

IDAT7215

Computer Programming for Product Development and Applications

Midterm Exam Guidelines

Dr. Zulfiqar Ali

Midterm Exam Guidelines

Exam Week (20-03-2025)

- Total Exam Time: 60 mins (1:00 Hrs)
- Exam Venue: Classroom
- It will be a closed-book exam (no cheat sheet allowed).
- All the materials covered till Week-7.
- 80 % of exam questions will be MCQs or True/False types.
- 20% of exam questions will be short questions.

Any Question



IDAT7215

Computer Programming for Product Development and Applications

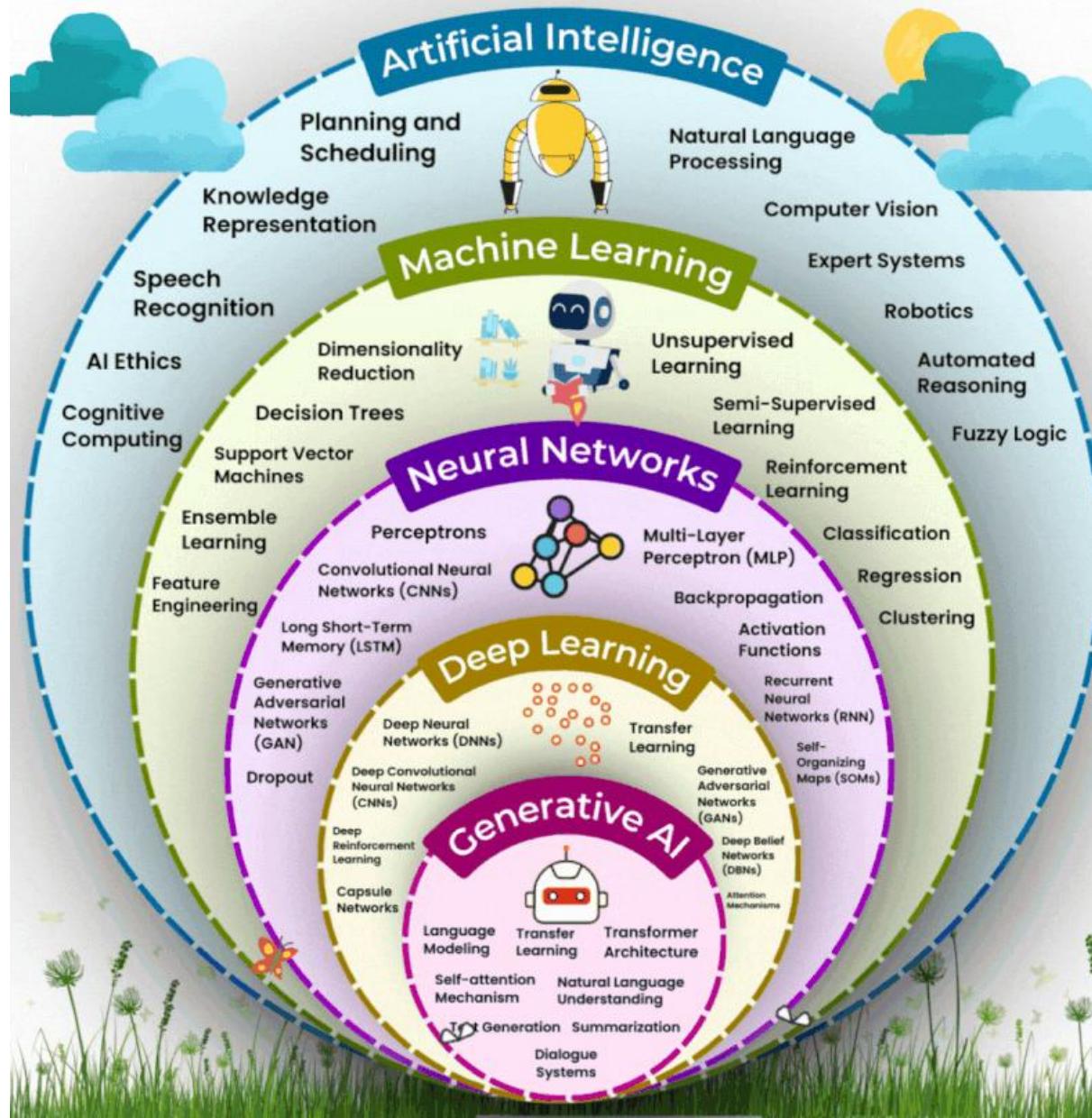
Lecture 6-1: Introduction to Machine Learning
Using Scikit-learn Python Library

Dr. Zulfiqar Ali

Outline

- Why Machine Learning
- What is Machine Learning
- Machine Learning vs Deep Learning
- Training and Test Data
- Types of Machine Learning
- Introduction to Scikit-Learn Library
- Supervised Machine Learning Algorithms

The AI Universe



Artificial Intelligence



Engineering of
making Intelligent
Machines and Programs

1950's

1960's

1970's

Machine Learning



Ability to learn
without being explicitly
programmed

1980's

1990's

2000's

2006's

Deep Learning



Learning based on
Deep Neural
Network

2010's

2012's

2017's

Why Machine Learning was Introduced

Statistics: How to efficiently train large complex models?

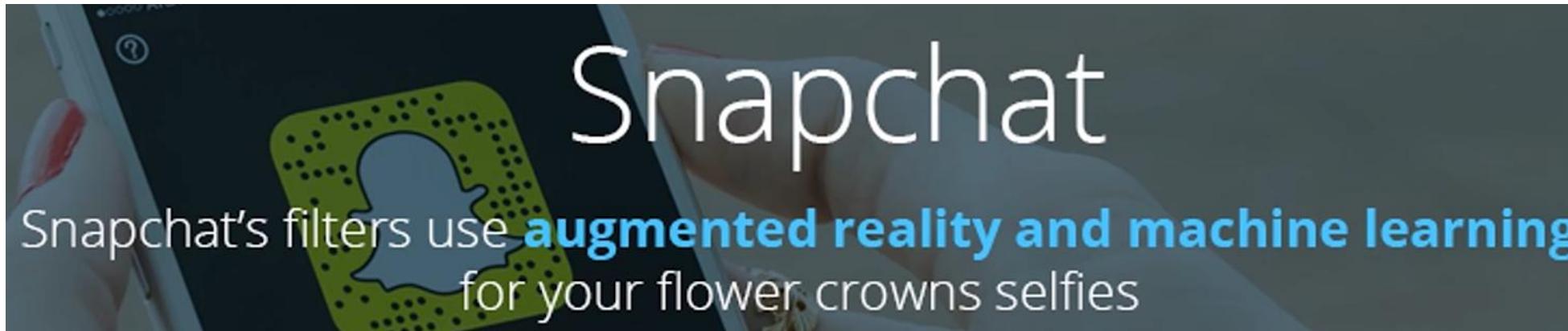
Computer Science & Artificial Intelligence: How to train more robust versions of the AI systems.

Neuroscience: How to design operational models of the brain.



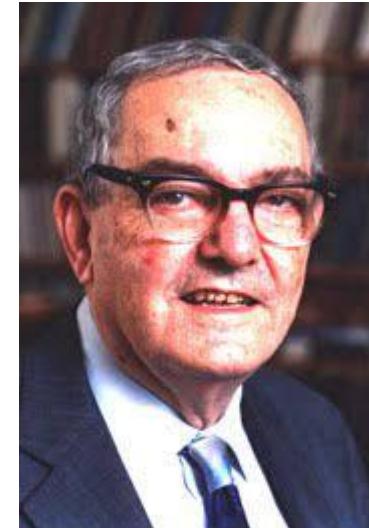
What is Machine Learning

Machine learning is a subset of AI techniques that **use statistical methods** to **enable machines to learn automatically** and improve from experience.



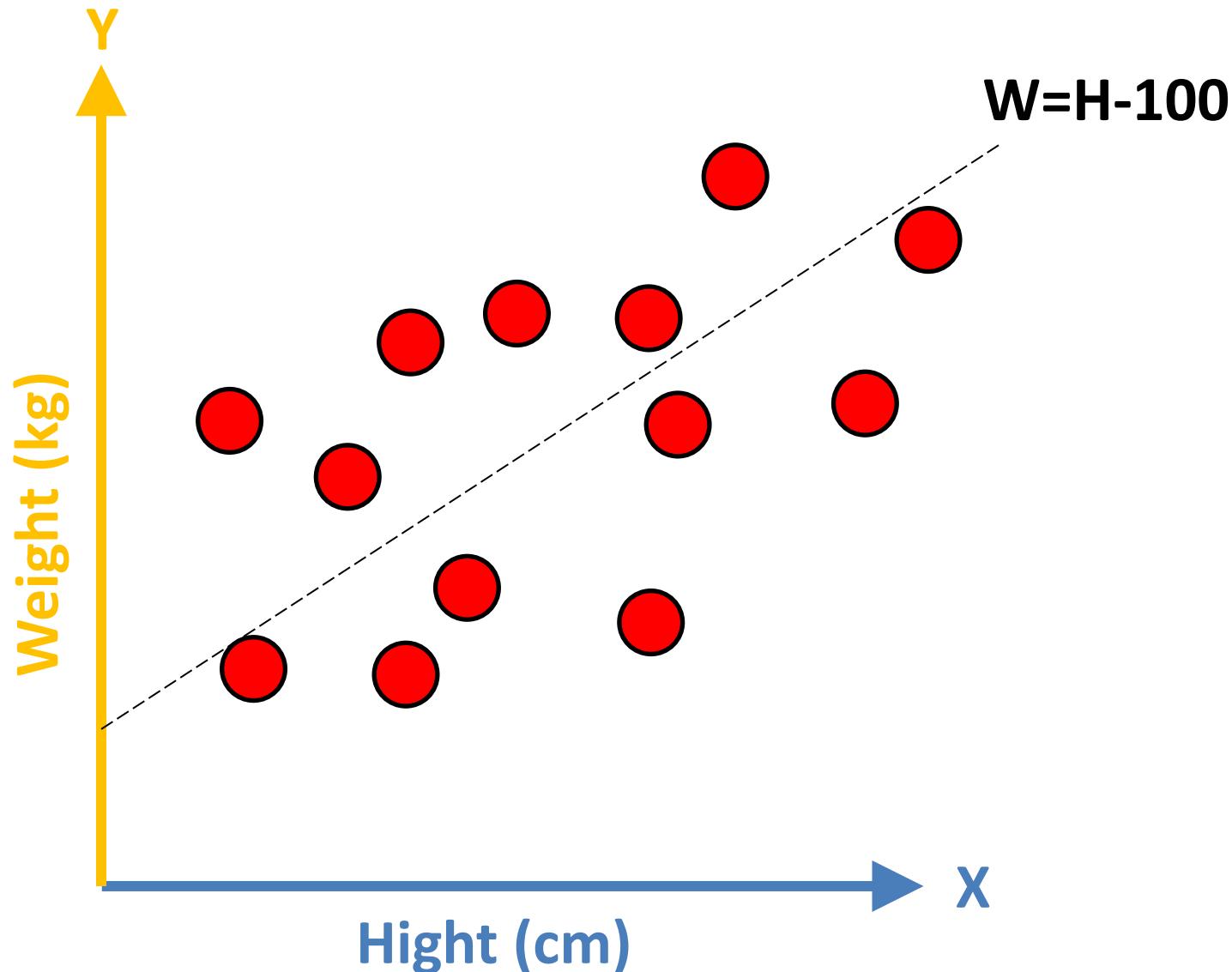
Machine Learning

- Herbert Alexander Simon: “ Learning is any process by which a **system** improves performance from **experience**.”
- “Machine Learning is concerned with **computer programs** that automatically improve their performance through experience.”

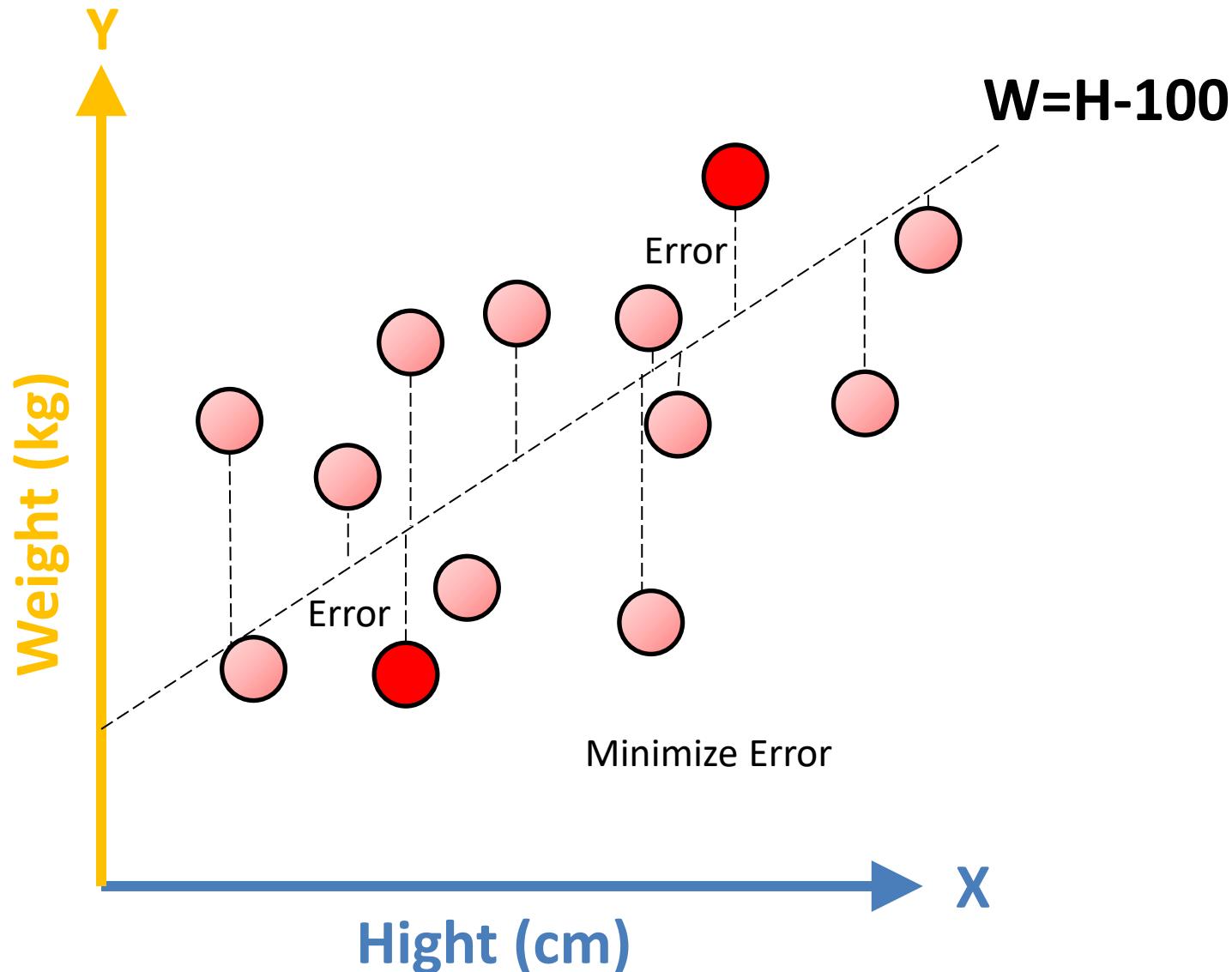


Herbert Simon
[Turing Award 1975](#)
[Nobel Prize in Economics 1978](#)

Example of Machine Learning



Example of Machine Learning



What is Deep Learning

Deep learning is a particular kind of machine learning that is **inspired by the functionality of brain cells called neurons** which led to the concept of artificial neural network.



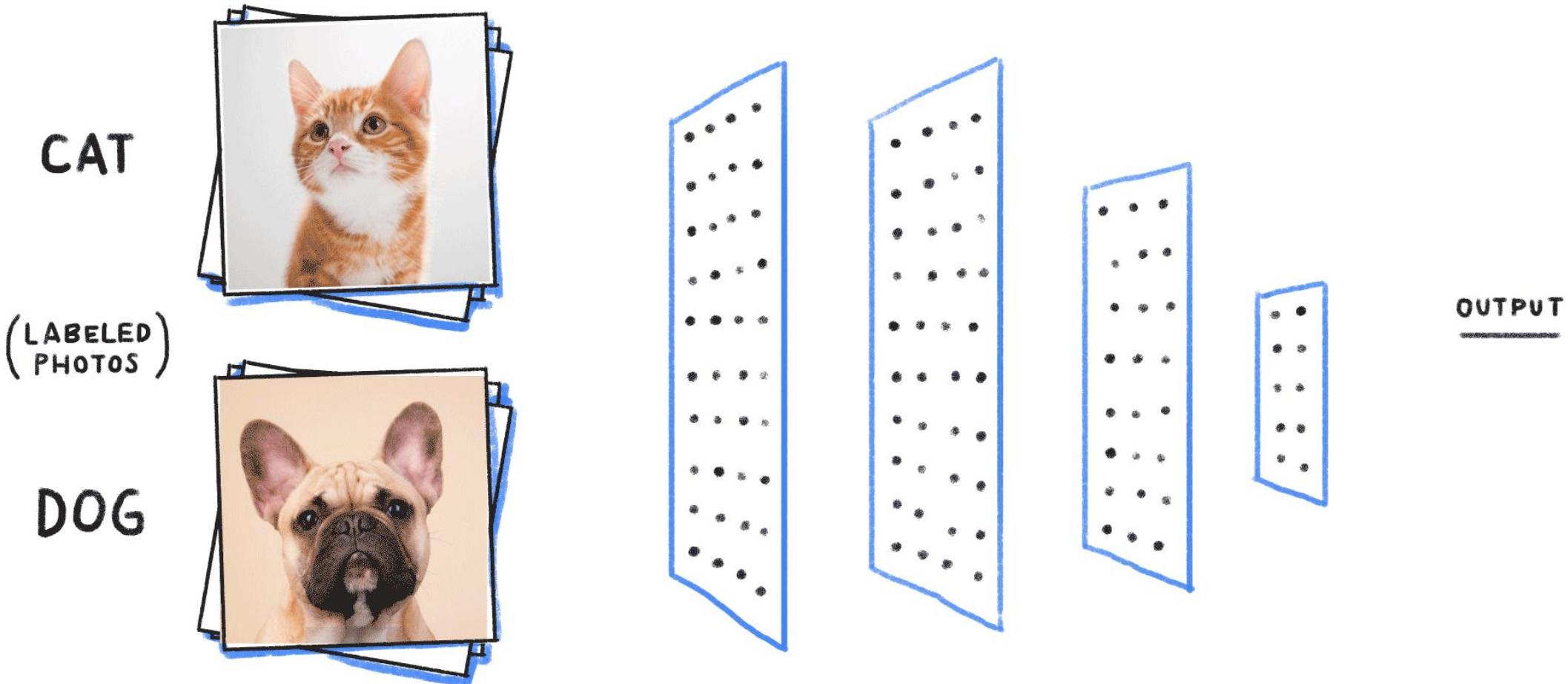
Example of Deep Learning



- Sides = 4 ✓
- Closed ✓
- Perpendicular ✓
- Equal sizes ✓

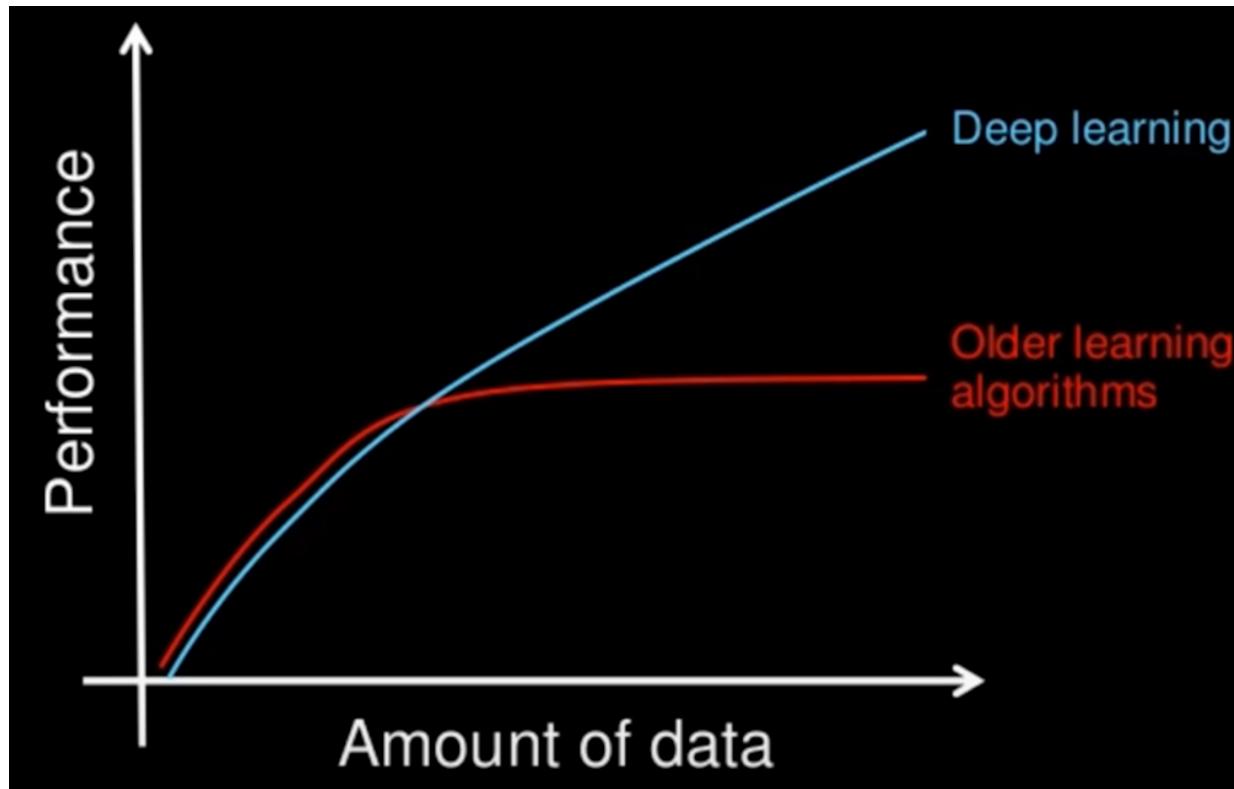
Well, it is nothing but a nested hierarchy of basic concepts.

Example of Deep Learning



Deep Learning Vs Machine Learning

Deep learning **is** a machine learning



Data dependency

Deep learning depend on the **high-end machine** where
Machine learning can work with a **low-end machine**

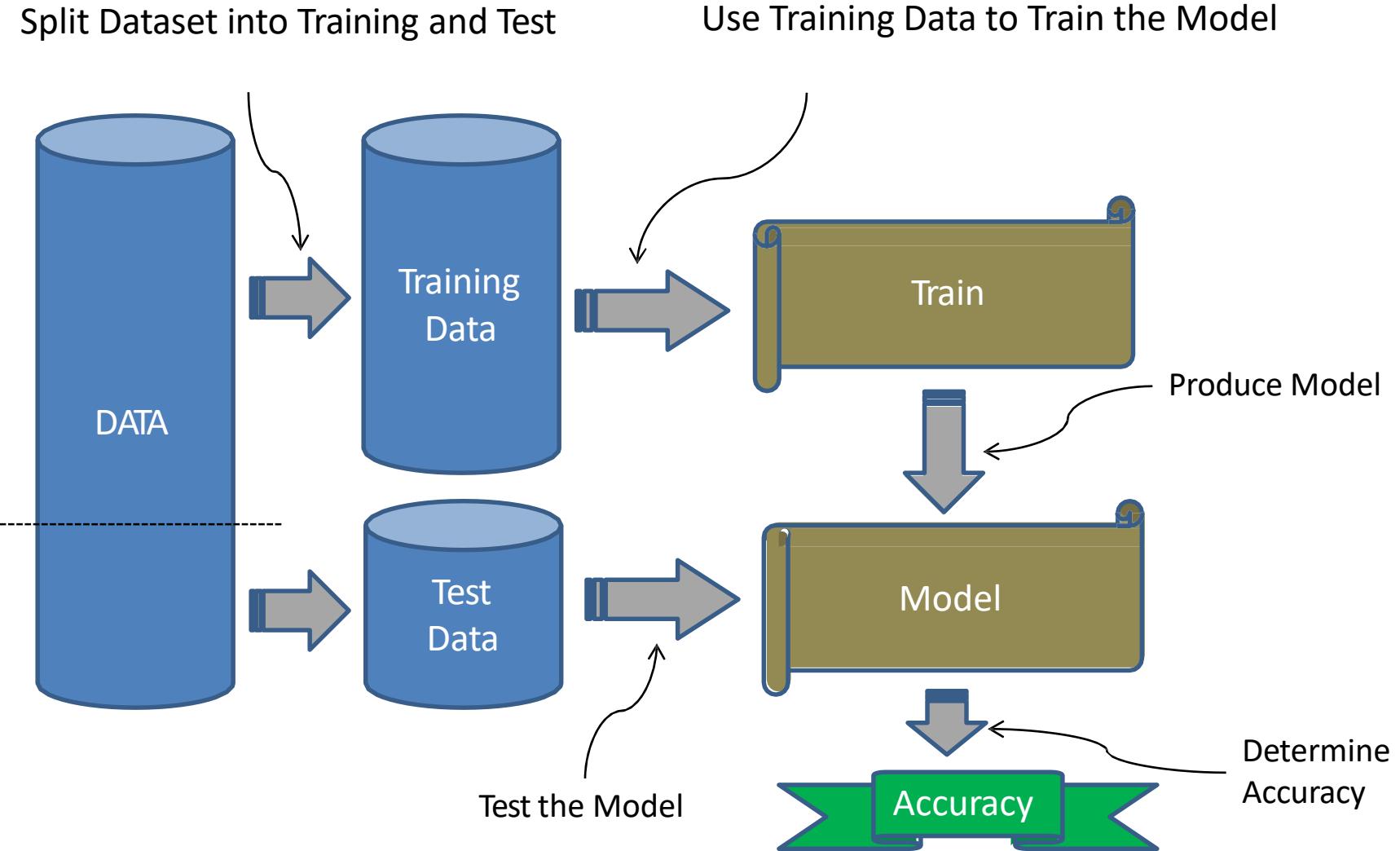


Hardware dependency

Splitting Datasets

- To use a dataset in Machine Learning, the dataset is split into a **training and test set**.
- The training set is **used to train the model**.
- The test set is **used to test the accuracy of the model**.
- Typically, **split 80% training, 20% test**.

It's About Training



Serial Splitting of the Dataset

The simplest method of splitting data is to **split it serially**.

- Take the **first 80% rows** and put them into the **training set**.
- Take the **remaining 20% rows** and put them into the **test set**.

```
import pandas as pd          # pandas library
dataset = pd.read_csv("Data.csv") # read in data as panda dataframe

nrows = dataset.shape[ 0 ]      # property shape[ 0 ] is the number of rows

train = dataset.iloc[ 1: int(nrows * .8) , : ]
test  = dataset.iloc[int(nrows * .8) +1, nrows, : ]
```

80% rows 20% rows all columns

Random Splitting of the Dataset

Another method is to pick rows at **random**.

