

Business Process Automation with VBA and Python

Mr. Eddie Chow / 7 December 2024

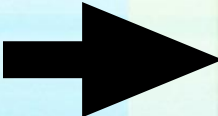


Table Of Contents

Introduction to business process automation

Business Process automation with VBA

Business process automation with Python



Introduction to project management for business process automation

Development and implementation of business process automation

Final Group Presentation





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 Introduction to Project Management in Business Process Automation
- 2 Methodologies in Project Management
- 3 Key Steps in Project Management
- 4 Understanding Business Process Automation
- 5 Integration of Project Management and Business Process Automation
- 6 Organizational Issues in Business Process Automation Projects
- 7 Project Scope and Requirements Gathering
- 8 Risk Management in Automation Projects
- 9 Change Management Best Practices
- 10 Performance Measurement and Key Performance Indicators (KPIs)
- 11 Case Studies of Successful Automation Projects
- 12 Conclusion and Future Directions

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

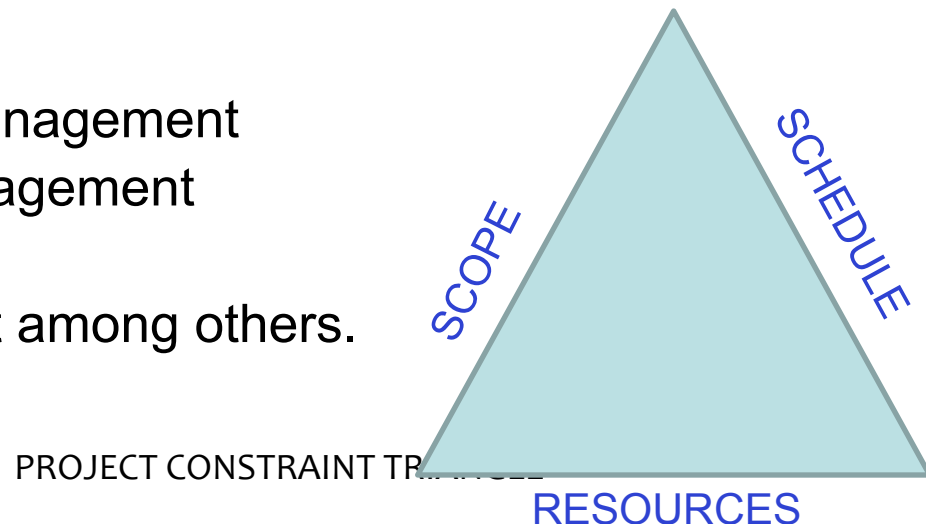
A project manager is a person who causes things to happen

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.



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

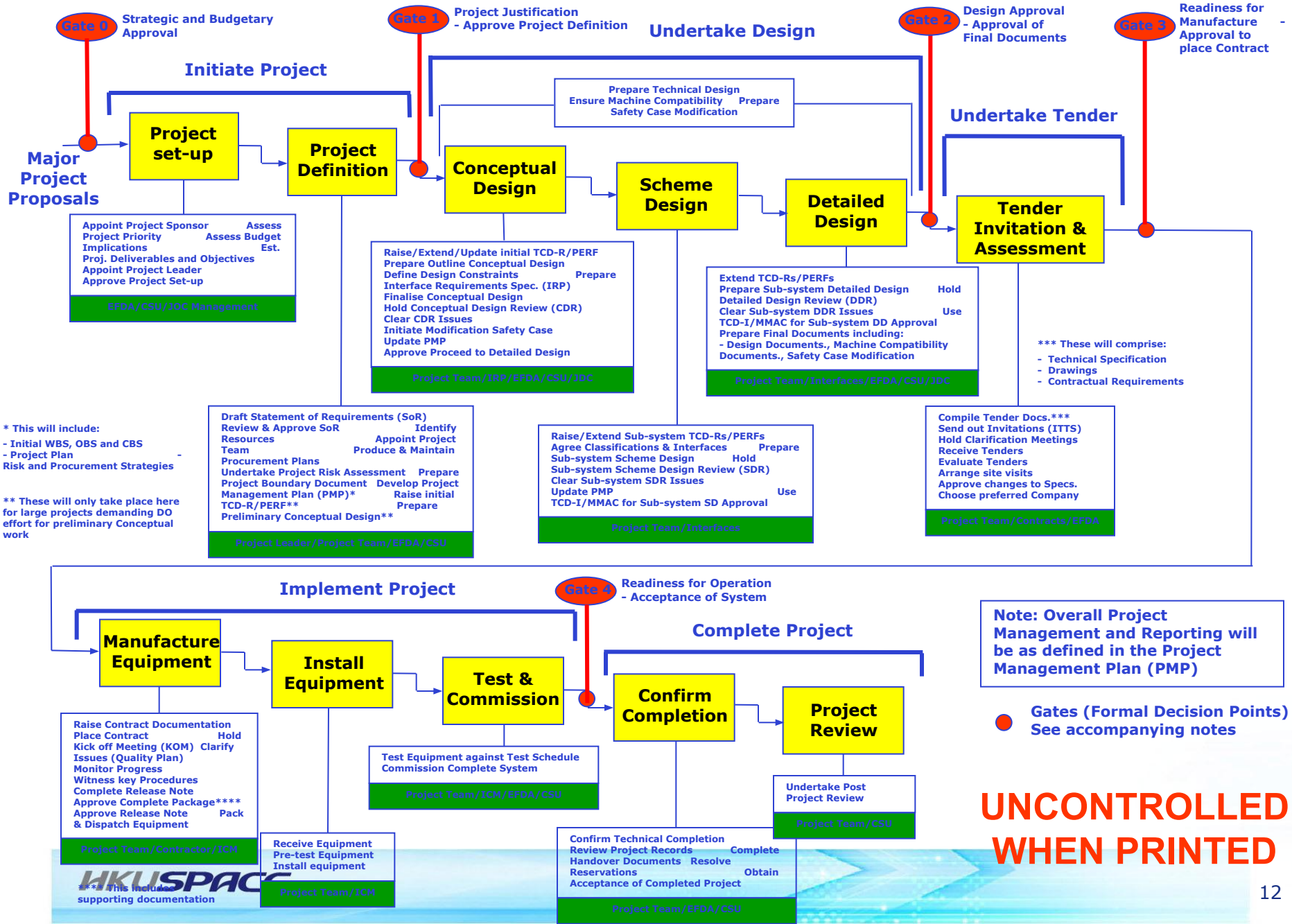
Define actions

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

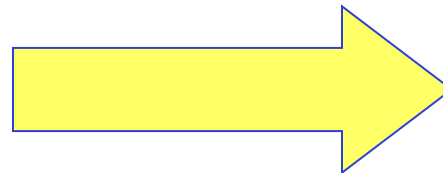
- Master Document for Project
- Defines the following:-
 - ⇒ Project Objectives, Scope, Deliverables
 - ⇒ Stakeholders (Internal & External)
 - ⇒ Work to be done (WBS)
 - ⇒ Project Organisation and Resources (OBS)
 - ⇒ Project Costings (CBS)
 - ⇒ Project Schedule
 - ⇒ Procurement/Contract Strategy
 -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

