colors)
```



Overlapping plots

Both of these show similar information, so it might be useful to overlap them.

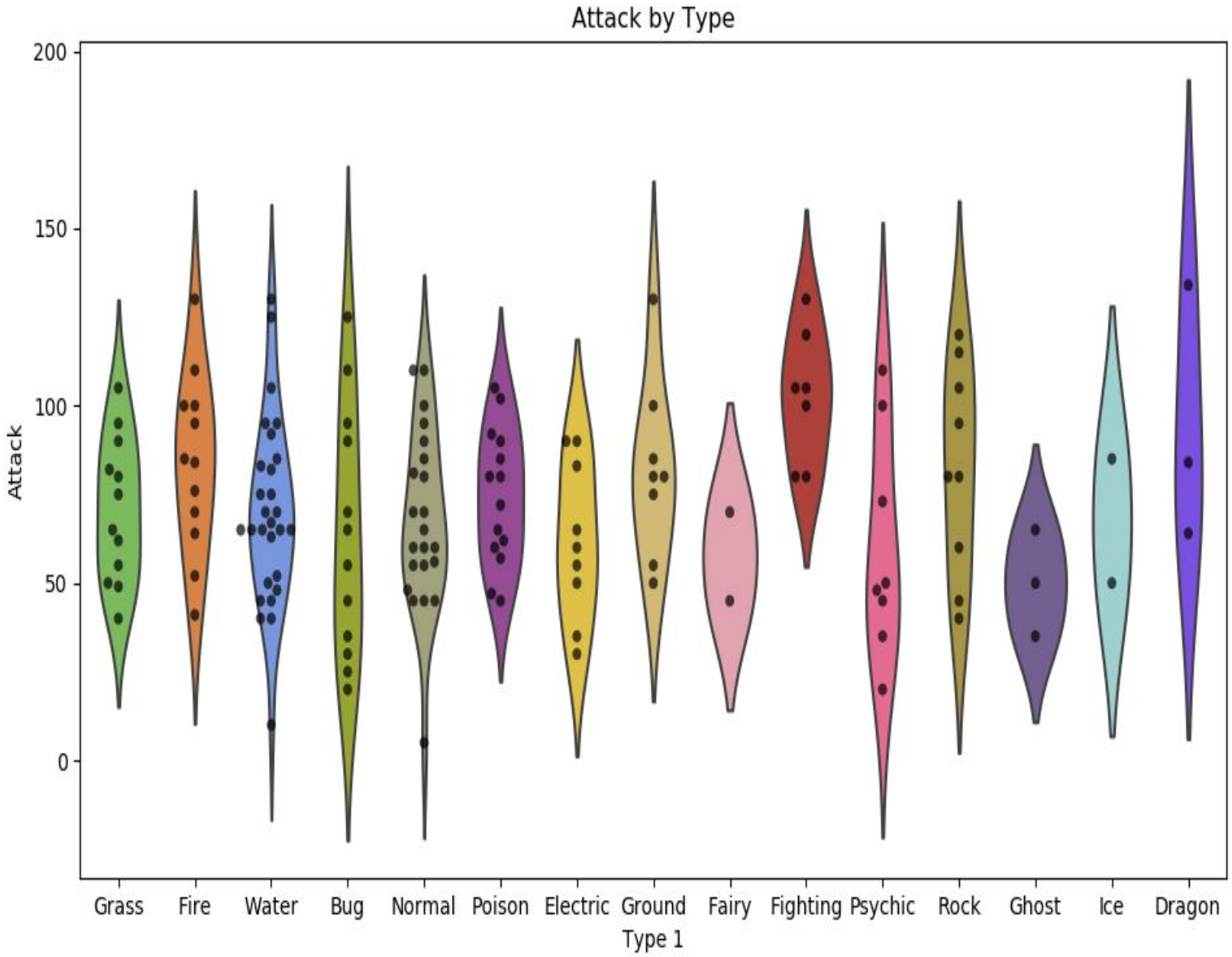
```
plt.figure(figsize=(10,6))
sns.violinplot(x='Type 1',
               y='Attack',
               data=df1,
               inner=None,
               palette=type_colors)
sns.swarmplot(x='Type 1',
              y='Attack',
              data=df1,
              color='k',
              alpha=0.7)
plt.title('Attack by Type')
```

Set size of print canvas.

Remove bars from inside the violins

Make bars black and slightly transparent

Give the graph a title


Attack by Type
Attack
200
150
100
50
0
Grass
Fire
Water
Bug
Normal
Poison
Electric
Ground
Fairy
Fighting
Psychic
Rock
Ghost
Ice
Dragon
Type 1

Data wrangling with Pandas

What if we wanted to create such a plot that included all of the other stats as well? In our current dataframe, all of the variables are in different columns:

```
print(df1.head())
```

```
        Name Type 1  Type 2  Total  ...  Sp. Def  Speed  Stage  Legendary
#                                   ...
1  Bulbasaur  Grass  Poison    318  ...       65     45      1      False
2    Ivysaur  Grass  Poison    405  ...       80     60      2      False
3   Venusaur  Grass  Poison    525  ...      100     80      3      False
4  Charmander   Fire     NaN    309  ...       50     65      1      False
5  Charmeleon   Fire     NaN    405  ...       65     80      2      False
```

If we want to visualise all stats, then we'll have to "melt" the dataframe.

```
stats_df = df1.drop(['Total', 'Stage', 'Legendary'], axis=1)
melted_df = pd.melt(stats_df,
                    id_vars=["Name", "Type 1", "Type 2"],
                    var_name="Stat")
print(melted_df.head())
```

We use the .drop() function again to re-create the dataframe without these three variables.

The dataframe we want to melt.

The variables to keep, all others will be melted.

A name for the new, melted, variable.

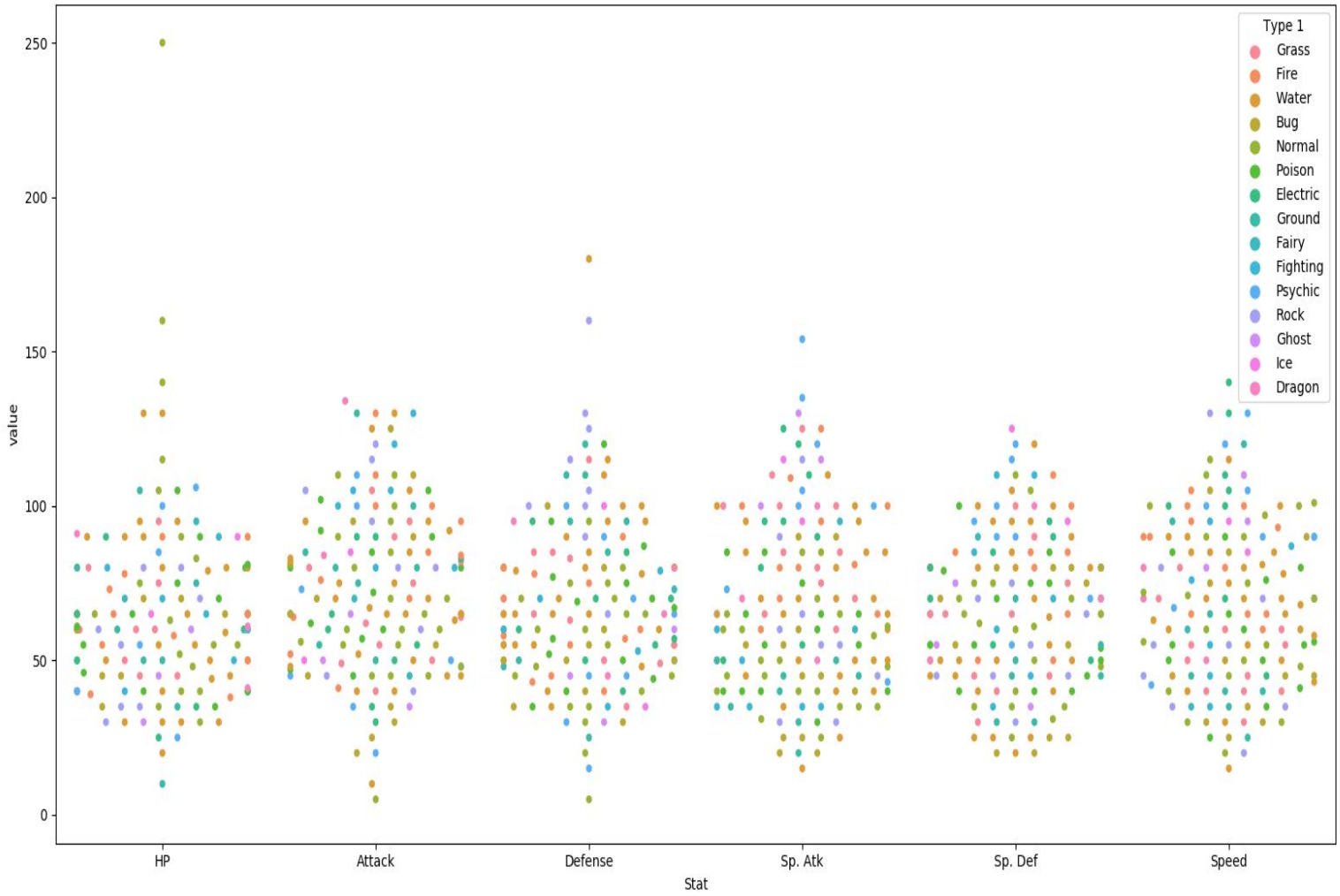
```
In [6]: runfile('C:/Users/lb690/Google Dri
pokemon_tutorial.py', wdir='C:/Users/lb690
          Name Type 1  Type 2 Stat  value
0    Bulbasaur  Grass  Poison   HP     45
1      Ivysaur  Grass  Poison   HP     60
2     Venusaur  Grass  Poison   HP     80
3   Charmander   Fire     NaN   HP     39
4   Charmeleon   Fire     NaN   HP     58
```

- All 6 of the stat columns have been "melted" into one, and the new Stat column indicates the original stat (HP, Attack, Defense, Sp. Attack, Sp. Defense, or Speed).
- It's hard to see here, but each pokemon now has 6 rows of data; hende the `melted_df` has 6 times more rows of data.

```
print( stats_df.shape )
print( melted_df.shape )
```

```
(151, 9)
(906, 5)
```

```
sns.swarmplot(x='Stat', y='value', data=melted_df,
hue='Type 1')
```



This graph could be made to look nicer with a few tweaks.