- Sci-kit learn has a built-in method

```
from sklearn.cross_validation import train_test_split  
  
ncols = dataset.shape[ 1 ]          # property shape[ 0 ] is the number of columns  
  
# Assume label is the last column in the dataset  
X = dataset.iloc[ :, :-1 ]           # X is all the features (exclude last column)  
y = dataset.iloc[ :, ncols ]         # Y is the label (last column)  
  
# Split the data, with 80% train and 20% test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

↑
split size

↑
seed for random
Number generator

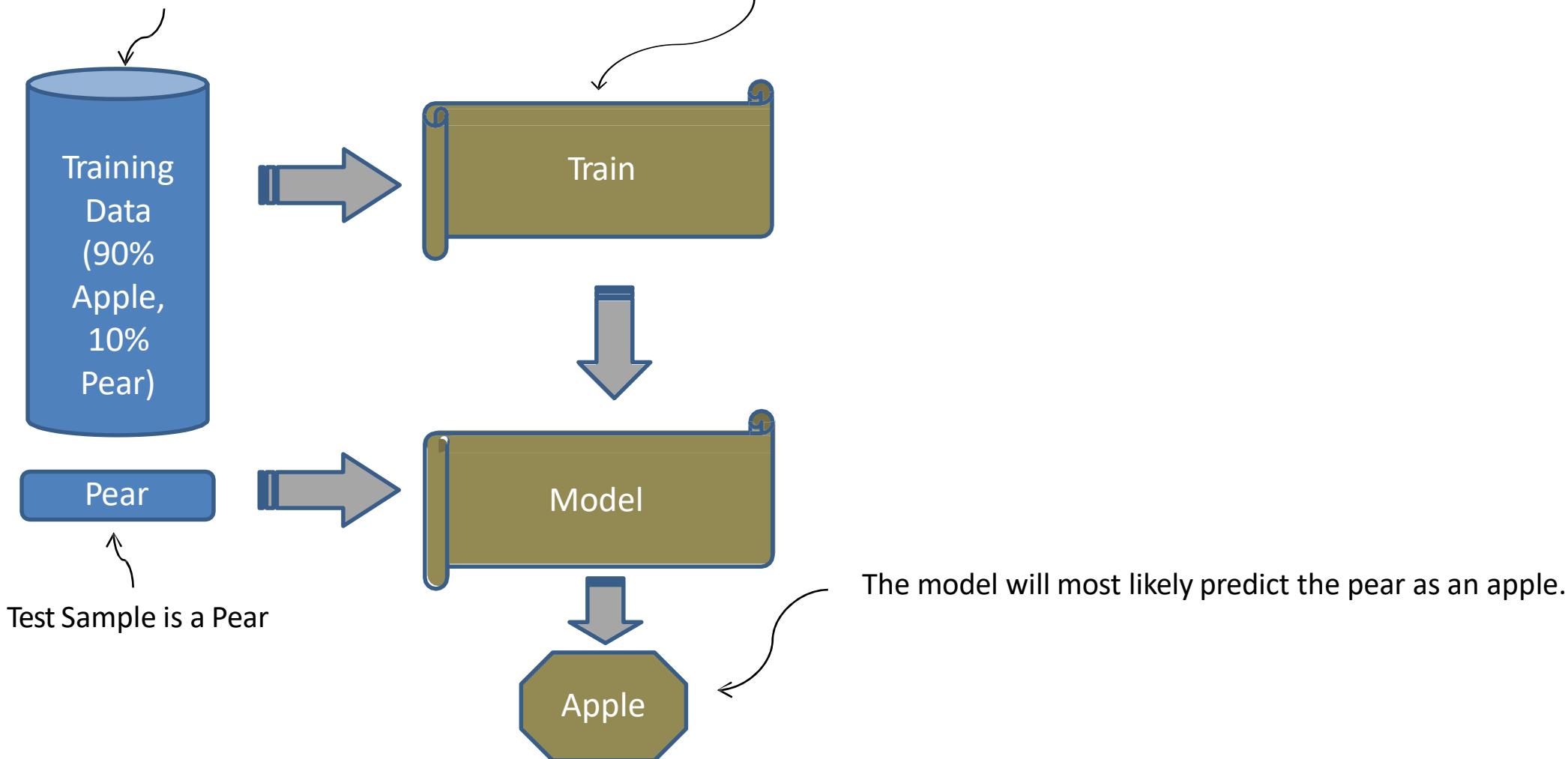
Data Imbalance-Overfitting

- If the training data is overly **unbalanced**, then the model will predict a non-meaningful result.
- For example, if the model is a binary classifier (e.g., apple vs. pear), and nearly all the samples are of the same label (e.g., apple), then the model will simply learn that everything is a label (apple).
- This is called **overfitting**. To prevent overfitting, there needs to be a fairly equal distribution of training samples for each classification, or range if the label is a real value.

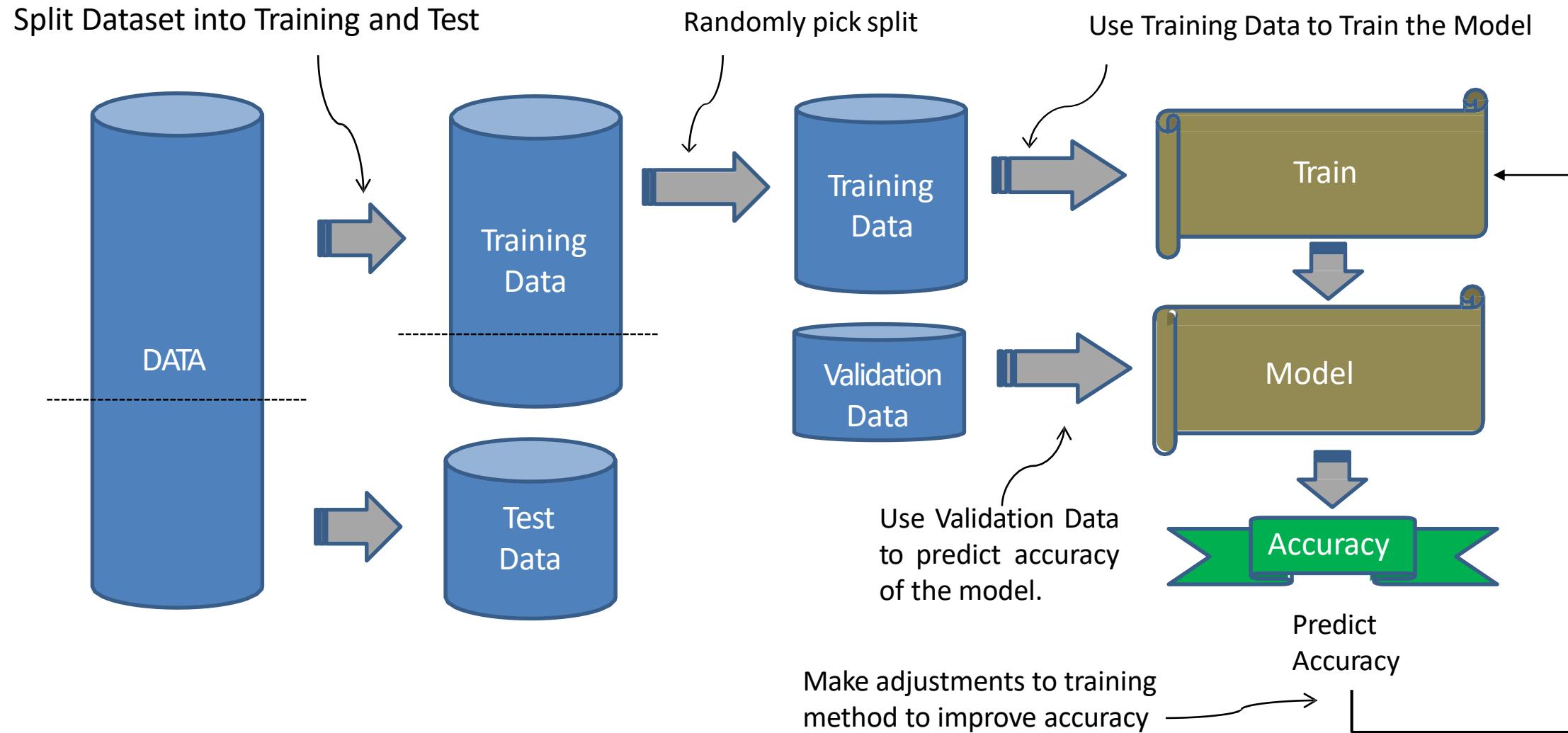
Data Imbalance-Overfitting

Nearly all samples are an Apple

In an imbalance, the model will fit itself to the imbalance, not the predictor.



Cross Validation



K-Fold Cross Validation

- K-Fold is a well-known form of cross-validation.
- Steps:
 1. Partition the dataset into **k equal-sized partitions**.
 2. Select **one partition** as the **validation data**.
 3. Use the **remaining k-1** as the **training data**.
 4. Train the model and determine accuracy from the validation data.
 5. Repeat the process **k** times, selecting a different partition each time for the validation data.
 6. Average the accuracy results.

Types of Machine Learning



Supervised Learning



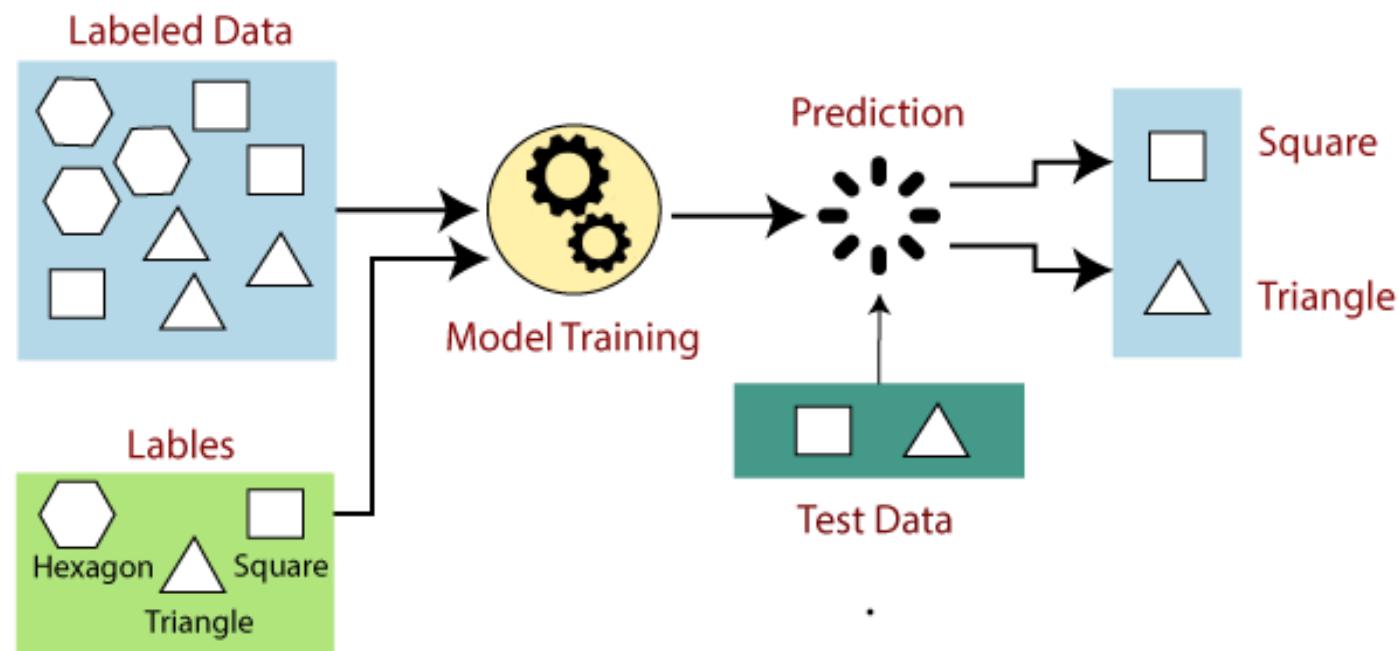
Unsupervised Learning



Reinforcement Learning

Supervised Learning

- The computer learns by making use of **labeled data**.
- In supervised learning, the algorithm “**learns**” from the training dataset by iteratively **making predictions** on the **data** and adjusting for the correct answer.



Example IRIS Dataset

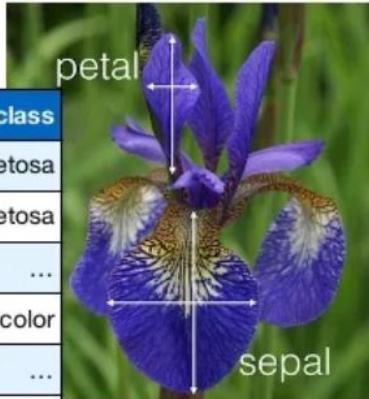
Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
...
50	6.4	3.2	4.5	1.5	vericolor
...
150	5.9	3.0	5.1	1.8	virginica

Features (attributes, dimensions)

IRIS

<https://archive.ics.uci.edu/ml/datasets/Iris>



Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \dots, y_N]$$

The loan data (Example)

Approved or not

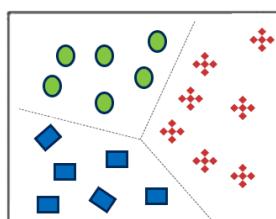
ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Supervised Learning

- The computer learns by making use of **labeled data**.
- In supervised learning, the algorithm “**learns**” from the **training dataset** by iteratively **making predictions** on the data and adjusting for the correct answer.

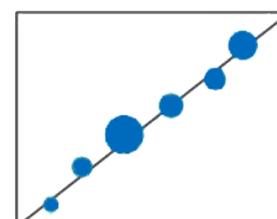
SUPERVISED LEARNING

EXAMPLES



Classification

Is this data input
red, blue or green?



Regression

What is the impact of
product price on number of
sales?
What is the impact of years
of experience on salary?

Applications of supervised Learning

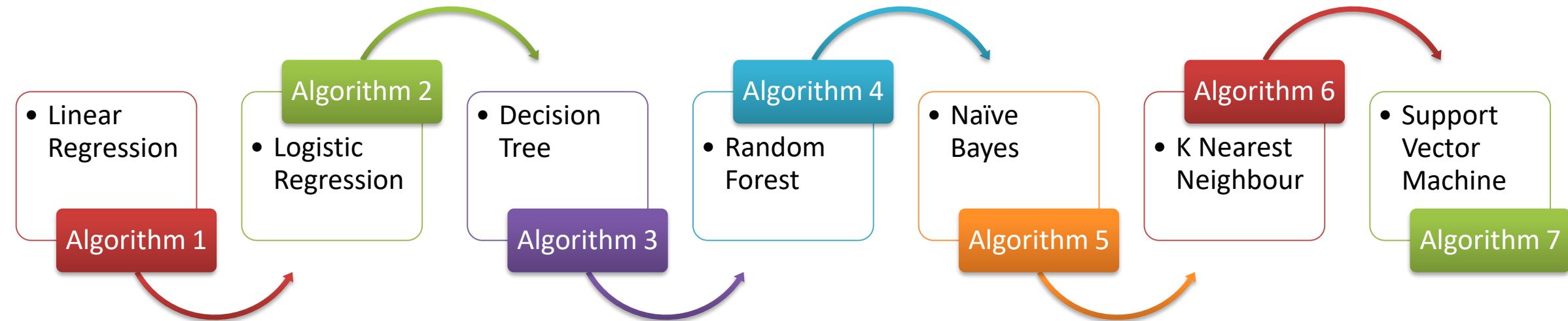


Risk Evaluation



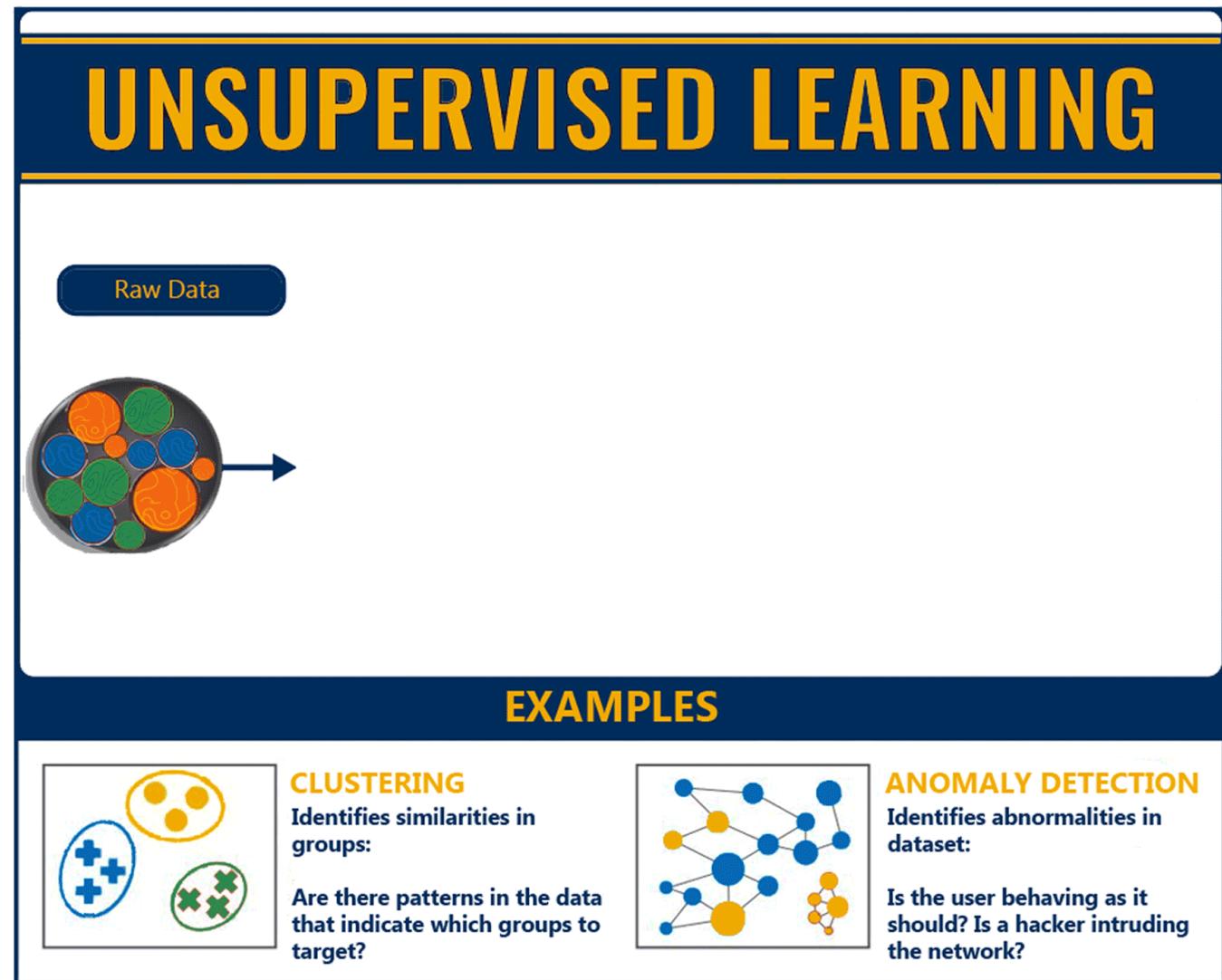
Forecast Sales

Popular Algorithms in Supervised Machine Learning

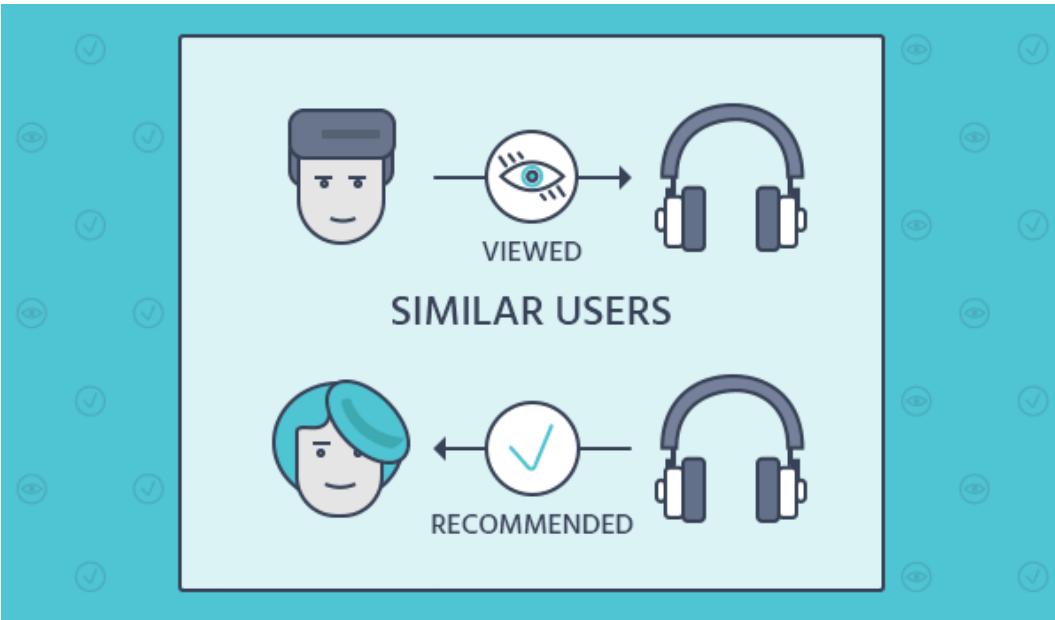


Unsupervised Learning

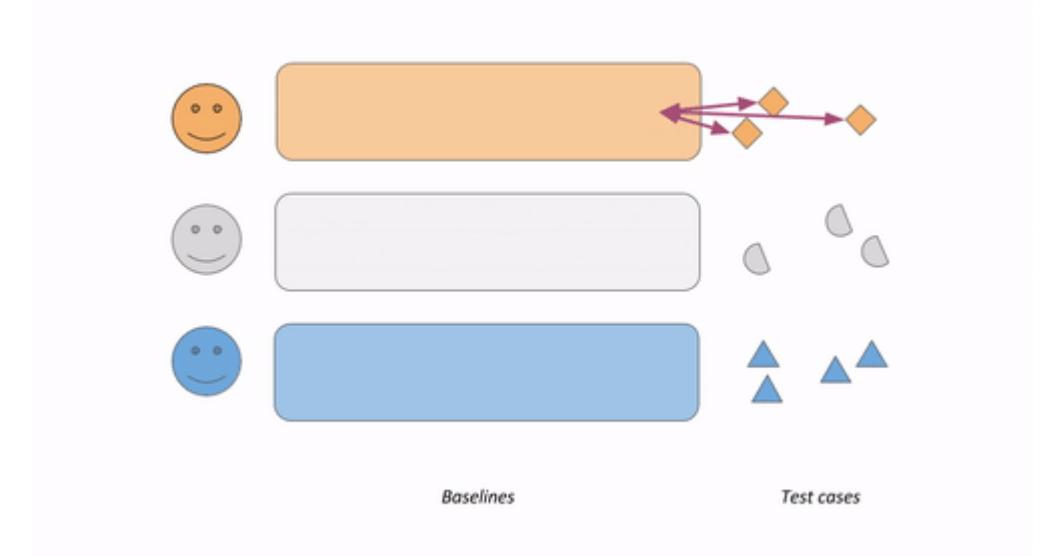
- Computer learns by making use of unlabeled data without any guidance.
- It uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention.



Applications of Unsupervised Learning



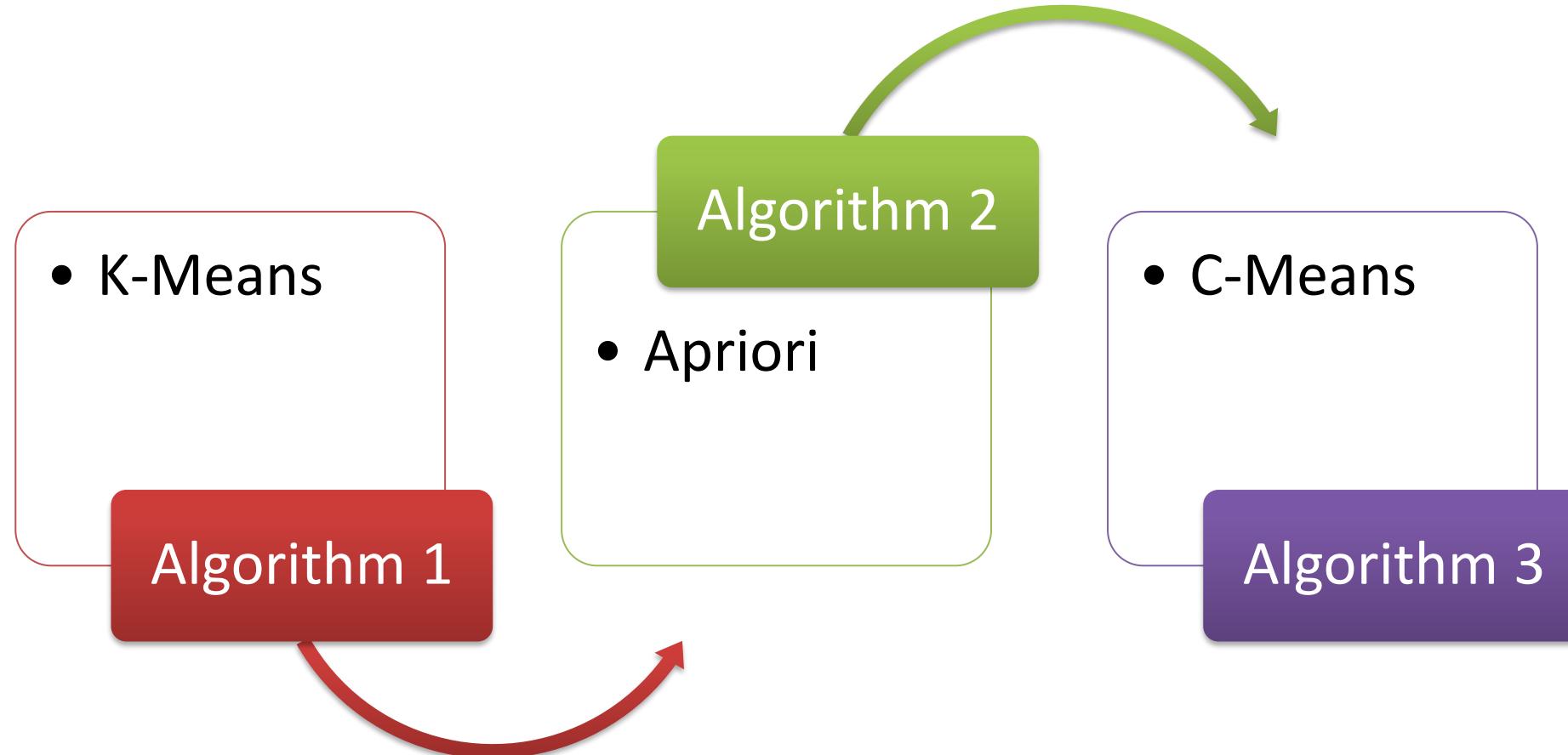
Recommendation Systems



Anomaly Detection

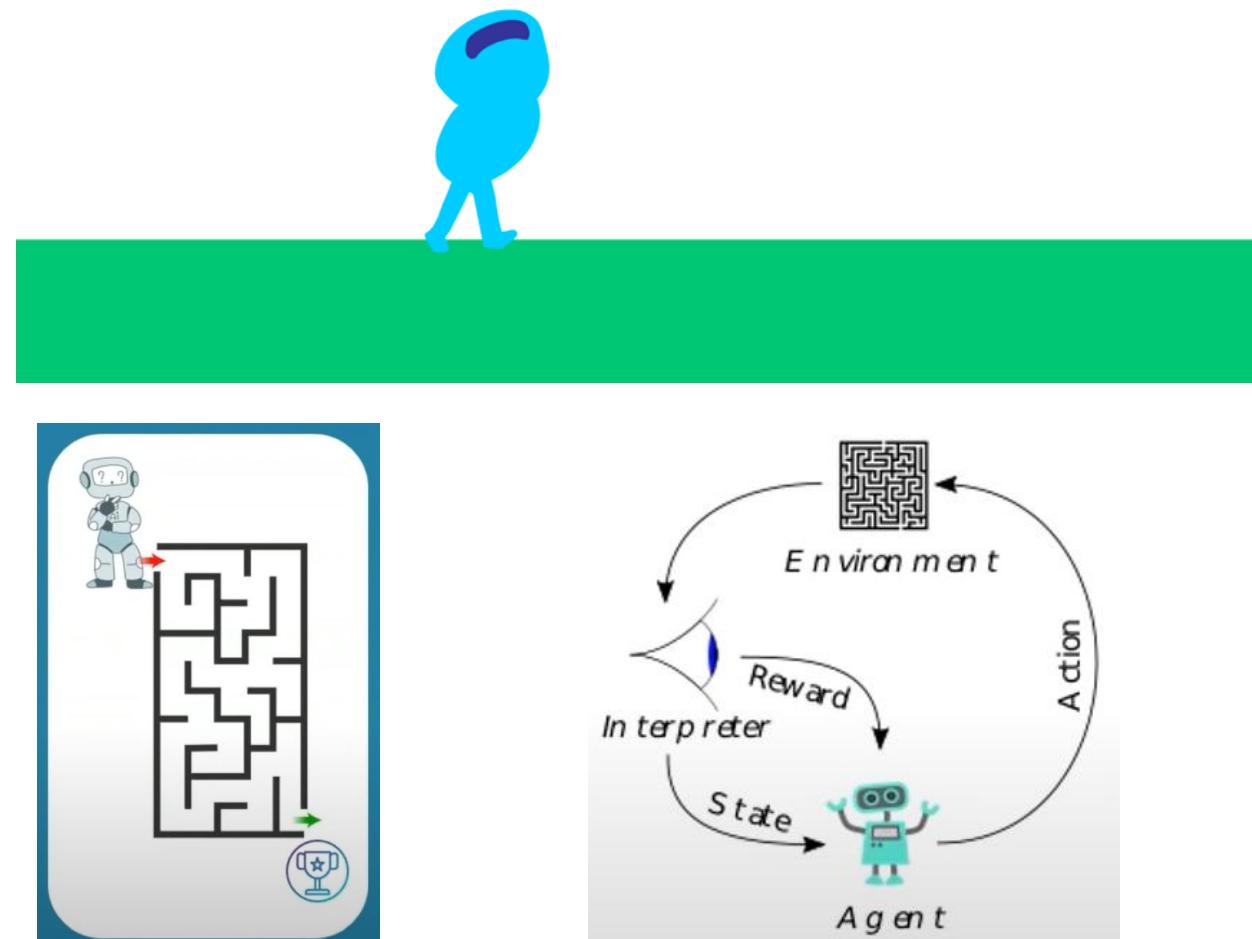
Anomaly detection (aka outlier analysis) is a step in data mining that identifies data points, events, and/or observations that deviate from a dataset's

Popular Algorithms in Unsupervised Machine Learning



Reinforcement Learning

- In Reinforcement learning an **agent** interacts with its environment by producing actions & discovers errors or rewards.
- In reinforcement learning there is **no predefined data given** and agents learn most of the time from the environment.
- No supervision



Applications of Reinforcement Learning

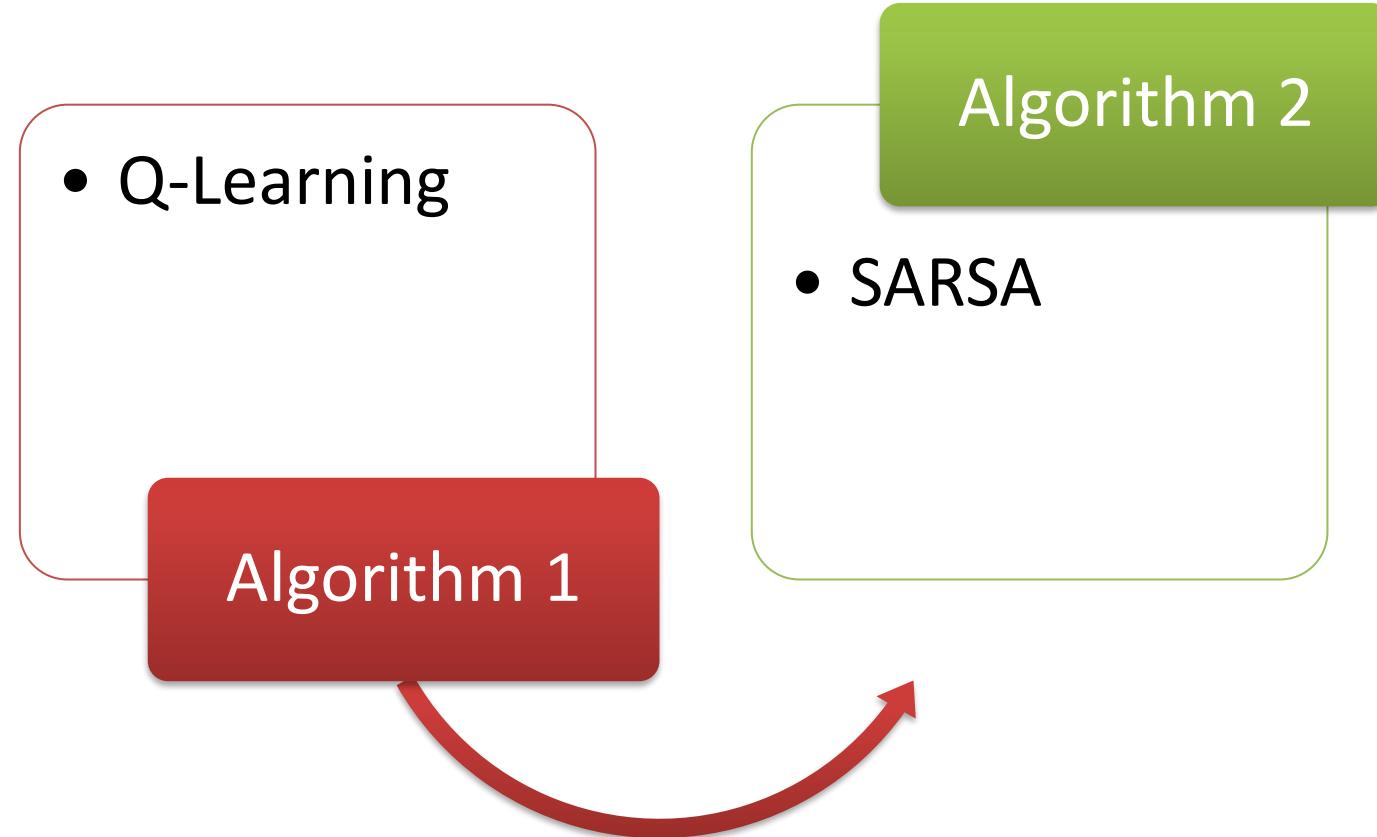


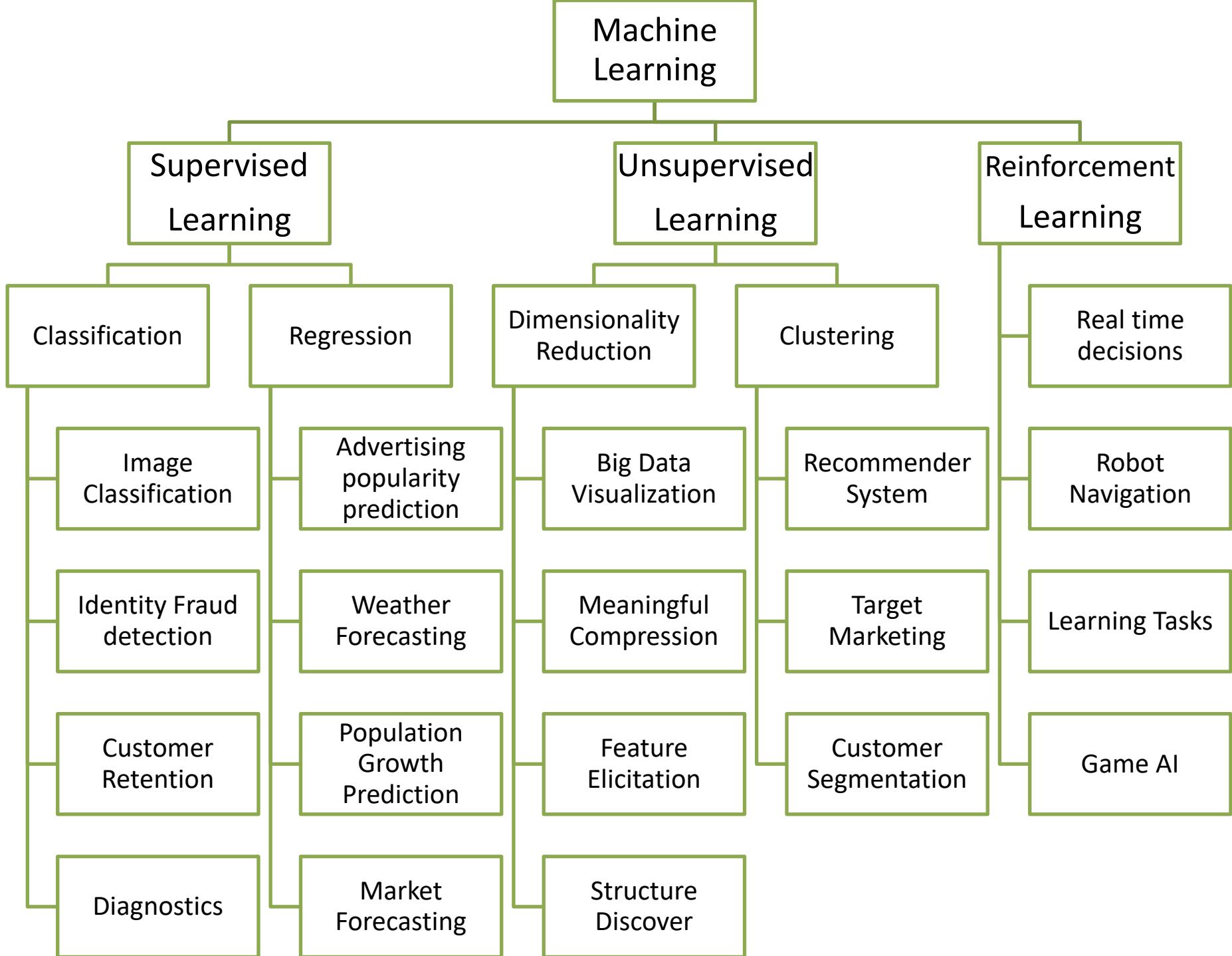
Self driving cars



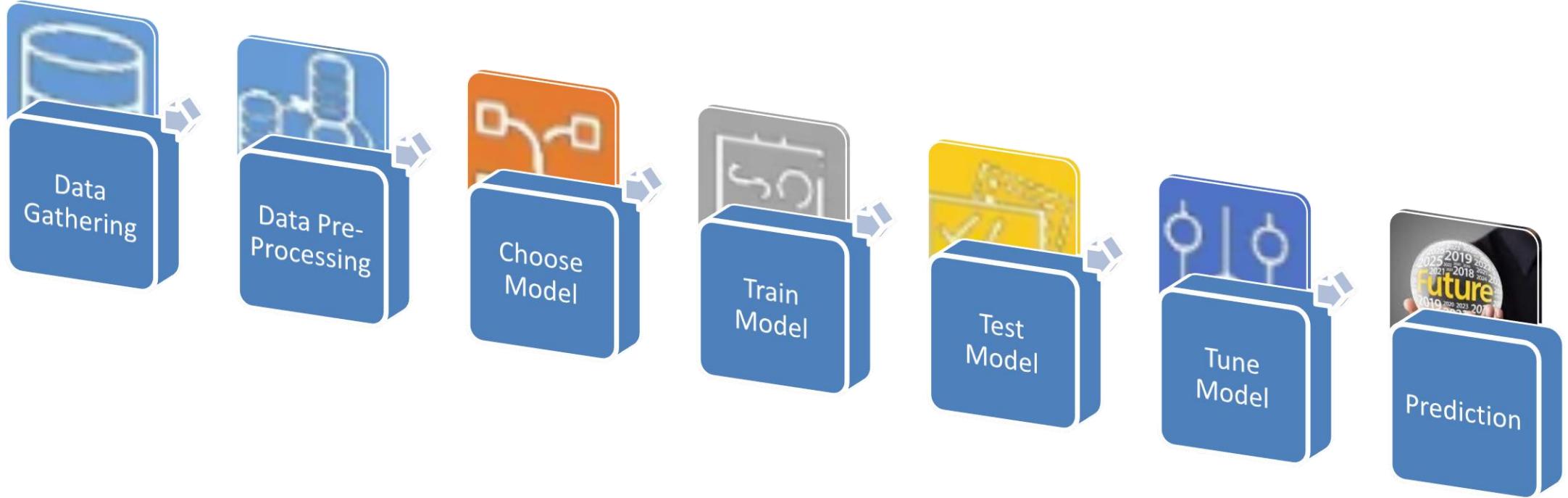
Gaming

Popular Algorithms in Reinforcement Machine Learning





Processes involved in Machine Learning



Introduction to SciKit-Learn library for machine learning algorithms

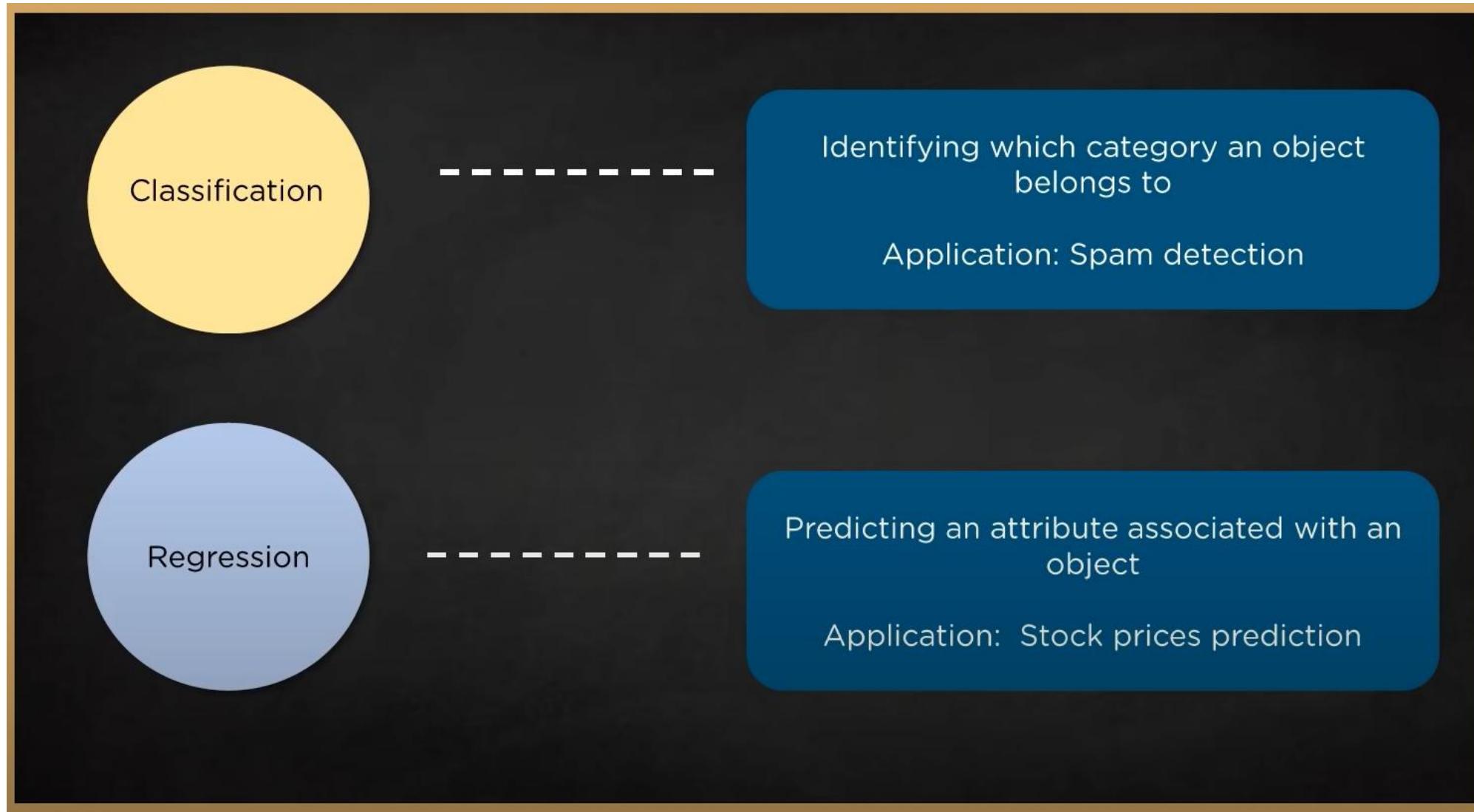


What is Scikit-learn?

- Popular machine learning toolkit in Python.
- Simple and efficient tool for data mining and data analysis.
- Built on NumPy, SciPy, and matplotlib.
- Open source, commercially usable-BSD license.



What we can achieve using scikit-learn



What we can achieve using scikit-learn

Clustering

Automatic grouping of similar objects into sets

Application: Customer segmentation

Model Selection

Comparing, validating and choosing parameters and models

Application: Improving model accuracy via parameter tuning



What we can achieve using scikit-learn

Dimensionality reduction



Reducing the number of random variables to consider

Application: To increase model efficiency

Pre-processing



Feature extraction and normalization

Application: Transforming input data such as text for use with machine learning algorithms

Python Scikit-learn Library

scikit-learn.org/stable/

City University BOOK DOWNLOADS conference educational websites Conferences IGA Reminder_links HKU Research IGA Google Scholar Gmail: Email from G... XE - The World's Fa... Home Feed Sci-Hub: removing... Other bookmarks

scikit learn Install User Guide API Examples Community More ▾

Go

scikit-learn

Machine Learning in Python

Getting Started Release Highlights for 1.1 GitHub

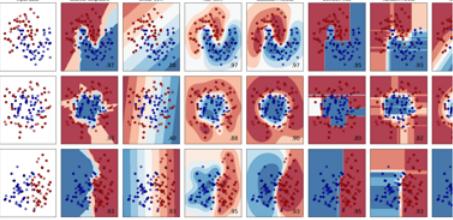
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



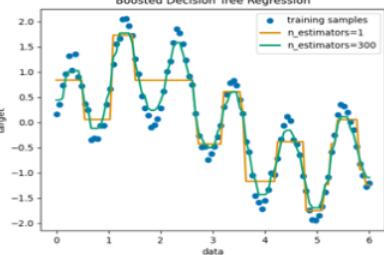
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



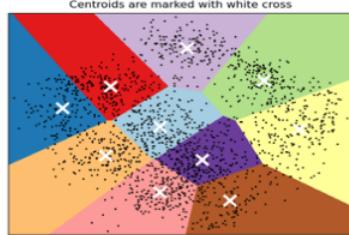
Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tun-

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

<https://scikit-learn.org/stable/>

 scikit-learn

Install User Guide API Examples Community More ▾

Prev Up Next

scikit-learn 1.1.2 Other versions

Please cite us if you use the software.

User Guide

1. Supervised learning

- 1.1. Linear Models
- 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- 1.4. Support Vector Machines
- 1.5. Stochastic Gradient Descent
- 1.6. Nearest Neighbors
- 1.7. Gaussian Processes
- 1.8. Cross decomposition
- 1.9. Naive Bayes
- 1.10. Decision Trees
- 1.11. Ensemble methods
- 1.12. Multiclass and multioutput algorithms
- 1.13. Feature selection
- 1.14. Semi-supervised learning
- 1.15. Isotonic regression
- 1.16. Probability calibration
- 1.17. Neural network models (supervised)

2. Unsupervised learning

3. Model selection and evaluation

4. Inspection

5. Visualizations

6. Dataset transformations

7. Dataset loading utilities

8. Computing with scikit-learn

9. Model persistence

10. Common pitfalls and recommended practices

1. Supervised learning

1.1. Linear Models

- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Generalized Linear Regression
- 1.1.13. Stochastic Gradient Descent - SGD
- 1.1.14. Perceptron
- 1.1.15. Passive Aggressive Algorithms
- 1.1.16. Robustness regression: outliers and modeling errors
- 1.1.17. Quantile Regression
- 1.1.18. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage and Covariance Estimator
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines

- 1.4.1. Classification

rg/stable/unsupervised_learning.html

VLOADS conference educational websites Conferences IGA Reminder_links HKU Research IGA Google Scholar Gmail: Email from G... XE - The World's Fa... R Home Feed Sci-Hub: re

 scikit-learn Install User Guide API Examples Community More ▾

Prev Up Next

scikit-learn 1.1.2
Other versions

Please cite us if you use the software.

User Guide

- 1. Supervised learning
- 2. Unsupervised learning**
- 2.1. Gaussian mixture models
- 2.2. Manifold learning
- 2.3. Clustering**
- 2.4. Biclustering
- 2.5. Decomposing signals in components (matrix factorization problems)
- 2.6. Covariance estimation
- 2.7. Novelty and Outlier Detection
- 2.8. Density Estimation
- 2.9. Neural network models (unsupervised)
- 3. Model selection and evaluation
- 4. Inspection
- 5. Visualizations
- 6. Dataset transformations
- 7. Dataset loading utilities
- 8. Computing with scikit-learn
- 9. Model persistence
- 10. Common pitfalls and recommended practices

Toggle Menu

2. Unsupervised learning

2.1. Gaussian mixture models

- 2.1.1. Gaussian Mixture
- 2.1.2. Variational Bayesian Gaussian Mixture

2.2. Manifold learning

- 2.2.1. Introduction
- 2.2.2. Isomap
- 2.2.3. Locally Linear Embedding
- 2.2.4. Modified Locally Linear Embedding
- 2.2.5. Hessian Eigenmapping
- 2.2.6. Spectral Embedding
- 2.2.7. Local Tangent Space Alignment
- 2.2.8. Multi-dimensional Scaling (MDS)
- 2.2.9. t-distributed Stochastic Neighbor Embedding (t-SNE)
- 2.2.10. Tips on practical use

2.3. Clustering

- 2.3.1. Overview of clustering methods
- **2.3.2. K-means**
- 2.3.3. Affinity Propagation
- 2.3.4. Mean Shift
- 2.3.5. Spectral clustering
- 2.3.6. Hierarchical clustering
- 2.3.7. DBSCAN
- 2.3.8. OPTICS
- 2.3.9. BIRCH
- 2.3.10. Clustering performance evaluation

2.4. Biclustering

- 2.4.1. Spectral Co-Clustering
- 2.4.2. Spectral Biclustering
- 2.4.3. Biclustering evaluation

How to import it on Jupyter Notebook