Project Planning



Project Planning

Adequate planning leads to the correct completion of work

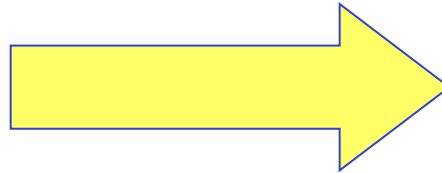


Plan

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

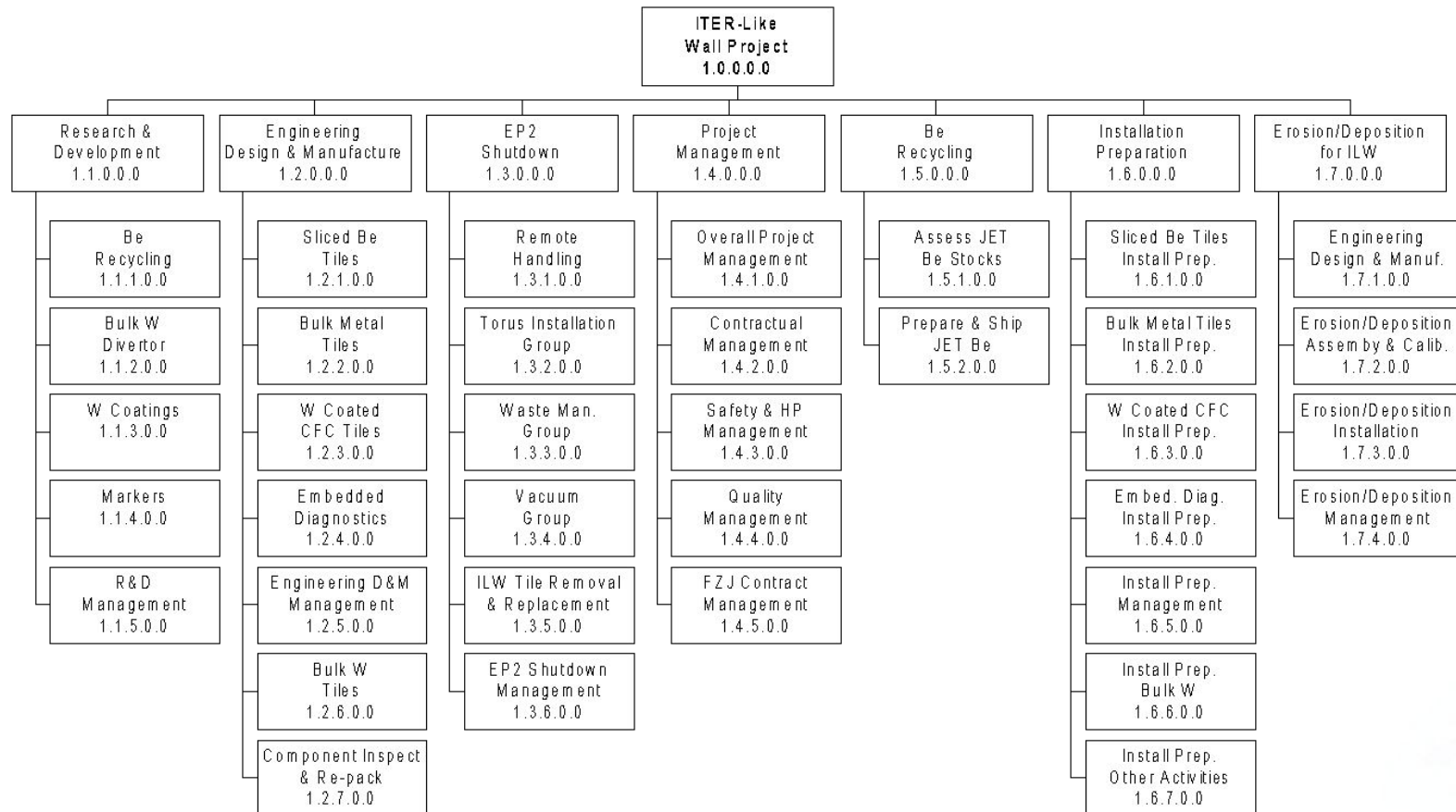
- It is the foundation on which everything else builds



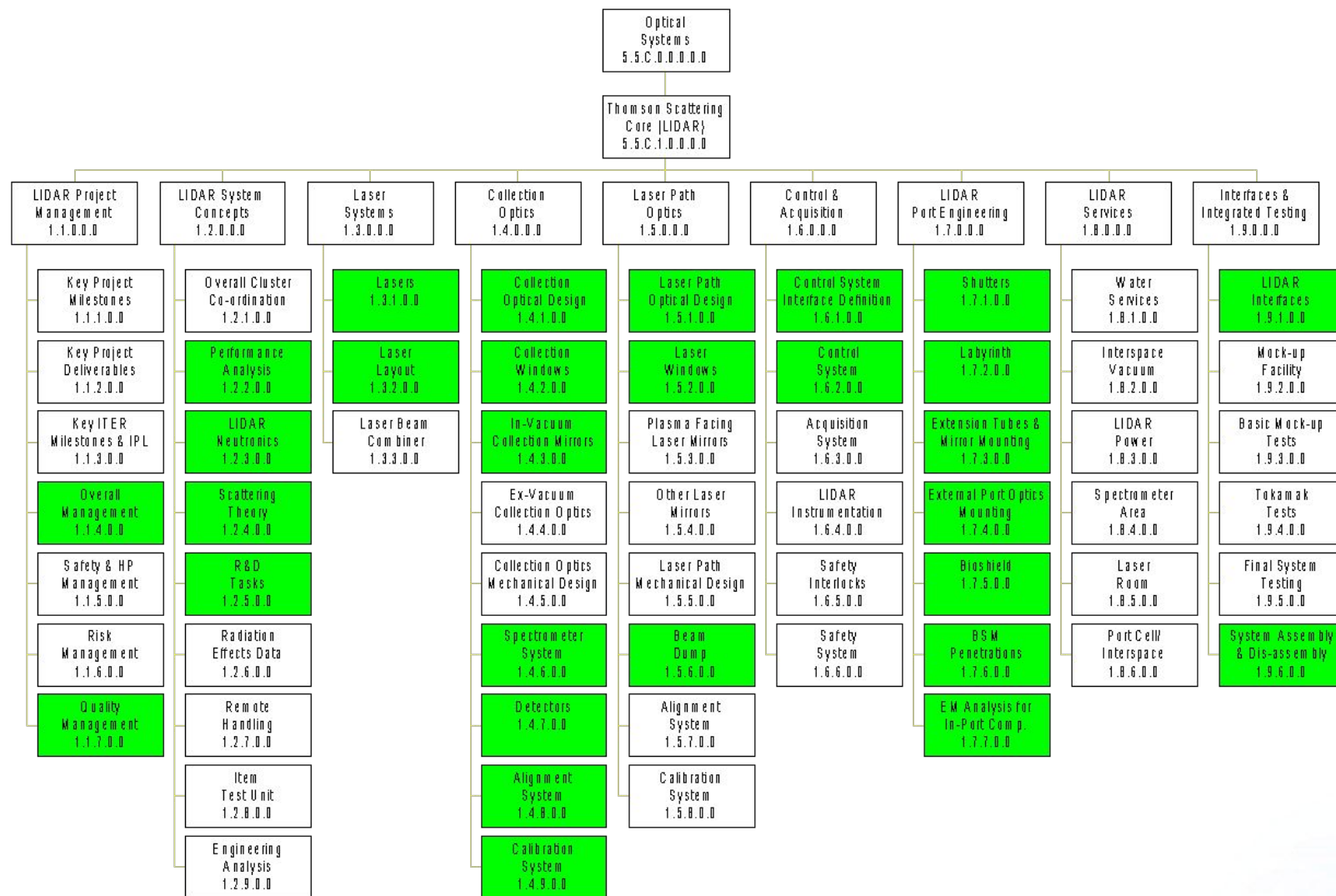
Work Breakdown Structure - Definition

“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

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
```

How to Manage Your Budget with a Simple Python Script

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

4. Main Function

```
if __name__ == "__main__":  
    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