```
plt.figure(figsize=(10,6))
sns.swarmplot(x='Stat',
              y='value',
              data=melted_df,
              hue='Type 1',
              split=True,
              palette=type_colors)
plt.ylim(0, 260)
plt.legend(bbox_to_anchor=(1, 1), loc=2)
```

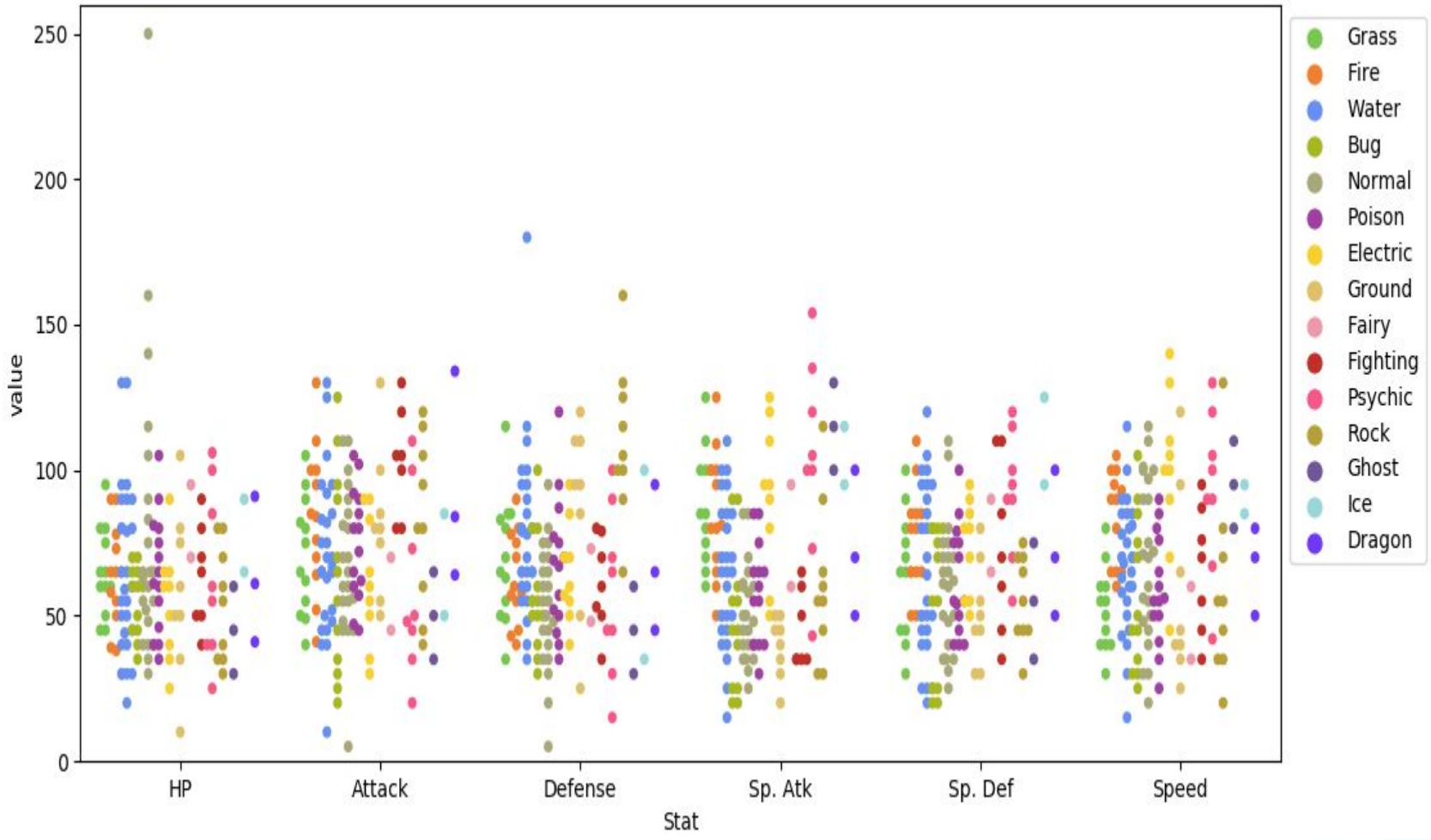
Enlarge the plot.

Separate points by hue.

Use our special Pokemon colour palette.

Adjust the y-axis.

Move the legend box outside of the graph and place to the right of it..


250
200
150
100
50
0
value
HP
Attack
Defense
Sp. Atk
Sp. Def
Speed
Stat
Grass
Fire
Water
Bug
Normal
Poison
Electric
Ground
Fairy
Fighting
Psychic
Rock
Ghost
Ice
Dragon

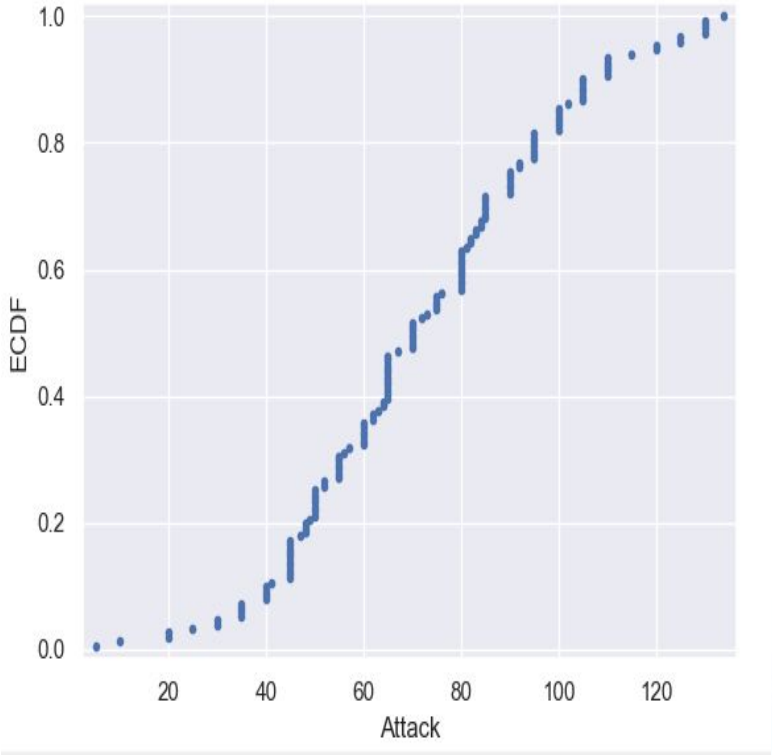
Plotting all data: Empirical cumulative distribution functions (ECDFs)

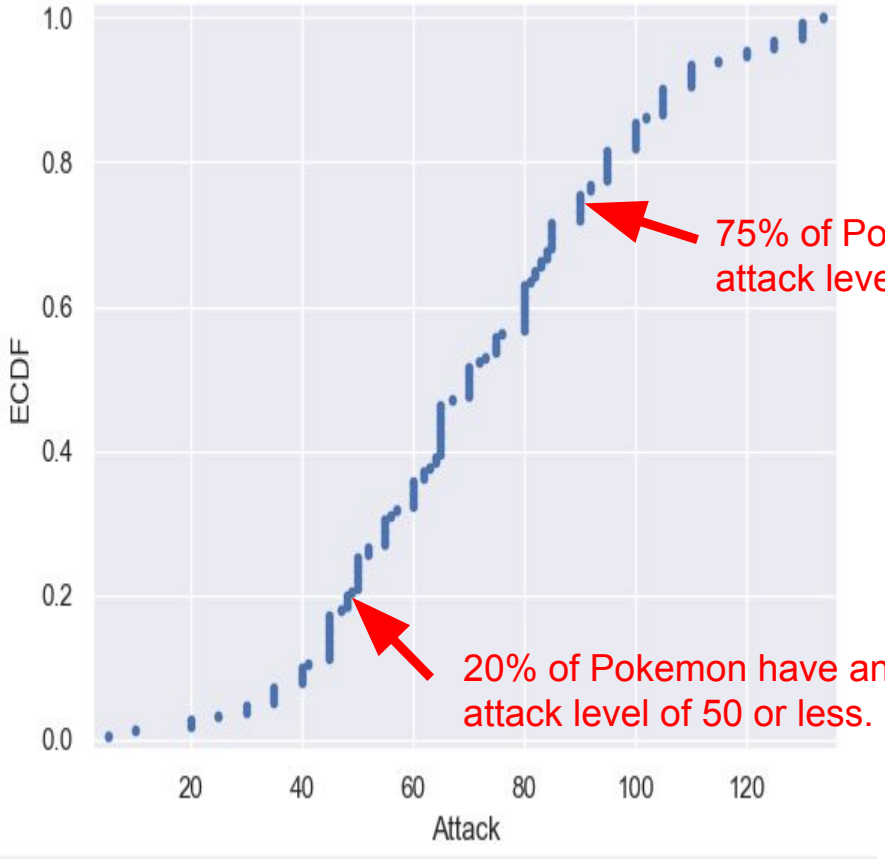
An alternative way of visualising a distribution of a variable in a large dataset is to use an ECDF.

Here we have an ECDF that shows the percentages of different attack strengths of pokemon.

An *x-value* of an ECDF is the quantity you are measuring; i.e. attacks strength.

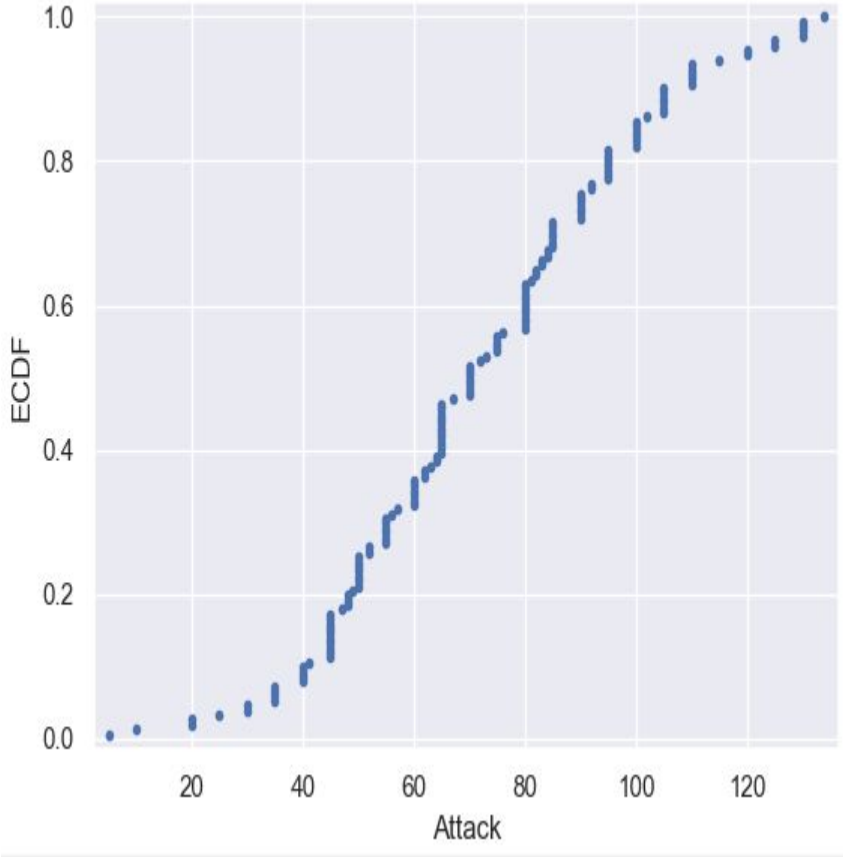
The *y-value* is the fraction of data points that have a value smaller than the corresponding x-value. For example…




1.0
0.8
0.6
0.4
0.2
0.0
ECDF
20
40
60
80
100
120
Attack
75% of Pokemon have an attack level of 90 or less
20% of Pokemon have an attack level of 50 or less.

Plotting an ECDF

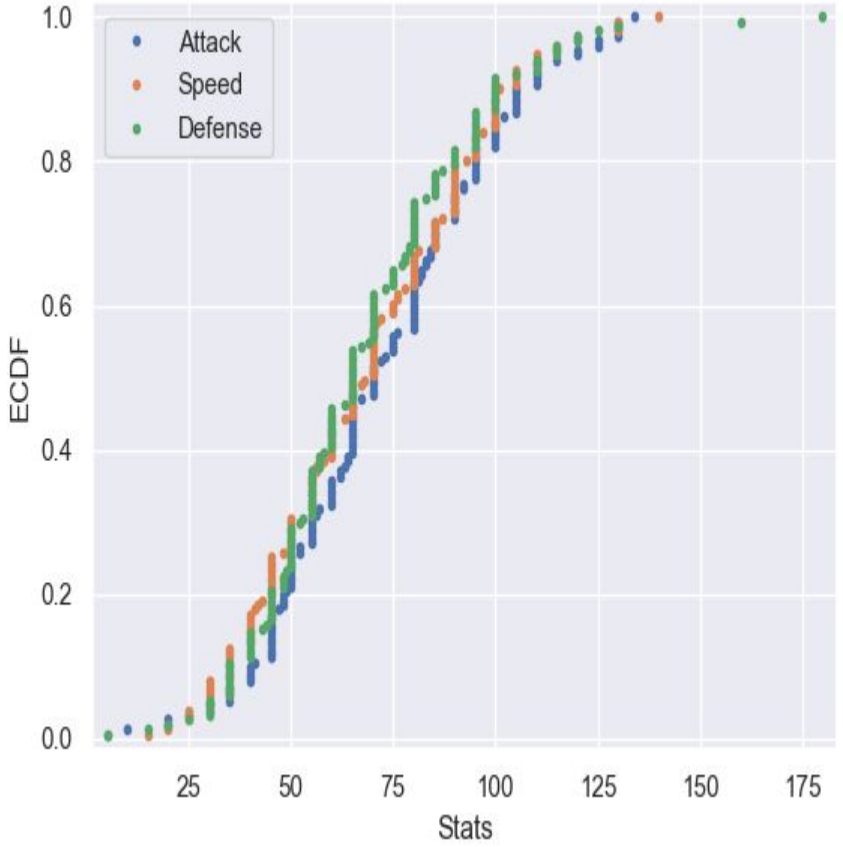
```
x = np.sort(df1['Attack'])
y=np.arange(1, len(x)+1)/len(x)
g = plt.plot(x, y, marker='.', linestyle='none')
g = plt.xlabel('Attack')
g = plt.ylabel('ECDF')
plt.margins(0.02)
plt.show()
```



You can also plot multiple ECDFs on the same plot.

As an example, here with have an ECDF for Pokemon attack, speed, and defence levels.

We can see here that defence levels tend to be a little less than the other two.



The usefulness of ECDFs

It is often quite useful to plot the ECDF first as part of your workflow.

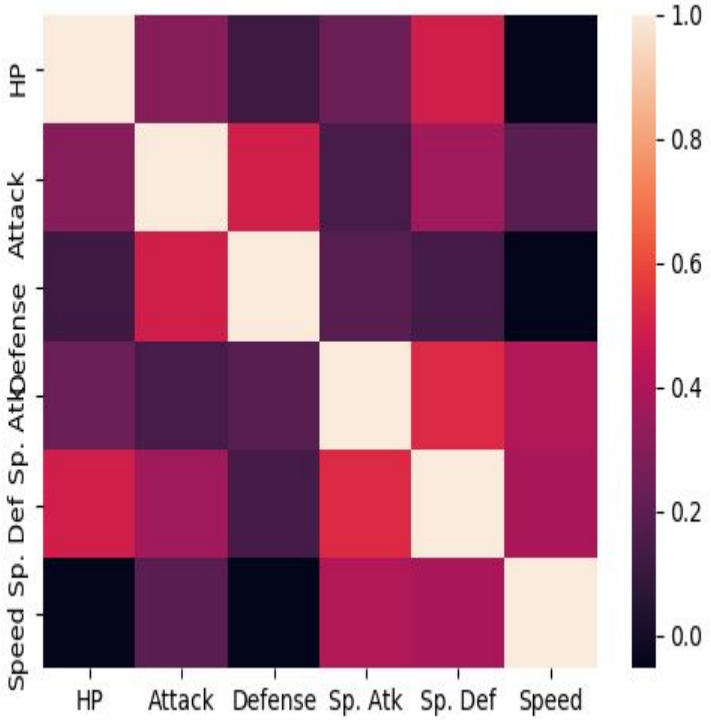
It shows all the data and gives a complete picture as to how the data are distributed.

Heatmaps

Useful for visualising matrix-like data.

Here, we'll plot the correlation of the stats_df variables