```
1 #Importing required packages.  
2 import pandas as pd  
3 import seaborn as sns  
4 import matplotlib.pyplot as plt  
5 from sklearn.ensemble import RandomForestClassifier  
6 from sklearn.svm import SVC  
7 from sklearn import svm  
8 from sklearn.neural_network import MLPClassifier  
9 #from sklearn.linear_model import SGDClassifier  
10 from sklearn.metrics import confusion_matrix, classification_report  
11 from sklearn.preprocessing import StandardScaler, LabelEncoder  
12 from sklearn.model_selection import train_test_split  
13 %matplotlib inline
```

Supervised Learning Algorithms

Linear and Logistic Regression

Regression

- Regression Analysis is a **predictive modeling technique**.
- It estimates the relationship between a **dependent (target)** and an **independent variable (predictor)**.

Uses of Regression

Three major uses for regression analysis are

- Determining the strength of predictors
- Forecasting an effect, and
- Trend forecasting.

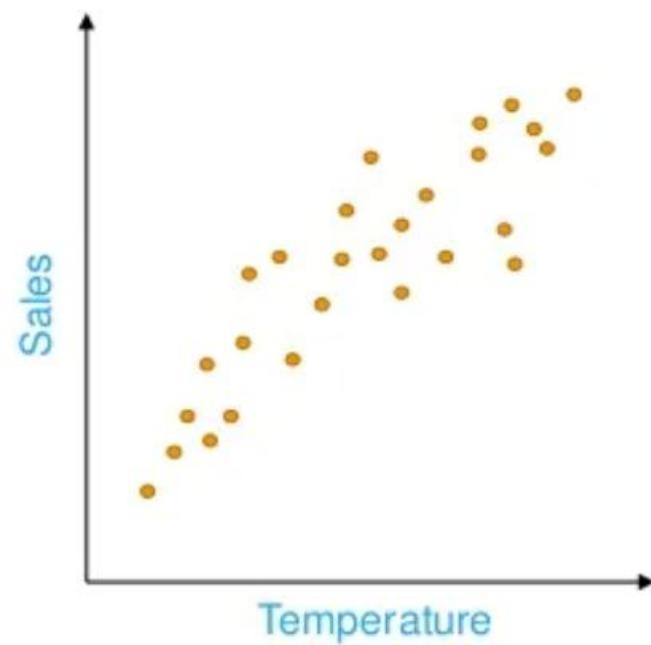
Types of Regression

- Linear Regression
- Logistic Regression
- Polynomial Regression

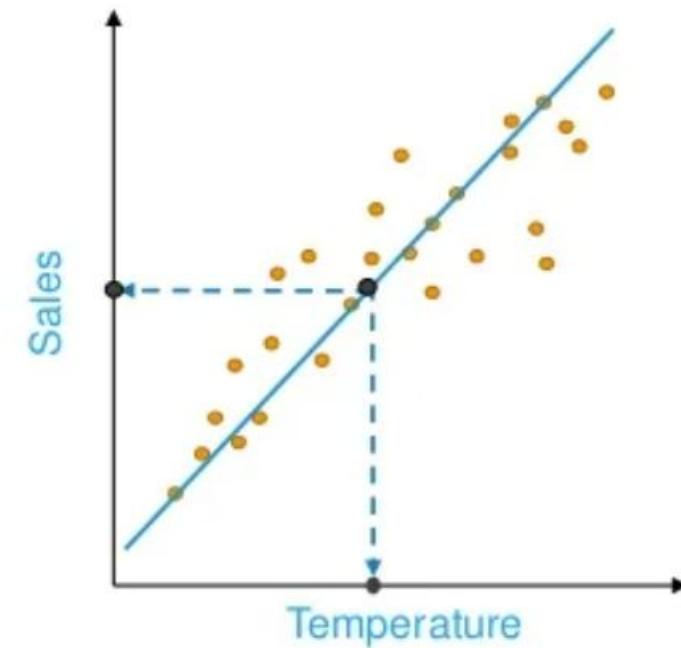
Basis	Linear Regression	Logistic Regression
Core Concept	The data is modelled using a straight line	The probability of some obtained event is represented as a linear function of a combination of predictor variables.
Used with	Continuous Variable	Categorical Variable
Output/Prediction	Value of the variable	Probability of occurrence of event
Accuracy and Goodness of fit	measured by loss, R squared, Adjusted R squared etc.	Accuracy, Precision, Recall, F1 score, ROC curve, Confusion Matrix, etc

Linear Regression

- Linear regression is a linear modelling approach to find relationship between one or more independent variables (predictors) denoted as X and dependent variable (target) denoted as Y.



Plotting the sales of ice cream based on temperature

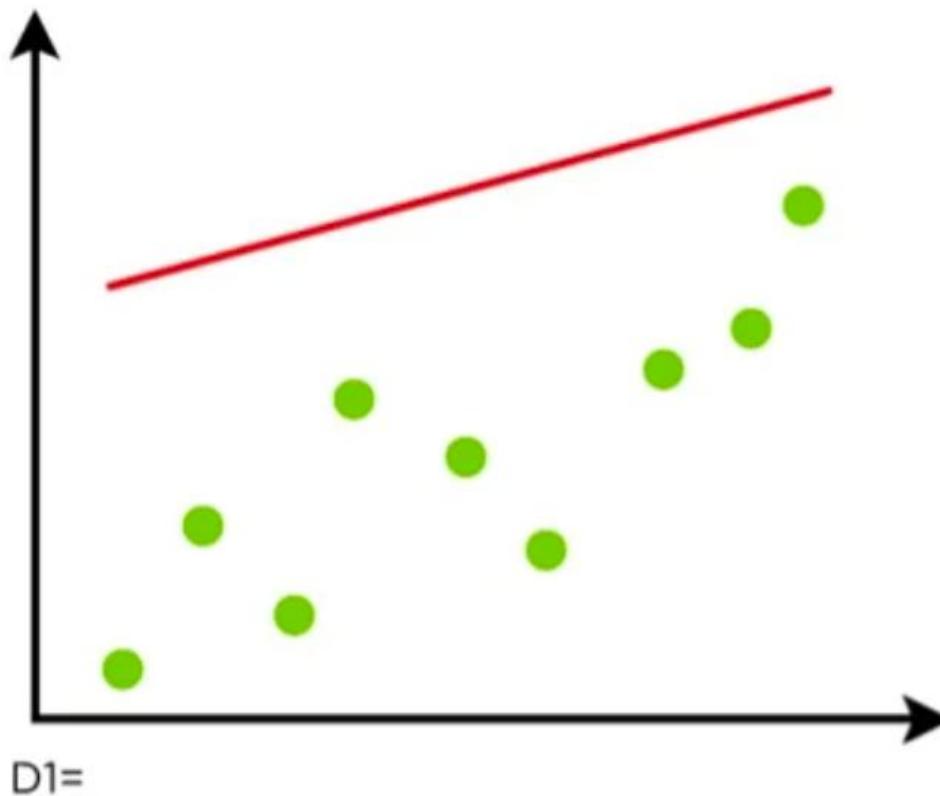


Regression line to predict the sales

Predict sales of Ice Cream based on temperature.

Linear Regression

- **Finding the best fit line:** The best fit line can be found by **minimizing the distance between all the data points and the distance to the regression line**. Ways to minimize this distance are the sum of squared errors, the sum of absolute errors, etc.

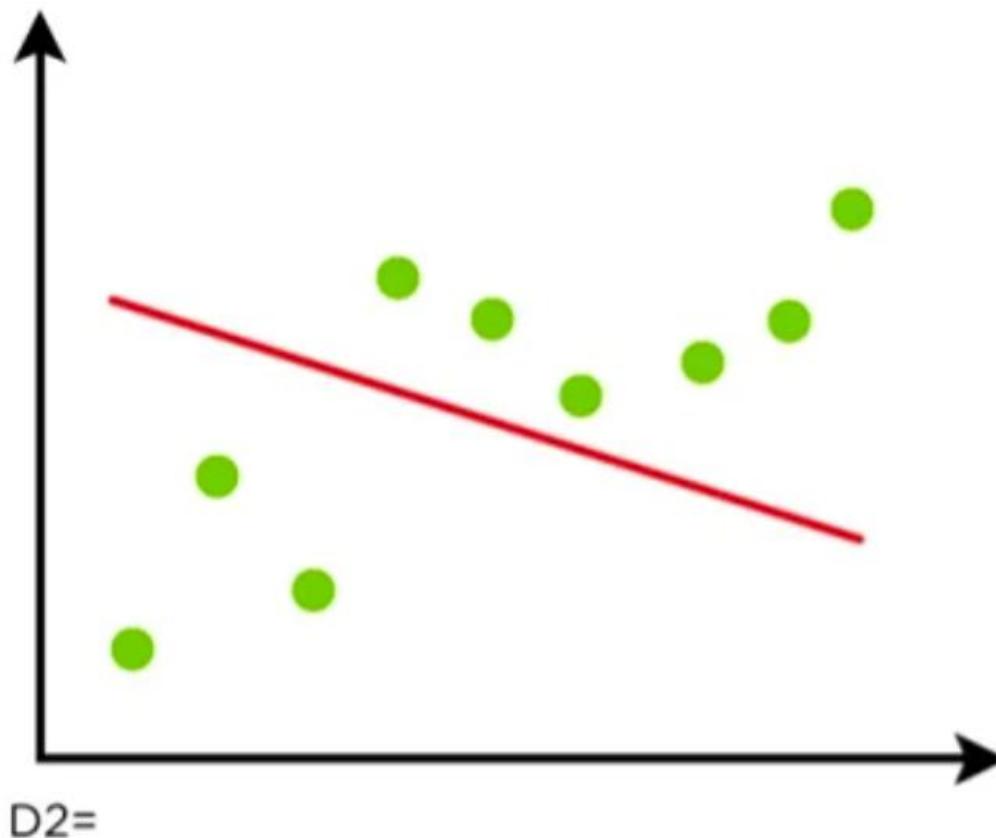


Not the best fit line

Distance between the data points
and the line is maximum

Linear Regression

- **Finding the best fit line:** The best fit line can be found by **minimizing the distance between all the data points and the distance to the regression line**. Ways to minimize this distance are the sum of squared errors, the sum of absolute errors, etc.

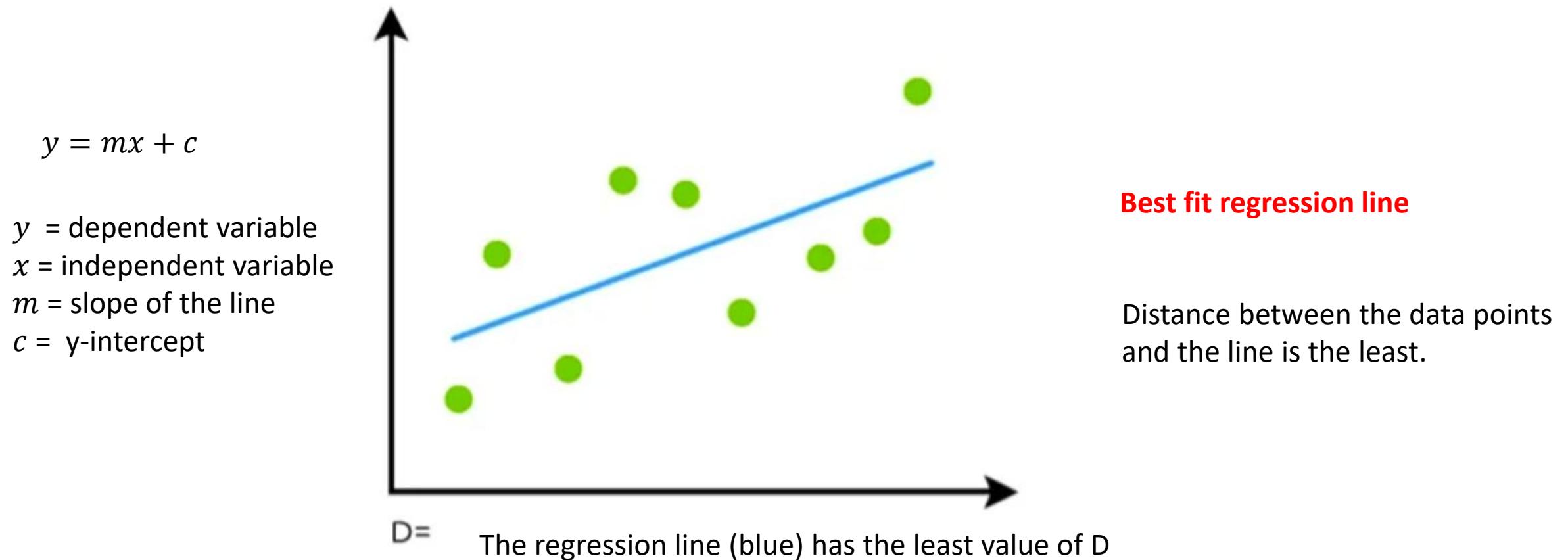


Not the best fit line

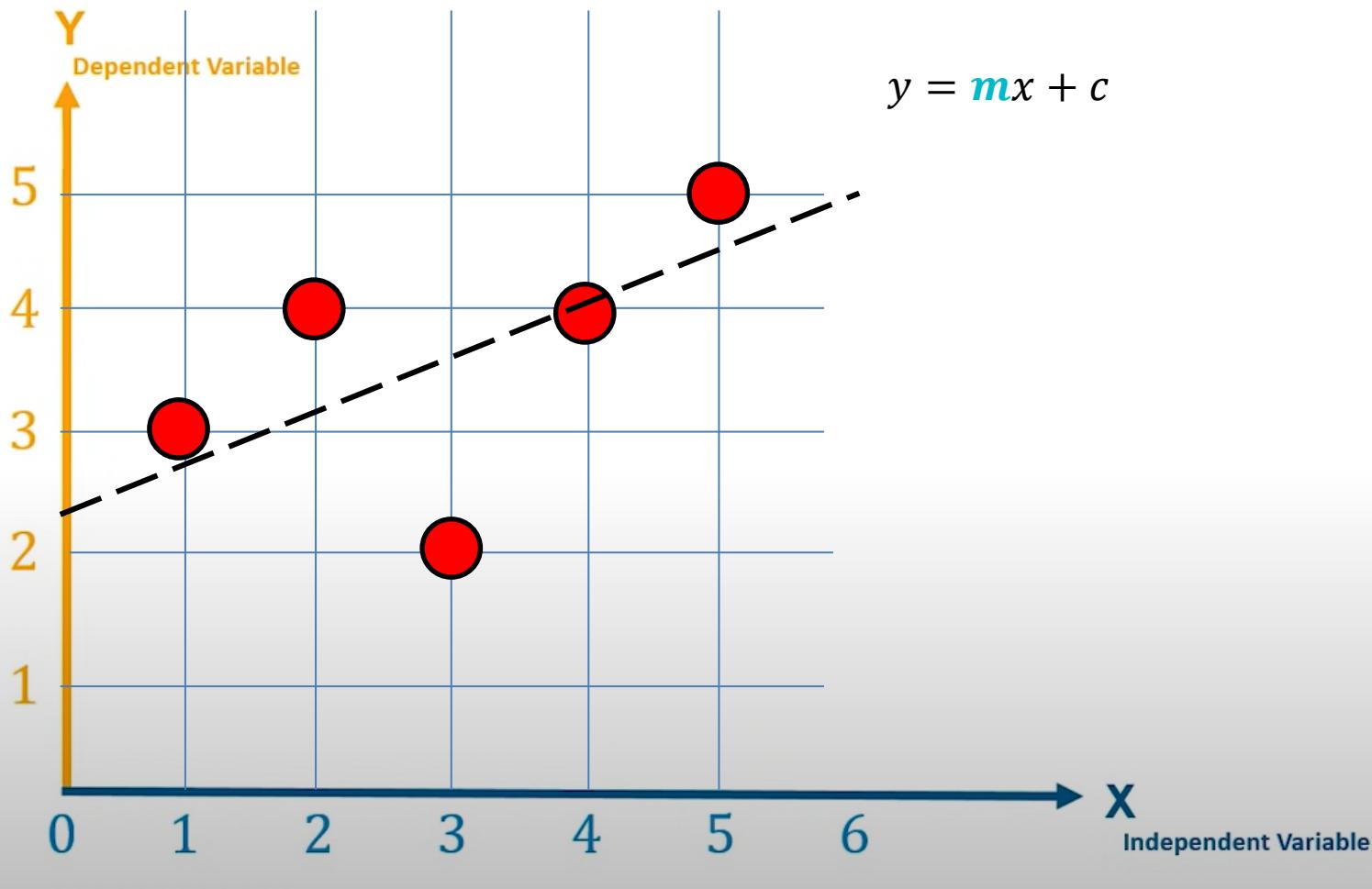
Distance between the data points
and the line can still be reduced.

Linear Regression

- **Finding the best fit line:** The best fit line can be found by **minimizing the distance between all the data points and the distance to the regression line**. Ways to minimize this distance are the sum of squared errors, the sum of absolute errors, etc.



Understanding Linear Regression



x	y
1	3
2	4
3	2
4	4
5	5

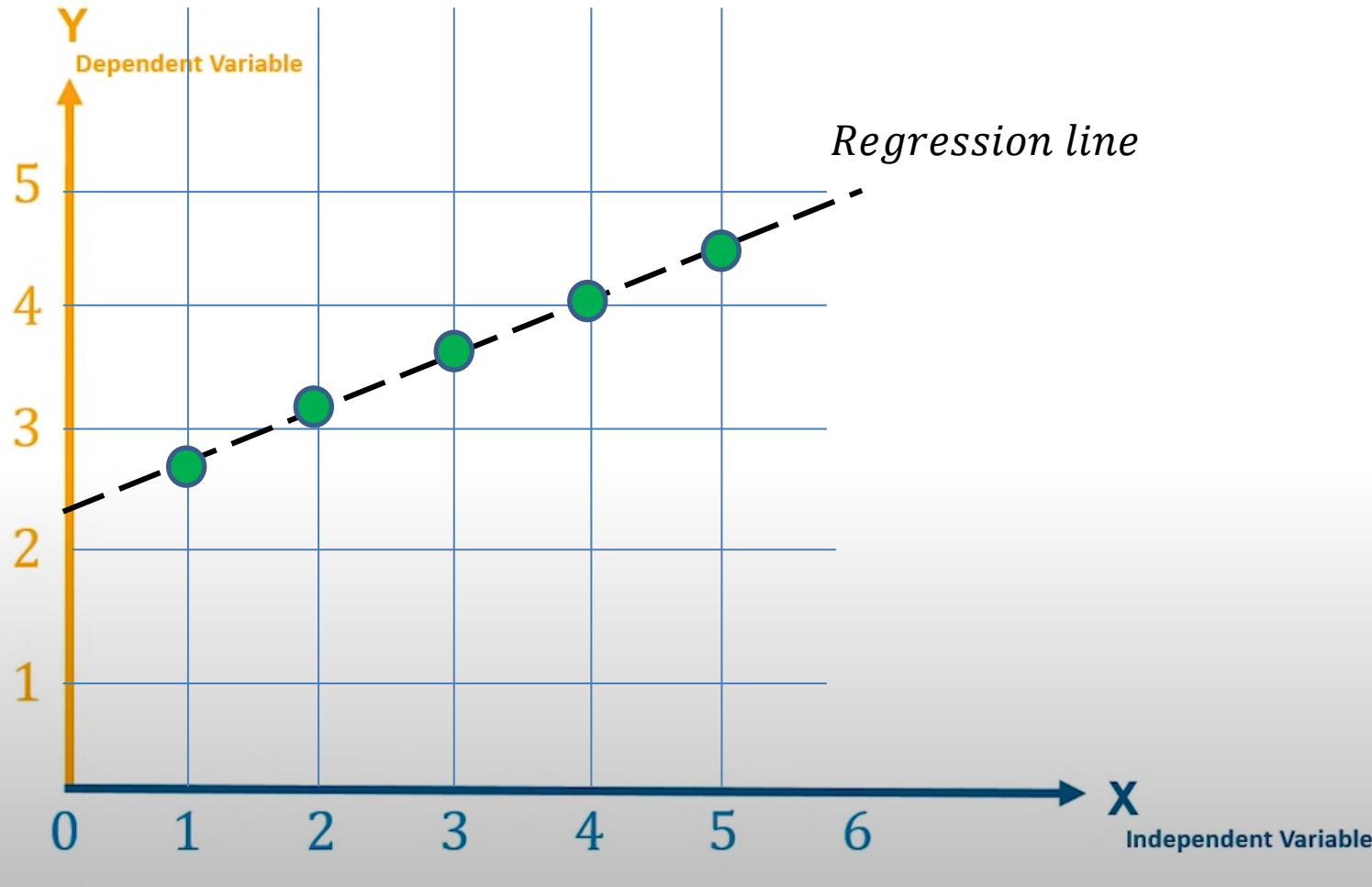
$$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2} = 0.4$$

\bar{x} = Mean of x

\bar{y} = Mean of y

What is R-square

- R-squared value is a statistical measure of how close the data are to the fitted regression line.
- It is also known as the coefficient of determination, or the coefficient of multiple determination.



x	y
1	3
2	4
3	2
4	4
5	5

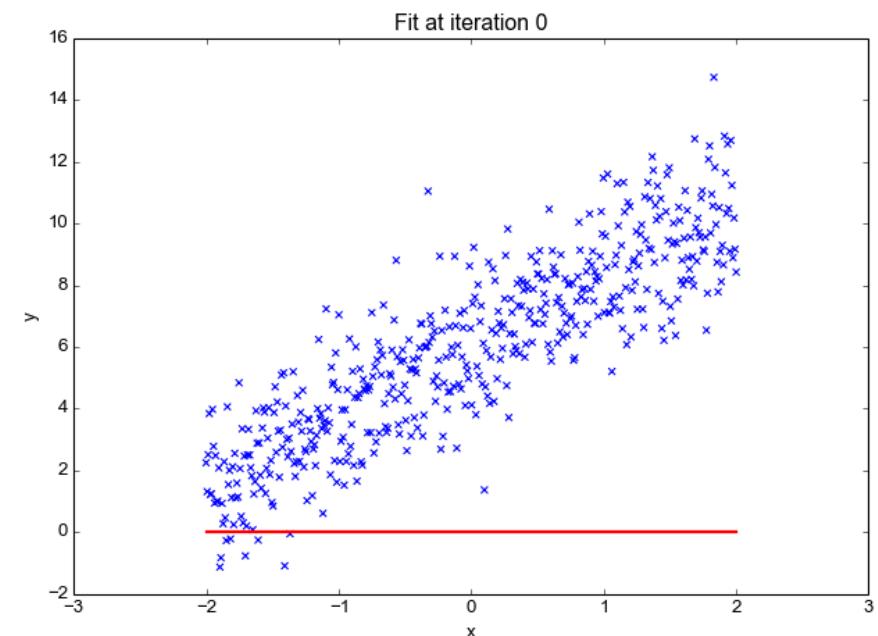
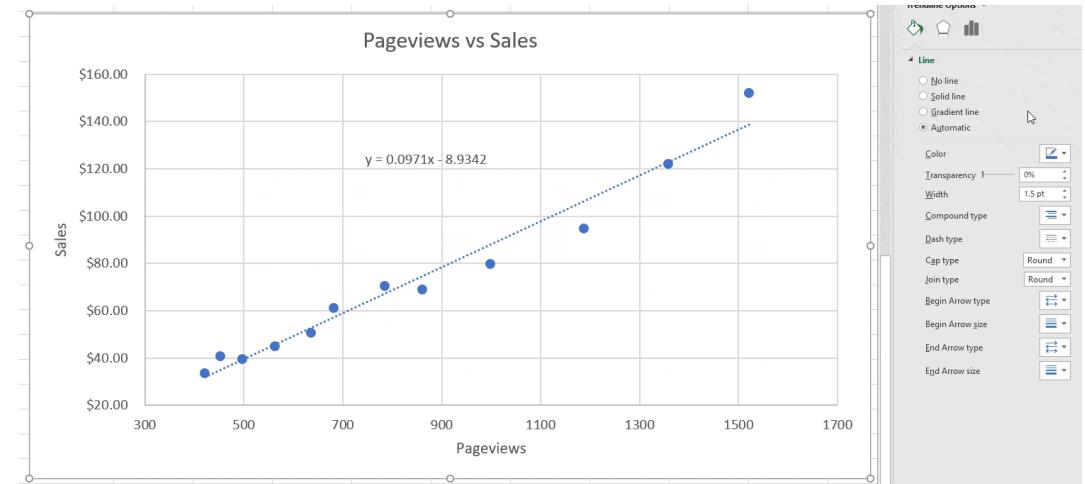
x	y_p
1	2.8
2	3.2
3	3.6
4	4.0
5	4.4

$$R^2 = \frac{\sum(y_p - \bar{y})^2}{\sum(y - \bar{y})^2}$$

$R^2 = 1$
Very good linear regression

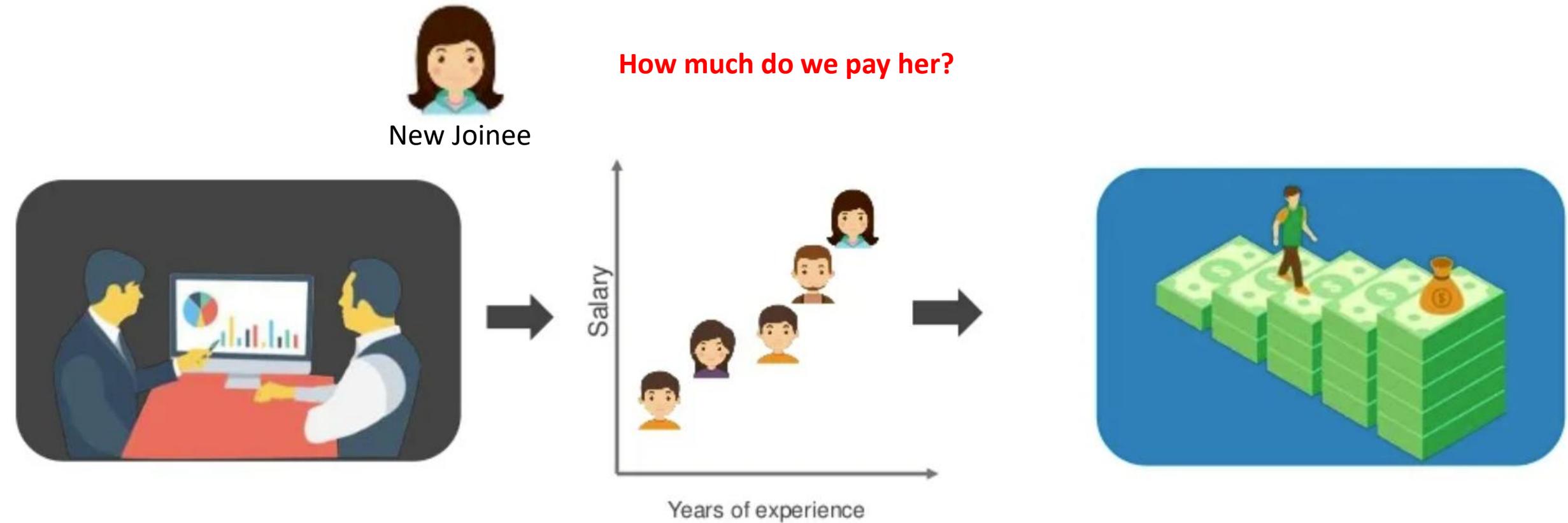
Application of Linear Regression

- Evaluating Trends and Sales Estimates.
- Analyzing the Impact of Price Changes.
- Assessment of risk in financial services and insurance domain.



Implementation of Linear Regression

- **Linear Regression:** Predict Employee Salary based on Years of Experience.



HR team trying to figure out how much to pay to a new joinee?

Employee data of Salary based on years of experience

Predict Salary

Implementation of Linear Regression

In [1]: # Step-1: Load the relevant Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]: # Step-2: Import the Dataset

```
salary_data=pd.read_csv("C:/Users/itsme_000/Desktop/HKI
x = salary_data.iloc[:, :-1].values
y = salary_data.iloc[:, 1].values
```

In [3]: # Step-3: Visualize the dataset

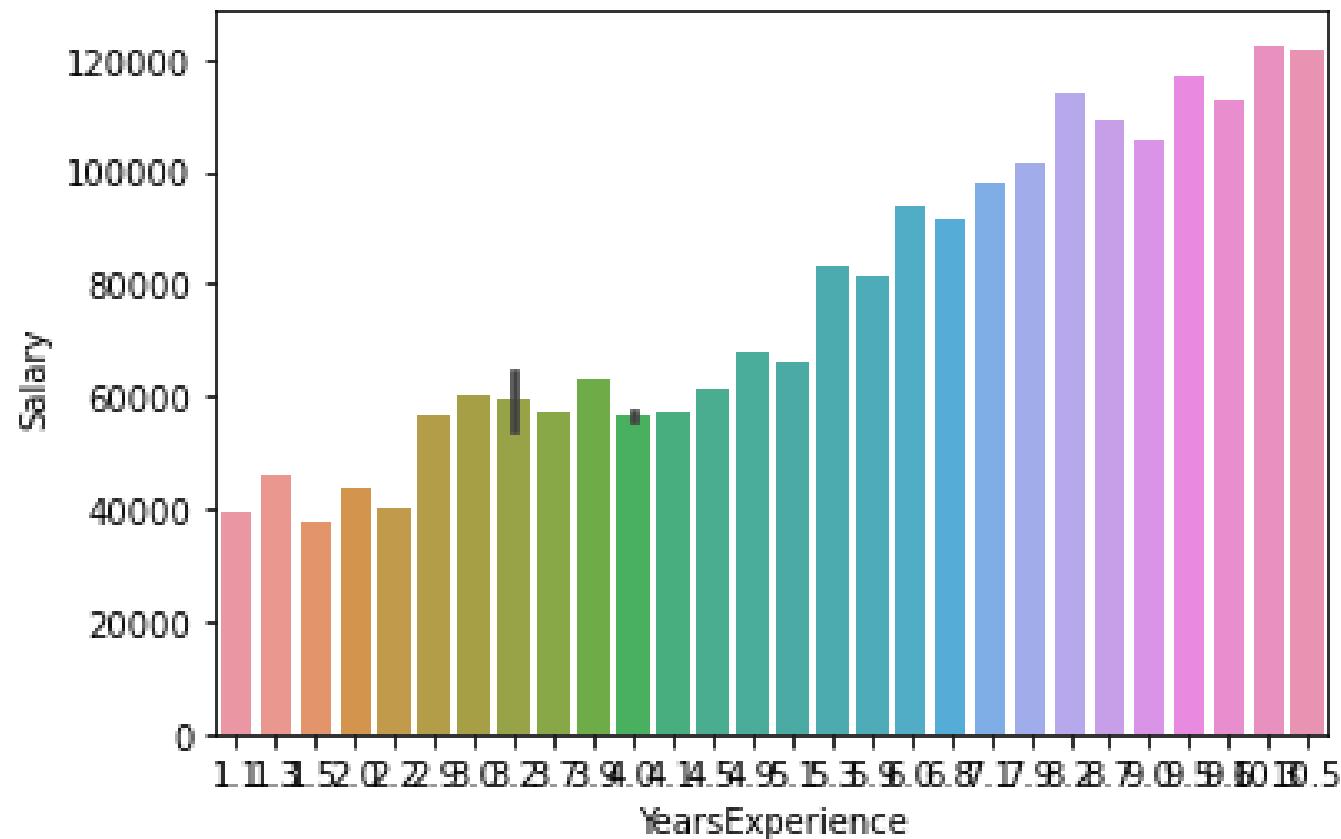
```
plt.plot(x, y, 'o')
plt.xlabel('Years of Experience')
plt.ylabel('Salary!')
plt.title('Salary vs Experience')
```



Implementation of Linear Regression