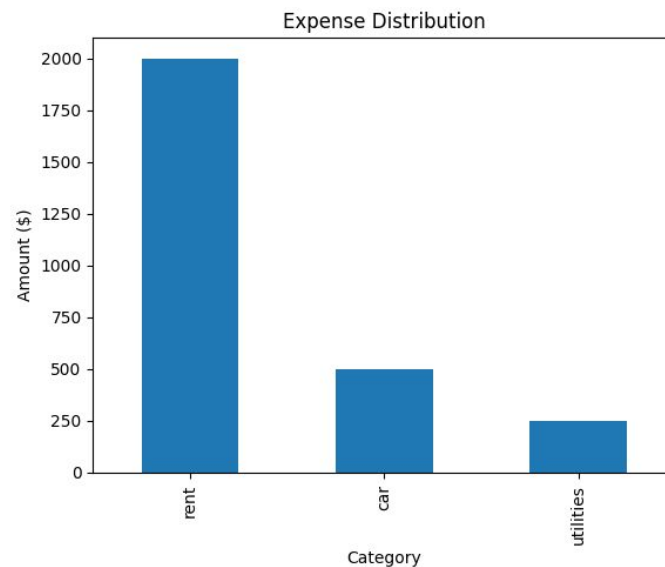
4. Main Function

Main Input

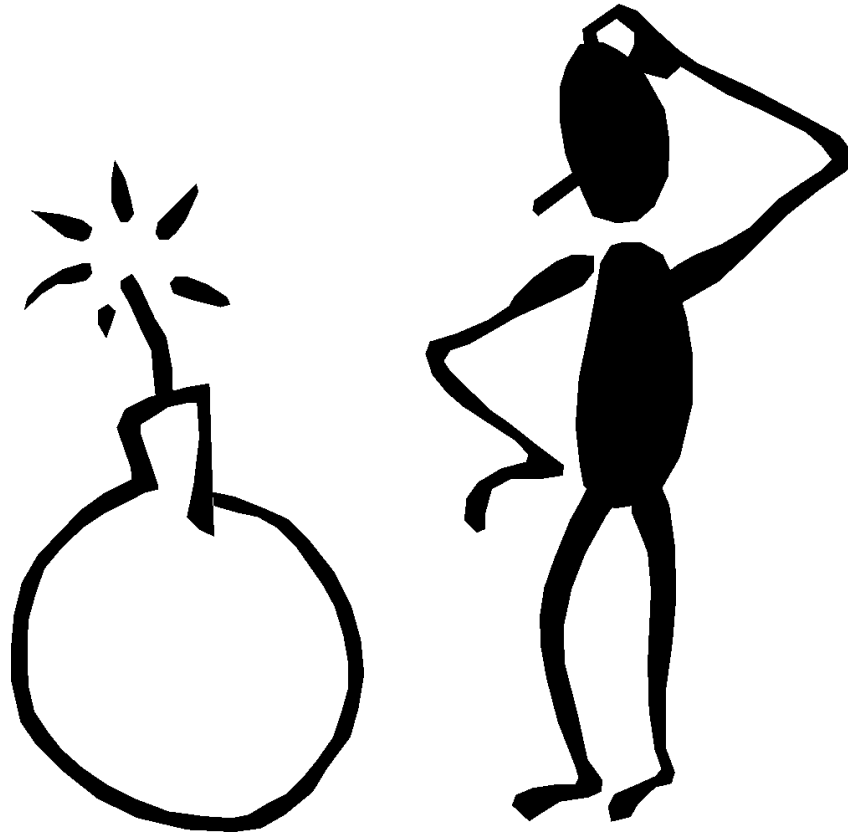
Enter your income: 10000
Enter expense category (or 'done' to finish): rent
Enter amount for rent: 2000
Enter expense category (or 'done' to finish): car
Enter amount for car: 500
Enter expense category (or 'done' to finish): utilities
Enter amount for utilities: 250
Enter expense category (or 'done'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

Result

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



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”

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



Risk Management Process

Identify Risks

Assess likelihood and impact

Rank risks and prioritize

Define risk management approach & actions

Implement actions

Monitor & review



Risk Management – Key Points

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 difficulties) and proactive

Project Monitoring and Control



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

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. Trello: A visual tool that organizes projects into boards and cards, ideal for agile workflows
4. Notion: Maintaining an internal database, collaborating, and managing tasks such as Creating project plans, Building roadmaps, Creating and storing important documents, guides, etc.
5. Wrike: Features task assignment, collaboration tools, and multiple viewing formats (list, board, Gantt) to manage workflows efficiently