```
corr = stats_df.corr()
sns.heatmap(corr)
```

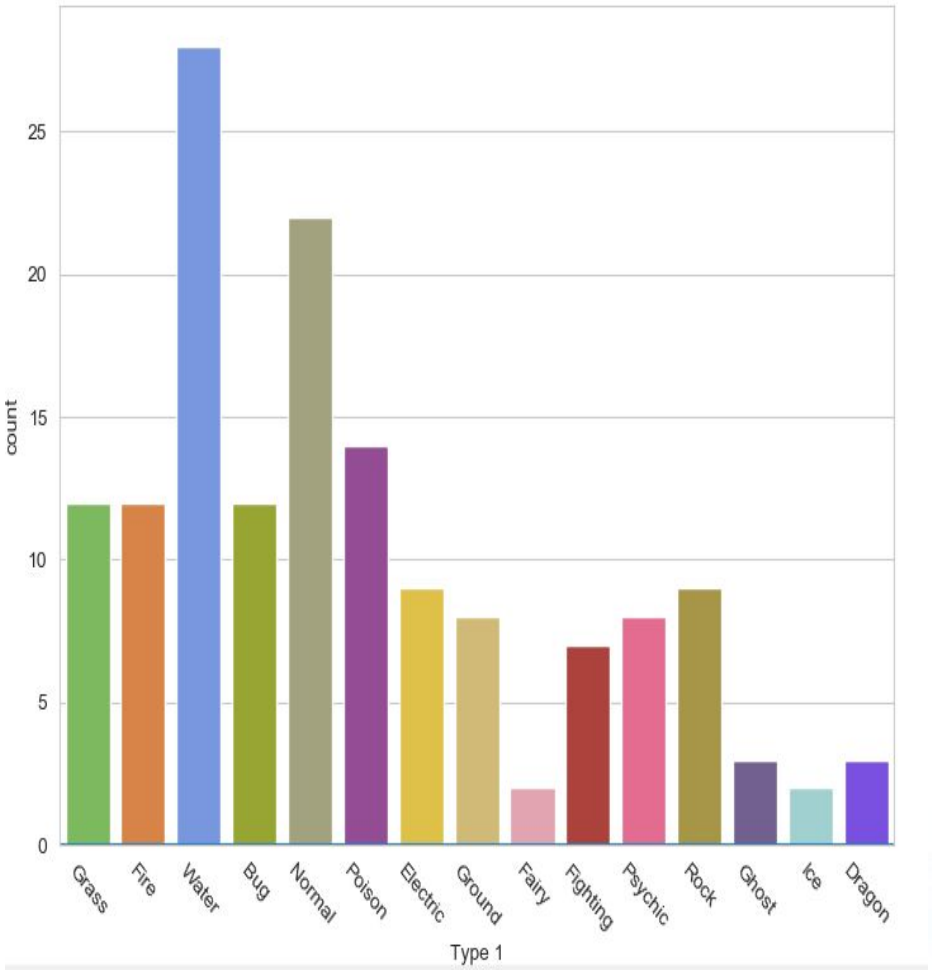


Bar plot

Visualises the distributions of categorical variables.

```
sns.countplot(x='Type 1', data=df1,
              palette=type_colors)
plt.xticks(rotation=-45)
```

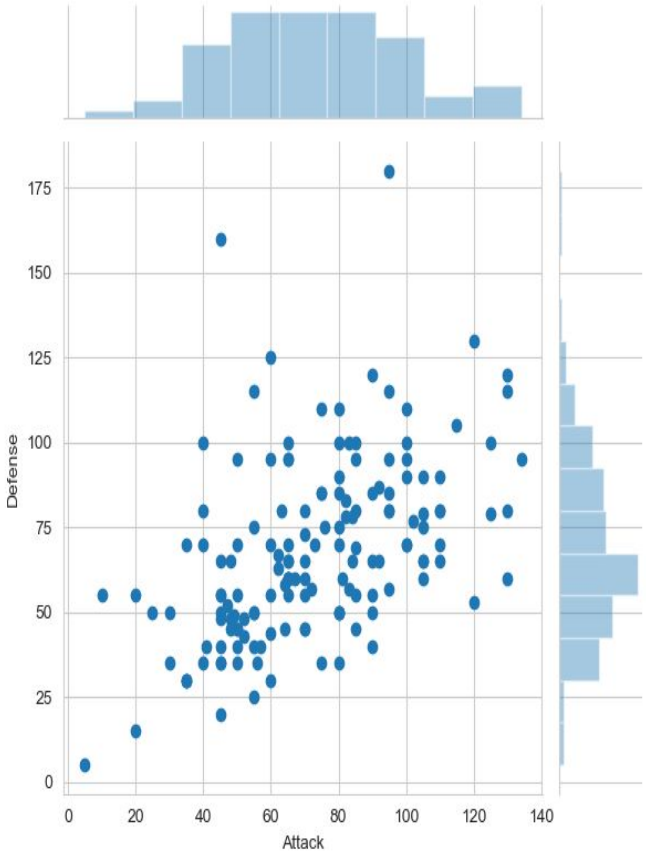
Rotates the x-ticks 45 degrees



Joint Distribution Plot

Joint distribution plots combine information from scatter plots and histograms to give you detailed information for bi-variate distributions.