```
In [4]: # Step-3: Visualize the dataset  
sns.barplot(x='YearsExperience', y='Salary', data=salary_data)
```

```
Out[4]: <AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>
```



Implementation of Linear Regression

Split the dataset into training and test dataset using sklearn

```
In [5]: # Step-4: Split the data into training and testing set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=1/3,random_state=0)
```

```
In [6]: # Step-5: Fit Simple Linear Regression to the training dataset
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
Out[6]: LinearRegression(n_jobs=1, normalize=False)
```

```
In [7]: # Step-6: Predict the test set results
y_pred = lr.predict(x_test)

y_pred
```

```
Out[7]: array([ 40835.10590871, 123079.39940819, 65134.55626083, 63265.36777221,
 115602.64545369, 108125.8914992 , 116537.23969801, 64199.96201652,
 76349.68719258, 100649.1375447 ])
```

Implementation of Linear Regression

```
# Step-7: Visualize the train set results:  
plt.scatter(x_train,y_train,color='blue')  
plt.plot(x_train,lr.predict(x_train),color='red')  
plt.title('Salary~Experience (Train set)')  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.show()
```



```
# Step-8: Visualize the test set results:  
plt.scatter(x_test,y_test,color='blue')  
plt.plot(x_train,lr.predict(x_train),color='red')  
plt.title('Salary~Experience (Test set)')  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.show()
```



Implementation of Linear Regression

```
# Step 9 : Calculting the residuals:
```

```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test,y_pred))
print('MSE:', metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_absolute_error(y_test,y_pred))))
```

MAE: 3426.4269374307123

MSE: 21026037.329511296

RMSE: 58.53568943329114

- *MSE* values closer to zero consider better.
- *RMSE* between 0.2 and 0.5 consider reasonable and the model can relatively predict the data accurately. However, close the score is to 0 the better the performing model is.

$$MAE = \frac{\sum_{i=1}^N |y_{test_i} - y_{pred_i}|}{N}$$

$$MSE = \frac{\sum_{i=1}^N (y_{test_i} - y_{pred_i})^2}{N}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (y_{test_i} - y_{pred_i})^2}{N}}$$

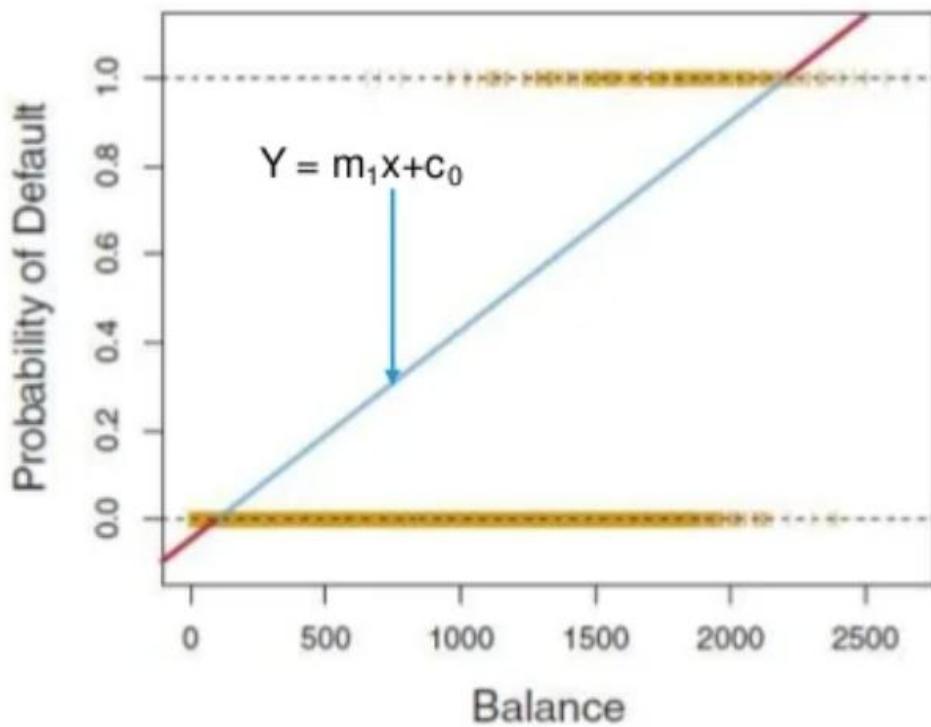
Logistic Regression

Logistic Regression is a classification algorithm that produces results in a **binary format** which is used to **predict the outcome of a categorical dependent variable**. So, the outcome should be **discrete/categorical** such as

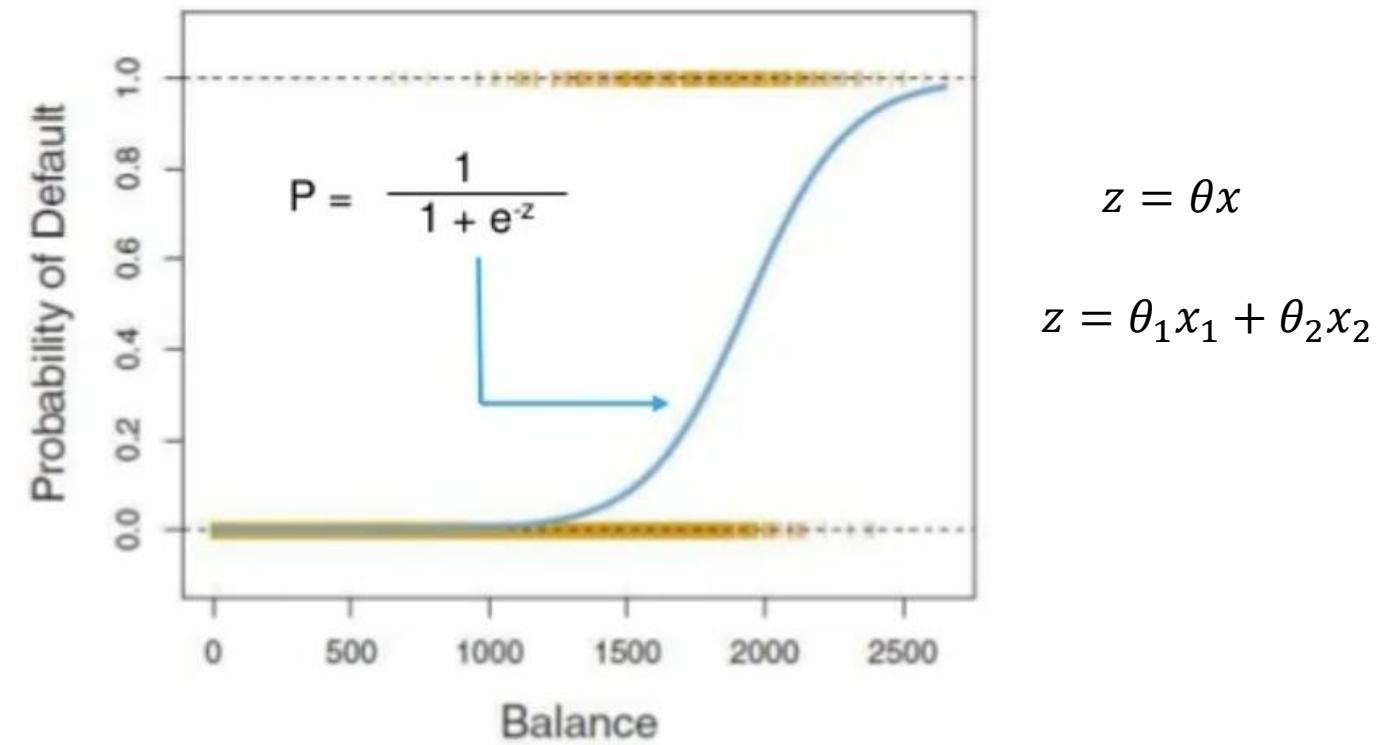


Logistic Regression

Plotting the Logistic Regression Curve: The logistic regression curve is known as the **sigmoid curve (S curve)**



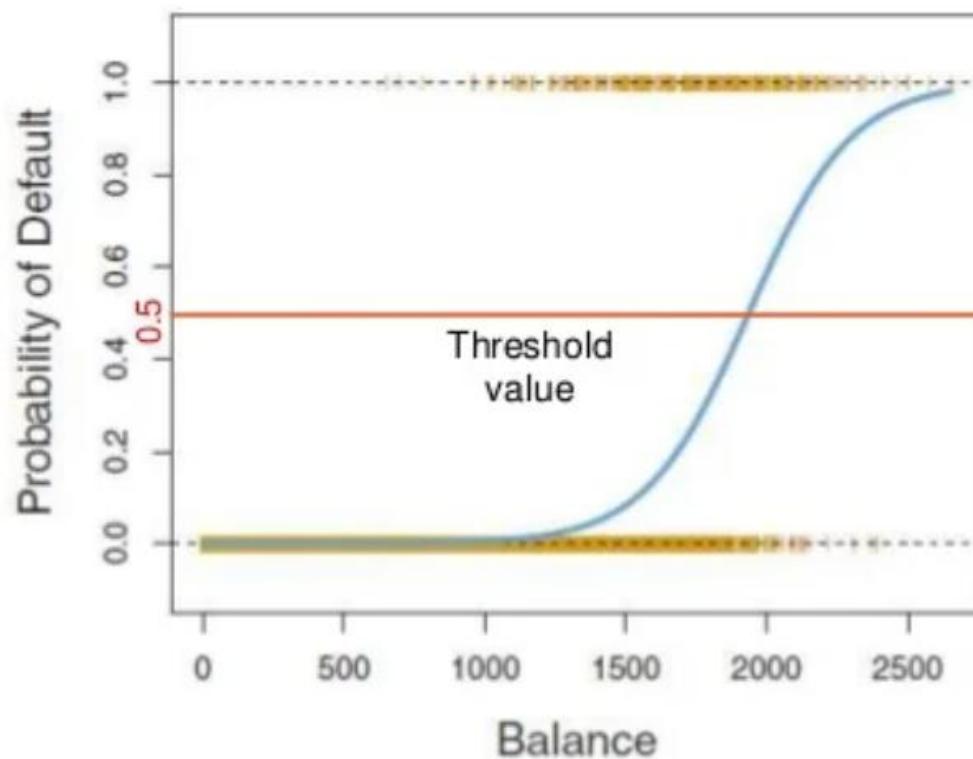
Predicted Y can exceed the range 0 and 1



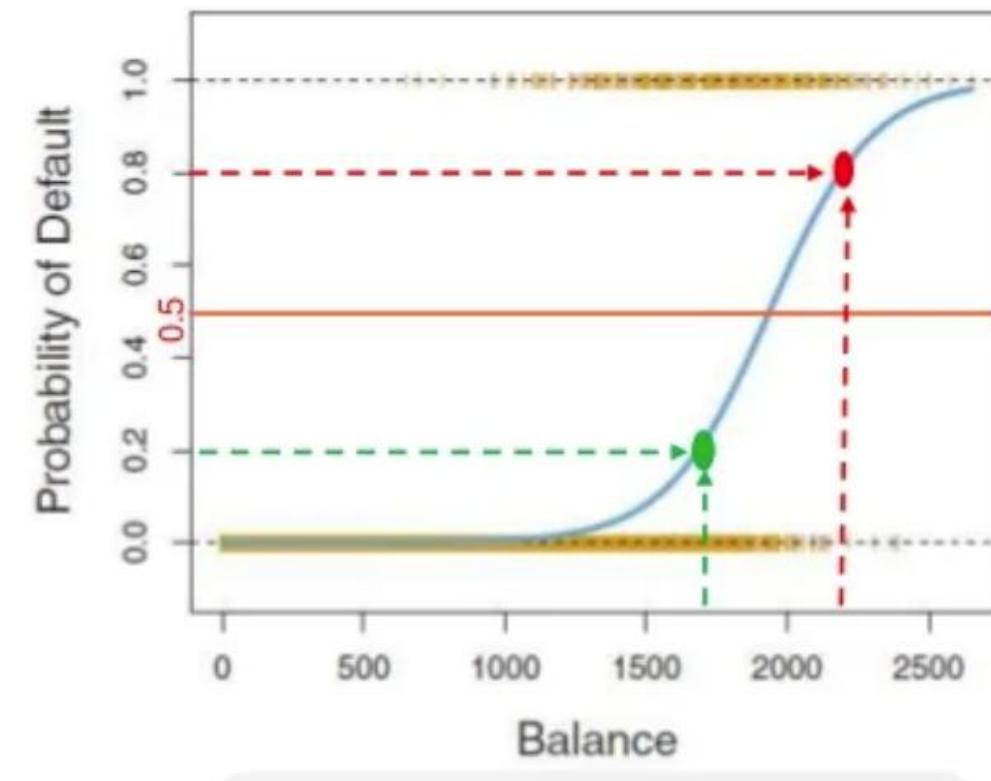
Predicted Y lies in the range 0 and 1

Logistic Regression

Plotting the Logistic Regression Curve: The logistic regression curve is known as the sigmoid curve (S curve)



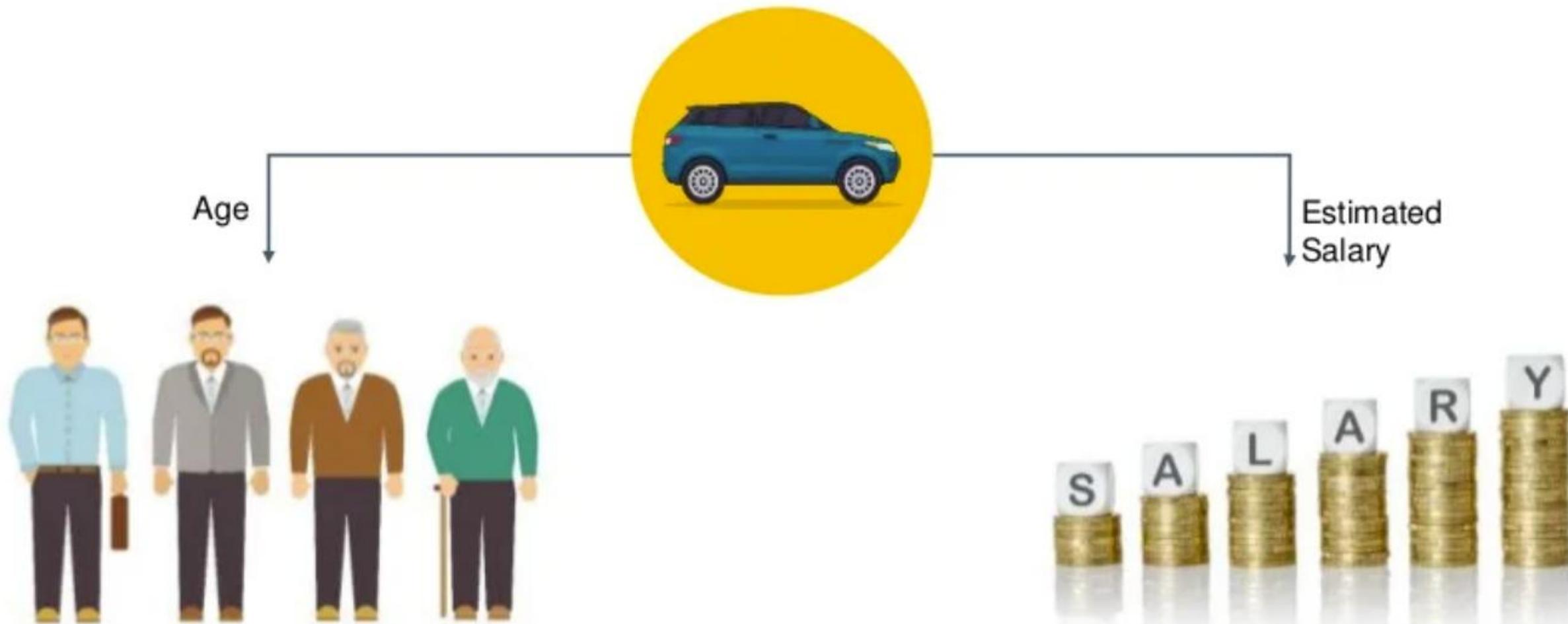
Cutoff point at 0.5, anything below it results in 0 and above is 1



The red data point will be default as it is above the threshold value of 0.5 and the green data point won't as it is below the threshold value

Implementation of Logistic Regression

- **Logistic Regression:** Predict if a person will buy a car based on their age and estimated salary.



Implementation of Logistic Regression

Step-1: Load the relevant libraries

```
In [1]: # Step-1: Load the relevant Libraries:  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline
```

Step-2: Import the dataset and extract the independent and dependent variables

```
In [2]: # Step-2: Import the Dataset & extract the independent and dependent variables:  
social_network=pd.read_csv("C:/Users/itsme_000/Desktop/HKU/Teaching/IDAT 7215 Computer Programming/
```

```
social_network.head(5)
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

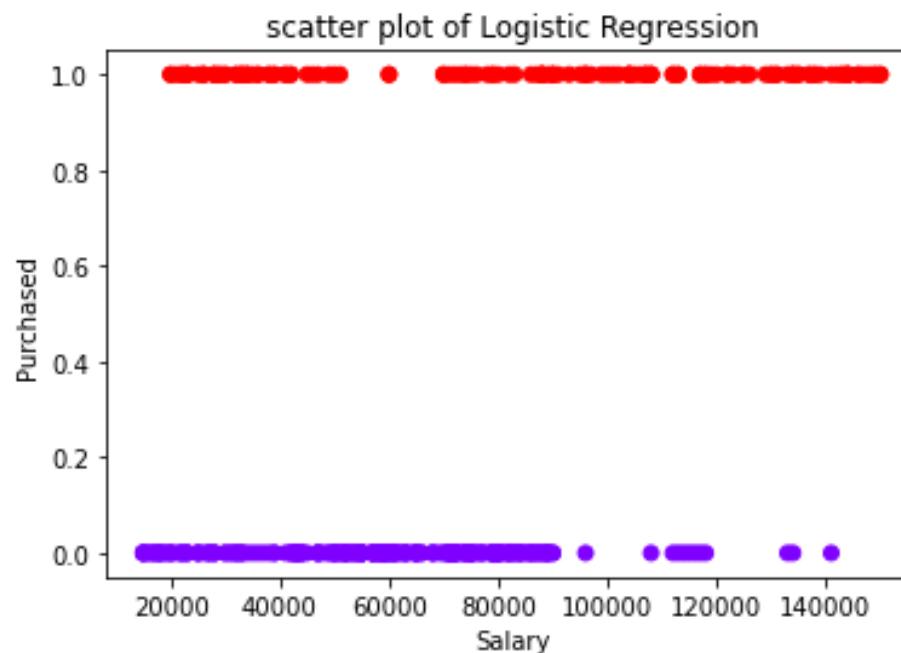
```
In [3]: x = social_network.iloc[:,[2,3]].values # Extracting the independent variables (age, salary)  
y = social_network.iloc[:,4].values # Extracting the dependent variables (purchased)
```

Implementation of Logistic Regression

Step-3: Visualize the dataset

```
In [4]: # Step-3: Visualize the dataset:  
# create scatter plot  
plt.scatter(x[:,1],y,c=y, cmap='rainbow')  
plt.xlabel('Salary')  
plt.ylabel('Purchased')  
plt.title('scatter plot of Logistic Regression')  
plt.show
```

```
Out[4]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Implementation of Logistic Regression

Step-4: Split the dataset into Training and Testing Set

```
In [6]: # Step-4: Split the dataset into Training and Testing set:  
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=1/3,random_state=0)
```

Step-5: Feature Scaling

```
In [7]: # Step-5: Feature Scaling:  
from sklearn.preprocessing import StandardScaler  
sc_x=StandardScaler()  
x_train=sc_x.fit_transform(x_train)  
x_test=sc_x.transform(x_test)
```

Feature scaling is a method used to **normalize the range of independent variables or features of data**. In data processing, it is also known as **data normalization** and is generally performed during the data preprocessing step.

Implementation of Logistic Regression

Step-6: Fit Logistic Regression to Training dataset:

```
In [8]: # Step 6: Fit Logistic Regression to Training dataset
from sklearn.linear_model import LogisticRegression
logR = LogisticRegression(random_state=0)
logR.fit(x_train, y_train)
```

```
Out[8]: LogisticRegression(random_state=0)
```

Step-7: Predicting the Test set results

```
In [9]: # Step 7: Predicting the Test set results:
y_pred = logR.predict(x_test)
y_pred
```

```
Out[9]: array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
0, 1], dtype=int64)
```

Implementation of Logistic Regression

Step-8: Display the confusion matrix:

```
In [10]: # Step 8: Display the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred)
cm
```

```
Out[10]: array([[79,  6],
                 [11, 38]], dtype=int64)
```

A **confusion matrix**, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.

Accuracy:

$$\frac{(TP+TN)}{P+N} = \frac{(79+38)}{134} = 0.87$$

Misclassification Rate:

$$\frac{(FP+FN)}{P+N} = \frac{(11+6)}{134} = 0.13$$

		Predicted condition	
		Total population = P + N	
		Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Confusion Matrix

Is accuracy is the best measure?

Accuracy may not be a good measure if the dataset is not balanced (both negative and positive classes have different numbers of data instances). We will explain this with an example.

Consider the following scenario: There are 90 people who are healthy (negative) and 10 people who have some disease (positive). Now let's say our machine learning model perfectly classified the 90 people as healthy but it also classified the unhealthy people as healthy. What will happen in this scenario? Let us see the confusion matrix and find out the accuracy.

The accuracy, in this case, is 90 % but this model is very poor because all the 10 people who are unhealthy are classified as healthy.

		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	90 TRUE NEGATIVE	0 FALSE POSITIVE
	POSITIVE	10 FALSE NEGATIVE	0 TRUE POSITIVE

Confusion matrix for healthy vs unhealthy people classification task.

Confusion Matrix Metrics

What does precision mean?

$$Precision = \frac{TP}{TP + FP}$$

Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e $TP = TP + FP$, this also means FP is zero. As FP increases the value of the denominator becomes greater than the numerator and precision value decreases (which we don't want).

		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	90	0
	POSITIVE	10	0
TRUE LABEL	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

Confusion matrix for healthy vs unhealthy people classification task.

Confusion Matrix Metrics

What does recall mean?

Now we will introduce another important metric called *recall*. *Recall* is also known as *sensitivity* or *true positive rate* and is defined as follows

$$Recall = \frac{TP}{TP + FN}$$

So ideally in a good classifier, we want both *precision* and *recall* to be one which also means *FP* and *FN* are zero. Therefore, we need a metric that takes into account both *precision* and *recall*.

		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	90	0
	POSITIVE	10	0

Confusion matrix for healthy vs unhealthy people classification task.

Confusion Matrix Metrics

What does F1 Score?

F1-score is a metric which takes into account both ***precision*** and ***recall*** and is defined as follows

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1 Score becomes 1 only when ***precision*** and ***recall*** are both 1. **F1 score** becomes high only when both ***precision*** and ***recall*** are high. **F1 score** is the harmonic mean of ***precision*** and ***recall*** and is a better measure than ***accuracy***.

		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	90	0
	POSITIVE	10	0
TRUE LABEL	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

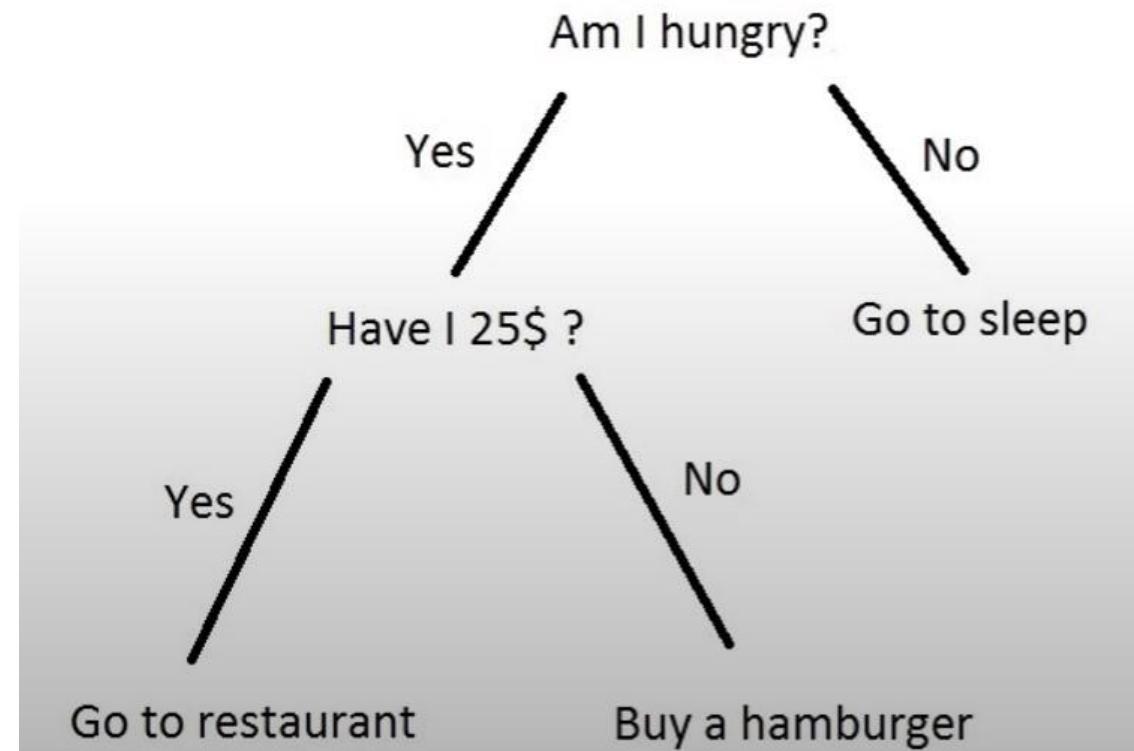
Confusion matrix for healthy vs unhealthy people classification task.

Decision Tree and Random Forest

Decision Tree

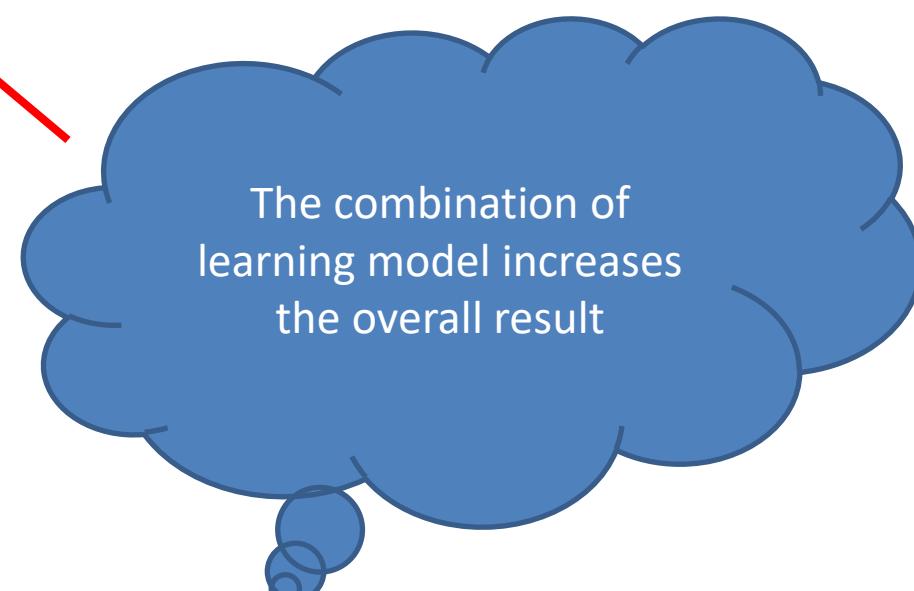
- Graphical representation of all the possible solutions to a decision.
- Decisions are based on **some conditions**.
- Decisions made **can be easily explained**.

Should the person go to a restaurant?



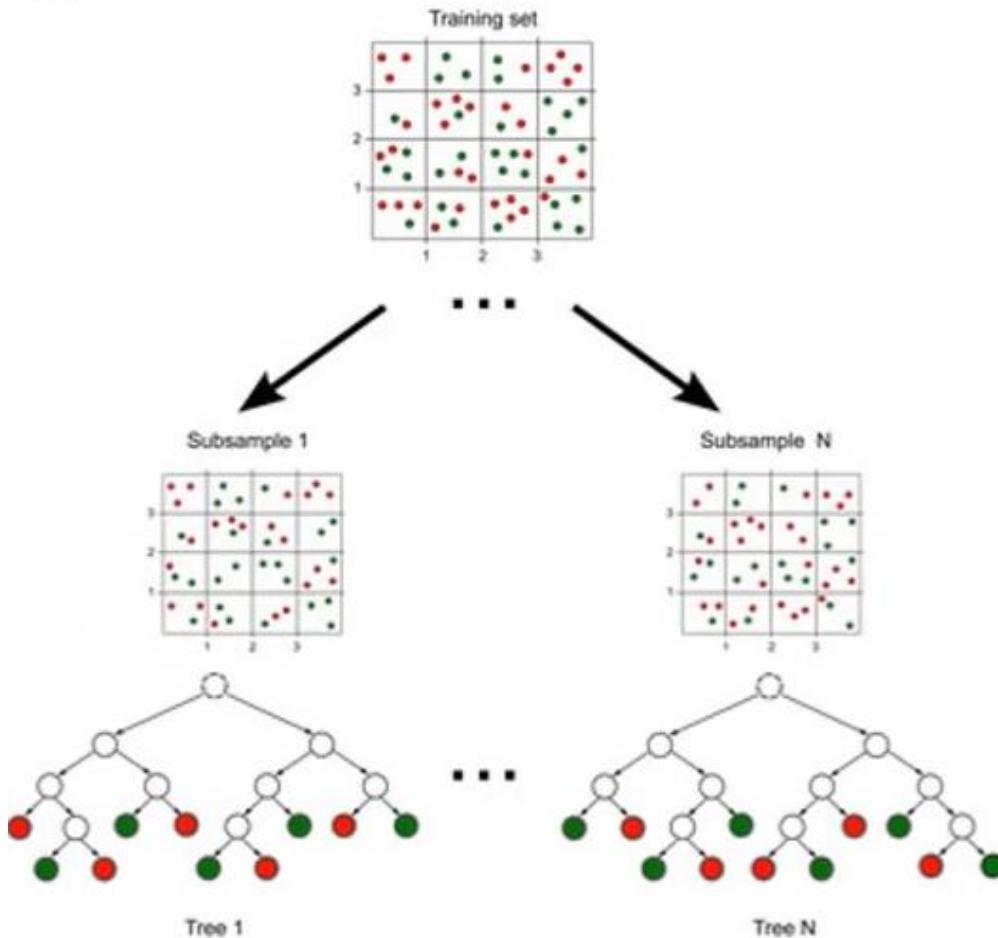
Random Forest

- Builds multiple decision trees and merges them together.
- More accurate and stable prediction.
- Random decision forests correct for decision trees' habit of overfitting to their training set.
- Trained with the “bagging” method.

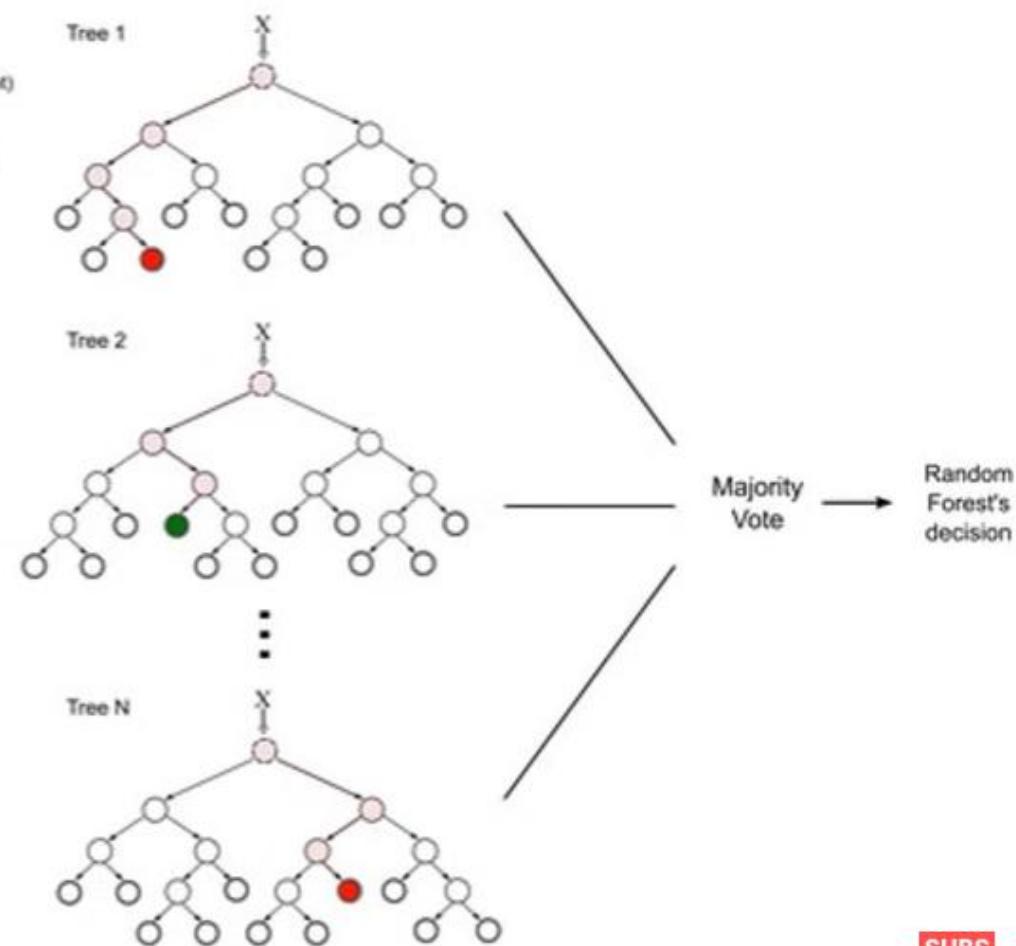


Random Forest

A



B

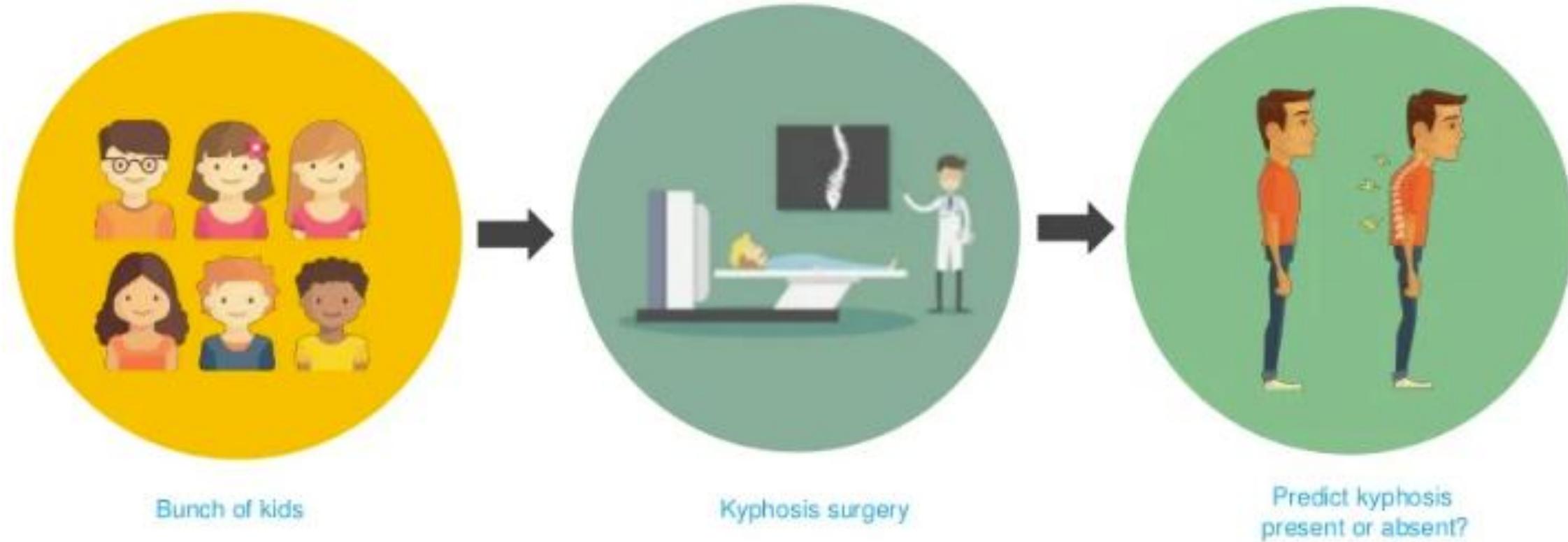


SUBSCRIBE

If the size of your dataset is huge then one single tree would lead to a overfit model

Implementation of Decision Tree

Decision Tree and Random Forest: Does Kyphosis exist after a surgery



Implementation of Decision Tree

Step-1: Load the relevant libraries