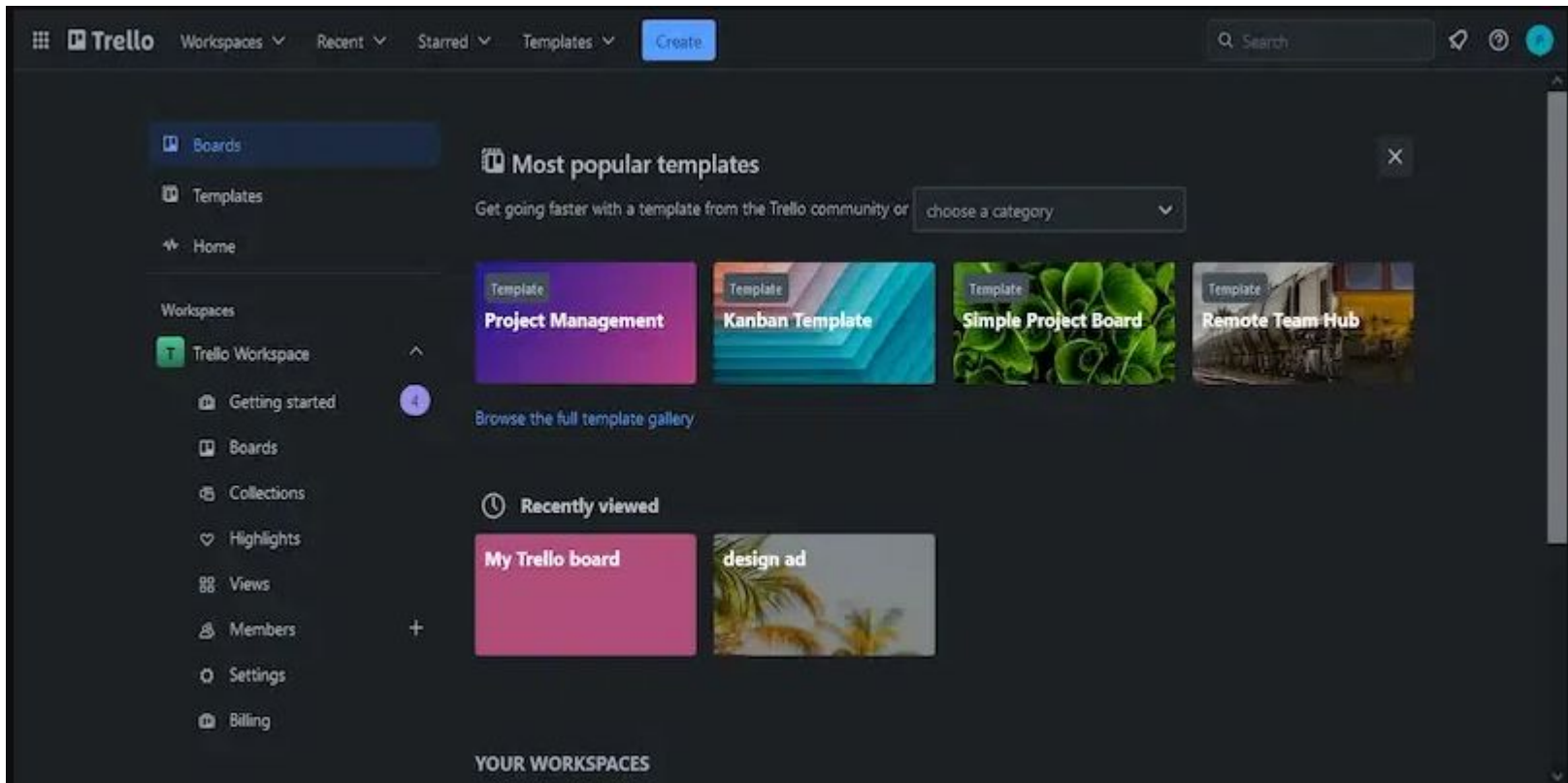


10 Ways to Use Trello for Project Management

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

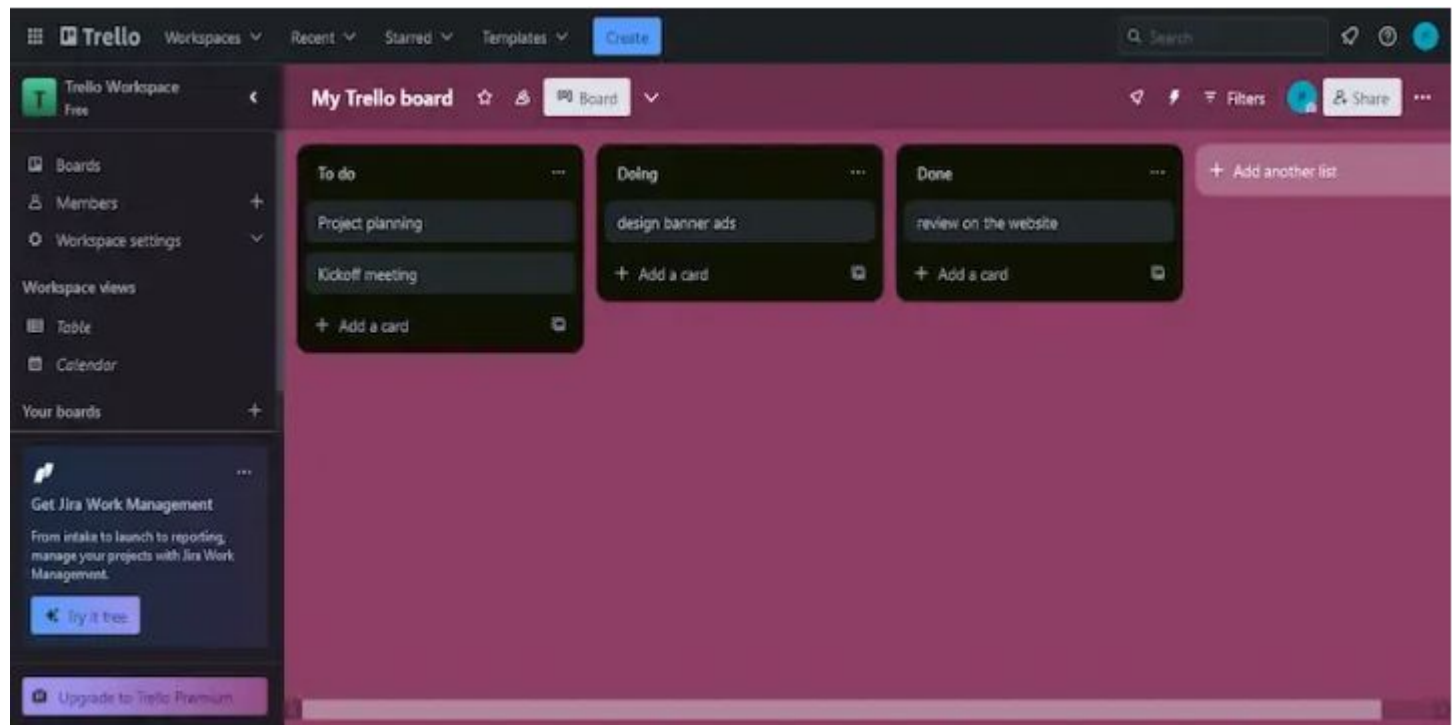
What is Trello?

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.

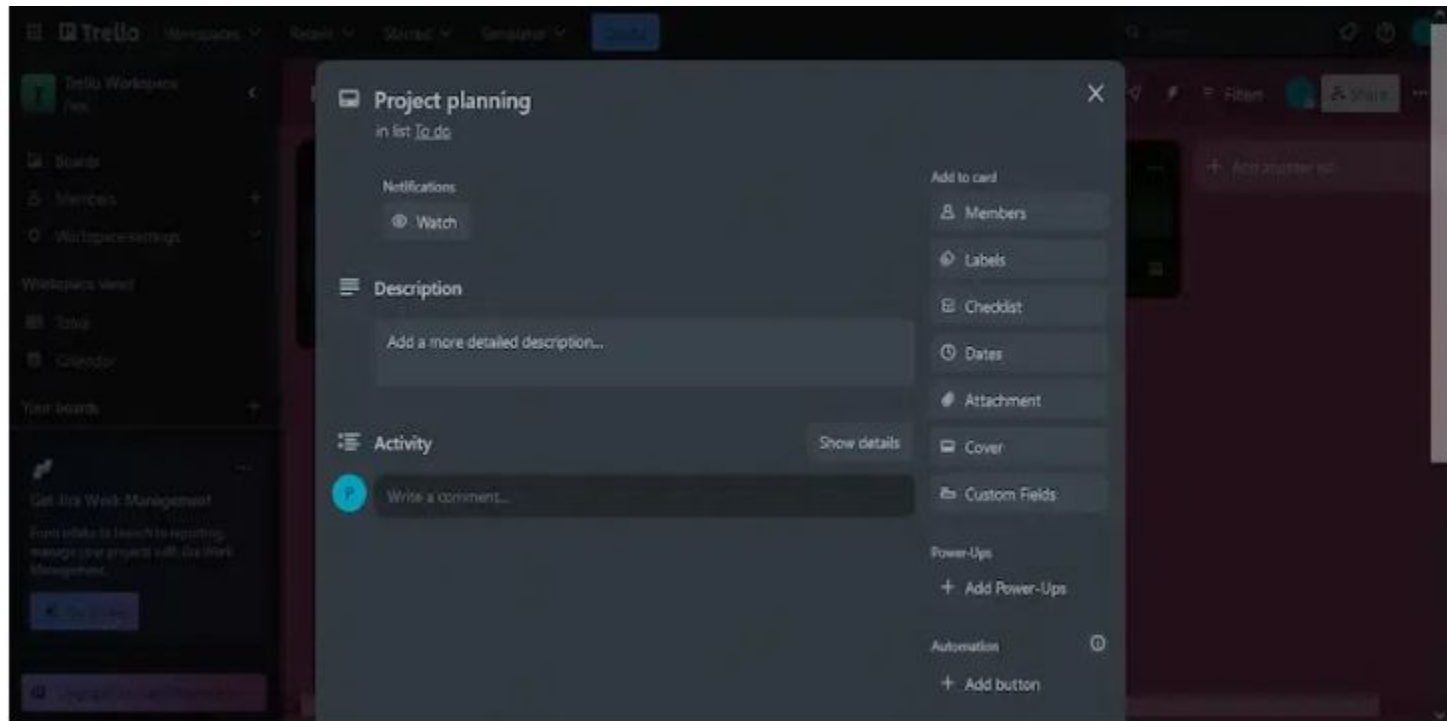


Exercise 1

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.