```
sns.jointplot(x='Attack',
              y='Defense',
              data=df1)
```



Introduction of Python Framework

Flask - Flask is a microframework for Python. The main purpose is to develop a strong web application base. As compared to Django, Flask is best suited for small and easy projects.

Django - Django is a high-level Python Web application development framework that encourages us to develop things rapidly. It's free and open source.

Web2Py - It is a free open source full-stack development framework in python which allows the user to develop things quickly. It is a cross-platform framework that supports all popular operating systems.

Commonly used Flash Framework

Comparison between Django & Flask

	django	Flask
Type of Framework	Full Stack Web Framework	WSGI framework
Flexibility	Feature-packed	Full flexibility
ORM Usage	Built-in ORM	SQLAlchemy is used
Design	Batteries-included	Minimalistic design
Working Style	Monolithic	Diversified

What is Dash / Python Flask Framework?

- **Dash** is **Python framework** for building web applications. It is open source, and its app run on the web browser

- It built on top of **Flask**, Plotly.js, React and React Js.

- It enables you to build dashboards using pure **Python**.

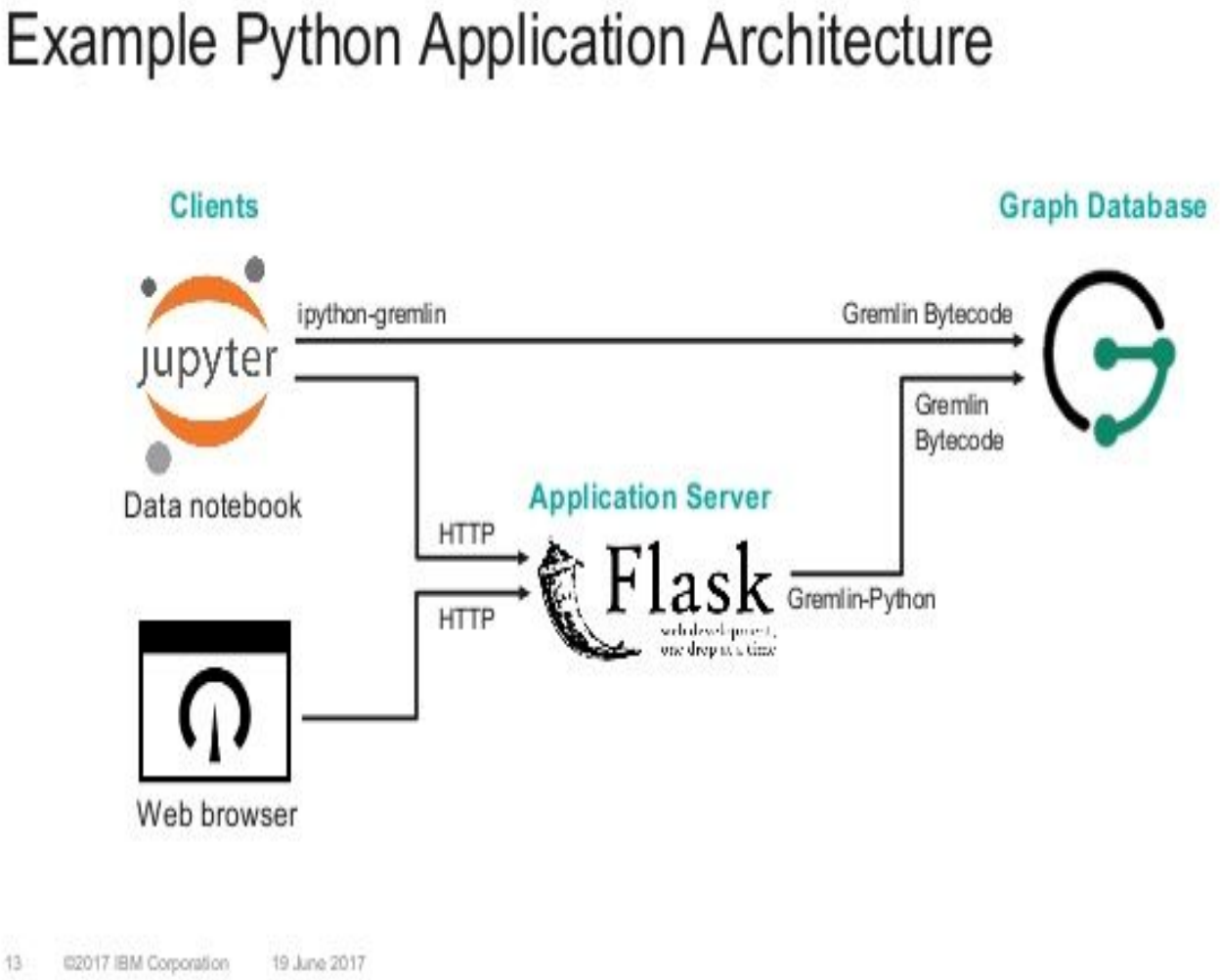
What is Dash / Python Flask Framework?

Every Dash App is composed of two main parts:

Layout = is used to define how all the content should be laid out on the application.

Callbacks = are used to make each of the desired components of the Dashboard interactive.

Python Flask Framework Architecture



What is a Project?

"**Unique process** consisting of a set of coordinated and controlled **activities** with **start** and **finish** dates, undertaken to achieve an objective conforming to specific requirements, including constraints of **time**, **cost**, **quality** and **resources**"

A Project is a planned set of activities
A Project has a scope
A Project has time, cost, quality and resource constraints

What is Project Management?

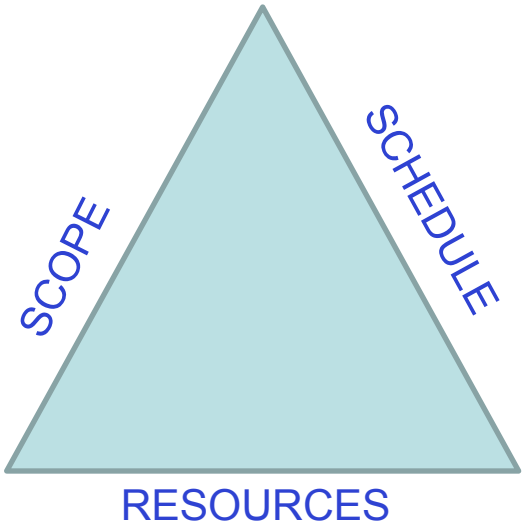
The art of organising, leading, reporting and completing a project through people

What is Project Management?

A project is a planned undertaking
A project manager is a person who causes things to happen
Therefore, project management is causing a planned undertaking to happen.

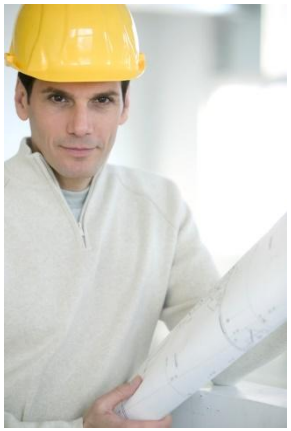
Project Management
The "Triple Constraint"

- The Project Manager / the researcher has certain constraints to battle with in delivering a project. These are referred to as "The Triple Constraint". They include:
  - Scope Management
  - Cost Management
  - Time Management
  - Quality Assurance
  - Human Resource Management
  - Communication Management
  - Risk Management
  - Conflict Management among others.



PROJECT CONSTRAINT TRIANGLE

A Good Project Manager



- Takes ownership of the whole project
- Is proactive not reactive
- Adequately plans the project
- Is Authoritative (**NOT** Authoritarian)
- Is Decisive
- Is a Good Communicator
- Manages by data and facts not uniformed optimism
- Leads by example
- Has sound Judgement
- Is a Motivator
- Is Diplomatic
- Can Delegate

Stakeholder Engagement process

Identify Stakeholders

Assess needs

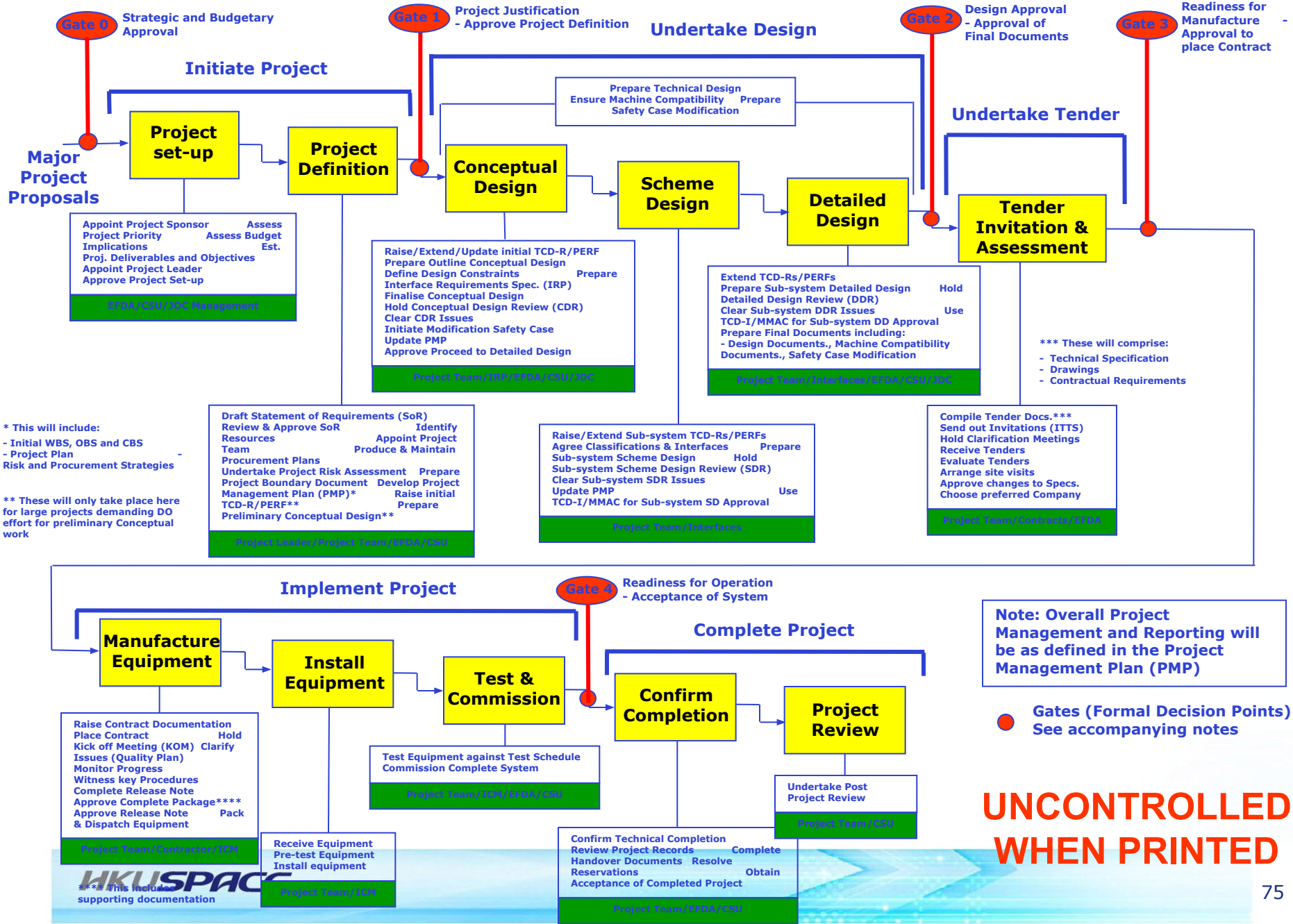
Define actions

Establish communication channels

Gather feedback

Monitor and review

The Project Process



Key Points in Project Set-up and Definition

- Create Project Management Plan (PMP)
- Be clear of scope and objectives
- Establish clear statement of what is to be done (WBS)
- Establish Risks to be Managed
- Establish Costs and Durations
- Establish Resources Required

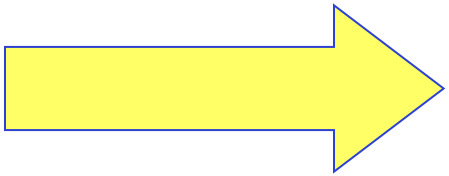
Project management Plan - PMP

- Master Document for Project
- Defines the following:-
  - Project Objectives, Scope, Deliverables
  - Stakeholders (Internal & External)
  - Work to be done (WBS)
  - Project Organisation and Resources (OBS)
  - Project Costings (CBS)
  - Project Schedule
  - Procurement/Contract Strategy
  - Risk Management
  - Quality management
  - Change Management

Project Planning

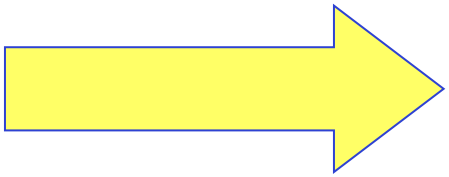
Project Planning

Adequate planning leads to the correct completion of work



Plan

Inadequate planning leads to frustration towards the end of the project & poor project performance



Project Start

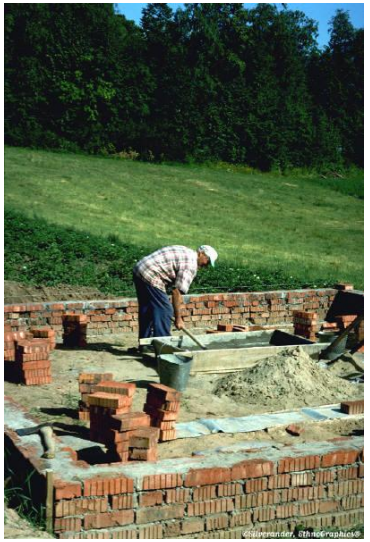
Project End

Work Breakdown Structure (WBS)

The Work Breakdown Structure is the foundation for effective project planning, costing and management.

It is the most important aspect in setting-up a Project

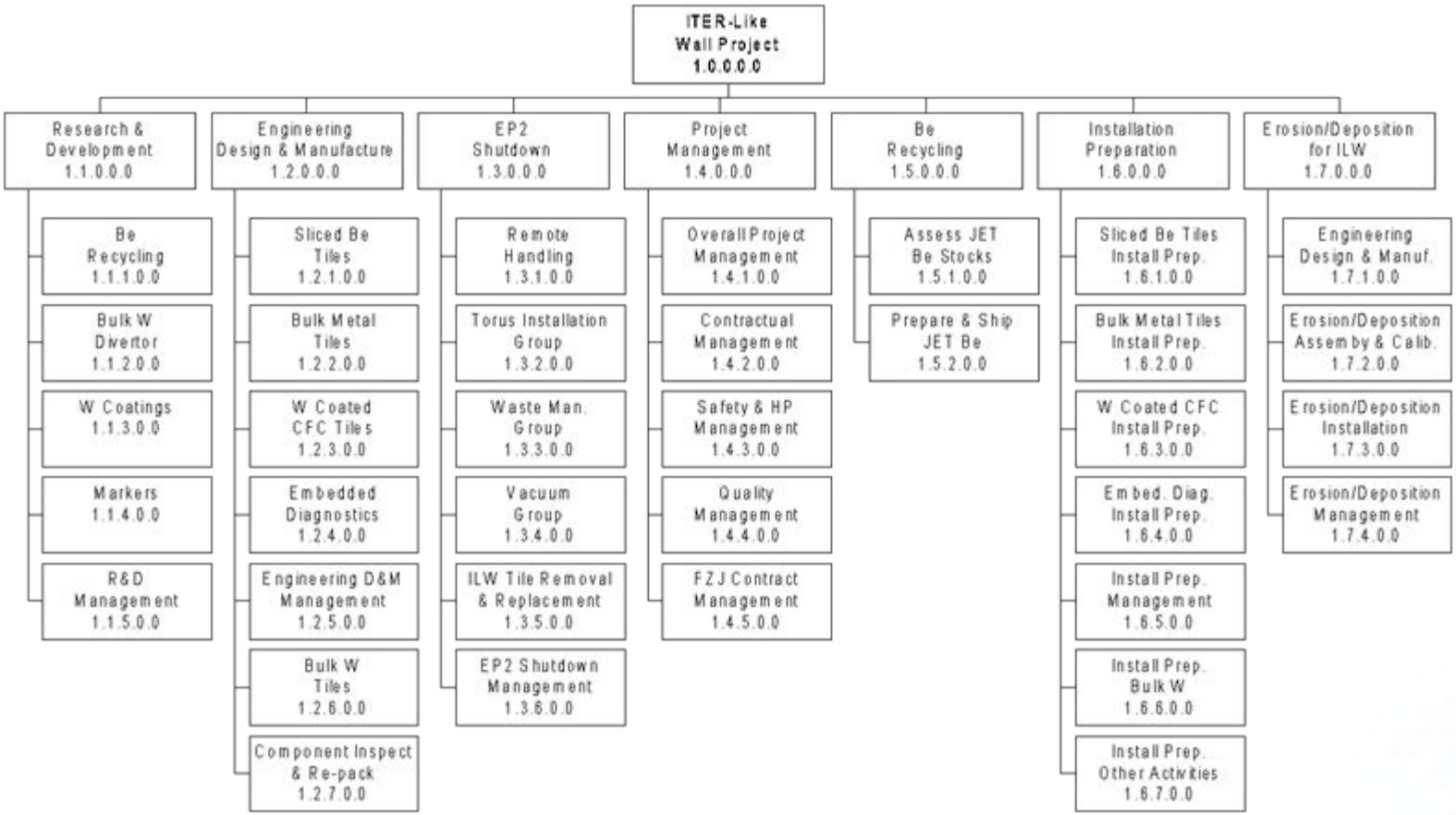
- It is the foundation on which everything else builds



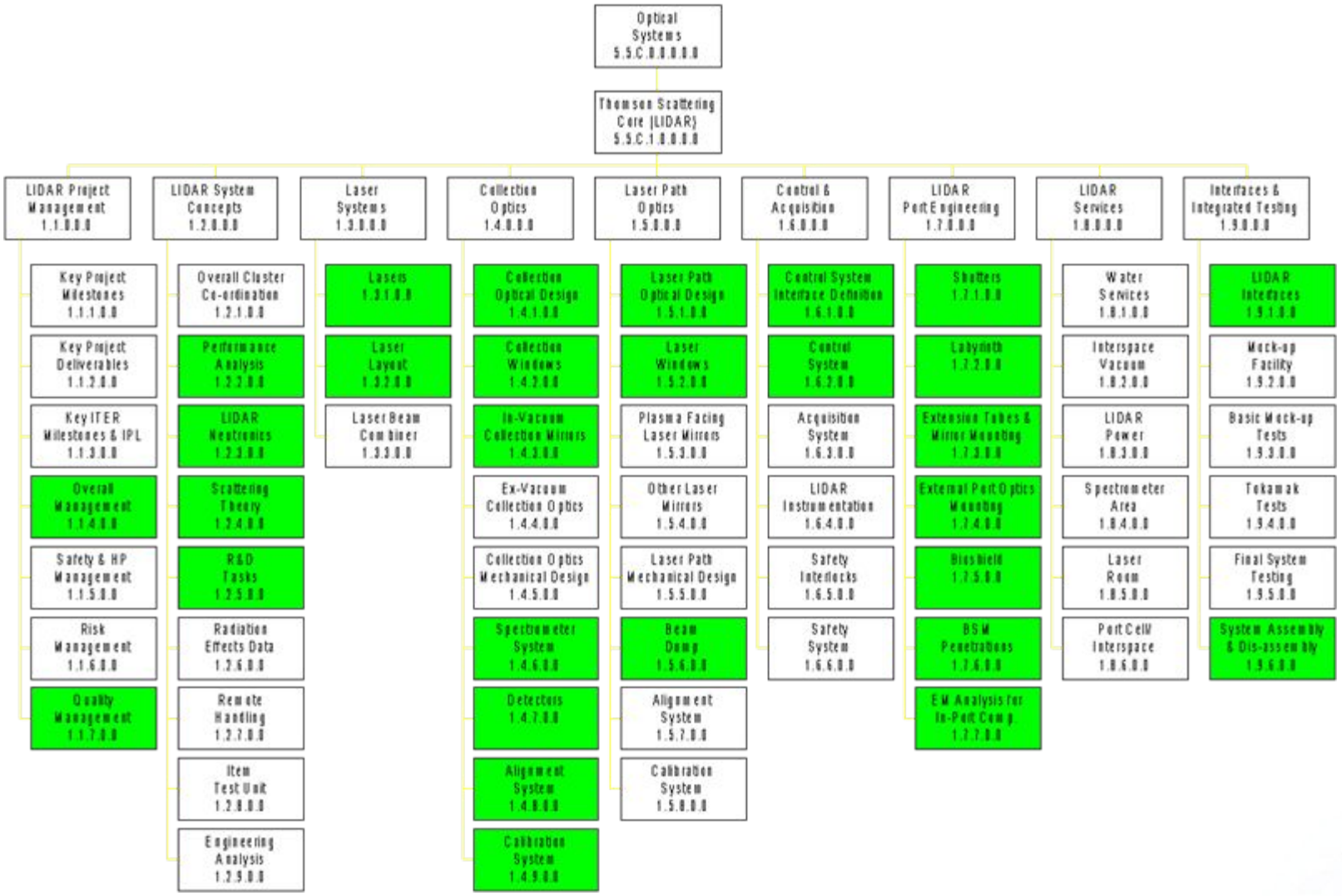
Work Breakdown Structure - Definition

"A Work Breakdown Structure (WBS) is a hierarchical (from general to specific) tree structure of deliverables and tasks that need to be performed to complete a project."

Example WBS - Top Level ILW Project



Example WBS - Top Level TSCL Project



How to Manage Your Budget with a Simple Python Script

Open Jupyter notebook

1. Getting User Input