```
In [1]: # Step-1: Load the relevant Libraries:  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline
```

Step-2: Import the dataset and extract the independent and dependent variables

```
In [2]: # Step-2: Importing the Dataset  
Kyphosis=pd.read_csv("C:/Users/itsme_000/Desktop/HKU/Teaching/IDAT 7215 Computer Programming/Jupyter Notebook/L  
Kyphosis.head()
```

Out[2]:

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

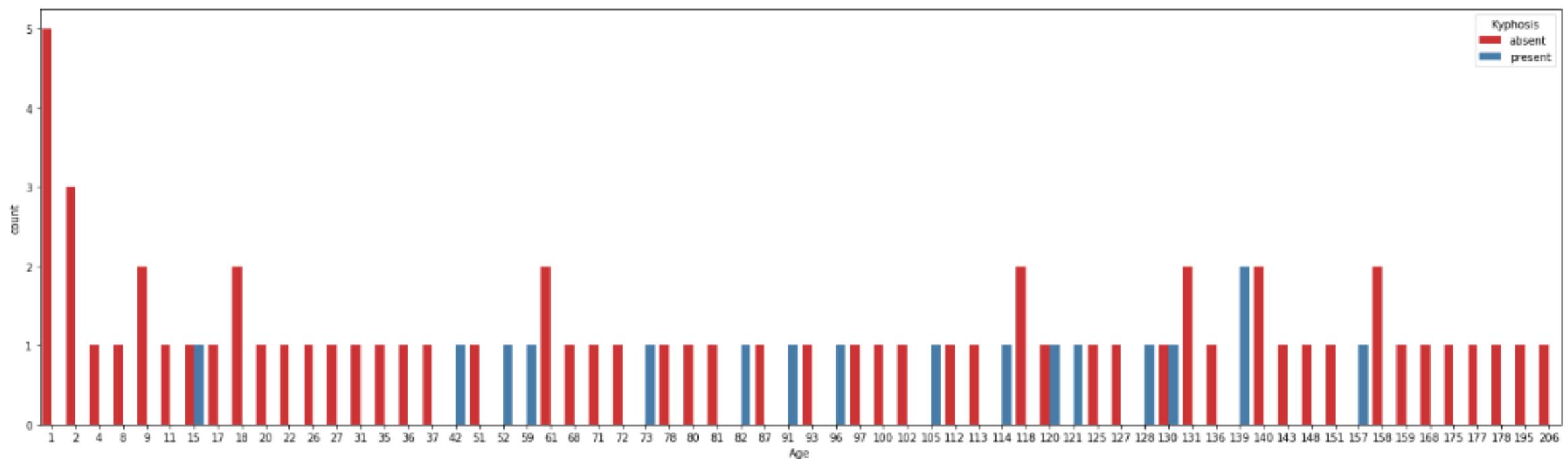
```
In [3]: # Extracting Dependent and Independent Variables  
x = Kyphosis.drop('Kyphosis',axis=1) # Extracting the independent variables  
y = Kyphosis['Kyphosis'] # Extracting the dependent variables
```

Implementation of Decision Tree

Step-3: Visualize the dataset

```
In [4]: # Step 3: Visualize the dataset  
plt.figure(figsize=(25,7))  
sns.countplot(x='Age', hue='Kyphosis', data=Kyphosis, palette='Set1')
```

```
Out[4]: <AxesSubplot:xlabel='Age', ylabel='count'>
```



Implementation of Decision Tree

Step-4: Split the dataset into Training and Testing Set

```
In [5]: # Step-4:Split the data into training and testing set  
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=100)
```

Step-5: Train Decision Tree

```
In [6]: # Step-5: Train Decision Tree|  
from sklearn.tree import DecisionTreeClassifier  
dtree=DecisionTreeClassifier()  
dtree.fit(x_train,y_train)
```

Out[6]: DecisionTreeClassifier()

Implementation of Decision Tree

Step-6: Predict the model:

```
In [7]: # Step-6: Predict the Model  
y_pred = dtree.predict(x_test)  
y_pred
```

```
Out[7]: array(['absent', 'absent', 'present', 'absent', 'present', 'absent',  
       'absent', 'absent', 'present', 'absent', 'absent', 'absent',  
       'absent', 'absent', 'present', 'present', 'absent', 'absent',  
       'present', 'present', 'absent', 'absent', 'present', 'absent',  
       'present'], dtype=object)
```

Step-7: Display the confusion matrix

Accuracy:

$$\frac{TP+TN}{P+N} = \frac{(14+1)}{25} = 0.6$$

Misclassification Rate:

$$\frac{FP+FN}{P+N} = \frac{(2+8)}{25} = 0.4$$

```
In [8]: # Step 7: Display the Confusion Matrix (To evaluate the model)  
from sklearn.metrics import classification_report,confusion_matrix  
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
absent	0.88	0.64	0.74	22
present	0.11	0.33	0.17	3
accuracy			0.60	25
macro avg	0.49	0.48	0.45	25
weighted avg	0.78	0.60	0.67	25

```
In [9]: # Confusion matrix  
cm=confusion_matrix(y_test, y_pred)|  
cm  
  
Out[9]: array([[14,  8],  
               [ 2,  1]], dtype=int64)
```

Implementation of Random Forest Tree

Step-8: Comparison with Random Forest

```
In [10]: # Step 9: Compare the model using Random Forest Tree  
from sklearn.ensemble import RandomForestClassifier  
rfc = RandomForestClassifier(n_estimators=100)  
rfc.fit(x_train,y_train)
```

```
Out[10]: RandomForestClassifier()
```

```
In [11]: rfc_pred=rfc.predict(x_test)
```

```
In [12]: # Confusion matrix  
cm_rfc=confusion_matrix(y_test, rfc_pred)  
cm_rfc
```

Accuracy:
$$\frac{(TP+TN)}{P+N} = \frac{(19+0)}{25} = 0.76$$

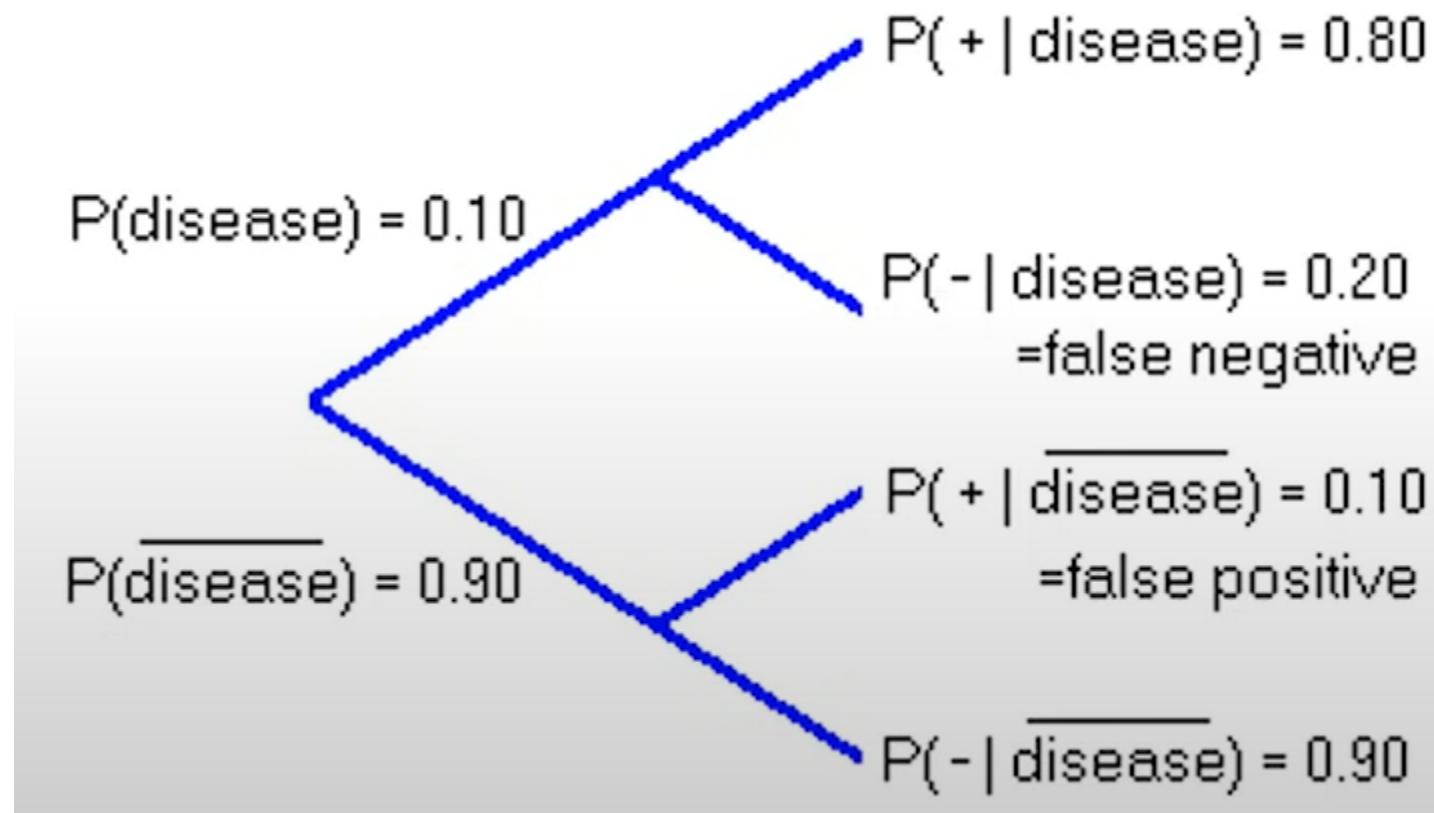
Misclassification Rate:
$$\frac{(FP+FN)}{P+N} = \frac{(3+3)}{25} = 0.24$$

```
Out[12]: array([[19,  3],  
                 [ 3,  0]], dtype=int64)
```

Naïve Bayes

What is Naïve Bayes

Naïve Bayes is a simple but surprisingly powerful **algorithm for predictive modeling**. It is a classification technique based on the Bayes theorem.



What is Naïve Bayes

Naïve Bayes is a simple but surprisingly powerful **algorithm for predictive modeling**. It is a classification technique based on the Bayes theorem.

Bayes' Theorem

Given a hypothesis H and evidence E , Bayes theorem states that the relationship between the probability of the hypothesis before getting the evidence $P(H)$ and the probability of the hypothesis after getting the evidence $P(H|E)$ is

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Bayes' Theorem Example



$$P(\text{King}) = \frac{4}{52} = \frac{1}{13}$$

$$P(\text{King}|\text{Face}) = ?$$

Apply Bayes Theorem

$$P(\text{King}|\text{Face}) = \frac{P(\text{Face}|\text{King}).P(\text{King})}{P(\text{Face})}$$

A king, queen, or jack in a deck of cards is known as a face card.

$$P(\text{Face}|\text{King}) = 1$$

$$P(\text{Face}) = \frac{12}{52} = \frac{3}{13}$$

$$P(\text{King}|\text{Face}) = \frac{1 \cdot 1/13}{3/13} = 1/3$$

Classification Steps

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No



Frequency Table		Play	
		Yes	No
Outlook	Sunny	2	3
	Overcast	4	0
	Rainy	3	2

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	Strong	6	2
	Weak	3	3

Classification Steps

Likelihood Table		Play		
		Yes	No	
Outlook	Sunny	2/9	3/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	3/9	2/5	5/14
		9/14	5/14	

$P(x|c) = P(\text{Sunny}| \text{Yes}) = 2/9 = 0.22$

$P(x) = P(\text{Sunny}) = 5/14 = 0.36$

$P(c) = P(\text{Yes}) = 9/14 = 0.64$

Likelihood of 'Yes' given Sunny is

$$P(c|x) = P(\text{Yes}|\text{Sunny}) = \frac{P(\text{Sunny}|\text{Yes}).P(\text{Yes})}{P(\text{Sunny})} = \frac{\frac{2}{9} \times \frac{9}{14}}{\frac{5}{14}} = 0.4$$

Likelihood of 'No' given Sunny is

$$P(c|x) = P(\text{No}|\text{Sunny}) = \frac{P(\text{Sunny}|\text{No}).P(\text{No})}{P(\text{Sunny})} = \frac{\frac{3}{5} \times \frac{5}{14}}{\frac{5}{14}} = 0.6$$

Classification Steps

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

$$P(\text{Yes|High}) = \frac{\frac{3}{9} \times \frac{9}{14}}{\frac{7}{14}} = 0.42$$

$$P(\text{No|High}) = \frac{\frac{4}{5} \times \frac{5}{14}}{\frac{7}{14}} = 0.58$$

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	Weak	6/9	2/5	8/14
	Strong	3/9	3/5	6/14
		9/14	5/14	

$$P(\text{Yes|Weak}) = \frac{\frac{6}{9} \times \frac{9}{14}}{\frac{8}{14}} = 0.75$$

$$P(\text{No|Weak}) = \frac{\frac{2}{5} \times \frac{5}{14}}{\frac{8}{14}} = 0.25$$

Classification Steps

Suppose we have a day with the following values

Outlook	=	Rain
Humidity	=	High
Wind	=	Weak
Play	=	?

Likelihood of 'Yes' on that Day = $P(Outlook = Rain|Yes) \times P(Humidity = High|Yes) \times P(Wind = Weak|Yes) \times P(Yes)$

$$= \frac{3}{9} \times \frac{3}{9} \times \frac{6}{9} \times \frac{9}{14} = 0.048$$

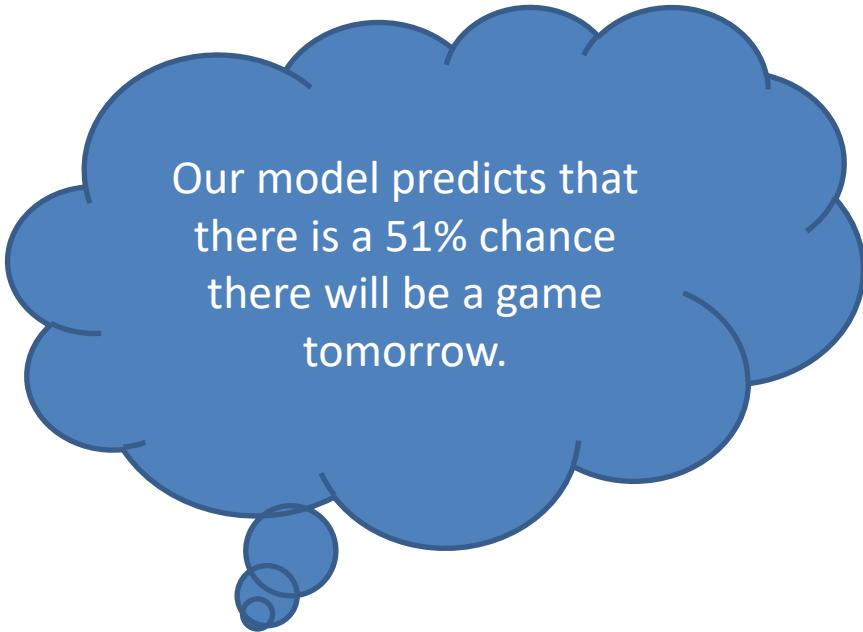
Likelihood of 'No' on that Day = $P(Outlook = Rain|No) \times P(Humidity = High|No) \times P(Wind = Weak|No) \times P(No)$

$$= \frac{2}{5} \times \frac{4}{5} \times \frac{2}{5} \times \frac{5}{14} = 0.046$$

Classification Steps

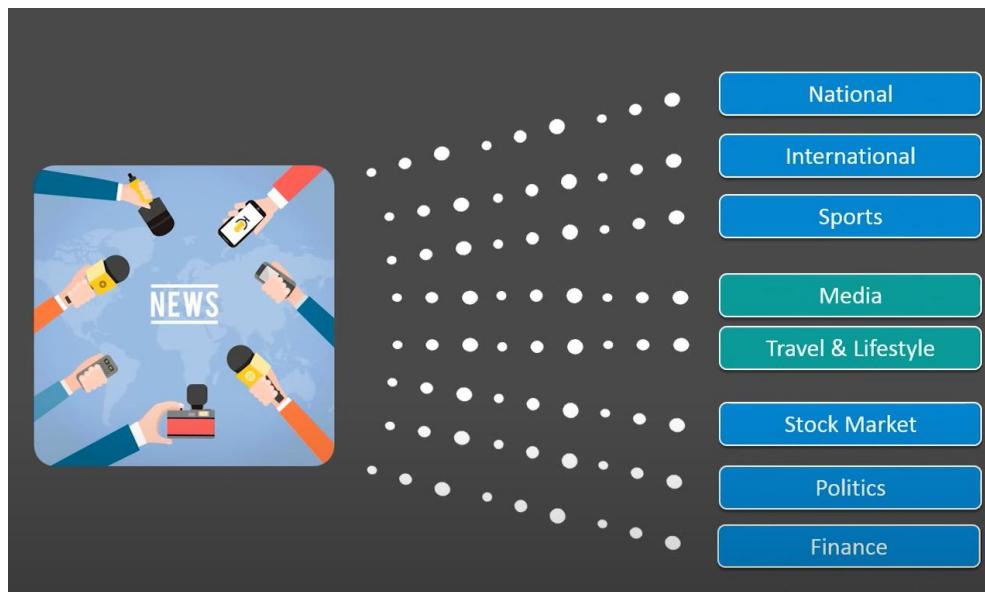
$$P(Yes) = \frac{0.048}{(0.048 + 0.046)} = 0.51$$

$$P(No) = \frac{0.046}{(0.048 + 0.046)} = 0.49$$

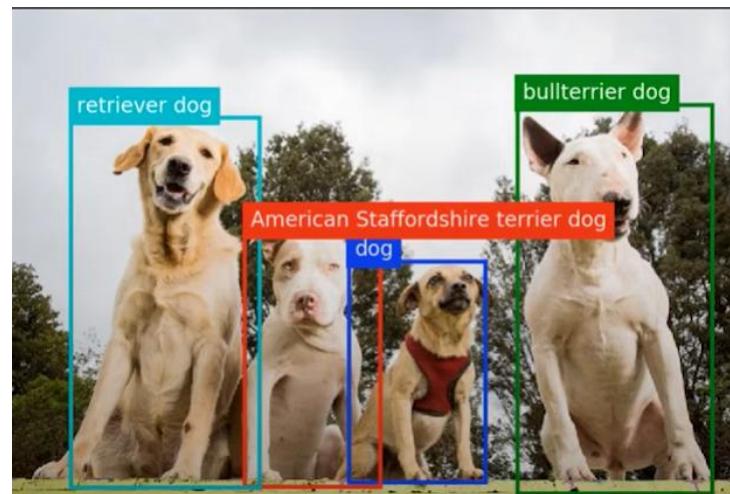


Our model predicts that there is a 51% chance there will be a game tomorrow.

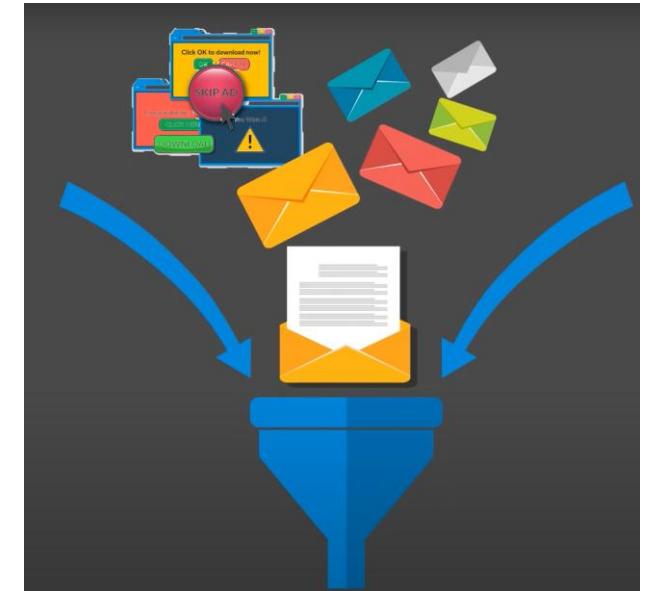
Applications of Naïve Base Algorithm



News Categorization



Object & Face Recognition



SPAM Filtering



Weather Prediction

Types of Naïve Bayes

1. **Gaussian:** It is used in classification and assumes that features follow a normal distribution.
2. **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x_i is observed over the n trials”.
3. **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

Implementation of Naïve Bayes

Load the library

```
In [1]: from sklearn import datasets  
from sklearn import metrics  
from sklearn.naive_bayes import GaussianNB
```

Load the IRIS Dataset

```
In [2]: dataset=datasets.load_iris() # Loading the IRIS Datasets
```

Fit a Gaussian Naïve Base Model on IRIS Dataset

```
In [3]: model = GaussianNB()  
model.fit(dataset.data, dataset.target)
```

```
Out[3]: GaussianNB()
```

```
In [4]: print(model)
```

```
GaussianNB()
```

Implementation of Naïve Bayes

Make Predictions

```
In [5]: expected = dataset.target  
predicted = model.predict(dataset.data)
```

```
In [6]: print(metrics.classification_report(expected, predicted))  
print(metrics.confusion_matrix(expected, predicted))
```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	1.00	1.00	1.00	50
1	0.94	0.94	0.94	50
2	0.94	0.94	0.94	50

accuracy			0.96	150
----------	--	--	------	-----

macro avg	0.96	0.96	0.96	150
-----------	------	------	------	-----

weighted avg	0.96	0.96	0.96	150
--------------	------	------	------	-----

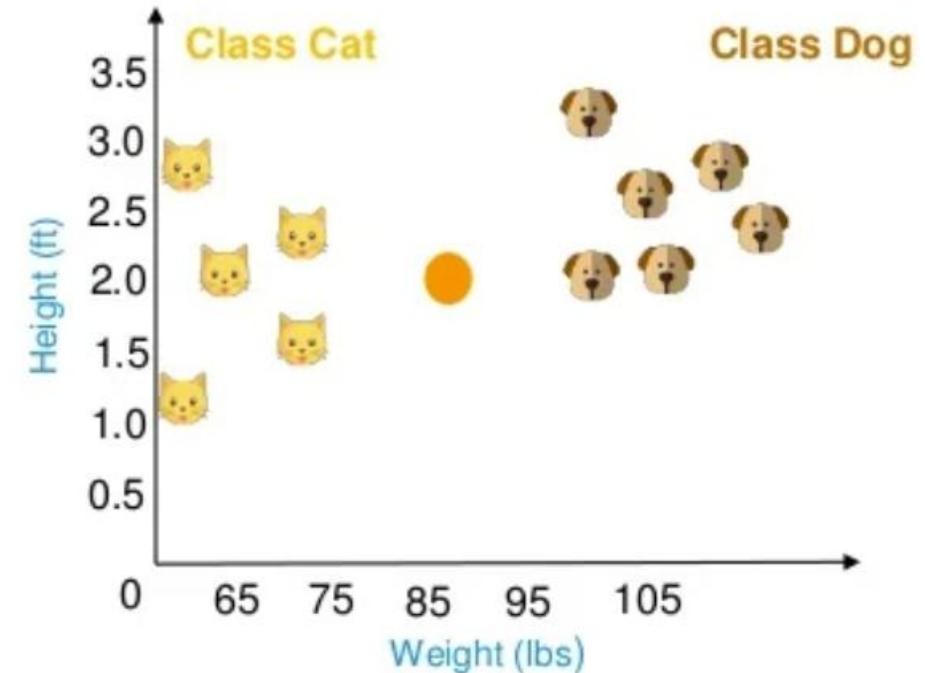
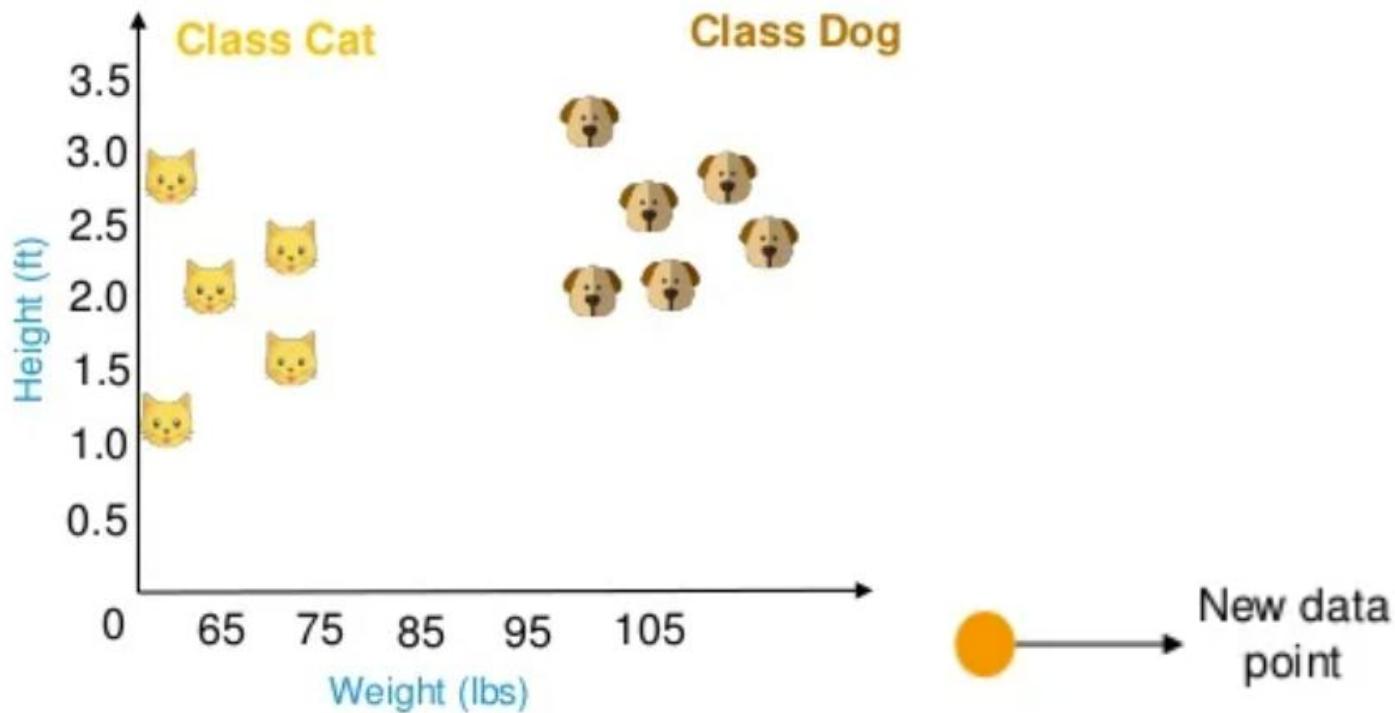
```
[[50  0  0]  
 [ 0 47  3]  
 [ 0  3 47]]
```

KNNs Algorithm

K Nearest Neighbors

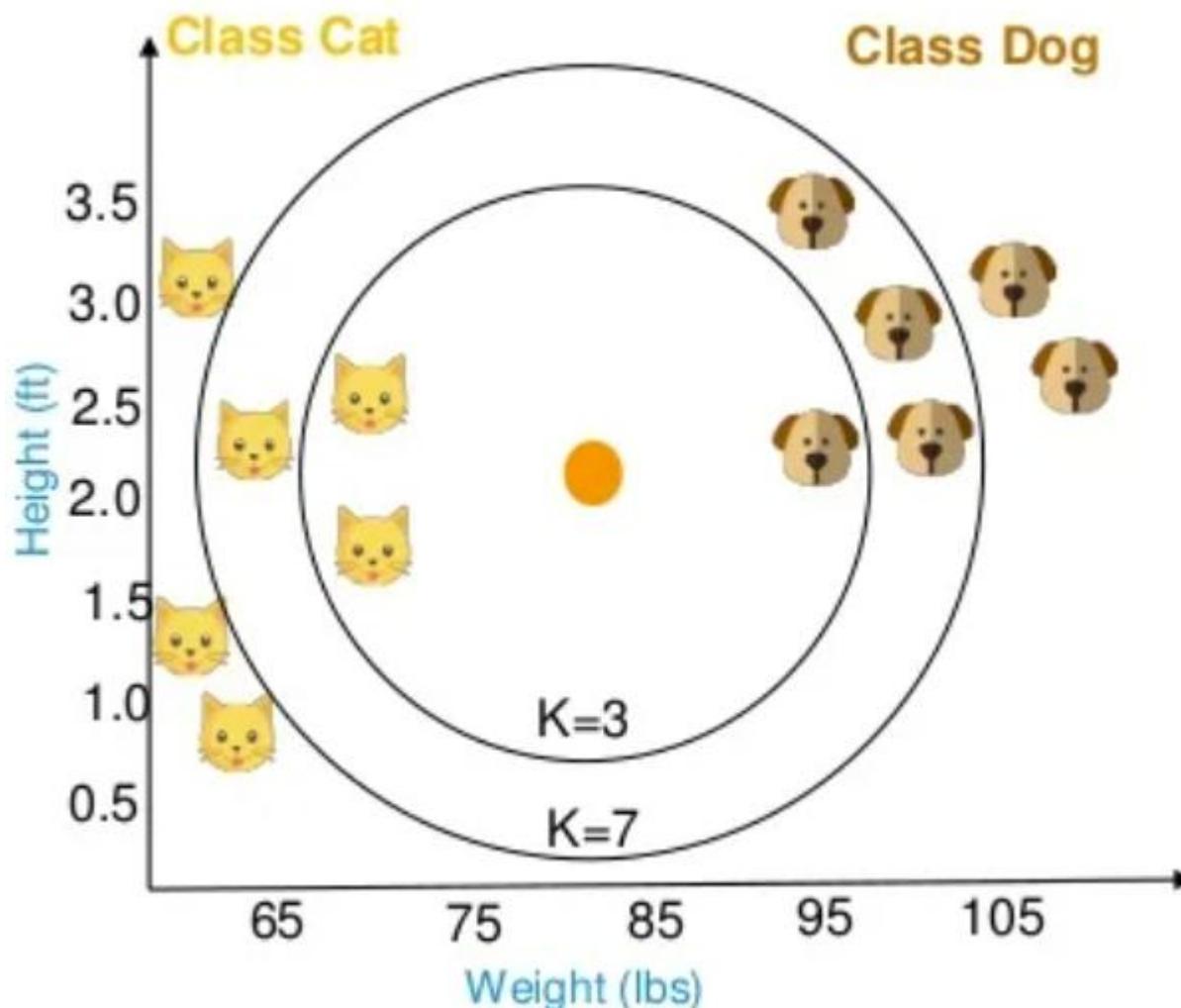
- KNN is a classification algorithm generally used to **predict categorical values**.
- It stores all the available cases and **classifies new cases based on a similarity measure**.

To find if a new data is



K Nearest Neighbors

- Choosing a K will define what class a new data point is assigned to.
- K is the number of nearest neighbors.