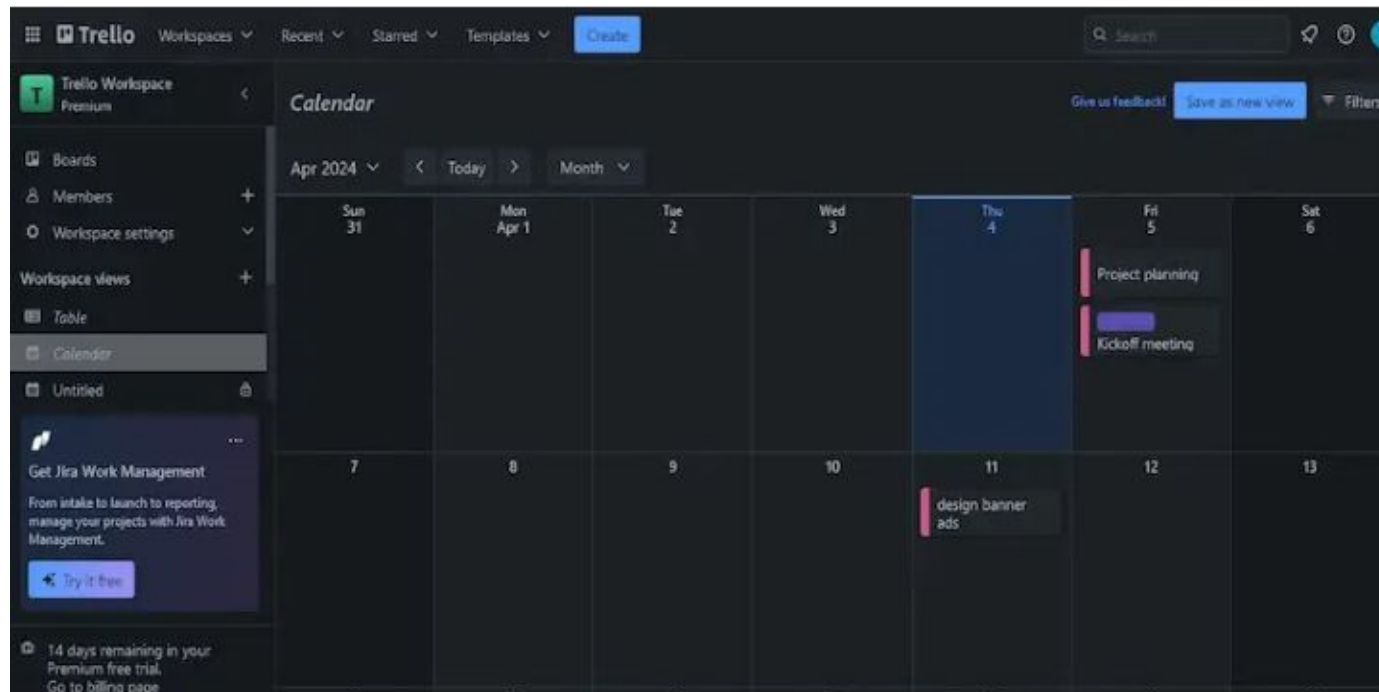
Using Trello Cards

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.



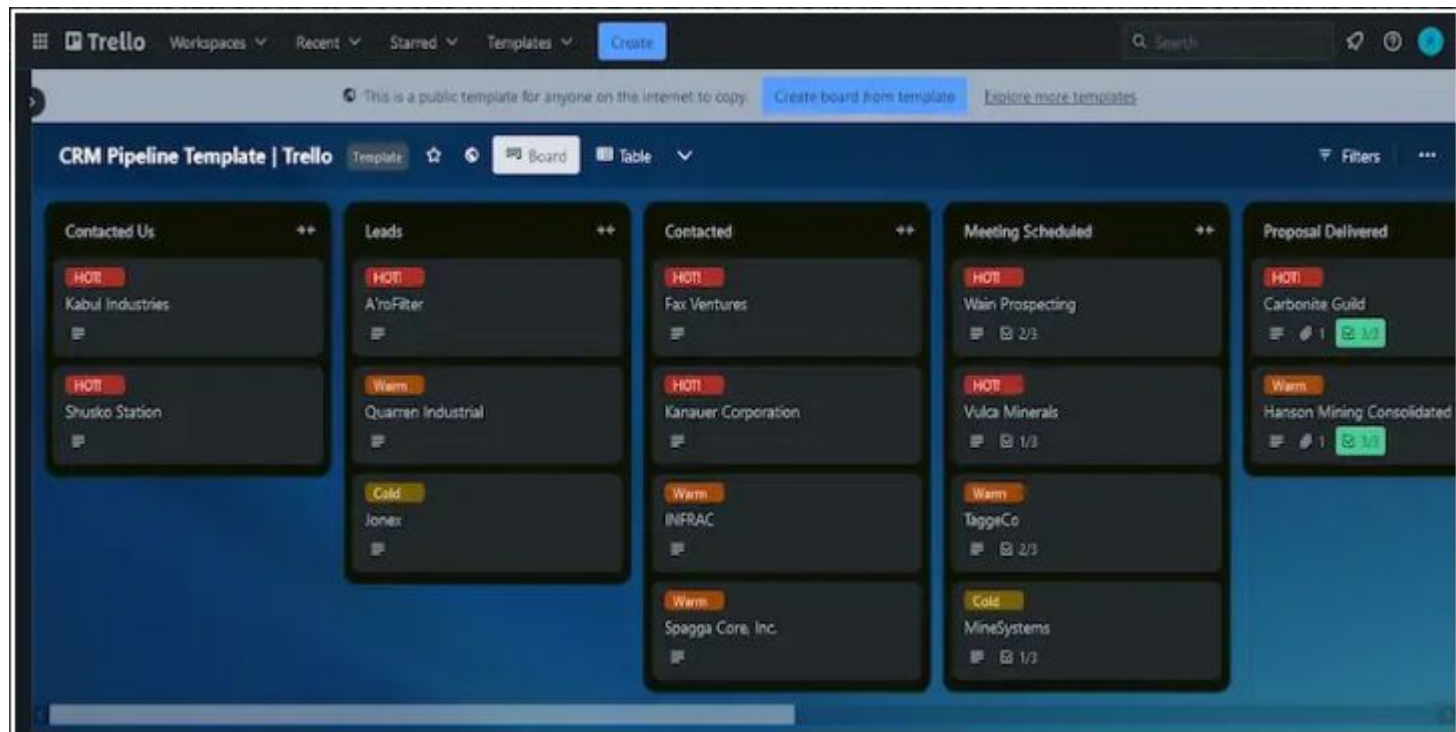
Calendar Planning

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

Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 1. Import the required libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing
import MinMaxScaler
from sklearn.model_selection
import train_test_split
from sklearn.metrics
import mean_squared_error
from keras.models
import Sequential
from keras.layers
import Dense, LSTM
```

Step 2: Load the finance data from <https://www.alphavantage.co>:

```
api_key = 'demo'
symbol = 'IBM'
url =
    f'https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol={symbol}&outputsize=f
    ull&apikey={api_key}'
response = requests.get(url)
data = response.json()
df = pd.DataFrame(data['Time Series (Daily)']).transpose()
df.index = pd.to_datetime(df.index)
df = df.sort_index()
```


Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 3 – Preprocess the data:

```
# Extract the closing prices
y = df['4. close'].values.astype(float)
# Normalize the closing prices
scaler = MinMaxScaler(feature_range=(0, 1))
y = scaler.fit_transform(y.reshape(-1, 1))
# Create the feature matrix
X = []
for i in range(60, len(df)):
    X.append(y[i-60:i, 0])
X = np.array(X)
# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y[60:], test_size=0.2, shuffle=False)
```

Step 4 – Define the model:

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

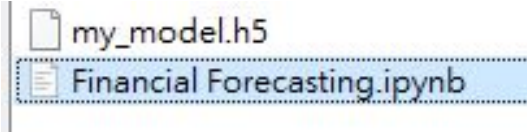
Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 5 – Train the model with 100 steps:

```
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_val, y_val))
```

Step 6 – Save the model in h5 format:

```
model.save('my_model.h5') # Save the model in HDF5 format
```



Step 7 – Evaluate the model:

```
# Evaluate the model on the validation set
```

```
y_pred = model.predict(X_val)
```

```
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
```

```
print('Root Mean Squared Error:', rmse)
```

Step 8 – Visualize the results:

```
# Visualize the results
```

```
y_pred = scaler.inverse_transform(y_pred)
```

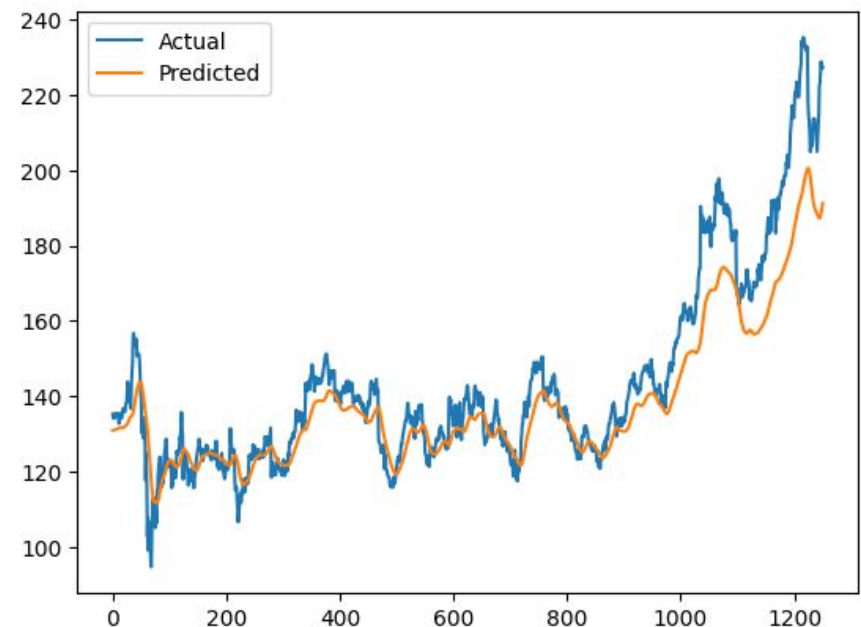
```
y_val = scaler.inverse_transform(y_val)
```