```
import pandas as pd
import matplotlib.pyplot as plt

def get_user_input():
    """Get user input for income and expenses."""
    income = float(input("Enter your income: "))
    # Expecting expenses to be a dictionary input
    expenses = {}
    while True:
      category = input("Enter expense category (or 'done' to finish): ")
      if category.lower() == 'done':
            break
      amount = float(input(f"Enter amount for {category}: "))
      expenses[category] = amount
    return income, expenses
```

How to Manage Your Budget with a Simple Python Script

2. Calculating the Budget

```
def calculate_budget(income, expenses):
    """Calculate total expenses and balance."""
    total_expenses = sum(expenses.values())
    balance = income - total_expenses
    return total_expenses, balance
```

3. Displaying the Budget Summary

```
def display_budget_summary(income, total_expenses, balance):
    """Display the budget summary."""
    print(f"Income: ${income:.2f}")
    print(f"Total Expenses: ${total_expenses:.2f}")
    print(f"Balance: ${balance:.2f}")
```

How to Manage Your Budget with a Simple Python Script

4. Plotting the Expenses

```
def plot_expenses(expenses):
    """Plot the expenses as a bar chart."""
    df = pd.DataFrame(list(expenses.items()), columns=['Category', 'Amount'])
    df.plot(kind='bar', x='Category', y='Amount', legend=False)
    plt.ylabel('Amount ($)')
    plt.title('Expense Distribution')
    plt.show()
```

How to Manage Your Budget with a Simple Python Script

5. Main Function

```
income, expenses = get_user_input()
total_expenses, balance = calculate_budget(income, expenses)
display_budget_summary(income, total_expenses, balance)
plot_expenses(expenses)
```

How to Manage Your Budget with a Simple Python Script

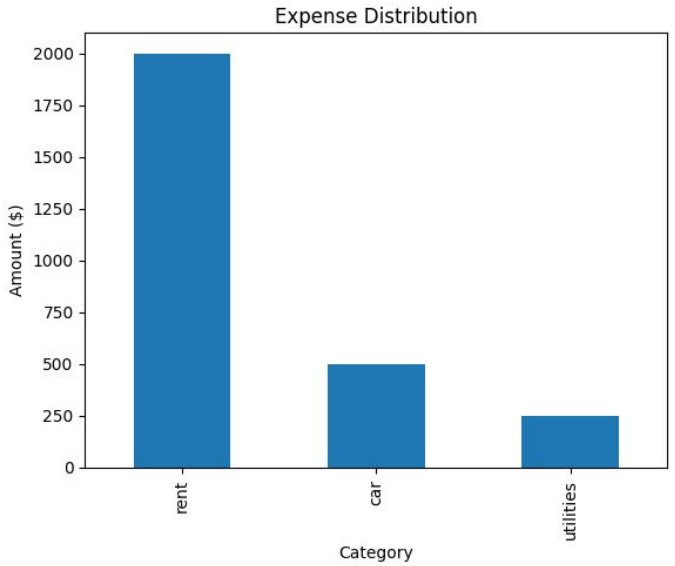
5. Main Function

Main Input

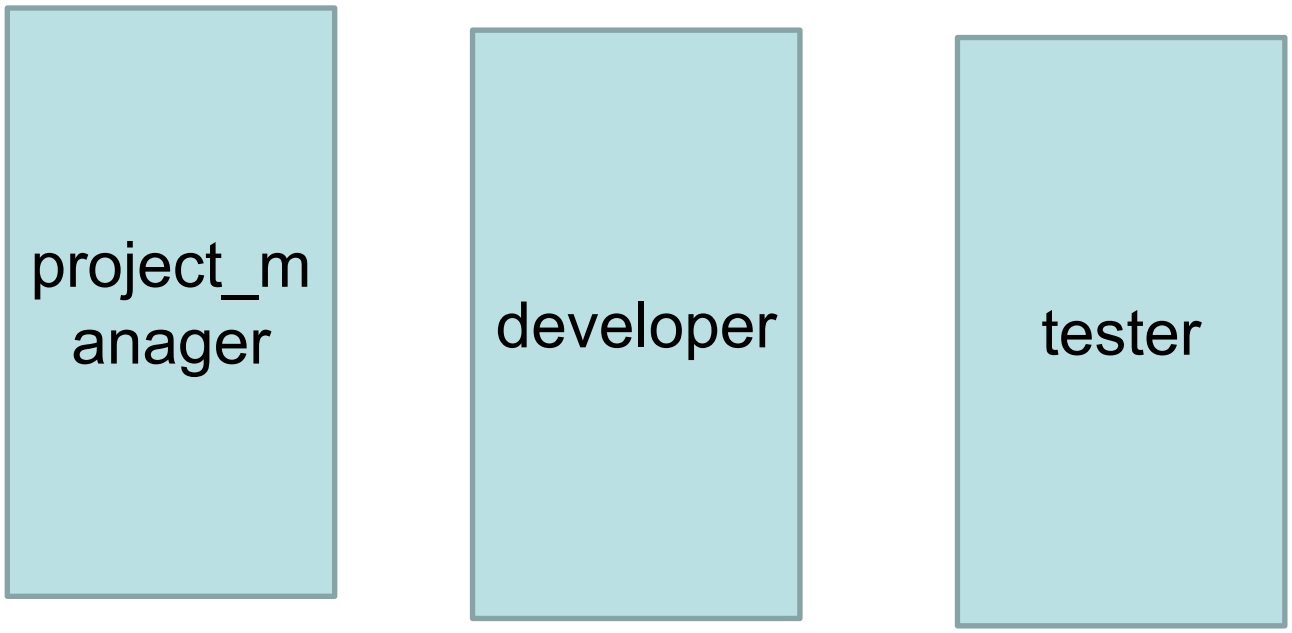
Enter your income: 10000
Enter expense category (or 'done' to finish): rent
Enter amount for rent: 2000
Enter expense category (or 'done' to finish): car
Enter amount for car: 500
Enter expense category (or 'done' to finish): utilities
Enter amount for utilities: 250
Enter expense category (or 'done' to finish): done

Result

Income: $10000.00
Total Expenses: $2750.00
Balance: $7250.00



Python PM Tutorial 1 – Set 3 Roles and Permission



Python PM Tutorial 1 – Set 3 Roles and Permission

```
class Project:
    def __init__(self, name):
        self.name = name
        self.tasks = []

    def add_task(self, task_name):
        self.tasks.append(task_name)
        print(f"Task '{task_name}' added to project '{self.name}'.")

    def view_tasks(self):
        print(f"Tasks in project '{self.name}':")
        for task in self.tasks:
            print(f"- {task}")

class Role:
    def __init__(self, name, permissions):
        self.name = name
        self.permissions = permissions
    def has_permission(self, permission):
        return permission in self.permissions
```

Python PM Tutorial 1 – Set 3 Roles and Permission

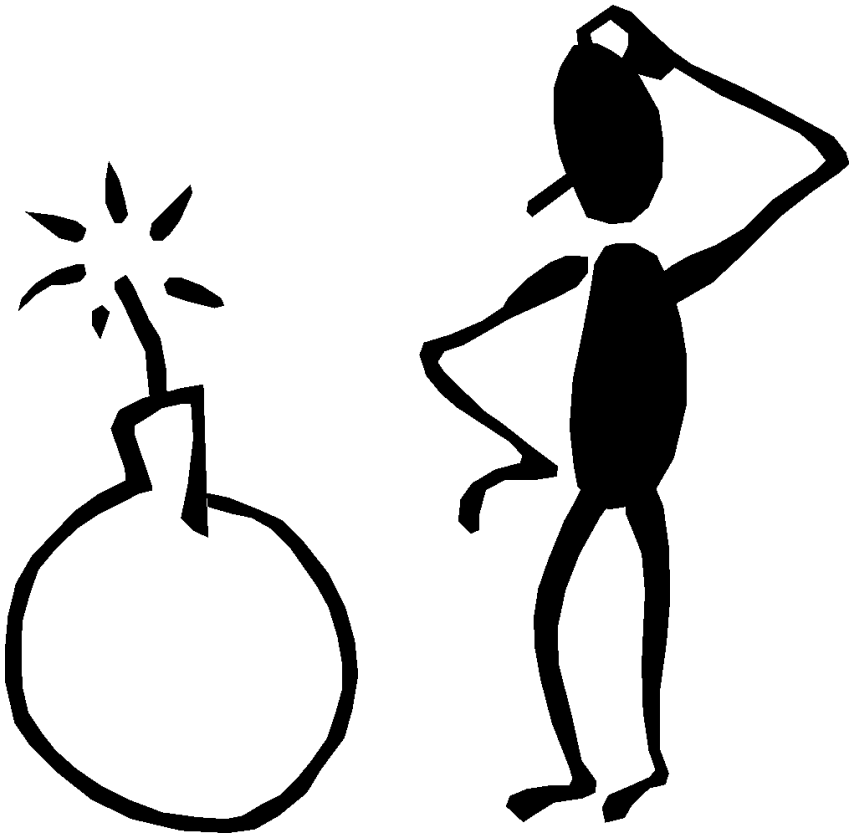
```
class User:
    def __init__(self, username, role, project):
            self.username = username
            self.role = role
            self.project = project

    def perform_action(self, action, task_name=None):
            if self.role.has_permission(action):
                if action == "add_task":
                  self.project.add_task(task_name)
                elif action == "view_tasks":
                  self.project.view_tasks()
                else:
                  print("Action not supported.")
            else:
                  print(f"User '{self.username}' does not have permission to '{action}'.")
```

Python PM Tutorial 1 – Set 3 Roles and Permission

Main Program

```
# Define roles and their permissions
developer_permissions = ["view_tasks", "add_task"] # Can view and add tasks
tester_permissions = ["view_tasks"]  # Can only view tasks
project_manager_permissions = ["view_tasks", "add_task"] # Can view and add tasks

developer_role = Role("Developer", developer_permissions)
tester_role = Role("Tester", tester_permissions)
project_manager_role = Role("Project Manager", project_manager_permissions)