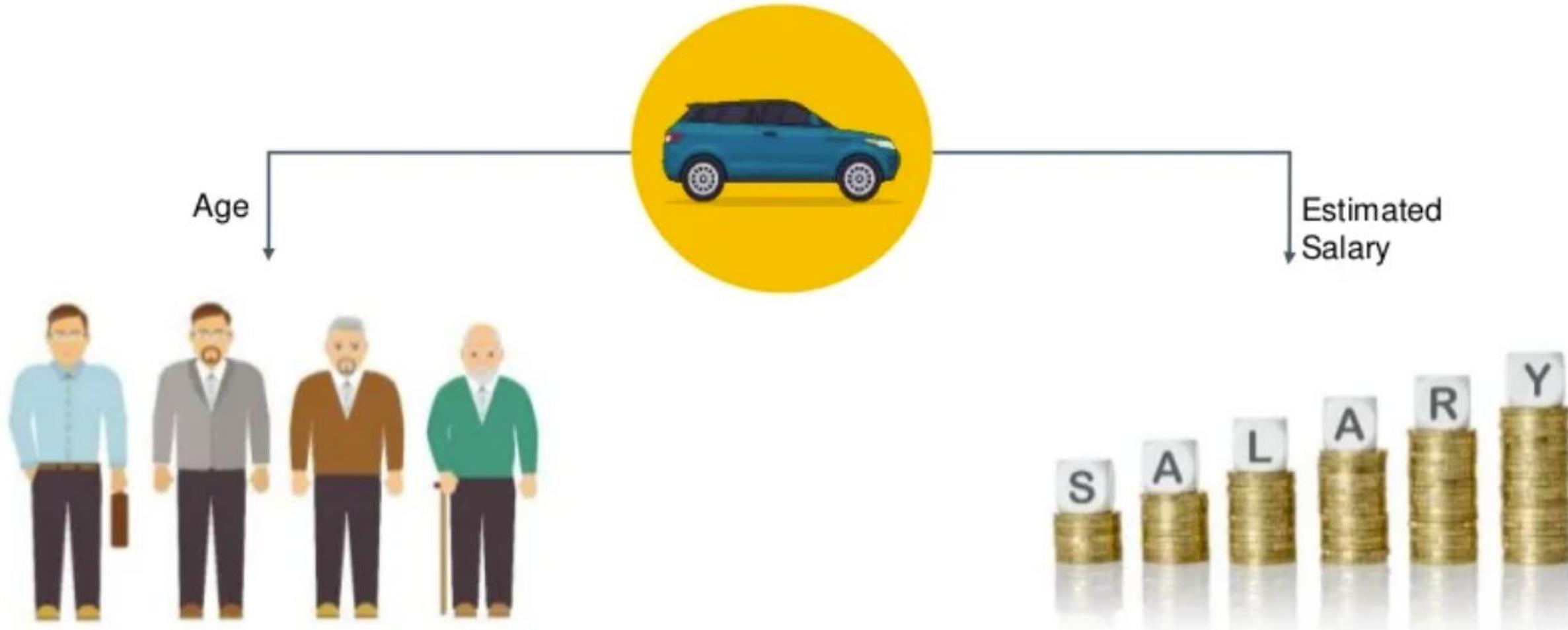


Calculate the least distance
from new points

If $K=3$ the new data point
belongs to class Cat

If $K=7$, the new data point
belongs to class Dog

Implementation of KNN Algorithm



Implementation of KNN Algorithm

Step-1: Load the relevant libraries

```
In [1]: # Step-1: Load the relevant Libraries:  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline
```

Step-2: Import the dataset and extract the independent and dependent variables

```
In [2]: # Step-2: Import the Dataset & extract the independent and dependent variables:  
social_network=pd.read_csv("C:/Users/itsme_000/Desktop/HKU/Teaching/IDAT 7215 Computer Programming/
```

```
social_network.head(5)
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

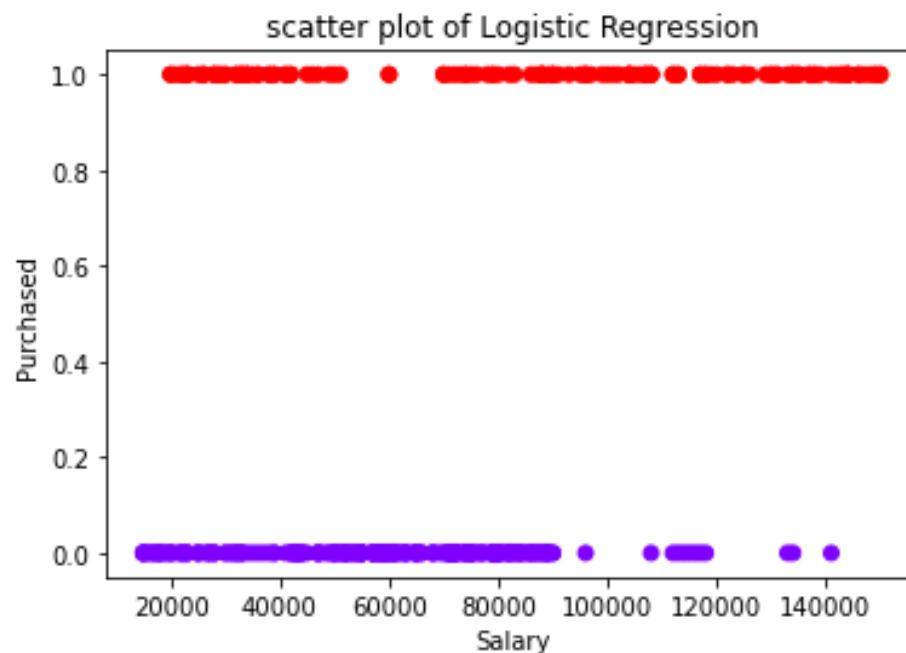
```
In [3]: x = social_network.iloc[:,[2,3]].values # Extracting the independent variables (age, salary)  
y = social_network.iloc[:,4].values # Extracting the dependent variables (purchased)
```

Implementation of KNN Algorithm

Step-3: Visualize the dataset

```
In [4]: # Step-3: Visualize the dataset:  
# create scatter plot  
plt.scatter(x[:,1],y,c=y, cmap='rainbow')  
plt.xlabel('Salary')  
plt.ylabel('Purchased')  
plt.title('scatter plot of Logistic Regression')  
plt.show
```

```
Out[4]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Implementation of KNN Algorithm

Step-4: Split the dataset into Training and Testing Set

```
In [6]: # Step-4: Split the dataset into Training and Testing set:  
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=1/3,random_state=0)
```

Step-5: Feature Scaling

```
In [7]: # Step-5: Feature Scaling:  
from sklearn.preprocessing import StandardScaler  
sc_x=StandardScaler()  
x_train=sc_x.fit_transform(x_train)  
x_test=sc_x.transform(x_test)
```

Implementation of KNN Algorithm

Step-6: Fit KNN model to Train dataset:

```
In [9]: # Step-6: Fit KNN to Train set:  
from sklearn.neighbors import KNeighborsClassifier  
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p=2)  
classifier.fit(x_train,y_train)
```

```
Out[9]: KNeighborsClassifier()
```

Step-7: Predicting the Test set results

```
In [13]: # Step 7: Predicting the Test set results:  
y_pred = classifier.predict(x_test)  
y_pred
```

```
Out[13]: array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,  
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,  
1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,  
0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,  
1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1], dtype=int64)
```

Implementation of KNN Algorithm

Step-8: Display the confusion matrix:

```
# Step 8: Display the Confusion Matrix (To evaluate the model)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred)
cm

array([[64,  4],
       [ 3, 29]], dtype=int64)
```

Accuracy:

$$\frac{(TP+TN)}{P+N} = \frac{(64+29)}{100} = 0.93$$

Misclassification Rate:

$$\frac{(FP+FN)}{P+N} = \frac{(3+4)}{100} = 0.07$$

		Predicted condition	
		Total population = P + N	
		Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

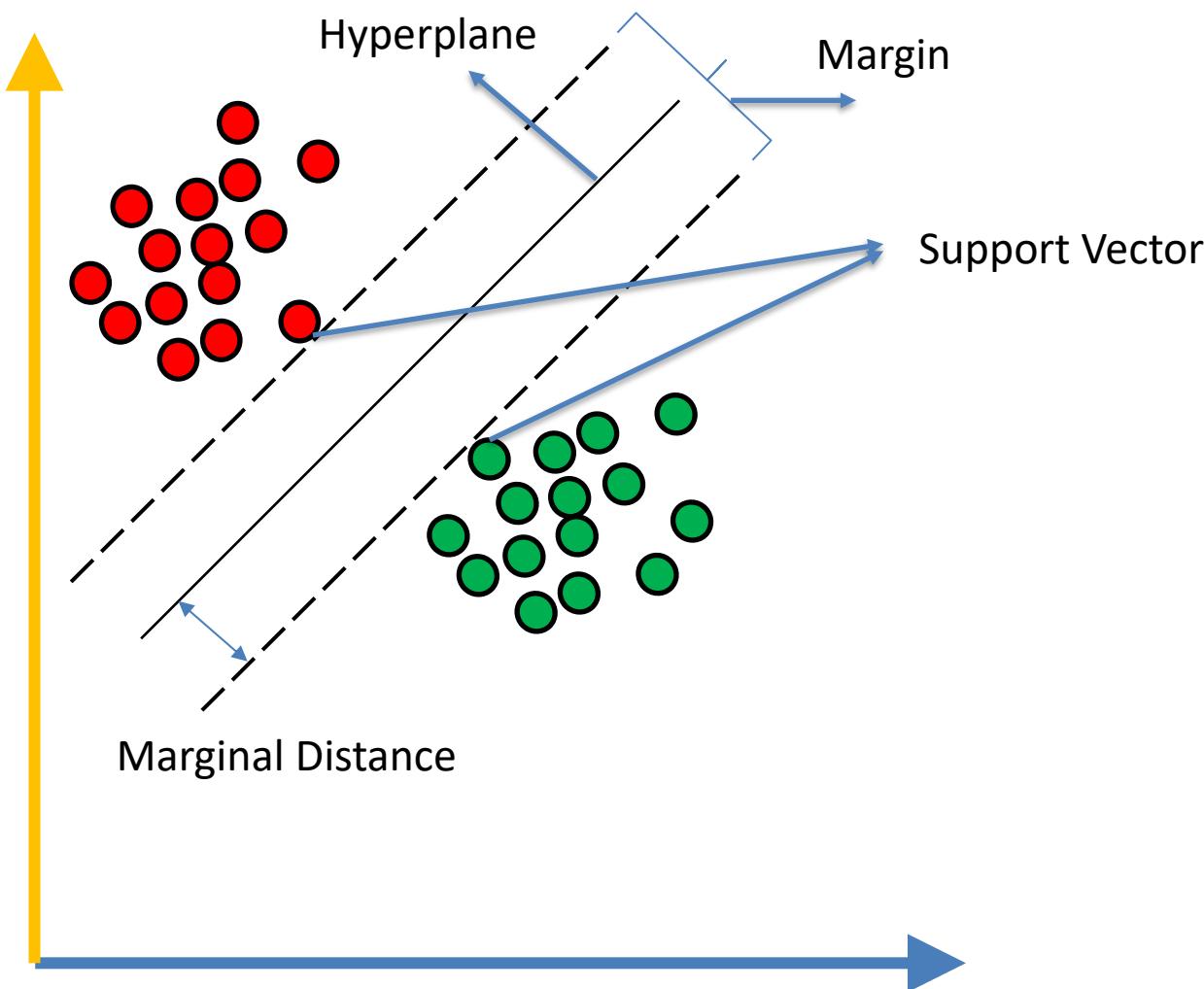
Support Vector Machine (SVM)

What is SVM

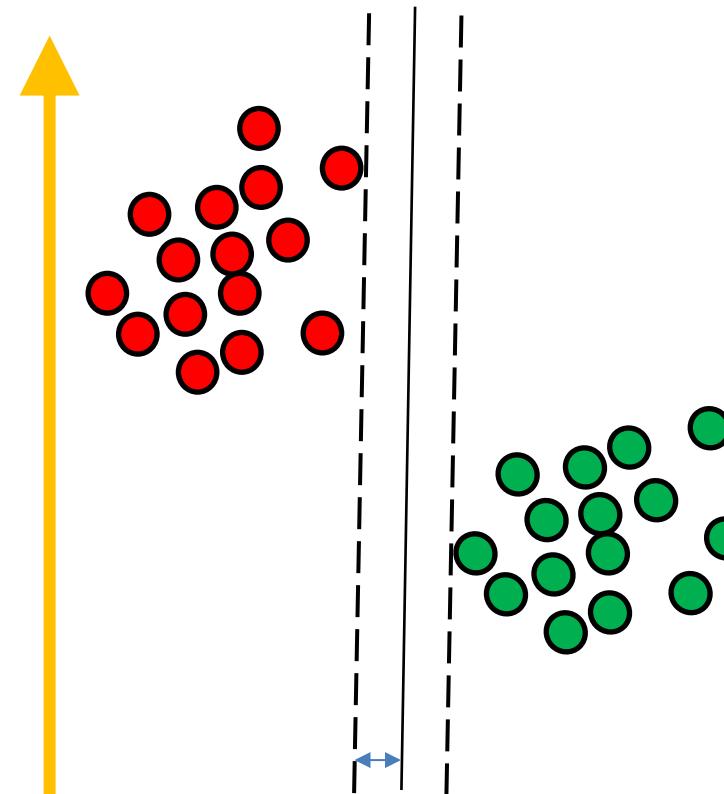
- Support Vector Machine (SVM) is a **discriminative classifier** that is formally designed by a **separative hyperplane**.
- Useful for solving both classification and regression-type problems.
- It is a representation of examples as **points in space** that are mapped so that the points of different categories are separated by a gap as wide as possible.
- The main **objective of SVM** is to segregate the given data in the best possible way.

How does SVM Work?

Linearly Separable data

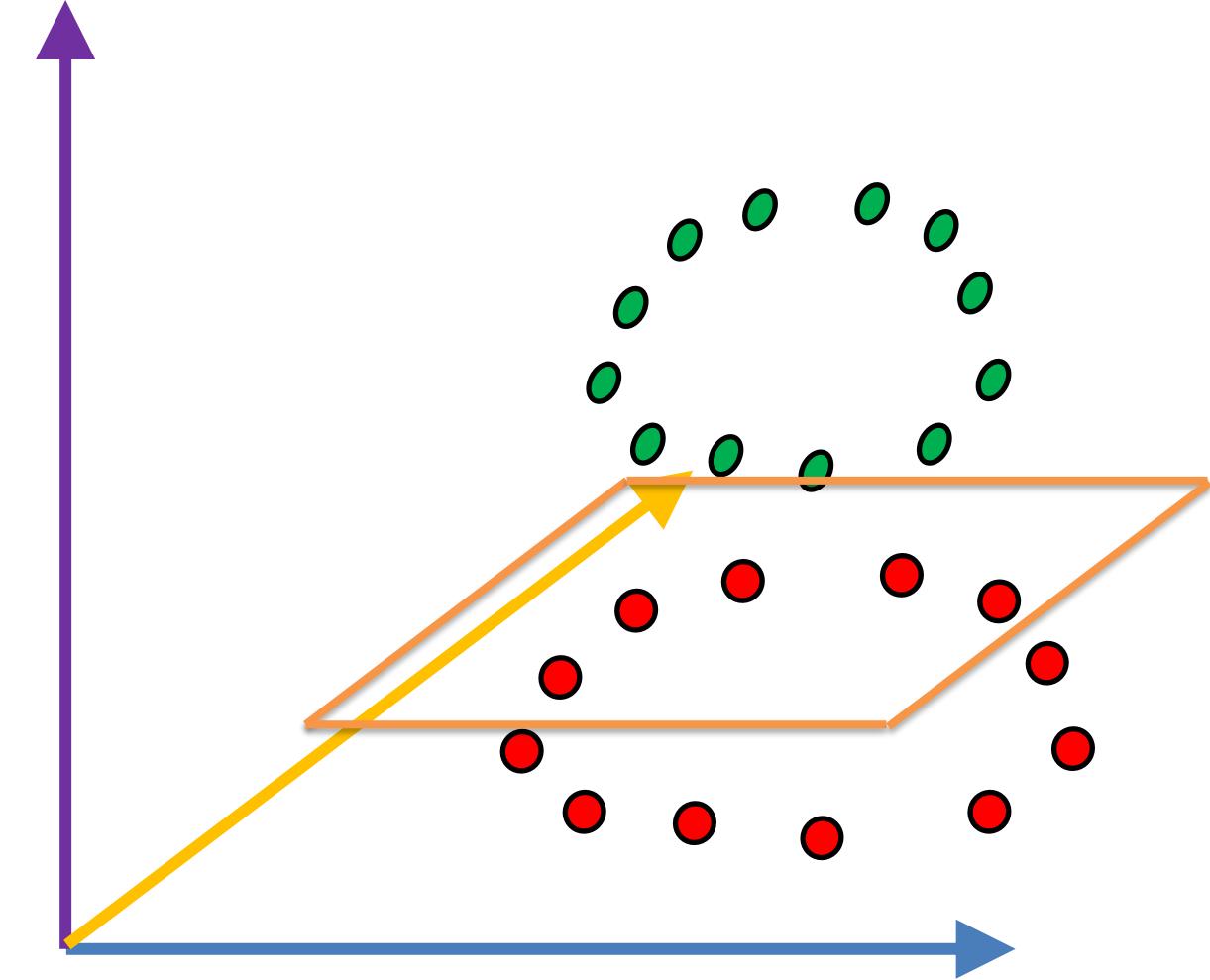
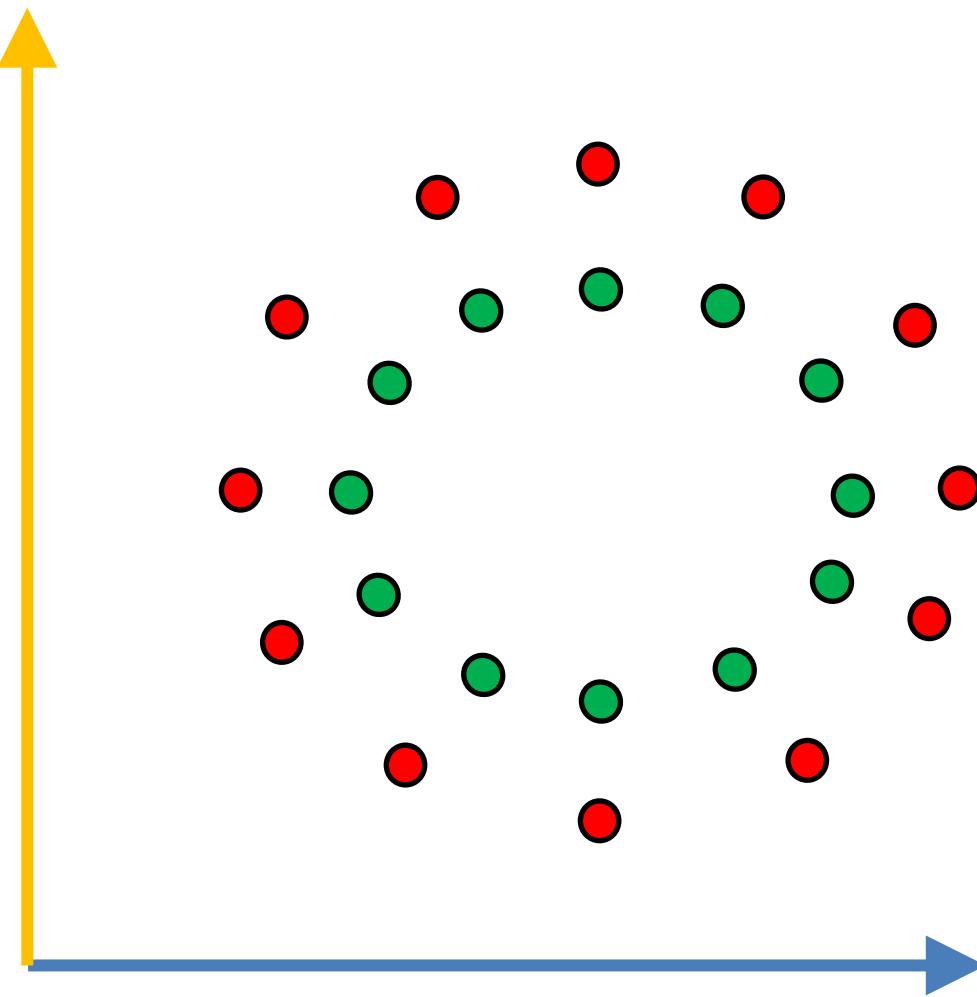


Choose a hyperplane that has the maximum marginal distance .



How does SVM Work?

Non-linearly Separable data



Convert the 2D dimensional problem into a 3D dimensional using SVM Kernels

SVM Kernels

Kernel does a transformation from lower dimension to higher dimension.

- Linear Kernels
- Polynomial Kernels
- Radial Basis Function (RBF) Kernel
- Sigmoid Kernel

SVM Use Cases

- Face detection
- Text and Hypertext Categorization
- Classification of Images
- Bioinformatics
- Remote Homology Detection
- Handwriting Detection
- Generalized Predictive Control

Implementation of SVM

IDAT7215 Lab 5 Tutorial- Supervised Machine Learning Algorithms

Support Vector Machine (SVM) on Breast Cancer Dataset Tutorial

```
In [1]: from sklearn import datasets  
        from sklearn.model_selection import train_test_split  
        from sklearn import svm  
        from sklearn import metrics
```

```
In [2]: cancer_data = datasets.load_breast_cancer()
```

Implementation of SVM

IDAT7215 Lab 5 Tutorial- Supervised Machine Learning Algorithms

Support Vector Machine (SVM) on Breast Cancer Dataset Tutorial

```
In [1]: from sklearn import datasets  
from sklearn.model_selection import train_test_split  
from sklearn import svm  
from sklearn import metrics
```

```
In [2]: cancer_data = datasets.load_breast_cancer()
```

```
In [3]: x_train, x_test, y_train, y_test = train_test_split(cancer_data.data,cancer_data.target,test_size=0.4,random_state=209)
```

```
In [4]: cls = svm.SVC(kernel='linear') # Choose the kernel
```

```
In [5]: # Train the model  
cls.fit(x_train, y_train)
```

```
Out[5]: SVC(kernel='linear')
```

```
In [6]: # Predict the response  
pred = cls.predict(x_test)
```

Implementation of SVM

```
In [11]: # Performance Measure
print("accuracy:", metrics.accuracy_score(y_test,y_pred=pred))      # Accuracy
print("precision:", metrics.precision_score(y_test,y_pred=pred))    # Precision
print("recall:", metrics.recall_score(y_test,y_pred=pred))          # recall
print(metrics.classification_report(y_test,y_pred=pred))           # Total Performance report
```

accuracy: 0.9254385964912281

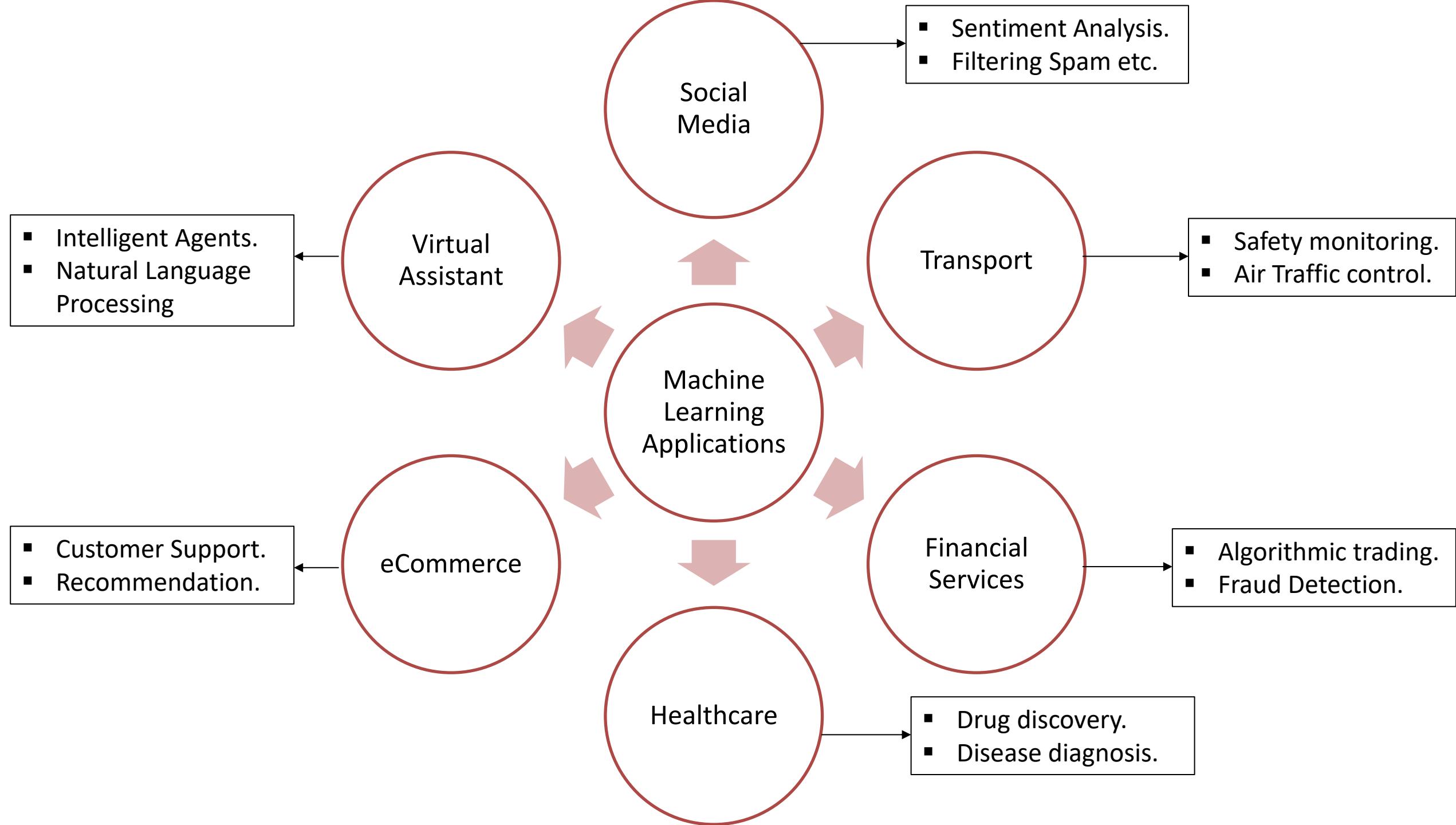
precision: 0.9333333333333333

recall: 0.9402985074626866

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.91	0.90	0.91	94
1	0.93	0.94	0.94	134

accuracy			0.93	228
macro avg	0.92	0.92	0.92	228
weighted avg	0.93	0.93	0.93	228



References

1. Alpaydin, E. (2020). Introduction to machine learning. MIT press.

Resources: Datasets

- UCI Repository: <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- UCI KDD Archive: <http://kdd.ics.uci.edu/summary.data.application.html>
- Statlib: <http://lib.stat.cmu.edu/>
- Delve: <http://www.cs.utoronto.ca/~delve/>

IDAT7215

Computer Programming for Product Development and Applications

Lecture 5-2: Expert Systems and Introduction to
PyKE Library

Dr. Zulfiqar Ali

Outline

- What is Expert System
- Components of Expert System
- Applications of Expert System
- Introduction to PyKE
 - Family Example
 - Weather Example



WHAT IS AN EXPERT SYSTEM ?

- An Expert System (ES) is an example of a **knowledge-based system**. It was introduced around 1965 at Stanford by *Edward Feigenbaum*.
- “ES is a **computer program** that uses artificial intelligence (AI) technologies to **simulate the judgment** and behavior of a **human** or an organization that has **expert knowledge** and **experience** in a particular field such as medical diagnosis, account, coding, and games, etc.”
- ES gathers data by asking the user **questions about the problem**. An initial set of questions can lead to further questions depending on the user's responses.
- The ES reasons what questions it needs to ask, based on the knowledge it is given. It will use the responses from users to rule out various possibilities that will allow it eventually reach a decision or diagnosis.



WHAT IS AN EXPERT SYSTEM ?

- We rely on **expert systems** to help us with knowledge and understanding in a particular field. We use them the same way as certain experts in our lives.
- For example, we will see a doctor if we have a health problem, or we will see a car mechanic if our car will not start.
- We can similarly use an expert system; it will **ask us questions** about our health problem or car problem and **provide us with a diagnosis or course of action**.
- ES is designed to try and **replicate the judgment of a human** that has **expert knowledge** in a certain field. By doing this they **can be used to replace or assist a human expert**.

Why Develop Expert System

- Human expertise:
 - In **short supply (few experts on task)**
 - Expensive to develop or hire
 - Hard to get a hold of in a hurry
 - ✓ Expertise difficult to apply quickly (complex tasks)
 - ✓ Impractical to have humans constantly on call
 - Can be lost
 - ✓ Retirement/Leave job
 - ✓ Death
- ES serves the following functions with respect to expertise:
 - Document/preserve
 - Reproduce/make available
 - Teach/disseminate
- Computers do not get bored/tired/frustrated/scared

Research Expert Systems



ELSEVIER

About Elsevier Products & Solutions Services Shop & Discover

Home > Journals > Expert Systems with Applications



ISSN: 0957-4174

Expert Systems with Applications

An International Journal

Publishing options: OA Open Access ↗ S Subscription ↗

↗ Guide for authors Track your paper ↴ Order journal ↴

Editor-in-Chief > Editorial board



Prof. Dr. Binshan Lin, PhD

Expert Systems With Applications is a refereed international journal whose focus is on exchanging information relating to **expert** and **intelligent systems** applied in **industry**, **government**, and **universities** worldwide. The thrust of the journal is to publish papers dealing with the design, development, testing, implementation,

[Read full aims & scope](#)

ⓘ CiteScore ↗

12.2

ⓘ Impact Factor ↗

8.665

ⓘ Top Readership

CN US GB

ⓘ Publication Time ↗

1.4 weeks

[View historical data and other metrics on Journal Insights.](#)

Submit your paper

ⓘ The Impact Factor of this journal is 8.665, ranking it 23 out of 276 in *Engineering, Electrical & Electronic*

ⓘ With this journal indexed in 7 international databases, your published article can be read and cited by researchers worldwide

[View articles](#)

Research Expert Systems



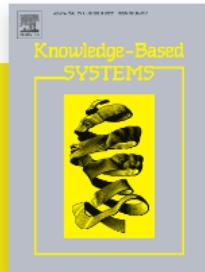
ScienceDirect®

Journals & Books



Register

Sign in



Knowledge-Based
SYSTEMS

Supports open access

12

CiteScore

8.139

Impact Factor

Articles & Issues ▾

About ▾

Publish ▾

Order journal ▾

Search in this journal

Submit your article ↗

Guide for authors ↗

Latest issue

Volume 256

In progress

28 November 2022

About the journal

Knowledge-based Systems is an international and interdisciplinary journal in the field of artificial intelligence. The journal will publish original, innovative and creative research results in the field, and is designed to focus on research in knowledge-based and other artificial intelligence ...

[View full aims & scope](#)

1.2 weeks

Publication Time



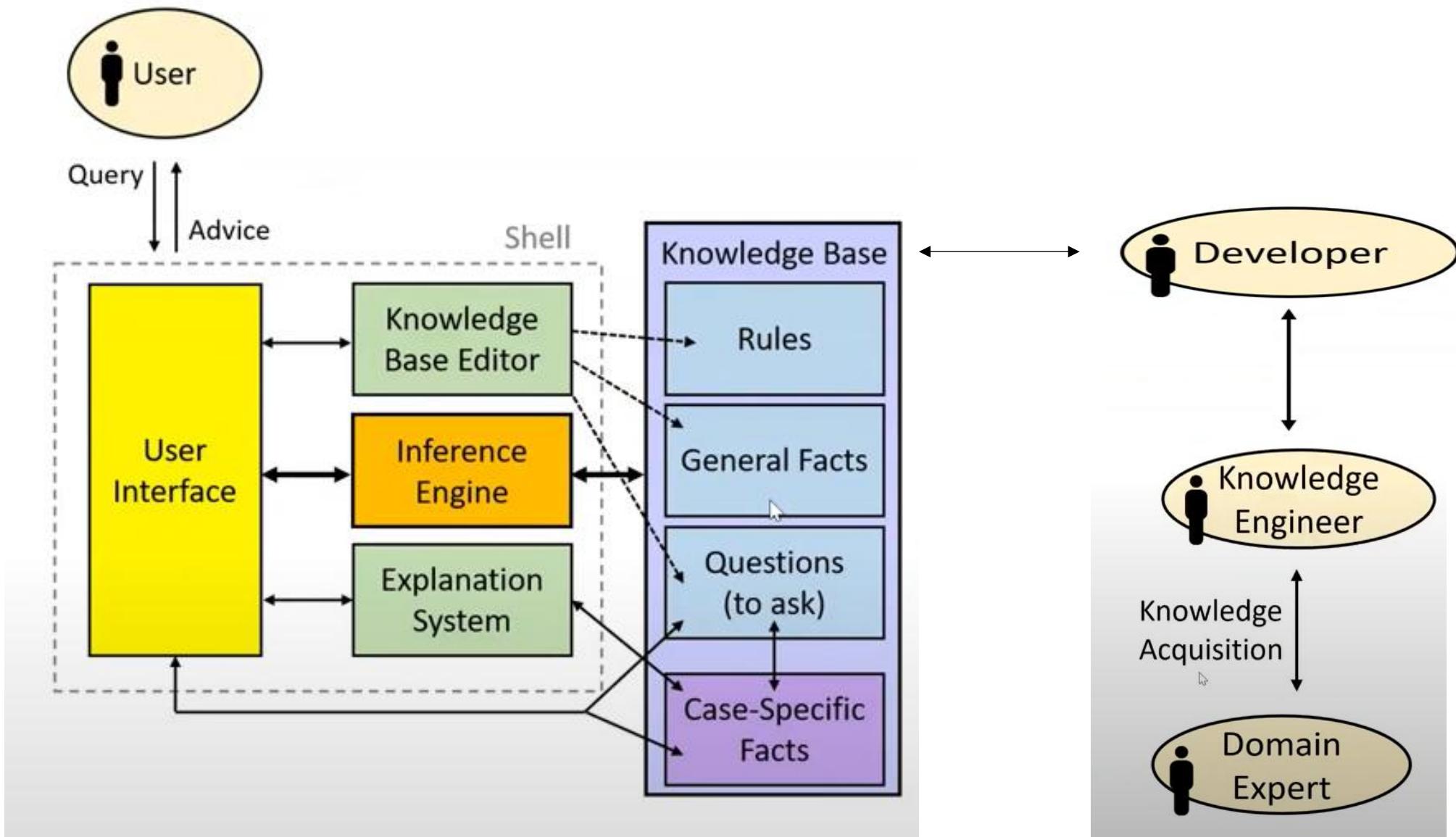
[View all insights](#)

FEEDBACK

ES Building Process

- First attempt unlikely to be very successful
 - Expert generally finds it **difficult to express exactly what knowledge and rules** they use to **solve a problem**.
 - Common sense/subconscious knowledge
 - ✓ Seems so obvious, they don't bother mentioning it.
- Knowledge Acquisition:
 - Gather/extract relevant knowledge.
- Knowledge Engineering
 - Build the expert system knowledge base
- Initial prototype developed and shown to expert
 - Check the system's performance and give feedback.
 - Knowledge base refinement.
- Iterative development from prototype
 - Feedback from experts and end users of the system.

The General Design of an Expert System



Knowledge Base

- The **developers** of the expert system will **interview a collection of experts** to build the **database of knowledge**.
- They will look to gain **two types of knowledge** from the experts.
- The first is **factual knowledge**. This is the knowledge that is widely shared.
- The second type is **heuristic knowledge**. This is the knowledge that is more personal and is acquired through a range of experiences and reasoning.

Knowledge Base

- Once the knowledge base is built, it can be used by the expert system to inform the question it needs to ask and **assist in providing the results**.
- Part of the knowledge base is the **rule base**. The rule base is a **set of rules** that will be **used to produce an output** or decision by the expert system. These **rules are used by the inference engine** as a base for **reasoning**, to obtain a solution to a problem or a decision.
- Each rule will contain two parts, the **IF** and the **THEN**. A rule can also have **multiple IF** parts that will be joined together by Boolean operators including **AND** and **OR**.
- Most ES will have a **knowledge-based editor built** into them. This allows the knowledge base to be checked for errors and edited and updated when needed.

Inference Engine

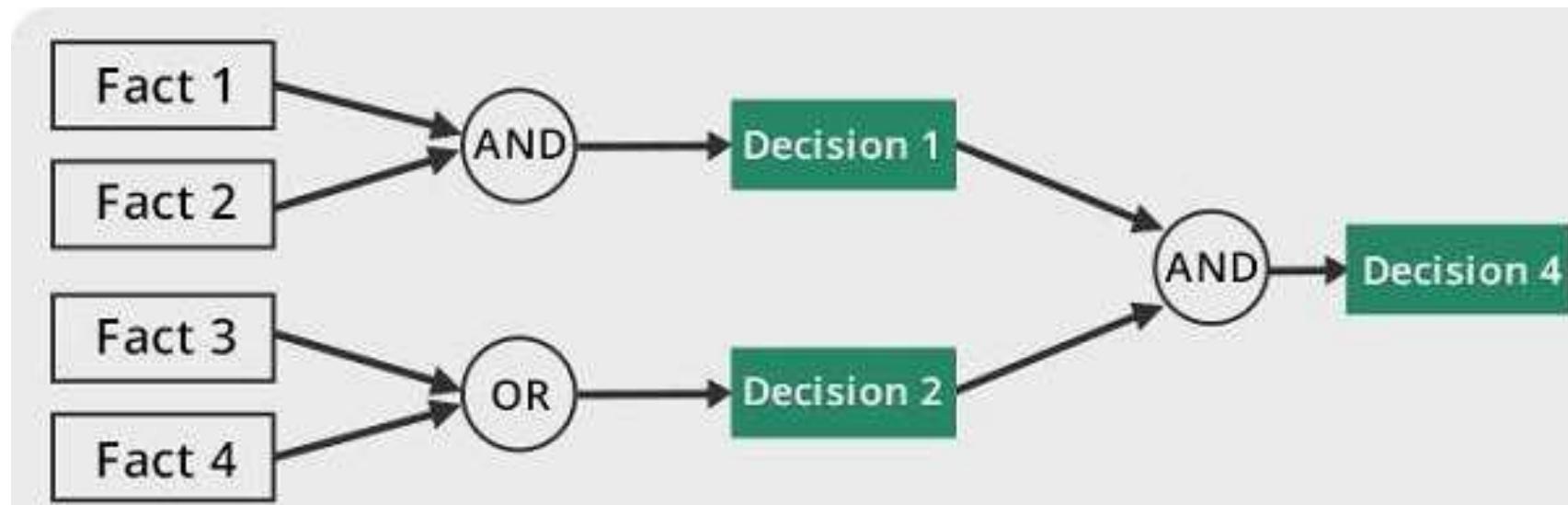
- It will ask the user questions and based on their answer it will follow a line of logic. This may then lead to further questions and eventually a final result.
- The inference engine is mostly a problem-solving tool. It organizes and controls steps to solve the problem. To do this it often uses a chaining method. It chains together what is known as IF-THEN rules to form a line of reasoning.
- The IF part of the rule is a condition, for example, IF I am hungry. The Then part of the rule is an action, for IF I am hungry THEN I will need to eat.