```
plt.plot(y_val, label='Actual')
```

```
plt.plot(y_pred, label='Predicted')
```

```
plt.legend()
```

```
plt.show()
```



Financial Forecasting with Machine Learning using Python (Numpy, Pandas, Matplotlib and Scikit-learn)

Step 9 – Make predictions:

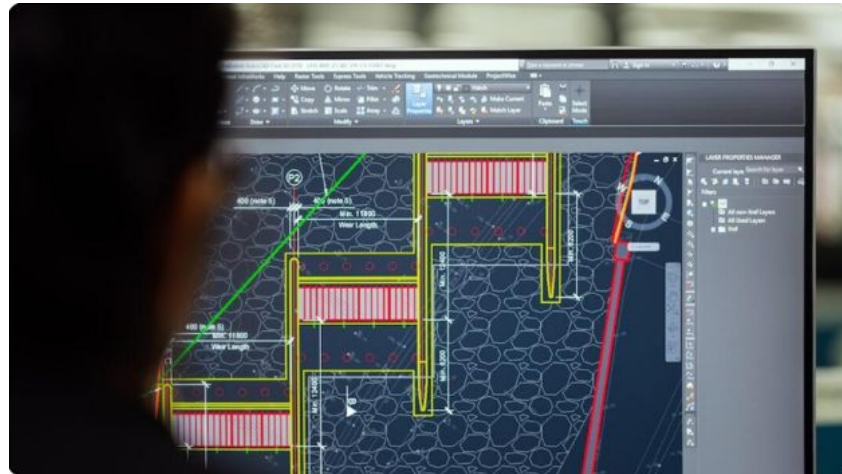
```
last_60_days = y[-60:]
last_60_days_scaled = scaler.transform(last_60_days.reshape(-1, 1))
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
y_pred = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred)
print('Predicted price:', y_pred[0][0])
```

```
1/1 [=====] - 1s 675ms/step
Predicted price: 40.51613
```

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

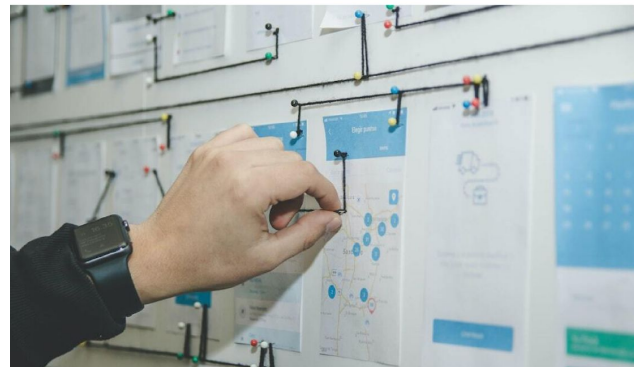
Creating Cohesion for Success

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

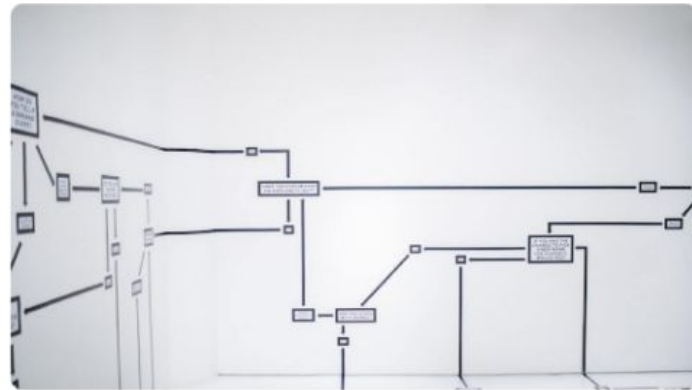
- **Definition:** Project management is the discipline of planning, organizing, and overseeing the successful execution of projects in the context of business process automation, which aims to optimize workflows and improve efficiency.
- **Importance:** The significance of project management in business 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.



Methodologies in Project Management

1. 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.

2. 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.

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.

- **Ittransition 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.

Exploratory data analysis (EDA)

Exploring your data is a crucial step in data analysis. It involves:

Organising the data set

Plotting aspects of the data set

Maybe producing some numerical summaries; central tendency and spread, etc.

“Exploratory data analysis can never be the whole story, but nothing else can serve as the foundation stone.”

- John Tukey.

Exploratory Data Analysis (EDA)

Why?

- EDA encompasses the “*explore* data” part of the data science process
- EDA is crucial but often overlooked:
 - If your data is bad, your results will be bad
 - Conversely, understanding your data well can help you create smart, appropriate models

Exploratory Data Analysis (EDA)

What?

1. Store data in data structure(s) that will be convenient for exploring/processing
(Memory is fast. Storage is slow)
2. Clean/format the data so that:
 - Each row represents a single object/observation/entry
 - Each column represents an attribute/property/feature of that entry
 - Values are numeric whenever possible
 - Columns contain atomic properties that cannot be further decomposed*

* Unlike food waste, which can be composted.
Please consider composting food scraps.

Exploratory Data Analysis (EDA)

What? (continued)

3. Explore **global** properties: use histograms, scatter plots, and aggregation functions to summarize the data
4. Explore **group** properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to follow-up in subsequent analysis.



EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

NOTE: The following music data are used purely for illustrative, educational purposes. The data, including song titles, may include explicit language. Harvard, including myself and the rest of the CS109 staff, does not endorse any of the entailed contents or the songs themselves, and we apologize if it is offensive to anyone in anyway.

EDA: without Pandas

top50.csv

Each row represents a distinct song. The columns are:

- **ID:** a unique ID (i.e., 1-50)
- **TrackName:** Name of the Track
- **ArtistName:** Name of the Artist
- **Genre:** the genre of the track
- **BeatsPerMinute:** The tempo of the song.
- **Energy:** The energy of a song - the higher the value, the more energetic.
- **Danceability:** The higher the value, the easier it is to dance to this song.
- **Loudness:** The higher the value, the louder the song.
- **Liveness:** The higher the value, the more likely the song is a live recording.
- **Valence:** The higher the value, the more positive mood for the song.
- **Length:** The duration of the song (in seconds).
- **Acousticness:** The higher the value, the more acoustic the song is.
- **Speechiness:** The higher the value, the more spoken words the song contains.
- **Popularity:** The higher the value, the more popular the song is.

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Mencia	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Grande	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful People	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

■
■
■

Q1: What are some ways we can store this file into data structure(s) using regular Python (not the Pandas library).

EDA: without Pandas

top50.c

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Mencia	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Grande	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful People	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Possible Solution #1: A 2D array (i.e., matrix)

Weaknesses:

- What are the row and column names? Need separate lists for them – clumsy.
- Lists are $O(N)$. We'd need 2 dictionaries just for column names

```
data = []  
col_name  
-> index  
index ->  
col_name
```

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	ness Live	Valence	Length	Acousticness	ness Speechi	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Possible Solution #2: A list of dictionaries

list

Item 1

= {"Track.Name": "Senorita", "Artist.Name": "Shawn Mendes", "Genre": "Canadian pop", ...}

Item 2

= {"Track.Name": "China", "Artist.Name": "Anuel AA", "Genre": "reggaeton flow", ...}

Item 3

= {"Track.Name": "Ariana Grande", "Artist.Name": "boyfriend", "Genre": "dance pop", ...}

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

```
f = open("../data/top50.csv", encoding = "ISO-8859-1")
column_names = f.readline().strip().split(",")[1:] # puts names in a list
cleaned_column_names = [name[1:-1] for name in column_names]
cleaned_column_names.insert(0, "ID")

dataset = []