# Create a project
my_project = Project("Awesome Project")
# Create users and assign roles and the project
john = User("john_doe", developer_role, my_project)
jane = User("jane_smith", tester_role, my_project)
peter = User("peter_jones", project_manager_role, my_project)

john.perform_action("add_task", "Implement user authentication")
jane.perform_action("view_tasks")
peter.perform_action("add_task", "Design database schema")
john.perform_action("view_tasks")
jane.perform_action("add_task", "Write unit tests") # This will fail, as the tester doesn't have permission to add tasks
```

Project Risk Management



Project Risk – Definition

"Project risk is an uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective"

"A combination of the probability of a defined **threat** or **opportunity (Likelihood)** and the magnitude of the consequences of the occurrence (Impact) defines a Risk Index"

Tasks	Threat	Reason
Data Collection	Delay 2 days	Business Scope Always changing
Data Cleaning	Delay 2 days	Business Scope Always changing
Machine Learning Model	Delay 0 days	Nil

Risk Impact

Threat → Scope → **Poor Quality Product**
Threat → Schedule → **Late Delivery**
Threat → Cost → **Overspend**

In addition there are health, safety and environmental threats that **must** be managed

Identify Risks
Assess likelihood and impact
Rank risks and prioritize
Define risk management approach & actions
Implement actions
Monitor & review

Make the management of risk integral to the way the project is managed
Ensure that cost and time contingencies are consistent with identified risks
Focus on the “significant few” – don’t try to manage too many risks
Be vigilant (be ready for possible problems and diffuculties)and proactive

Project Monitoring and Control

Project Monitoring

Typical Monitoring Activities

- regular reviews of progress against schedule using WBS as basis (Plan against Baseline)
- regular review of actual costs (O/P from SAP) against budgeted costs and Earned Value at WBS level
- regular review of resource loading
- regular progress meetings with project team
- regular meetings with contractors
- production of periodic progress reports
- risk reviews
- inspections/ audits

Project Control

Typical Control Activities

- assign responsibilities at Work Package level
- staged authorisation of work to be done
- staged release of budgets (staged release of WBS(e) numbers)
- ensure PM has a ‘Management Reserve’ under his control
- seek corrective action reports when WPs go ‘off track’ (overrunning or overspending)
- release Management Reserve carefully

Project Monitoring and Control Summary

Monitor against the plan – status regularly
Take a factual approach to decisions
Identify management action early
Check that defined controls are being applied – correct if necessary
Apply change control

Automation on Stakeholder Engagement process

1. **Identify Stakeholders**
   - Mapping Stakeholders: Begin by identifying all relevant stakeholders, both internal (employees, management) and external (customers, partners, investors). Create a comprehensive stakeholder map to visualize their roles and influence.
   - Prioritization: Classify stakeholders based on their level of interest and influence. This will help prioritize engagement efforts and tailor strategies accordingly
2. **Assess Needs**
   - Conduct Surveys and Interviews: Gather information on stakeholders' needs, expectations, and concerns through surveys or one-on-one interviews. This qualitative data is essential for understanding what stakeholders value most.
   - Analyze Data: Use analytics tools to interpret the data collected. Identify common themes and specific requirements that can guide your engagement strategy.
3. **Define Actions**
   - Develop Engagement Strategies: Based on the assessed needs, define specific actions tailored to each stakeholder group. This could include personalized communication plans, involvement in decision-making processes, or targeted information sessions.
   - Set Clear Goals: Establish measurable objectives for each action to ensure accountability and track progress over time.

Automation on Stakeholder Engagement process

4. Establish Communication Channels

- Choose Appropriate Platforms: Select the most effective communication channels for engaging with different stakeholders. Options may include email newsletters, social media updates, webinars, or dedicated project management tools
- Automate Communication: Utilize automation tools to streamline communication efforts. For example, set up automated email campaigns or notifications to keep stakeholders informed without manual effort

5. Gather Feedback

- Implement Feedback Mechanisms: Create structured methods for collecting feedback from stakeholders after key interactions or milestones. This could involve follow-up surveys or feedback forms.
- Encourage Open Dialogue: Foster an environment where stakeholders feel comfortable sharing their thoughts and concerns. Use tools like online forums or chatbots to facilitate ongoing conversations.

6. Monitor and Review

- Track Engagement Metrics: Regularly monitor key performance indicators (KPIs) related to stakeholder engagement, such as response rates, satisfaction levels, and participation in events.
- Evaluate and Adjust Strategies: Conduct periodic reviews of your engagement strategies based on the feedback received and the metrics tracked. Make necessary adjustments to improve effectiveness and ensure alignment with stakeholder needs

Methodologies in Project Management

1. Waterfall Project Management
   - Waterfall project management follows a linear and sequential process where each phase must be completed before the next begins. It suits projects with fixed requirements and clear objectives.
2. Agile Project Management
   - Agile project management is an iterative approach that emphasizes flexibility, customer collaboration, and rapid delivery through short cycles called sprints, allowing teams to adapt quickly to changing requirements
3. Hybrid Project Management
   - Hybrid project management combines Agile and Waterfall methodologies, allowing teams to use structured planning while maintaining flexibility for iterative development, making it ideal for projects needing both predictability and adaptability.
4. Other Notable Methodologies
   - Other methodologies include Scrum, an Agile framework focusing on short cycles and team roles, and Lean, which maximizes value by minimizing waste and improving processes throughout the project lifecycle.

Successful Automation Projects: Case Studies

- **UiPath Automation Case Studies**

1. Enerjisa (Energy): Improved internal processes and customer service, leading to enhanced operational efficiency and customer satisfaction.
2. Suncoast Credit Union (Financial Services): Utilized AI and automation to improve member trust and accelerate growth, streamlining banking processes.
3. MongoDB (Technology): Saved over 150,000 working hours and $1.5 million by automating time-consuming data management tasks.

- **Kawasaki Robotics Case Studies**

1. German Brewery (Food & Beverage): Implemented robotic palletizing systems to reduce labor costs and increase production capacity.
2. Automated Bottle Production Line (Food & Beverage): Enhanced material handling and palletizing processes, resulting in streamlined operations and reduced production times.

- **Itransition RPA Use Cases**

1. Heritage Bank (Banking): Automated 80 processes across departments, significantly improving operational efficiency and reducing customer interaction processing times.
2. Thermo Fisher Scientific (Biotechnology): Achieved a 70% reduction in invoice processing time for 824,000 invoices annually, enhancing accuracy in financial operations.

Successful Automation Projects: Case Studies

- **Nividous RPA Case Studies**

1. Manufacturer Invoice Processing: Automated data extraction from invoices, saving over $90,000 annually and speeding up processing times
2. Healthcare Patient Claims Processing: Reduced handling time by 70% through automation of data extraction and claims submission, improving cash flow management.

- **Industrial Automation Insights**

1. JR Automation & Advanced Drainage Systems (Manufacturing): Developed a flexible pipe sorting system that improved operational efficiency while reducing manual labor needs.
2. Snowboard Manufacturing Digital Transformation: Streamlined manufacturing processes using automation technologies, increasing productivity and reducing time-to-market.

Tools for doing Project Management Plan

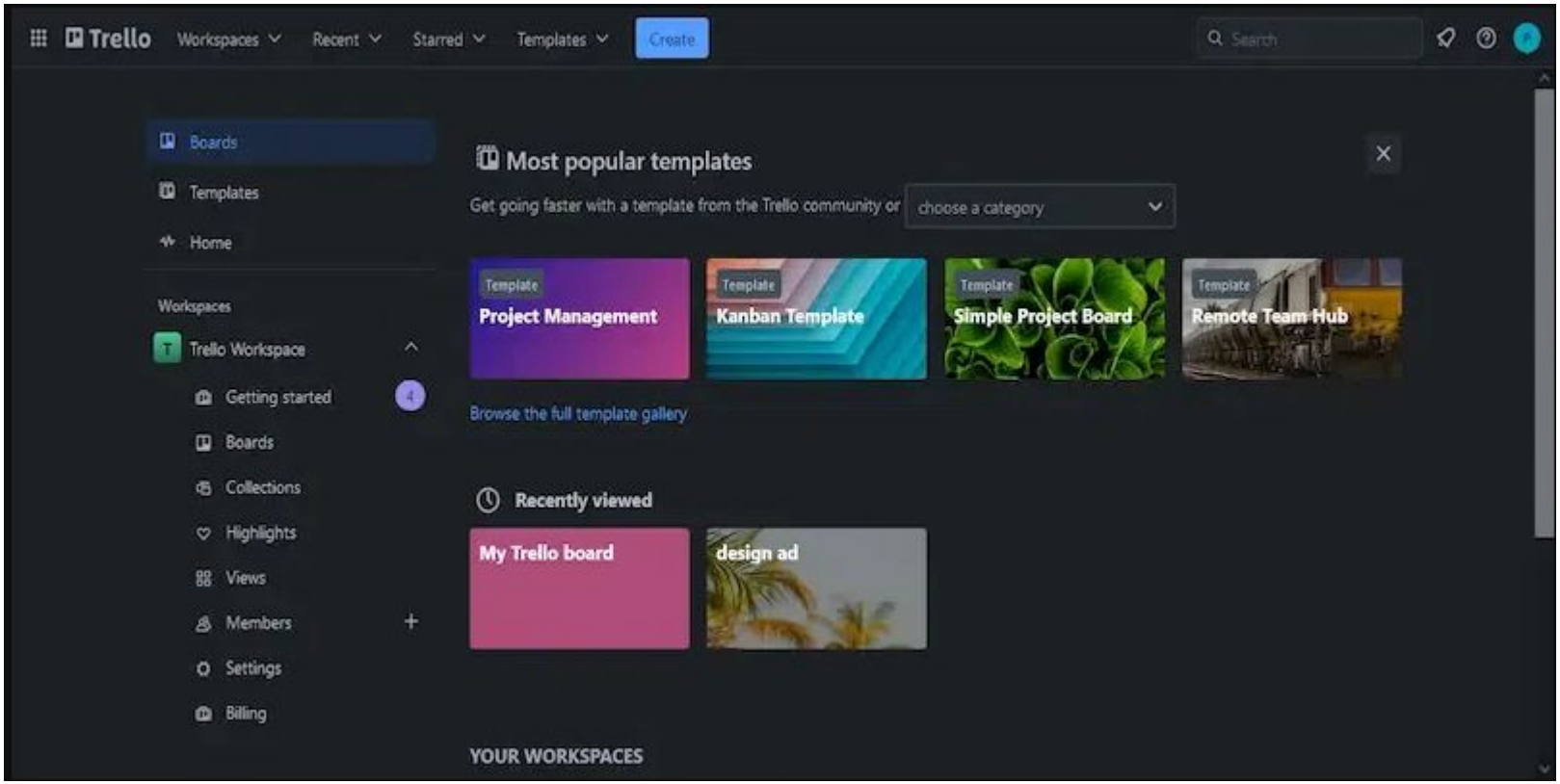
1. Gantt Charts: visualize timelines, track progress, and manage tasks effectively.
2. Asana: Offers flexible task management with various visualization options like lists, Kanban boards, and Gantt charts
3. Jiro : Jira is amongst the most established, powerful, and feature-complete project management tools available. It is straightforward to set up and available both on the cloud and on premises.