Inference Engine

- There are **two main methods** that can be used to obtain a result. The appropriate one will depend on whether the **expert system** is designed to **produce a final result** i.e., a diagnosis or course of action, or if it begins with a **known conclusion** i.e., goal.
- If the process starts with a set of conditions and chaining moves towards a final conclusion, this is called **forward chaining**.
- If the process starts with a known conclusion, but the path to it is unknown, the chaining is called **backward chaining**.

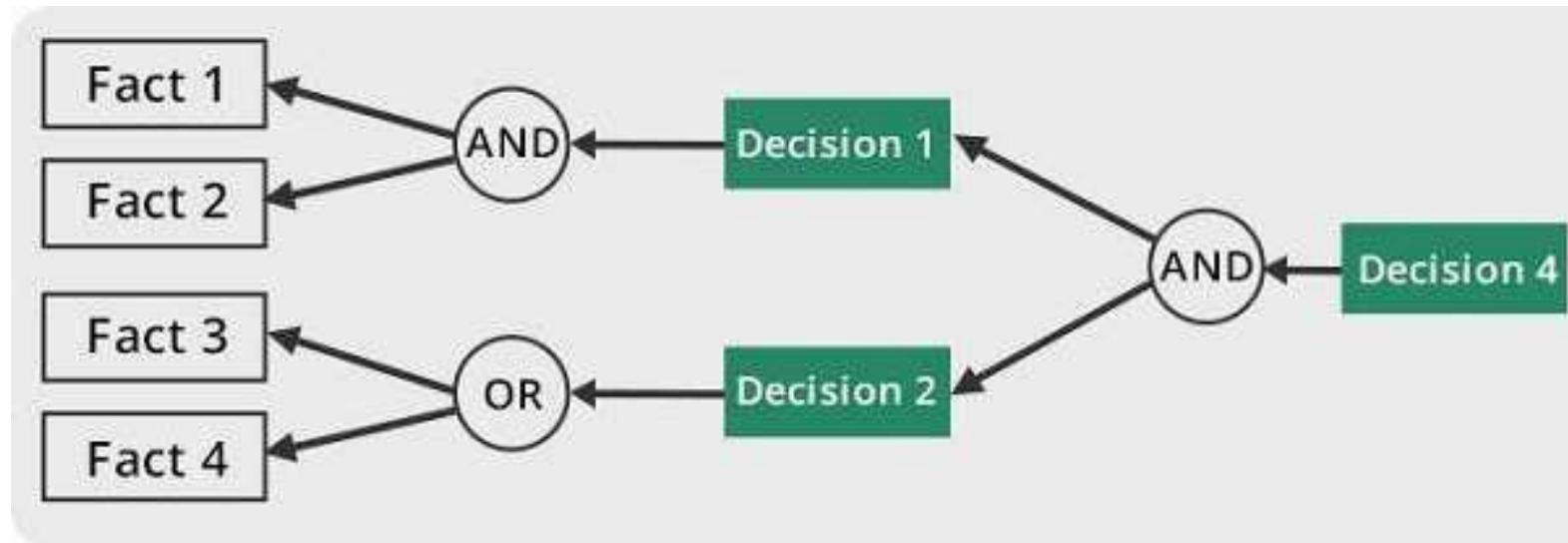
Inference Engine-Forward Chaining

- In a forward chaining system, the ES will **take the data input** and **match it to the knowledge and rules** it contains. It will keep doing this until I can reach an end goal or outcome. Therefore, we can say a **forward chaining system is data-driven**.
- Data is gathered about the problem and then the system infers what it can from the data to reach a conclusion.



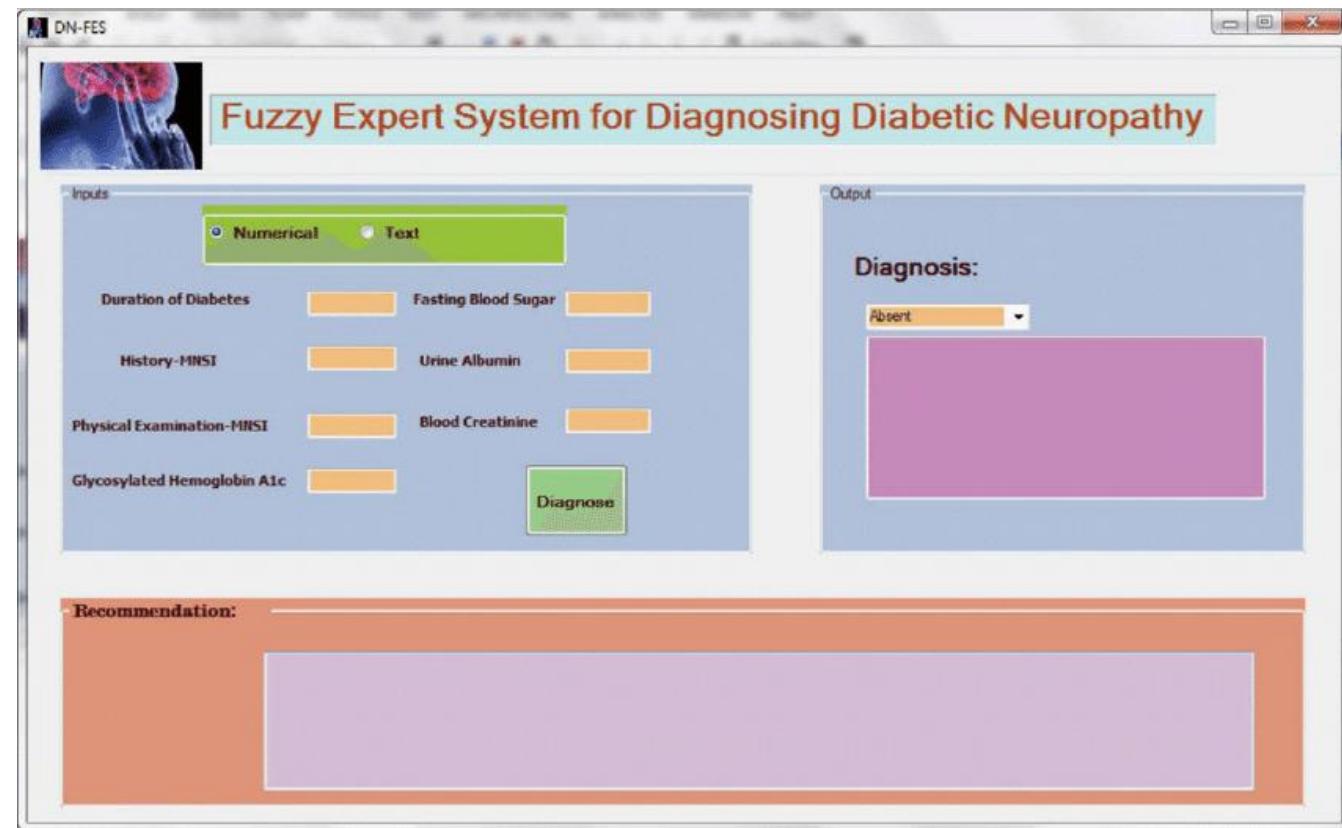
Inference Engine-Backward Chaining

- With this strategy, an expert system finds out the **answer to the question, “Why”**
- On the basis of what has already happened, the Inference Engine tries to **find out which conditions could have happened** in the past for this result.



User Interface

- The user interface is the way the user **interacts** with the expert system.
- They are **often graphical** in nature and have a **range of selection processes** or typing methods to allow the user to provide responses.



Example

The chart is titled "IS IT A COLD or FLU?" in large, bold, white letters on a yellow background. It features green, spiky virus-like icons around the title and the table border. The table has a green header row with three columns: "Signs and Symptoms", "Cold", and "Flu". The rows represent different symptoms: "Symptom onset", "Fever", "Aches", "Chills", "Fatigue, weakness", "Sneezing", "Stuffy nose", "Sore throat", "Chest discomfort, cough", and "Headache". For each symptom, the "Cold" column lists the frequency and the "Flu" column lists the frequency.

Signs and Symptoms	Cold	Flu
Symptom onset	Gradual	Abrupt
Fever	Rare	Usual
Aches	Slight	Usual
Chills	Uncommon	Fairly common
Fatigue, weakness	Sometimes	Usual
Sneezing	Common	Sometimes
Stuffy nose	Common	Sometimes
Sore throat	Common	Sometimes
Chest discomfort, cough	Mild to moderate	Common
Headache	Rare	Common

#FIGHT FLU

Flu=0

Cold=0

IF onset="Abrupt" flu=flu+1

ELSE cold=cold+1

IF fever == TRUE flu=flu+1

ELSE cold = cold +1

IF chills = TRUE flu = flu +1

ELSE cold=cold+1

...

IF fever> cold print("It's probably the flu")

ELSE print("It's probably a cold")

Demo: Identify a sport you would enjoy

- Try this via clicking the weblink

The screenshot shows a web browser window with the following details:

- Address Bar:** magicmonktutorials.com/IT/expertsystem/html/con7.html
- Toolbar:** Includes standard browser icons for back, forward, search, and refresh.
- Bookmark Bar:** Contains links to various websites like City University, BOOK DOWNLOADS, conference, educational websites, Conferences IGA, Reminder_links, HKU, Research IGA, Google Scholar, Gmail, XE - The World's Fa..., Home Feed, and Sci-Hub: removing... A total of 15 items are listed.
- Content Area:**
 - A message in blue text: "Based on the responses you have made, the most appropriate conclusion is:"
 - The result: "Ballroom dancing"
 - A link: "[Identify a sport you would enjoy](#)"
 - Navigation links at the bottom: "Title Page", "Search", "Knowledge Base", "Decision Tree", "Decision Table", "Attributes List", and "Documentation".

This Expert System Web created with ES-Builder 3.0 Expert System Shell.
©2009 McGoo Software.

<https://magicmonktutorials.com/IT/expertsystem/html/con7.html>

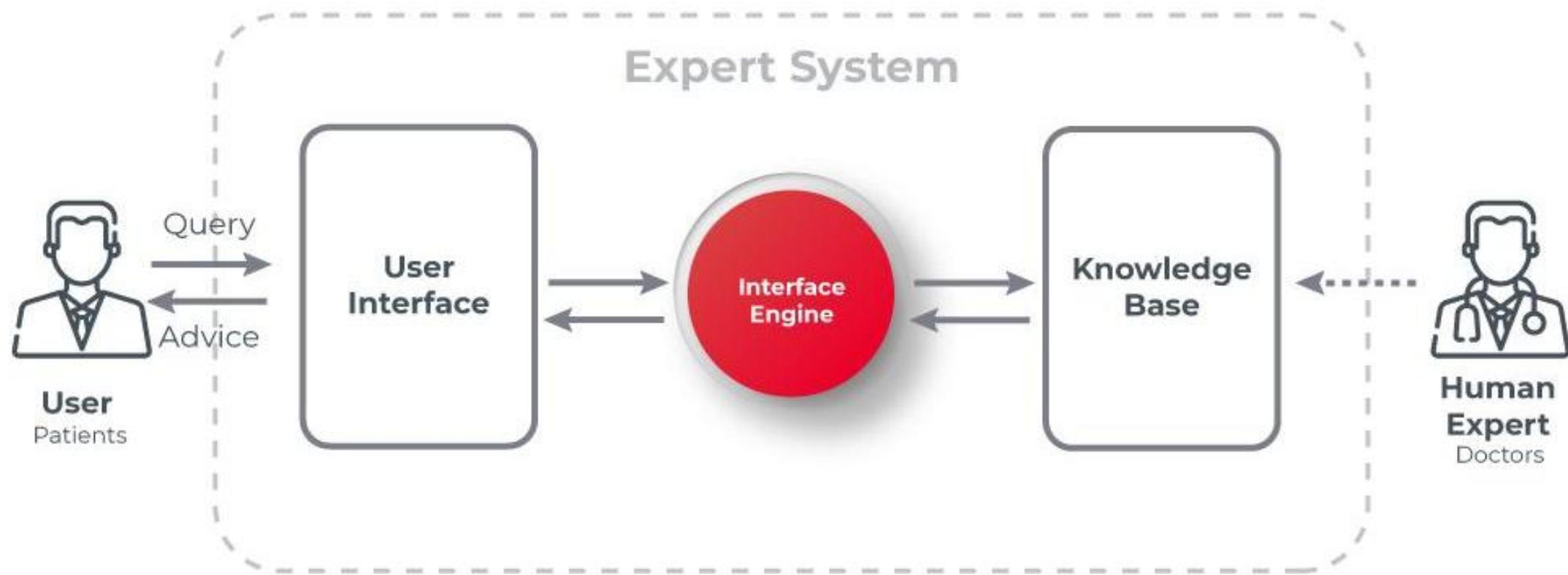
Shallow Expert System vs Deep Expert System

- Shallow Expert Systems are first-generation architecture expert systems, and which lack causal knowledge (knowledge of the underlying causes and effects in a system) while Deep Expert Systems are second-generation architecture expert systems that alleviate first-generation architecture expert systems' limitations.
- Shallow Expert Systems use surface knowledge which can only represent expertise while Deep Expert Systems use deep knowledge which can represent complex structured objects using object models.
- Shallow Expert Systems use rule-based reasoning while Deep Expert Systems combine both rule-based reasoning and model-based reasoning.
- Shallow Expert Systems use heuristics or heuristic rules or rules of thumb, which do not guarantee optimal solutions while Deep Expert Systems use the A* algorithm which guarantees optimality by finding solutions in reasonable computation time, in terms of, number of transformations and steps of a search.

Applications of Expert System

- Medical diagnosis.
- Providing financial advice.
- Manufacturing Production & Maintenance
- Troubleshooting computer and printer issues.
- Identifying items, for example, plants and birds.
- Using a telephone help desk
- Playing chess

MYCIN: Early Medical Diagnostic System (1970s)



MYCIN: Early Medical Diagnostic System (1970s)

- MYCIN was the earliest designed expert system at Stanford University in the 1970s.
- Its job was to diagnose and recommend treatment for certain blood infections.
- MYCIN was written in LISP Programming Language.
- ~ 600 rules to diagnose the presence of certain bacteria and recommend antibiotics.
- Combined rules in a more sophisticated way (Bayesian inference)
- Never used in practice due to ethical concerns.
- Inspired the creation of development environments for others who DID make expert systems that were put into use.

MYCIN: Early Medical Diagnostic System (1970s)

- MYCIN represented its knowledge as a set of **IF-THEN rules** with **certainty factors**. The following is an English version of one of MYCIN's rules:

IF the infection is **primary-bacteremia**

AND the site of the culture is one of the **sterile sites**

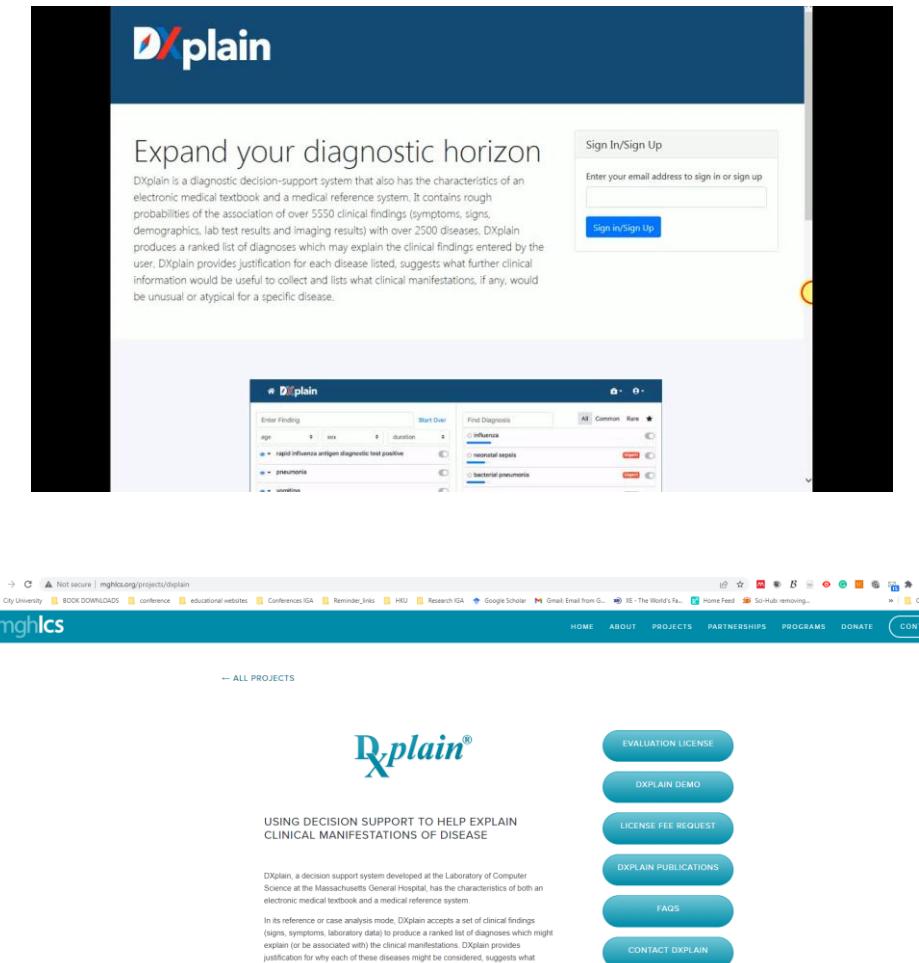
AND the suspected portal of entry is the **gastrointestinal tract**

THEN there is suggestive evidence (0.7) that **infection is bacteroid.**

- The 0.7 is roughly the **certainty** that the **conclusion will be true** given the evidence. If the evidence is uncertain, the **certainties** of the bits of evidence will be combined with the **certainty of the rule** to give the **certainty of the conclusion**.

Other Medical Expert System

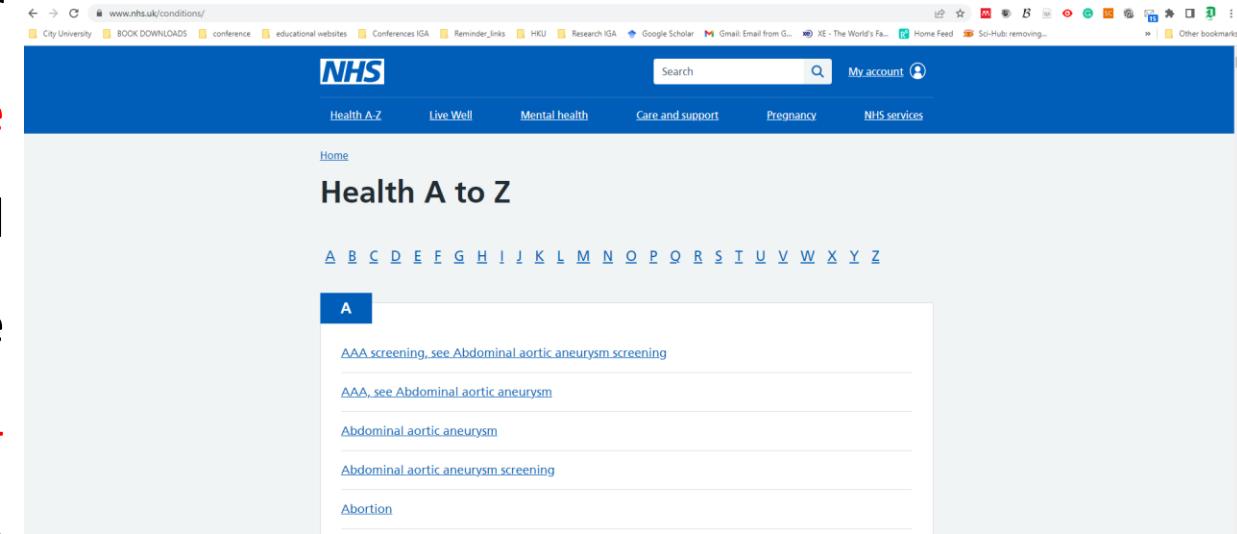
- **DXPLAIN:** It is used for diagnosis. DXplain provides **justification for why each of these diseases might be considered**, suggests what further clinical information would be useful to collect for each disease, and lists **what clinical manifestations**.
- **CaDet:** It is for early **cancer detection**. Clinical data related to **early cancer detection** and to cancer risk factors was collected and incorporated into the database, together with **heuristic rules** for evaluating this data.



<http://www.mghlcs.org/projects/dxplain>

Example- National Health Service (NHS) UK

- The National Health Service (NHS) is the medical system in the UK. The NHS has a website that **allows users to enter the symptoms of their illness or ailment and it will provide them with a possible diagnosis.** This may help a person to understand which possible illnesses they may have. It can be used by professionals within the NHS to double-check a diagnosis they are giving, or to aid them with any areas of weakness they have in their medical knowledge.



<https://www.nhs.uk/conditions/>

Banking and Financial Sector:

- An Expert System that helps bank managers in making decisions on granting loans.
- An Expert System that advises bank managers in giving housing loans.
- An Expert System that advises insurance companies on the risks involved in insuring a customer or a company.
- An Expert System that helps banks decide on whether a customer is entitled to a credit card or not.
- An Expert System that identifies computer fraud and controls it.

Application of Expert System Production Scheduling

- Due to the changing production style and highly dynamic variations in production requirement Expert System Manufacturing is one of the widely applied used areas.
- One of the most common uses of ES in manufacturing **is Scheduling and Production planning.**
- The first application of an Expert System in job scheduling is **Intelligent Scheduling and Information System (ISIS)**, 1984.

Application of Expert System Production Scheduling

- Some Scheduler using Expert System technology

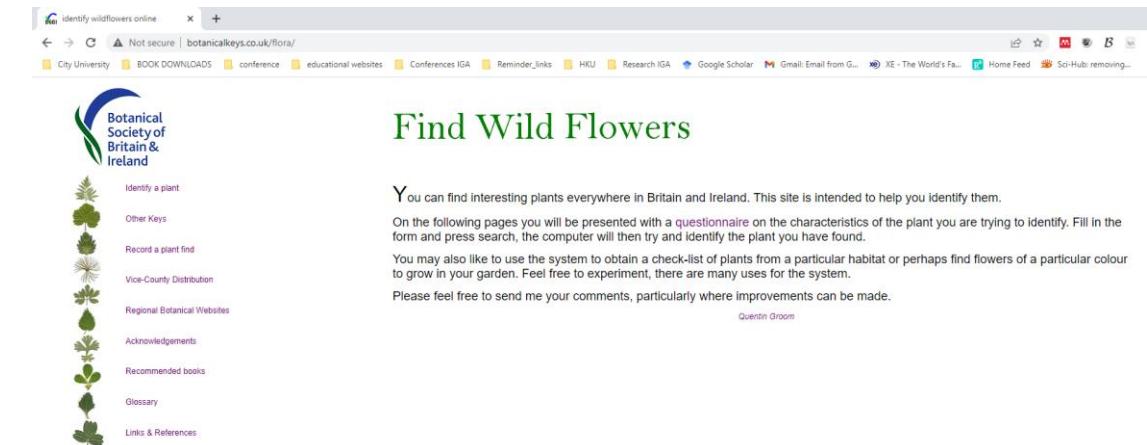
<i>Project</i>	<i>Group</i>	<i>Domain</i>
MASCOT	Parunak, 1993 ITI	Manufacturing scheduling and control
DAS	Burke and Prosser, 1991 University of Strathclyde	Manufacturing scheduling
ABACUS	McEleney <i>et al.</i> , 1998 UCB, UMIST	Manufacturing scheduling
MetaMorph II	Shen <i>et al.</i> , 1998 University of Calgary	Intelligent manufacturing production
SFA	Parunak, 1996 NCMS	Manufacturing scheduling and control
A case based expert system for generative computer-aided process planning with manufacturing uncertainty	Wong, 1997 MSERC	Manufacturing planning
IAO	Kwok and Norrie, 1994 University of Calgary	Intelligent manufacturing

Application of Expert System Production Scheduling

- Advantages of using Expert System in Manufacturing According to *Byrd (1995)*
 - Reduction in time to complete the tasks
 - Increase in production
 - Reduction in stuff
 - More effective use of resource

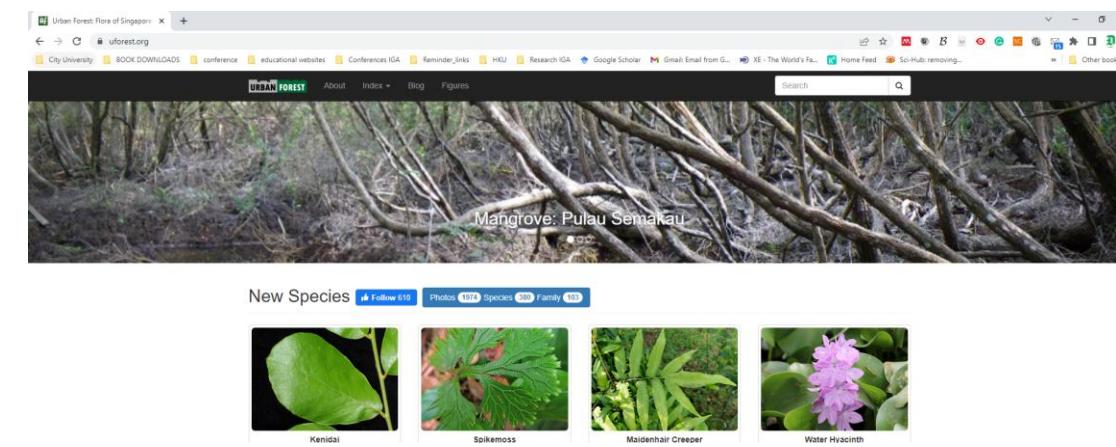
Botanical Keys –UK (Plant identification)

- The website Botanical keys allow a user to enter a range of characteristics about a plant. The system will then identify what the plant is. Users can use this to find out what plants are growing in their garden or identify a plant they have seen when out on a walk in the countryside or through a park.



<http://www.botanicalkeys.co.uk/flora/>

- Urban Forest is a non-profit website in Singapore that contains information about flora. From Singapore and Southeast Asia-users can visit the site to help them with plant identification.



<https://uforest.org/>

Expert System

Advantages	Disadvantages
<ul style="list-style-type: none">They provide answers to questions that are outside a user's knowledge and experience.	<ul style="list-style-type: none">They do not have the addition of common sense that humans have. This means that their response can be a logical one and cannot have a creative approach.
<ul style="list-style-type: none">They can aid professionals in areas where their knowledge and experience are a little weaker.	<ul style="list-style-type: none">Errors in the knowledge base or inference engine will produce errors in the results. Therefore, they are only as good as the data and rules they are given.
<ul style="list-style-type: none">As they use a logical process they are consistent in the answers that they give.	<ul style="list-style-type: none">They are not able to automatically adapt to changing environments. This requires the knowledge base to be changed.
<ul style="list-style-type: none">They will not forget to ask a question, as all the rules and knowledge is put into place to make sure that all data needed is gathered.	<ul style="list-style-type: none">They are expensive to produce as they require a great deal of time and effort from experts and developers.

Installing Pyke library in the Anaconda Distribution

conda-forge / packages / pyke 1.1.1



Python Knowledge Engine and Automatic Python Program Generator

[Conda](#) [Files](#) [Labels](#) [Badge](#)

www.nature.com/scientificreports/

 License: [MIT](#)

Installers

Info: This package contains files in non-standard labels

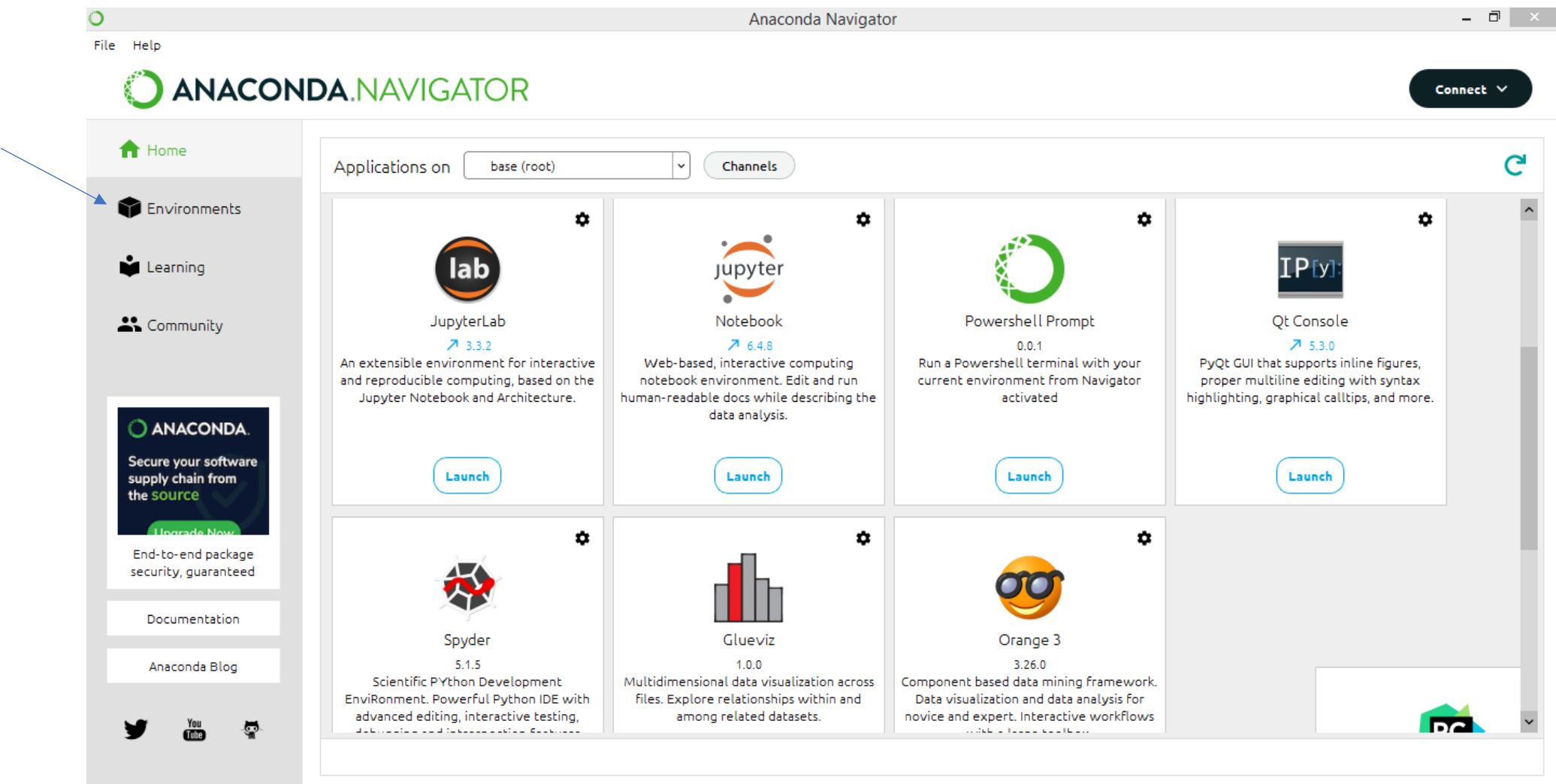
conda install ?

-  linux-64 v1.11
-  win-32 v1.11
-  noarch v1.11
-  win-64 v1.11
-  osx-64 v1.11

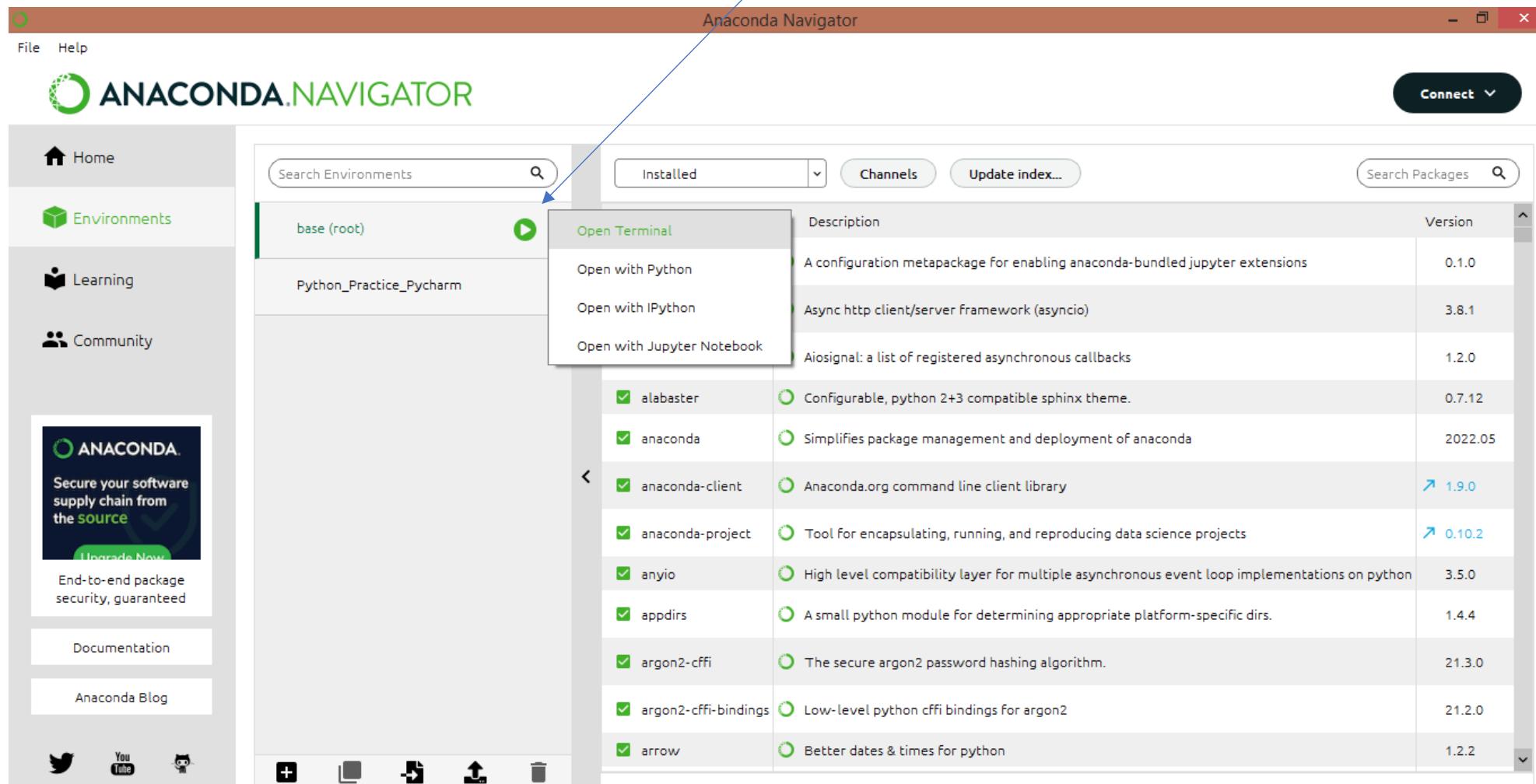
To install this package run one of the following

```
conda install -c conda-forge pyke
```

<http://pyke.sourceforge.net/>



Open Terminal



ANACONDA.NAVIGATOR

[Home](#)[Environments](#)[Learning](#)[Community](#)Search Environments base (root) 

Installed

Channels

Update index...

C:\WINDOWS\system32\cmd.exe - conda install -c conda-forge pyke

```
C:\Users\Dr. Ali>rem proj-data is installed because its license was copied over
(base) C:\Users\Dr. Ali>    conda install -c conda-forge pyke
Collecting package metadata (current_repodata.json): done
Solving environment: /
Warning: 8 possible package resolutions (only showing differing packages):
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, anaconda/win-64::certifi-2021.10.8-py39haa95532_2, anaconda/win-64::openssl-1.1.1n-h2bbff1b_0
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, anaconda/win-64::certifi-2021.10.8-py39haa95532_2, defaults/win-64::openssl-1.1.1n-h2bbff1b_0
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, anaconda/win-64::openssl-1.1.1n-h2bbff1b_0, defaults/win-64::certifi-2021.10.8-py39haa95532_2
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, defaults/win-64::openssl-1.1.1n-h2bbff1b_0, defaults/win-64::certifi-2021.10.8-py39haa95532_2
- anaconda/win-64::openssl-1.1.1n-h2bbff1b_0
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, defaults/win-64::openssl-1.1.1n-h2bbff1b_0, defaults/win-64::certifi-2021.10.8-py39haa95532_2
- anaconda/win-64::ca-certificates-2022.3.29-haa95532_1, defaults/win-64::openssl-1.1.1n-h2bbff1b_0, defaults/win-64::certifi-2021.10.8-py39haa95532_2
anyio - defaults/win-64::ca-certificates-2022.3.29-haa95532_1, defaults/win-64::certifi-2021.10.8-py39haa95532_2, defaults/win-64::openssl-1.1.1n-h2bbff1b_0
appdirs - anaconda/win-64::certifi-2021.10.8-py39haa95532_2, anaconda/win-64::openssl-1.1.1n-h2bbff1b_0, defaults/win-64::ca-certificates-2022.3.29-haa95532_1
argon2 - anaconda/win-64::certifi-2021.10.8-py39haa95532_2, defaults/win-64::ca-certificates-2022.3.29-haa95532_1, defaults/win-64::openssl-1.1.1n-h2bbff1b_0done
arrow
astroid
astropy
```

A abstract syntax tree for python with inference support.