# iterates through each line of the .csv file
for line in f:
    attributes = line.strip().split(",")

    # constructs a new dictionary for each line
    dataset.append(dict(zip(cleaned_column_names, attributes)))
```


EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q2: Write code to print all songs (Artist and Track name) that are longer than 4 minutes (240 seconds):

```
for song in dataset:
    if int(song["Length."]) > 240:
        print(song["Artist.Name"], "-", song["Track.Name"], "is", song["Length."], "seconds long")
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q3: Write code to print the most popular song (artist and track) – if ties, show all ties.

```
max_score = -1
most_populars = set()
for song in dataset:
    if int(song["Popularity"]) > max_score:
        most_populars = set([str(song["Artist.Name"]) + "-" + song["Track.Name"]])
        max_score = int(song["Popularity"])
    elif int(song["Popularity"]) == max_score:
        most_populars.add(str(song["Artist.Name"]) + "-" + song["Track.Name"])
print(most_populars)
```

EDA: list of dictionaries

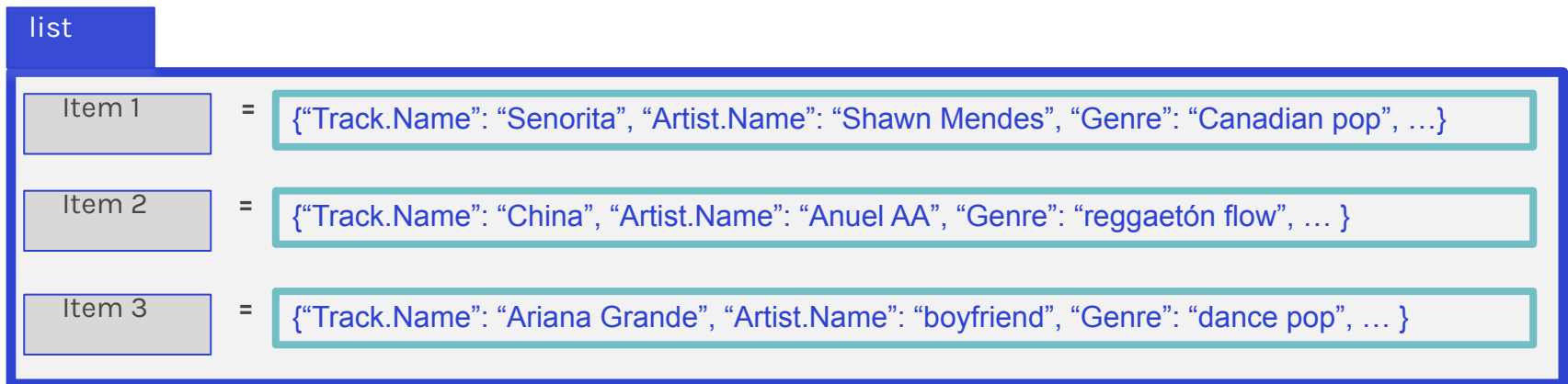
Possible Solution #2: A list of dictionaries

Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).



Cumbersome to move dictionaries around in a list. Problematic even if we don't move the dictionaries.

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q5: How could you check for null/empty entries? This is only 50 entries. Imagine if we had 500,000.

list	
Item 1	= {"Track.Name": "Senorita", "Artist.Name": "Shawn Mendes", "Genre": "Canadian pop", ...}
Item 2	= {"Track.Name": "China", "Artist.Name": "Anuel AA", "Genre": "reggaetón flow", ... }
Item 3	= {"Track.Name": "Ariana Grande", "Artist.Name": "boyfriend", "Genre": "dance pop", ... }

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q6: Imagine we had another table* below (i.e., .csv file). How could we combine its data with our already-existing *dataset*?

spotify_aux.

	TrackName	ArtistName	ExplicitLanguage	
1	Senorita	Shawn Menco	TRUE	
2	China	Anuel AA	FALSE	
3	boyfriend (w	Ariana Grande	TRUE	
4	Beautiful People	Ed Sheeran	FALSE	

* 3rd column is made-up by me. Random values. Pretend they're accurate.

EDA: with Pandas!



Kung Fu Panda is property of DreamWorks and Paramount Pictures

Lecture Outline

- Exploratory Data Analysis (EDA):
 - Without Pandas (part 1) – These slides
 - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides
- Web Scraping with Beautiful Soup (part 4) – Mix

EDA: with Pandas

What / Why?

- Pandas is an *open-source* Python library (anyone can contribute)
- Allows for high-performance, easy-to-use data structures and data analysis
- Unlike NumPy library which provides multi-dimensional arrays, Pandas provides 2D table object called **DataFrame** (akin to a spreadsheet with column names and row labels).
- Used by *a lot* of people

EDA: with Pandas

How

- import **pandas** library (convenient to rename it)
- Use **read_csv()** function

```
import pandas as pd  
dataframe = pd.read_csv("yourfile.csv")
```

Common Panda functions

High-level viewing:

- `head()` – first N observations
- `tail()` – last N observations
- `columns()` – names of the columns
- `describe()` – statistics of the quantitative data
- `dtypes()` – the data types of the columns

Common Panda functions

Accessing/processing:

- `df["column_name"]`
- `Df.column_name`
- `.max()`, `.min()`, `.idxmax()`, `.idxmin()`
- `<dataframe> <conditional statement>`
- `.loc[]` – label-based accessing
- `.iloc[]` – index-based accessing
- `.sort_values()`
- `.isnull()`, `.notnull()`

Common Panda functions

Grouping/Splitting/Aggregating:

- `groupby()`, `.get_groups()`
- `.merge()`
- `.concat()`
- `.aggegate()`
- `.append()`

Lecture Outline

- Exploratory Data Analysis (EDA):
 - Without Pandas (part 1) – These slides
 - With Pandas (part 2) – Mostly Jupyter Notebook
- Data concerns (part 3) – These slides
- Web Scraping with Beautiful Soup (part 4) – Mix

Data Concerns

When determining if a dataset is sound to use, it can be useful to think about these four questions:

- Did it come from a trustworthy, authoritative source?
- Is the data a complete sample?
- Does the data seem correct?
- **(optional)** Is the data stored efficiently or does it have redundancies?

Data Concerns: the format

- Often times, there may not exist a single dataset that contains all of the information we are interested in.
- May need to merge existing datasets
- Important to do so in a sound and efficient format

Data Concerns: the format

For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

SpotifySongID, # of Streams, Date			
200	2789179,	42003,	06-01
	▪		
200	3819390,	89103,	06-01
	▪		
200	4492014,	52923,	06-02
	▪		
200	8593013,	189145,	06-02
	▪		

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

SpotifySongID, Artist, Track, [10 acoustic features]			
50	2789179,	Billie Eilish, bad guy, 3.2, 5.9, ...	
	▪		
50	3901829,	Outkast, Elevators, 9.3, 5.1, ...	
	▪		

50 x 13

Data Concerns: the format

For example, say we have two datasets:

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

SpotifySongID, # of Streams, Date

200	2789179,	42003,	06-01
	•		
	•		
200	3819390,	89103,	06-01
	•		
	•		
200	4492014,	52923,	06-02
	•		
	•		
200	8593013,	189145,	06-02
	•		
	•		

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

SpotifySongID, Artist, Track, [10 acoustic features]

50	2789179,	Billie Eilish, bad guy, 3.2, 5.9, ...
	•	
	•	
50	3901829,	Outkast, Elevators, 9.3, 5.1, ...
	•	
	•	