4. Trello: A visual tool that organizes projects into boards and cards, ideal for agile workflows
5. Notion: Maintaining an internal database, collaborating, and managing tasks such as Creating project plans, Building roadmaps, Creating and storing important documents, guides, etc.
6. Monday.com : monday.com is a highly visual and intuitive project management tool that offers a range of customization options.
7. Wrike: Features task assignment, collaboration tools, and multiple viewing formats (list, board, Gantt) to manage workflows efficiently

10 Ways to Use Trello for Project Management

1. Use Trello boards
2. Using Trello Cards
3. Calendar Planning
4. Email Replacement
5. Productivity Metrics
6. Automate Repetitive Email Tasks
7. CRM Tool
8. Arranging Meetings
9. Project Chat Channels
10. Time Tracking

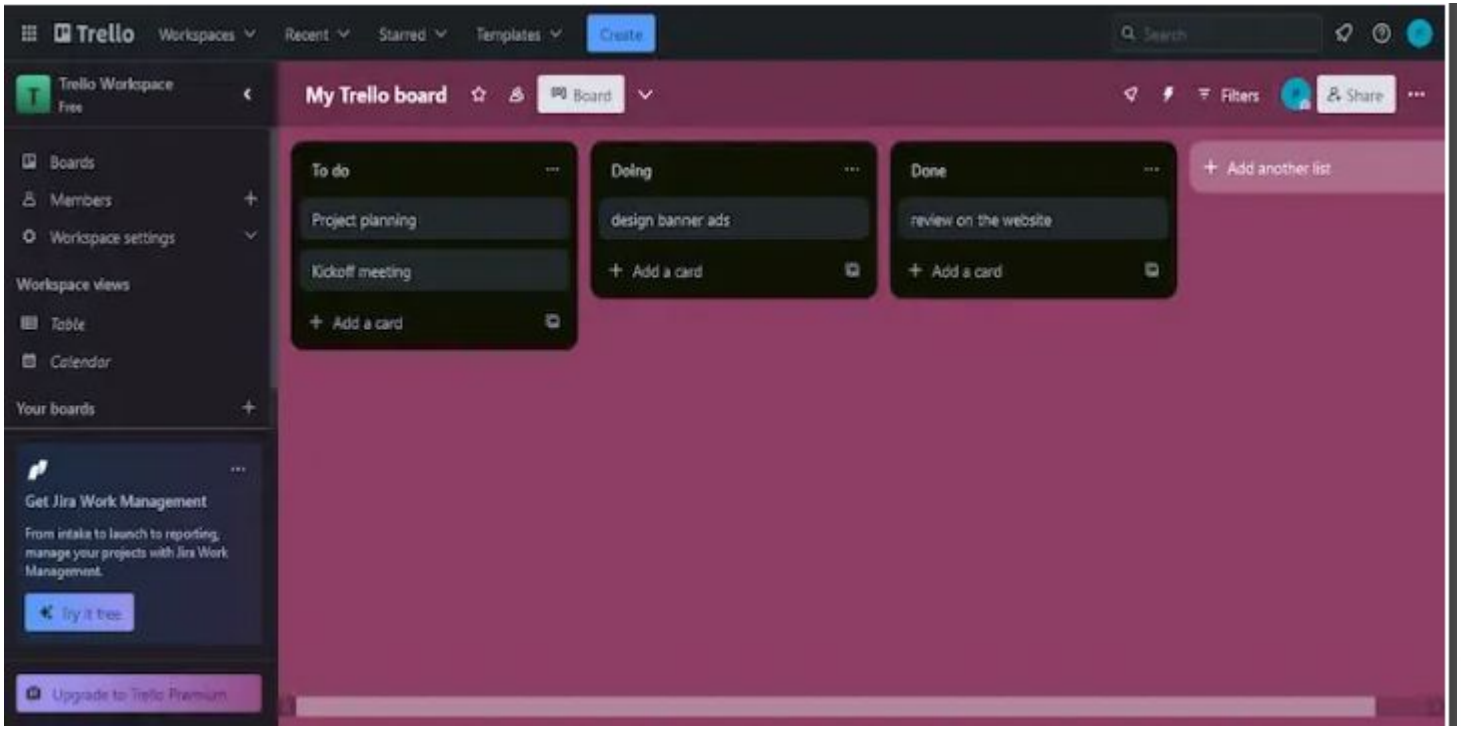
What is Trello?

Trello is a task and project management tool that's comprised of Boards, Lists, and Cards.



Trello Board

Trello boards are the advanced version of bulletin boards that are used to organize thoughts. The specialty of Trello is that you are not restricted to using a single board and can use multiple boards as complex projects will require multiple boards. It also provides different privacy options: **personal**, **private**, **team**, **organization**, and **public visibility**. Your board will show you **what is planned to be done**, and **task statuses,** and help you indicate capacity limitations.

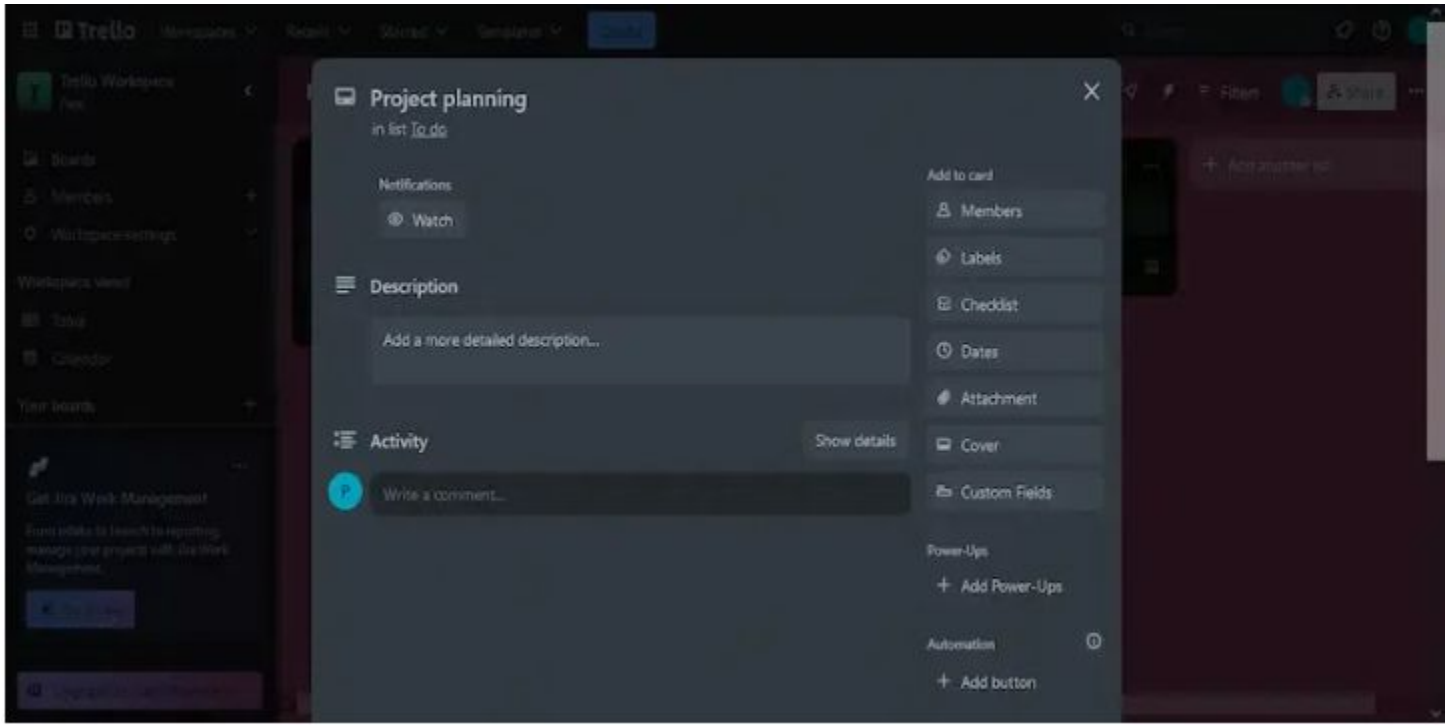


Exercise 1

1. **Sign Up and Create an Account**: Go to Trello's website and sign up for a free account.
2. **Create a Board**: Click on the "**Create new board**" button and give your board a name.
3. **Add Lists**: Inside your board, create lists to represent different stages of your workflow (e.g., To Do, In Progress, Done).
4. **Create Cards**: Add cards under each list for individual tasks. Click on a card to add details, due dates, attachments, and comments.
5. **Collaborate**: Invite team members by clicking on the "**Invite**" button and assigning them to specific cards.
6. **Move Cards**: Drag and drop cards between lists to reflect the progress of tasks.
7. **Use Power-Ups**: Enhance your board with **Power-Ups** (integrations and additional features) for added functionality.

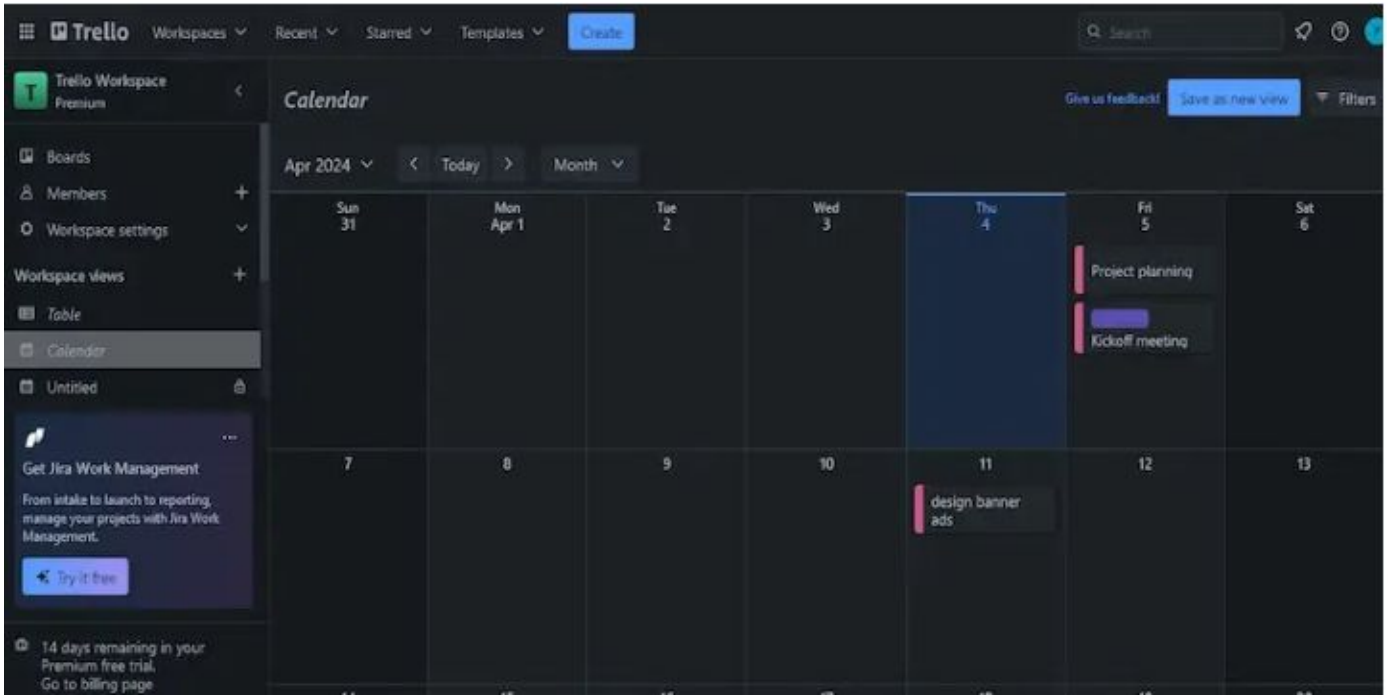
Using Trello Cards

Once the lists on the board are **completed**, cards need to be added. These Cards are the **building block of Trello** which provides details of the tasks within the project. Clicking a card expands to show detailed information like the example shown below.



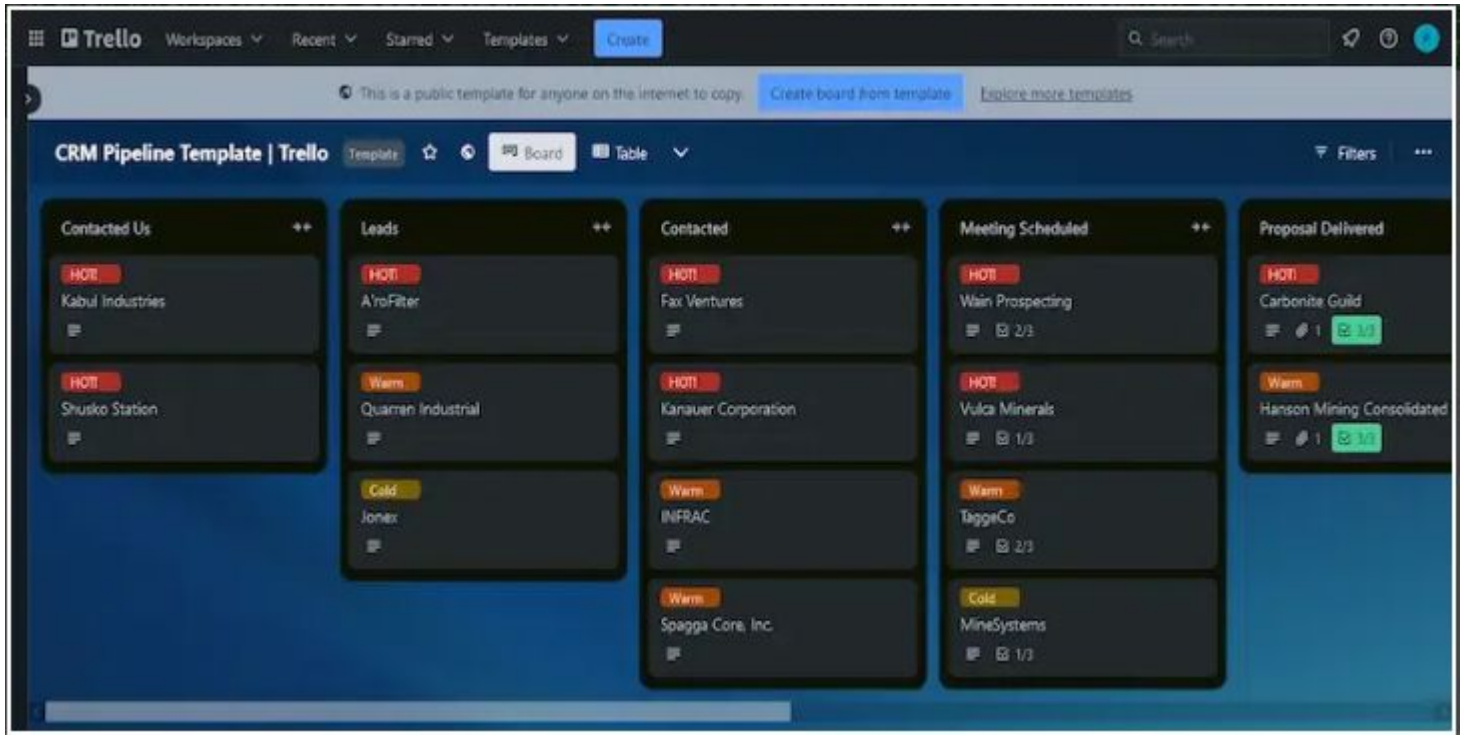
Calendar Planning

Trello's built-in view makes it a calendar planner, allowing you to see your deadlines visually, organized by due date. Lists act as timeframes ("**This Week**"), while color-coded labels highlight priorities. Optional Power-Ups further enhance functionality, letting you sync with external calendars. This makes Trello a powerful yet flexible way to manage your time, no matter how complex your schedule gets. By leveraging these features, you can effectively use Trello for project management, ensuring that all tasks are tracked and completed on time.



CRM Tool

Trello can be an essential tool in managing Customer relationships. Each customer has a Trello card, similar to a digital folder holding all their info, notes, and deal value. Move the cards around, and you can instantly see where leads are within the pipeline - easy, at first glance, to understand what's in the queue for your team. And it's flexible for your team to work together on pushing these deals forward.



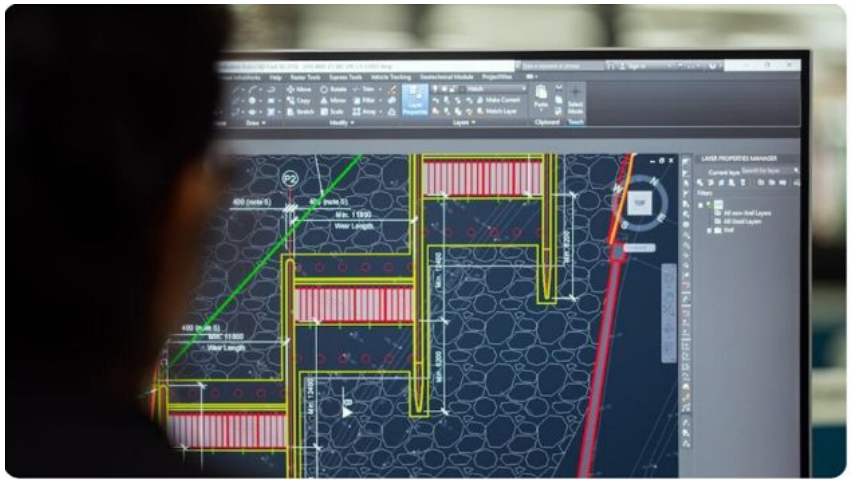
Benefits of Using Automation in Project Management

1. **Establishing Risks to be Managed**
   - Real-Time Risk Monitoring: Automation allows for continuous monitoring of project parameters, enabling teams to identify potential risks as they arise rather than after the fact. This proactive approach enhances the ability to mitigate risks effectively before they escalate
   - Data Analysis and Predictive Insights: Automated systems can analyze large volumes of data to identify trends and patterns that may indicate emerging risks. This predictive capability allows project managers to prepare for potential challenges and implement strategies accordingly
2. **Establishing Costs and Durations**
   - Accurate Cost Estimation: Automation tools can analyze historical data and current project metrics to provide more accurate cost estimations. This helps in budgeting and financial planning by reducing guesswork and improving reliability
   - Streamlined Reporting: Automated reporting features generate cost and duration reports quickly, providing stakeholders with timely insights into project status without manual compilation efforts. This enhances transparency and facilitates informed decision-making
3. **Establishing Resources Required**
   - Resource Optimization: Automation helps in tracking resource allocation and utilization across projects. By analyzing resource usage patterns, organizations can optimize their workforce and materials, ensuring that resources are used efficiently
   - Scalability: Automated systems can easily scale to accommodate changes in resource requirements as projects grow or shrink. This adaptability ensures that organizations do not overcommit or underutilize their resources

Understanding Business Process Automation

Defining the Future of Efficiency

- Definition: Business process automation (BPA) refers to the use of technology to automate repetitive, manual tasks in business operations to increase efficiency and accuracy.
- Benefits: The advantages of BPA include reduced operational costs, increased speed and accuracy of processes, and the ability to reallocate human resources to more strategic tasks.
- Tools: Various tools such as workflow automation software, robotic process automation, and AI are instrumental in facilitating BPA by removing bottlenecks in business operations.
- Examples: Common applications include automating invoice processing, customer relationship management (CRM), and supply chain management, demonstrating BPA's versatility across industries.



Integration of Project Management and Business Process Automation

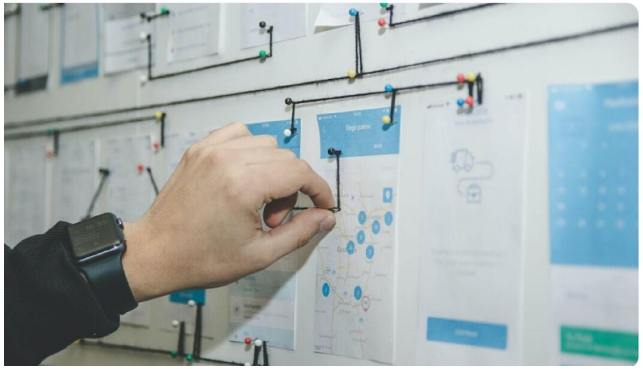
Creating Cohesion for Success

Scope	Description
Synergy	The integration of project management with BPA leads to improved planning, execution, and monitoring of initiatives, enhancing overall project success.
Process Mapping	Visualizing workflows and processes aids in identifying inefficiencies and opportunities for automation, aligning project management initiatives with business objectives.
Best Practices	Employing best practices such as stakeholder engagement, risk management, and iterative development can significantly improve project outcomes in automation.
Alignment	Ensuring that project management frameworks align with automation goals fosters a culture of continuous improvement and operational excellence within the organization.

Integration of Project Management and Business Process Automation

Understanding the Fundamentals

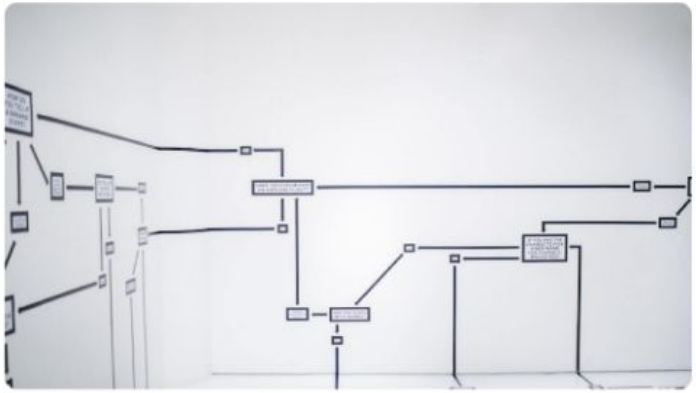
- Definition: Project management is the discipline of planning, organizing, and overseeing the successful execution of projects in the context of business process automation, which aims to optimize workflows and improve efficiency.
- Importance: The significance of project management in business process automation lies in its ability to streamline processes, enhance productivity, and ensure resources are effectively utilized to meet business objectives.
- Goals: The primary objectives encompass enhancing efficiency, reducing costs, improving quality and ensuring successful project delivery that meets stakeholder expectations.
- Overview: An effective project management approach integrates all phases of automation projects, from inception through to evaluation, thus providing a comprehensive roadmap for implementation.



Key Steps in Project Management

The Phases of Project Execution

- Initiation: Defining the project scope, objectives, and stakeholders to establish a shared understanding of the project goals and outcomes.
- Planning: Creating a detailed project plan that outlines tasks, timelines, resources, and budget, ensuring a clear pathway toward project completion.
- Execution: Implementing the project plan, coordinating teams, and managing resources to deliver the project output effectively.
- Monitoring: Continuous oversight of project progress, ensuring alignment with goals, addressing any deviations, and making necessary adjustments to stay on track.
- Closing: Finalizing all project elements, conducting evaluations, and documenting lessons learned to inform future projects and ensure comprehensive closure.



Project Management Tasks Priority Tutorial

We are going to conduct an booking management System. In working out the project, we apply python to create 2 sprint within 2 weeks that include 8 tasks