50 x 13

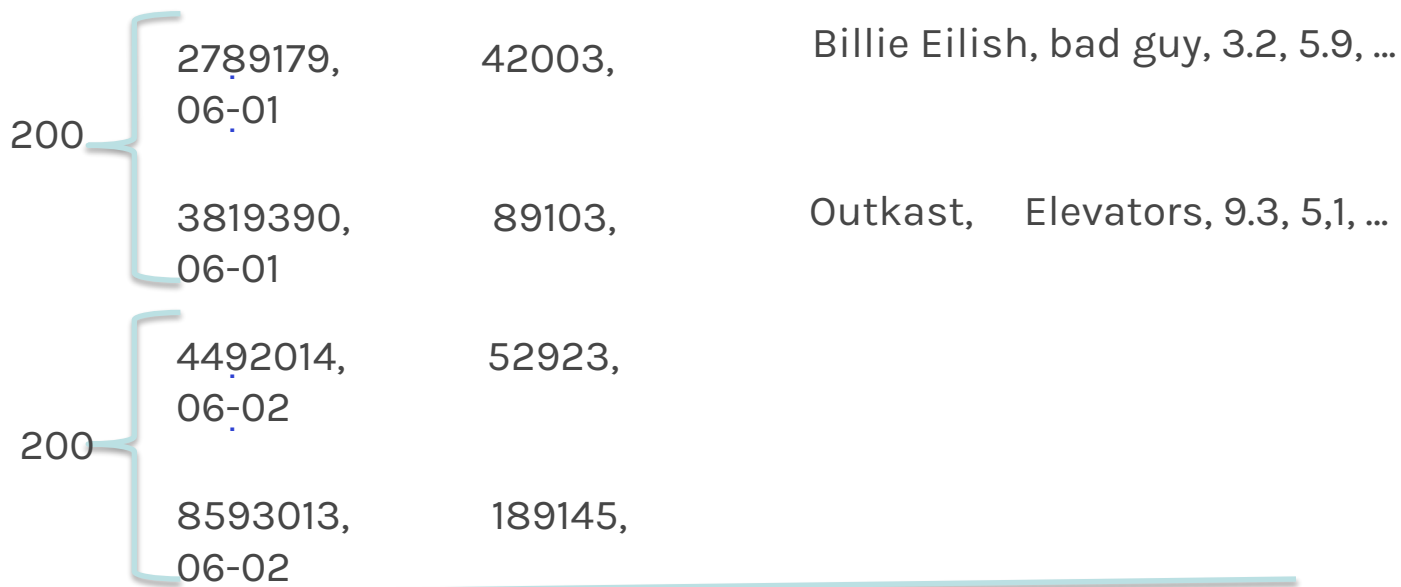
We are interested in determining if songs with high danceability are more popular during the weekends of June than weekdays in June. **What should our merged table look like? Concerns?**

Data Concerns: the format

This is wasteful, as it has 10 acoustic features, artist, and track repeated many times for each unique song.

Datasets Merged (poorly)

SpotifySongID, # of Streams, Date, Artist, Track, [10 acoustic features]



2789179,	42003,	Billie Eilish, bad guy, 3.2, 5.9, ...
06-01		
3819390,	89103,	Outkast, Elevators, 9.3, 5.1, ...
06-01		
4492014,	52923,	
06-02		
8593013,	189145,	
06-02		

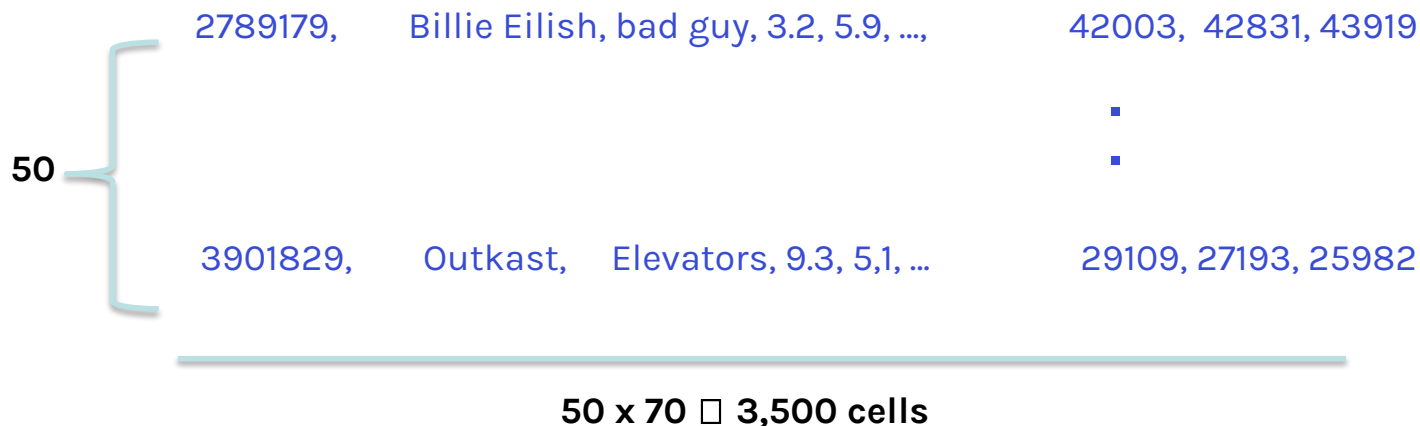
6,000 x 15 = 90,000 cells

Data Concerns: the format

Some rows may have null values for # of Streams (if the song wasn't popular in June)

Datasets Merged (better)

SpotifySongID, Artist, Track, [10 acoustic features], 06-01 Streams, 06-02 Streams



2789179,	Billie Eilish, bad guy, 3.2, 5.9, ...,	42003, 42831, 43919
3901829,	Outkast, Elevators, 9.3, 5.1, ...	29109, 27193, 25982

50 x 70 = 3,500 cells



Data Concerns: the format

- Is the data correctly constructed (or are values wrong)?
- Is there redundant data in our merged table?
- Missing values?

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.



Flask

django

WEB2PY

Commonly used Flash Framework

Key features of Django



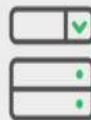
Authentication



Security



Object-relational mapping



Templates



Forms

django

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?

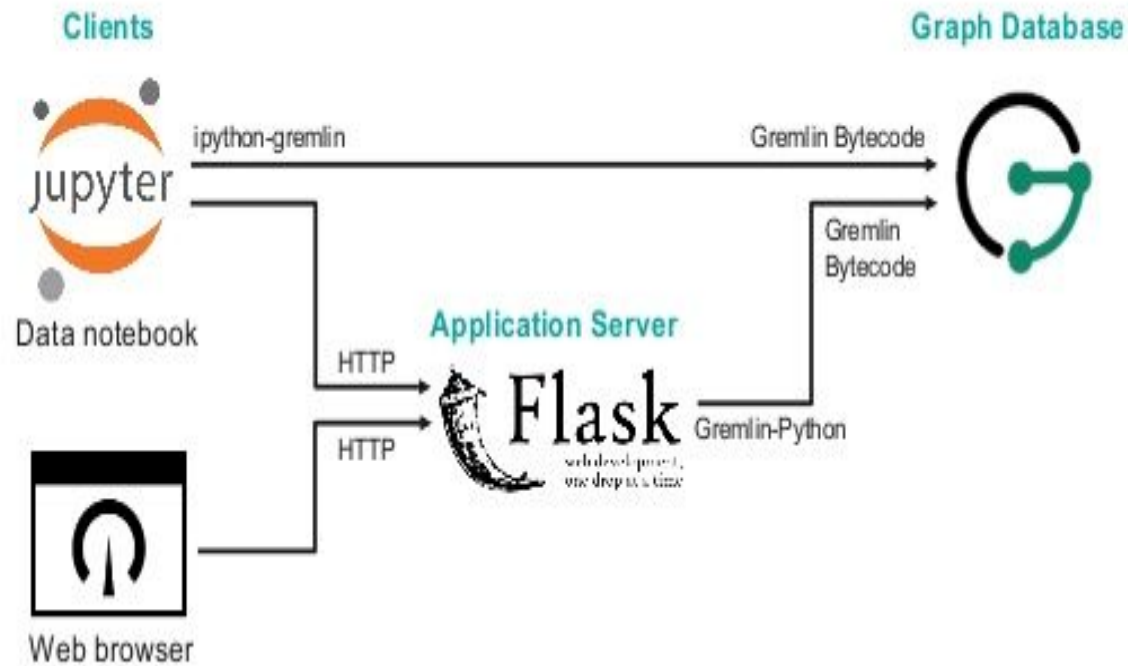
Every Dash App is composed of two main parts:

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

Example Python Application Architecture





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

THANK YOU