```
# First install required packages
import pandas as pd
import os
import plotly.express as px
from datetime import datetime

CSV_FILE = "bookingsystem_tasks.csv“
# Initial sample tasks for an booking management system
tasks = [
    # Sprint 1 - Core Features
    {'Task': 'User Authentication', 'Start': '2023-01-01', 'Finish': '2023-01-05', 'Priority': 'Must-Have', 'Sprint': 1},
    {'Task': 'Property Listing CRUD', 'Start': '2023-01-03', 'Finish': '2023-01-07', 'Priority': 'Must-Have', 'Sprint': 1},
    {'Task': 'Booking System Core', 'Start': '2023-01-06', 'Finish': '2023-01-10', 'Priority': 'Must-Have', 'Sprint': 1},
    {'Task': 'Basic Search Filters', 'Start': '2023-01-08', 'Finish': '2023-01-12', 'Priority': 'Should-Have', 'Sprint': 1},

    # Sprint 2 - Enhanced Features
    {'Task': 'Payment Integration', 'Start': '2023-01-15', 'Finish': '2023-01-19', 'Priority': 'Must-Have', 'Sprint': 2},
    {'Task': 'Review System', 'Start': '2023-01-17', 'Finish': '2023-01-21', 'Priority': 'Should-Have', 'Sprint': 2},
    {'Task': 'Recommendation Engine', 'Start': '2023-01-20', 'Finish': '2023-01-24', 'Priority': 'Could-Have', 'Sprint':
2},
    {'Task': 'Social Media Login', 'Start': '2023-01-22', 'Finish': '2023-01-26', 'Priority': 'Could-Have', 'Sprint': 2}
]
```

Project Management Tasks Priority Tutorial

(Continue)…

We define function that save all tasks into csv file named “bookingsystem_tasks.csv”

```
def save_tasks_to_csv(tasks, csv_file):
    df = pd.DataFrame(tasks)
    if not os.path.exists(csv_file):
        df.to_csv(csv_file, index=False)
        print(f"Created new CSV file with sample tasks at {csv_file}")
    else:
        print(f"CSV file already exists at {csv_file}. Overwriting...")
        df.to_csv(csv_file, index=False)
        print(f"Updated tasks saved to {csv_file}")

#Main Program
save_tasks_to_csv(tasks, CSV_FILE)
```

Task	Start	Finish	Priority	Sprint
User Authentication	1/1/2023	5/1/2023	Must-Have	1
Property Listing CRUD	3/1/2023	7/1/2023	Must-Have	1
Booking System Core	6/1/2023	10/1/2023	Must-Have	1
Basic Search Filters	8/1/2023	12/1/2023	Should-Have	1
Payment Integration	15/1/2023	19/1/2023	Must-Have	2
Review System	17/1/2023	21/1/2023	Should-Have	2
Recommendation Engine	20/1/2023	24/1/2023	Could-Have	2
Social Media Login	22/1/2023	26/1/2023	Could-Have	2

Project Management Tasks Priority Tutorial

Task 2 : We are going to generate and display gantt chart based on previous tasks.

```
csv_file = "bookingsystem_tasks.csv"

#define function to load tasks from csv
def load_tasks_from_csv(csv_file):
    if not os.path.exists(csv_file):
        print(f"Error: CSV file not found at {csv_file}. Please run save_tasks.py first.")
        return None
    else:
        df = pd.read_csv(csv_file)
        df['Start'] = pd.to_datetime(df['Start'])
        df['Finish'] = pd.to_datetime(df['Finish'])
        return df
```

Project Management Tasks Priority Tutorial

(Continue)… Here we define function that generate gantt chart and save as png

```
def display_gantt_chart(df):
    # Create a figure with a secondary axis for the Gantt chart
    fig = go.Figure()

    # Define colors for priorities
    color_map = {
        'Must-Have': '#FF4C4C',  # Red - Critical
        'Should-Have': '#FFA500', # Orange - Important
        'Could-Have': '#4CAF50'   # Green - Nice-to-have
    }

    # Iterate over each task and add it to the Gantt chart
    for index, row in df.iterrows():
        fig.add_trace(go.Scatter(
            x=[row['Start'], row['Finish']],
            y=[index, index],
            mode='lines',
            line_shape='hv',
            line_color=color_map[row['Priority']],
            hoverinfo='text',
            hovertext=f"Task: {row['Task']}, Priority: {row['Priority']}" #or
            #hovertext=f"Priority: {row['Priority']}"
        ))
```

Project Management Tasks Priority Tutorial

(Continue)… Here we define function that generate gantt chart and save as png

```
    # Set layout properties
    fig.update_layout(
        title="Housing Booking System Project Schedule",
        xaxis_title="Date",
        yaxis_title="Task",
        yaxis=dict(
            tickmode='array',
            tickvals=list(range(len(df))),
            ticktext=df['Task']
        ),
        xaxis=dict(
            type='date'
        )
    )

    fig.show()
```

Project Management Tasks Priority Tutorial

(Continue)… Main Program to run function to generate chart

```
#Display Gantt Chart
df = load_tasks_from_csv(CSV_FILE)
if df is not None:
    display_gantt_chart(df)
```

HKUSPACE
香港大學專業進修學院
HKU School of Professional and Continuing Education

THANK YOU