Community-developed python library for astronomy

PyKE

python knowledge engine

Home

[Home](#)
[About Pyke](#)
[Logic Programming](#)
[Knowledge Bases](#)
[Pyke Syntax](#)
[Using Pyke](#)
[Examples](#)
[PyCon 2008 Paper](#)

[Pyke Project Page](#)
Please Make a Donation:
[Support this project](#)
Hosted by:


Welcome to Pyke

Release 1.1

Pyke introduces a form of [Logic Programming](#) (inspired by [Prolog](#)) to the Python community by providing a knowledge-based inference engine (expert system) written in 100% Python.

Unlike Prolog, Pyke integrates with Python allowing you to invoke Pyke from Python and intermingle Python statements and expressions within your expert system rules.

Pyke was developed to significantly raise the bar on code reuse. Here's how it works:

1. You write a set of Python functions, and a set of Pyke [rules](#) to direct the configuration and combination of these functions.
2. These functions refer to Pyke [pattern variables](#) within the function body.
3. Pyke may instantiate each of your functions multiple times, providing a different set of constant values for each of the pattern variables used within the function body. Each of these instances appears as a different function.
4. Pyke then automatically assembles these customized functions into a complete program (function call graph) to meet a specific need or use case. Pyke calls this function call graph a [plan](#).

In this way, Pyke provides a way to radically customize and adapt your Python code for a specific purpose or use case.

Doing this essentially makes Pyke a very high-level compiler. And taking this approach also produces dramatic increases in performance.

And Pyke is very successful at this, providing order of magnitude improvements in:

- Code adaptability (or customization),
- Code reuse and
- Performance

Pyke does not replace Python, nor is meant to compete with Python. Python is an excellent general purpose programming language, that allows you to "program in the small".

Pyke builds upon Python by also giving you tools to directly [program in the large](#).

Oh, and Pyke uses Logic Programming to do all of this. So if you're interested in Logic Programming or Expert Systems, well Pyke has that too...

Pyke on Google Groups

Please join [Pyke](#) on Google Groups for questions and discussion!

FAQ

There is also an [FAQ](#) list on the sourceforge [wiki](#), to make it easy to contribute.

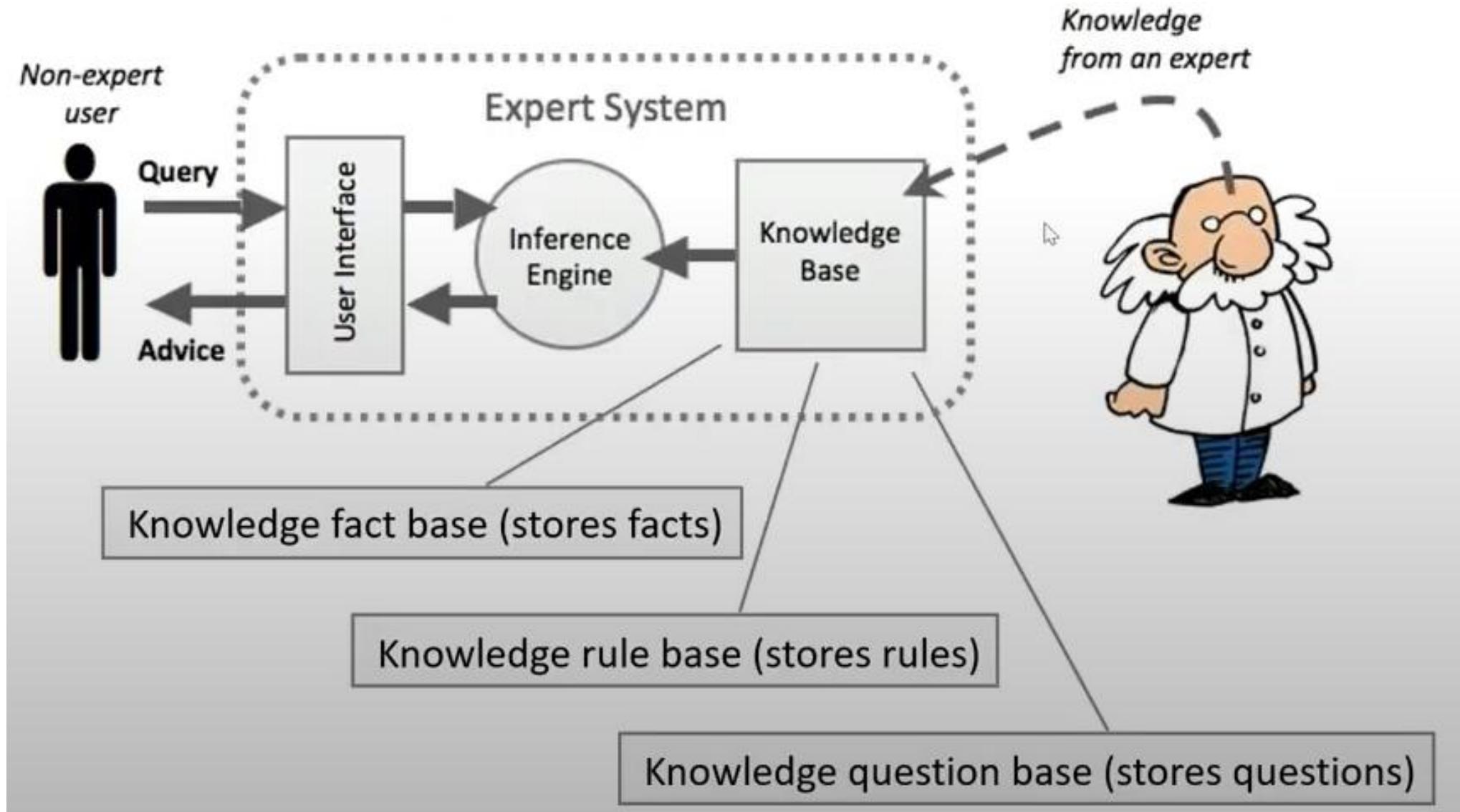
More:

[About Pyke](#)
[What pyke does for you, its features, steps to using pyke and installation.](#)
[Logic Programming Tutorial](#)
[A tutorial on logic programming in Pyke, including statements, pattern matching and rules.](#)
[Knowledge Bases](#)
[Knowledge is made up of both facts and rules. These are gathered into named repositories called knowledge bases.](#)
[Pyke Syntax](#)
[The syntax of Pyke's three different kinds of source files.](#)
[Using Pyke](#)
[How your Python program calls Pyke.](#)
[Examples](#)
[An overview of the examples provided with Pyke.](#)
[Applying Expert System Technology to Code Reuse with Pyke](#)
[Paper presented at the PyCon 2008 conference in Chicago.](#)

What is PyKE

- Python Knowledge Engine (PyKE)
- An **expert system interpreter and coding framework** that we will use in this course **to learn how to build and run a simple expert system.**
- Integrates a form of logic programming into python.
 - Provides an **inference engine** that :
 - Applies **rules to facts to establish** additional facts (forward chaining)
 - And/or prove goals (backward chaining)
 - Optionally **assemble Python functions** into customized call graphs, called plans..
- http://pyke.sourceforge.net/about_pyke/index.html

What is PyKE



Why PyKE

- PyKE was chosen because it **operates exclusively in Python** (and unlike PyCLIPS, it works in Python 3).
- Probably the most commonly used expert system framework today is called CLIPS, however, this is a C coding language interpreted system
 - If you want to (optionally) check out CLIPS, see the link below
 - Link: CLIPS website <https://www.clipsrules.net/>

Using Pyke- The simples use of PyKE

- IDAT7215 Lab 5 Tutorial- PyKE Installation and Family Relations Expert System Test.ipynb

Pyke Instructions

1. Download pyke3-1.1.1.zip from <https://sourceforge.net/projects/pyke/files/pyke/1.1.1/>
2. Unzip pyke3-1.1.1.zip (if you are unfamiliar with how to unzip a file see <https://www.wikihow.com/Unzip-a-File>)
3. Check the unzipped folder to ensure its contents include folders by the name of (build, doc, examples, experimental, pyke, Test) along with a number of other files including README.txt. (Note when working with newly installed code, the README file includes important information on how to get started with and run the code)
4. Copy the folder with these PyKE files to where ever you wish to work from (i.e. your working directory).
5. Take a look at the pyke example folder '/examples/family_relations/'.
 - Within that folder you should see 7 files including 'driver.py'
 - There are three kinds of Pyke source files:
 - .kfb files define fact bases
 - .krb files define rule bases
 - .kqb files define question bases
 - Open and read the README files in this family_relations folder for an explanation of the different files and a set of examples that may be run.
6. Copy your unique path to this '/family_relations/' into the 'sys.path.append()' command in subtask 3. I've left how I did this on my computer as an example.
7. If these steps have been completed successfully then you should be able to run the commands in the cells below...

Using Pyke- The simples use of Pyke

- Step 1: Create an engine object

- From pyke import knowledge_engine
 - my_engine = knowledge_engine.engine(__file__)

- Step 2: Activate rule bases

- my_engine.activate('rule_base_file')

- Step 3: Prove goals

- from pyke import goal
 - my_goal=goal.compile('goal')

Load PyKE Driver

```
In [1]: #Run this cell
import sys
# The following command points your notebook to the location of a folder outside your working directory that you want to import.
sys.path.append('C:/Users/Dr. Ali/Lab 04-Expert System/pyke3-1.1.1/pyke-1.1.1/examples/family_relations')
#sys.path.append('yourpath')
import driver
```

```
import contextlib
import sys
import time

from pyke import knowledge_engine, krb_traceback, goal

# Compile and load .krb files in same directory that I'm in (recursively).
engine = knowledge_engine.engine(__file__)

fc_goal = goal.compile('family.how_related($person1, $person2, $relationship)')

def fc_test(person1 = 'bruce'):
    ...
        This function runs the forward-chaining example (fc_example.krb).
    ...
    engine.reset()      # Allows us to run tests multiple times.

    start_time = time.time()
    engine.activate('fc_example')  # Runs all applicable forward-chaining rules.
    fc_end_time = time.time()
    fc_time = fc_end_time - start_time

    print("doing proof")
    with fc_goal.prove(engine, person1=person1) as gen:
        for vars, plan in gen:
            print("%s, %s are %s" % \
                  (person1, vars['person2'], vars['relationship']))
prove_time = time.time() - fc_end_time
print()
print("done")
engine.print_stats()
print("fc time %.2f, %.0f asserts/sec" % \
      (fc_time, engine.get_kb('family').get_stats()[2] / fc_time))
```

driver.py

Test PyKE: Family-Forward Chaining

Test 1

```
In [2]: # Run this cell
driver.fc_test('michael_k')

doing proof
michael_k, amanda are ('father', 'daughter')
michael_k, tammy are ('father', 'daughter')
michael_k, crystal are ('father', 'daughter')

done
family: 9 fact names, 94 universal facts, 6920 case_specific facts
fc_example: 20 fc_rules, 6772 triggered, 892 rerun
fc_example: 0 bc_rules, 0 goals, 0 rules matched
          0 successes, 0 failures
fc time 0.35, 19964 asserts/sec
```

Test 2

In this second test, we simply demonstrate what happens when a name is given that doesn't appear in the fact base.

```
In [3]: # Run this cell
driver.fc_test('ryan')

doing proof

done
family: 9 fact names, 94 universal facts, 6920 case_specific facts
fc_example: 20 fc_rules, 6772 triggered, 892 rerun
fc_example: 0 bc_rules, 0 goals, 0 rules matched
          0 successes, 0 failures
fc time 0.33, 21235 asserts/sec
```

Knowledge Bases- Fact Bases and Rule bases

1. Fact base stores facts in the form of a knowledge fact base (.kfb) file

- Two types of facts:
 - Case Specific facts: will be **deleted when the inference engine is rested** to prepare for another run of the inference engine.
 - Universal facts: **never deleted**.

2. Rule base stores rule in the form a of knowledge rule base (.krb) file.

- A single rule base may contain both forward chaining rules and backward chaining rules.

Knowledge Bases- Fact Bases and Rule bases

- **Question base** stores questions for end users in the form of a knowledge question base (**.kqb**) file
- The .kqb file contains all the information about the question needed to ask the question, validate the answer, and output the appropriate review text.
- Questions can be of various types (e.g., yes/no, true/false, multiple choice)
- Example Question

- Who_father(\$ans)

Who is user father?

\$ans = select_1

1. Joe 2. Steve 3. Bill 4. Dave

PyKE: What is statement

- A statement is a generalized expression of knowledge, also just called a **fact**.
- Three components
 1. Name of a knowledge base
 2. Name of knowledge entity
 - The relationship between the arguments.
 3. State arguments
 - Can be any simple python data value (numbers, strings, None, True or False) or tuples of these values (including nested tuples)
 - Change of order arguments would change the meaning of a statement

```
family.son_of(son, father, mother)
```

`family.son_of(Bruce, Thomas, Norma)`

Semantics: “Bruce is the son of Thomas (his father) and Norma(his mother).”

PyKE: Pattern Matching

- Literal Patterns:

- Look exactly like data
- Only match themselves.
- Serve as inputs to Pyke.
- Used when asking “ Is statement X true?”

```
family.son_of(son, father, mother)
```

- Pattern Variables:

- Serve as output parameters.
- Start with a \$.
- Either bound to a value (literal pattern) or unbound.
- Used when asking “ Who are the sons of Thomas and Norma?”

```
family.son_of($son, father, mother)
```

- Anonymous pattern variables:

- Start with \$ and an underscore (\$_).
- Never bound to values.
- Used when asking “Who are Norma’s sons?” and don’t care who are father is.

```
family.son_of($son, $_father, mother)
```

PyKE: Rules

- Rules have two parts:
 - If part (containing a list of statements called the **premises**)
 - Then part (containing a list of statements called the **conclusions**)
- Each of these if and then parts contain **one or more** facts or goals

Syntax: If [A], [B], and [C], Then [D] and [E]

Semantics: If A, B and C are true, then D and E are true.

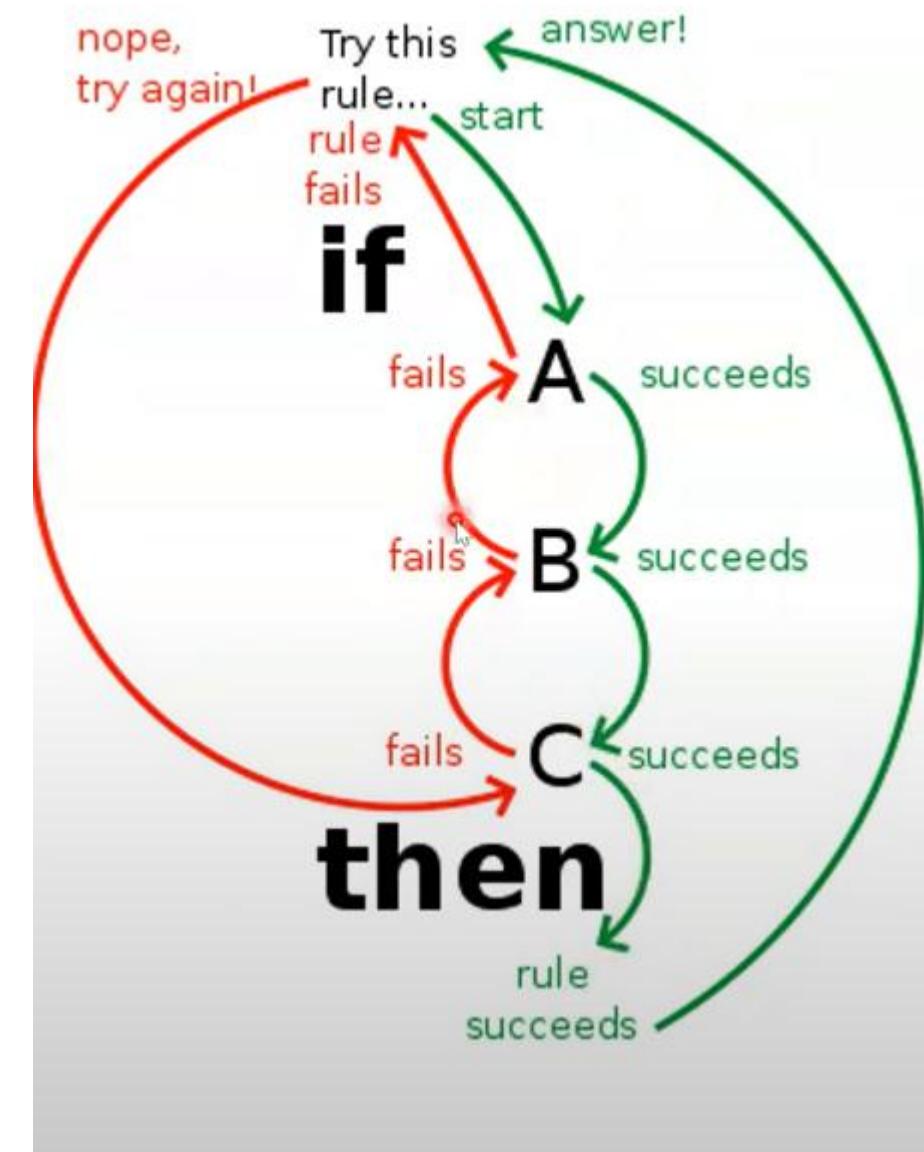
- In order to make the rule succeed, Pyke tries to match all statements with facts within the if clause through a process called **backtracking**.

PyKE: Backtracking

While processing the list premises (A, B, C in the graph)

Within a rule's if clause:

- If succeeds at proving a premise:
 - Proceed down to the **next premise in the list**.
 - Trying to proceed down from the last premise in the list (when it succeeds) causes the rule to succeed.
- If fails at proving a premise:
 - Back up to the **prior premise in the list** and try to find another solution for it.
 - Trying to back up from the **first premise in the list** (when it fails) causes the rule to fail.



PyKE: Forward Chaining Rules

- Use “**Foreach**”, “**Assert**” rather than “**If**”, “**Then**”
- Steps:
 1. Pyke starts with the *if* clause of the **first rule** and checks to see if it **matches the known facts**
 2. If so, it **proceeds to the then clause** of that rule (firing the rule) to **assert a new fact**.
 3. These **new facts may fire other forward-chaining rules** by matching their if clause.
 4. The process keeps going until no more rules could be fired.

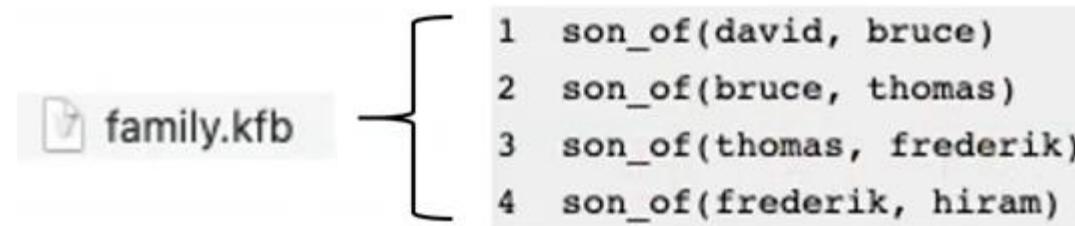
PyKE: Backward Chaining Rules

- Use “**Use**”, “**When**” rather than “**Then**”, “**If**”
- Steps:
 1. Pyke starts by **finding a rule** whose then part **matches the goal**.
 2. Pyke then tries to **prove all the subgoals** in the if part of that rule
 - Some of **these subgoals** are **matched against facts**
 - Others are matched with then part of other backward-chaining rules.
 3. If **all of the subgoals are proved**, the rule succeeds, and the original goal is proven.
Otherwise, the rule fails, and Pyke tries to find another rule, and so on.

PyKE Family ES
Family_relations

Forward Chaining Rules Example-Fact Base

- Facts are stored in a knowledge fact base called **family.kfb** in the form:
 - `son_of($son, $father)`
 - Semantics: first person is the son of the second person.



- We want to derive father-son relationships in the form :
 - `father_son($father, $son, $prefix)`
 - Where:
 - \$son: is the name of the son (e.g., david)
 - \$father: is the name of the father (e.g., bruce)
 - \$prefix: is a tuple of prefixes before the 'father' and 'son' titles to indicate the number of generations (e.g., () for direct father_son relationship, (grand), etc).

Forward Chaining Rules Example-Face Base

- Rules are stored in a knowledge rule base called **fc_example.krb** in the form:



```
1 direct_father_son
2     foreach
3         family1.son_of($son, $father)
4     assert
5         family1.father_son($father, $son, ())
6 grand_father_son
7     foreach
8         family1.father_son($father, $grand_son, ())
9         family1.father_son($grand_father, $father, ())
10    assert
11        family1.father_son($grand_father, $grand_son, (grand))
```

PyKE: Forward Chaining Rules Example

- **Step 1: Direct_father_son**
- First, we establish the direct father-son relationships (\$prefix is ())

```
1 son_of(david, bruce)
2 son_of(bruce, thomas)
3 son_of(thomas, frederik)
4 son_of(frederik, hiram)
```

```
1 direct_father_son
2 foreach
3     family1.son_of( david | bruce )
4     assert
5         family1.father_son($father, $son, ())
```

- When the rule fires matching line 3 to: son_of(david, bruce)
- It runs line 5 to assert: father_son(bruce, david, ())

Backward Chaining Rules Example- Rule Base

- Backward Chaining Rules are stores in a knowledge rule base

```
1 direct_father_son
2     use father_son($father, $son, ())
3     when
4         family2.son_of($son, $father)

5 grand_father_son
6     use father_son($grand_father, $grand_son, (grand))
7     when
8         father_son($father, $grand_son, ())
9         father_son($grand_father, $father, ())

10 great_grand_father_son
11    use father_son($gg_father, $gg_son, (great, $prefix1, *$rest_prefixes))
12    when
13        father_son($father, $gg_son, ())
14        father_son($gg_father, $father, ($prefix1, *$rest_prefixes))
```

```
15 brothers
16     use brothers($brother1, $brother2)
17     when
18         father_son($father, $brother1, ())
19         father_son($father, $brother2, ())
20         check $brother1 != $brother2

21 uncle_nephew
22     use uncle_nephew($uncle, $nephew, $prefix)
23     when
24         brothers($uncle, $father)
25         father_son($father, $nephew, $prefix1)
26         $prefix = ('great',) * len($prefix1)

27 cousins
28     use cousins($cousin1, $cousin2, $distance, $removed)
29     when
30         uncle_nephew($uncle, $cousin1, $prefix1)
31         father_son($uncle, $cousin2, $prefix2)
32         $distance = min(len($prefixes1), len($prefixes2)) + 1
33         $removed = abs(len($prefixes1) - len($prefixes2))
```

Backward Chaining Rules Example- Fact Base

- For this example, these are the starting set of facts stores in fact base- **family2.kfb**

```
1 son_of(tim, thomas)
2 son_of(fred, thomas)
3 son_of(bruce, thomas)
4 son_of(david, bruce)
```

- And we want to know who fred's nephew are. So we'd ask

uncle_nephew(fred, \$nepher,\$prefix)

Step1: find a rule whose use part matches the goal

- Our goal:
 - **uncle_nephew(fred, \$nephew, \$prefix)**
- The first rule found by the inference engine:

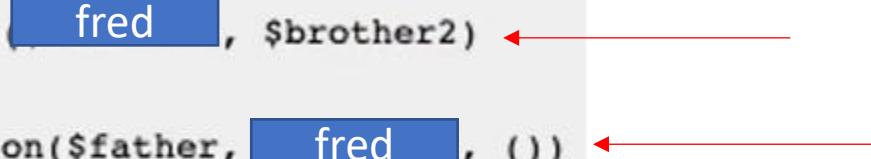
```
21 uncle_nephew
22     use uncle_nephew( fred , $nephew, $prefix) ←
23     when
24         brothers( fred  $father) ←
25         father_son($father, $nephew, $prefix1)
26         $prefix = ('great',) * len($prefix1)
```

- The first subgoal to be proved:
 - **brothers(fred, \$father)**

Step2: prove the 1st subgoal in the uncle_nephew rule

- The subgoal:
 - **brothers(fred, \$father)**
- The first rule found by the inference engine:

```
15 brothers
16     use brothers( fred , $brother2) ←
17     when
18         father_son($father, fred , ())
19         father_son($father, $brother2, ())
20         check $brother1 != $brother2
```



- The first subgoal to be proved:
 - **father_son(\$father, fred,())**

Step3: prove the 2nd subgoal in the brother's rule

- The subgoal:
 - **father_son(\$father,fred,())**
- The first rule found by the inference engine:

```
1 direct_father_son
2     use father_son( thomas , fred )()
3     when
4         family2.son_of fred $ thomas )
```

Success

- The second fact in the fact base matches:
 - **Son_of(fred, thomas)**



PyKE Weather ES
Simple_bc_all

A simple ES with ‘Questions’

- [IDAT 7215 - Expert System \(Weather\).ipynb](#)
- A very simple set of codes has been provided as a template for how to create your own backward chaining expert system.
- Specifically, this system decides what you should bring when walking out of the house (i.e. an umbrella, a raincoat, or nothing)
- This is in the Lab 04-Expert System folder “simple_bc_all “

Weather Example: First Without Questions

```
In [1]: #Run this cell
import sys
# The following command points your notebook to the location of a folder outside your working directory that you want to import.
#sys.path.append('C:/Users/Dr. Ali/Desktop/Lab 05/simple_bc_all')
sys.path.append('C:/Users/Dr. Ali//Lab 04-Expert System/simple_bc_all')

#sys.path.append('yourpath')
import driver_simple

driver_simple.bc_test()

doing proof
You should bring: raincoat

done
```

Weather Example: First Without Questions

Open driver_simple.py in folder simple_bc_all

```
import contextlib
import sys

from pyke import knowledge_engine
from pyke import krb_traceback

engine = knowledge_engine.engine(__file__)

def bc_test():

    engine.reset()      # Allows us to run tests multiple times.

    engine.activate('bc_simple_rules') #STUDENTS: you will need to edit the name of your rule file here

    print("doing proof")

    try:
        with engine.prove_goal('bc_simple_rules.what_to_bring($bring)') as gen: #STUDENTS: you will need to edit this line
            for vars, plan in gen:
                print("You should bring: %s" % (vars['bring'])) #STUDENTS: you will need to edit this line
    except Exception:
        # This converts stack frames of generated python functions back to the
        # .krb file.
        krb_traceback.print_exc()
        sys.exit(1)

    print()
    print("done")
    #engine.print_stats()
```

Weather Example: Fact & Ryke KB's

```
bc_simple_rules.krb x facts.kfb x
```

```
# Weather facts
raining(True)
windy(True)
```

```
# bc_simple_rules.krb
wear_rain_protection
use rain_protection(True)
when
facts.raining(True)

what_toBring_raincoat
use what_toBring(raincoat)
when
rain_protection($protection)
facts.windy(True)

what_toBring_umbrella
use what_toBring(umbrella)
when
rain_protection($protection)
facts.windy(False)

what_toBring_nothing
use what_toBring(nothing)
when
facts.raining(False)
facts.windy(False)
```

Weather Example: With Questions

```
[2]: driver_simple.bc_test_questions()
```

```
doing proof
```

```
Is it raining? (y/n) y
```

```
Is it windy? (y/n) y
```

```
You should bring: umbrella
```

```
Are any of the following disasters currently occurring?
```

1. Forest Fire
2. Tornado
3. Hurricane
4. Pandemic
5. None of the above

```
? [1-5] 1      ↵
```

```
You should bring: marshmellos
```

```
done
```

Weather Example: With Questions

```
def bc_test_questions():

    engine.reset()      # Allows us to run tests multiple times.

    engine.activate('bc_simple_rules_questions') #STUDENTS: you will need to edit the name of your rule file here

    print("doing proof")

    try:
        with engine.prove_goal('bc_simple_rules_questions.what_to_bring($bring)') as gen: #STUDENTS: you will need to edit this line
            for vars, plan in gen:
                print("You should bring: %s" % (vars['bring'])) #STUDENTS: you will need to edit this line

    except Exception:
        # This converts stack frames of generated python functions back to the
        # .krb file.
        krb_traceback.print_exc()
        sys.exit(1)

    print()
    print("done")
```

Weather Example: Questions and Rules

The diagram illustrates the mapping between the `questions.kqb` file (left) and the `bc_simple_rules.krb` file (right). Red arrows indicate the flow from the question definitions in `questions.kqb` to the corresponding rule definitions in `bc_simple_rules.krb`.

questions.kqb:

```
1 # questions.kqb
2
3 any_disasters($sans)
4     Are any of the following disasters currently occurring?
5     ---
6     $sans = select_1
7         1: Forest Fire
8         2: Tornado
9         3: Hurricane
10        4: Pandemic
11        5: None of the above
12        ! Thank goodness
13
14 is_raining($sans)
15     Is it raining?
16     ---
17     $sans = yn
18
19 is_windy($sans)
20     Is it windy?
21     ---
22     $sans = yn
23
```

bc_simple_rules.krb:

```
1 # bc_simple_rules.krb
2
3 no_rain
4     use what_toBring(no_rain_gear)
5     when
6         questions.is_raining(False)
7
8 what_toBring_raincoat
9     use what_toBring(raincoat)
10    when
11        questions.is_raining(True)
12        questions.is_windy(False)
13
14 what_toBring_umbrella
15     use what_toBring(umbrella)
16     when
17        questions.is_raining(True)
18        questions.is_windy(True)
19
20 what_toBring_marshmallows
21     use what_toBring(marshmallows)
22     when
23         questions.any_disasters($sans)
24         check $sans in (1,)
25
26 what_toBring_kite
27     use what_toBring(kite)
28     when
29         questions.any_disasters($sans)
30         check $sans in (2,3)
31
32 what_toBring_tissues
33     use what_toBring(tissues)
34     when
35         questions.any_disasters($sans)
36         check $sans in (4,)
```

IDAT7215

Computer Programming for Product Development and Applications

Lecture 5-1: Introduction to Artificial Intelligence

Dr. Zulfiqar Ali

Outline

- What is AI?
- History of AI
- Application of AI
- Search Algorithms for AI

What is intelligence

- The ability to **learn** or **understand** from experience.
- The ability to **acquire** and **retain** knowledge.
- The ability to **respond** quickly and successfully to a new situation.
- The ability to **use reason** to solve problems.

What is human intelligence

- It is a composition of abilities like



Learning



Reasoning



Perceiving



Understanding of
Language



Feeling

Ravi Kumar B N, Asst.Prof,CSE,BMSIT

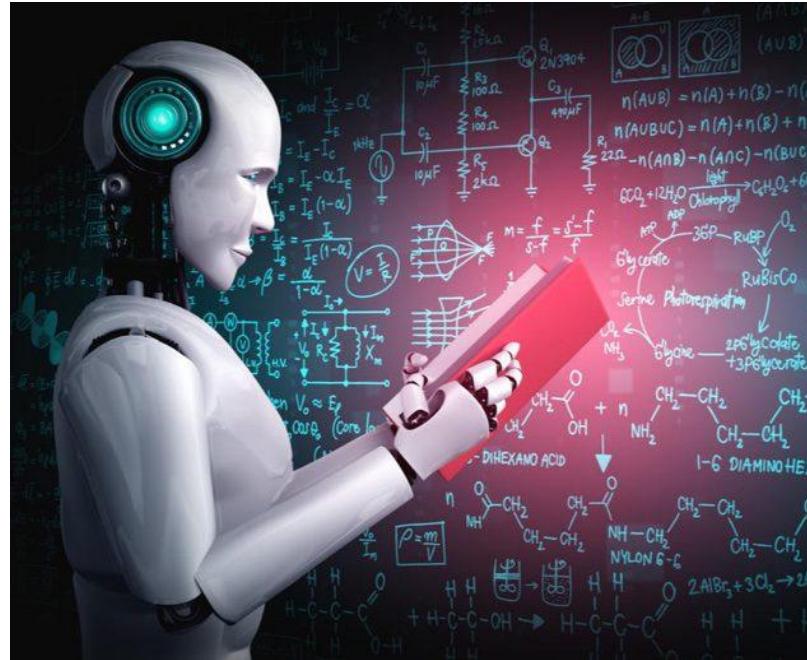
- If intelligence is **learning, understanding, retaining, responding, and using reason** than what is Artificial Intelligence (AI) ?

Quick Answer from Academia

- Modeling **human cognition** using computers.
- Study of making computer do things which at the moment humans are better at..
- Making computers do things **which require intelligence**.

What is Artificial Intelligence (AI)

- **Artificial Intelligence (AI):** The simulation of human intelligence by machines.
 - The ability to **solve problems**.
 - The ability to **act rationally**.
 - The ability to **act like humans**.
- Computers with the ability to mimic or duplicate the functions of human brain.
- The central principles of AI include:
 - Reasoning, knowledge, planning, learning, and communication.
 - Perception and the ability to move and manipulate objects.
 - It is the **science and engineering of making intelligent machines**. Especially intelligent computer.



What's Involved in Intelligence?

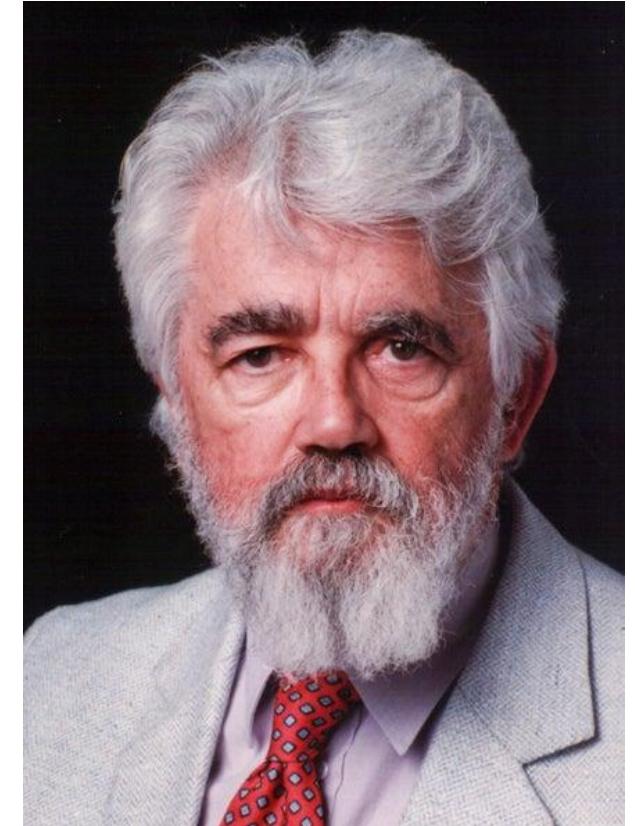
- Ability to interact with the real world
 - To perceive, understand, and act
 - ✓ E.g., speech recognition and understanding
- Searching the best solution
- Reasoning and Planning
 - Modelling the external world-delivery robot
 - Solving new problems, planning, and making decisions.
 - Ability to deal with unexpected problems, uncertainties.

- Learning and Adaption

- We are continuously learning and adapting.
- Our internal models are always being updated.
 - ✓ E.g., a baby learning to categorize and recognize animals.

John McCarthy

- (September 4, 1927 – October 24, 2011) was an American Computer Scientist and Cognitive Scientist.
- McCarthy was one of the **founders of the discipline of Artificial Intelligence**.
- He coined the term “ Artificial Intelligence (AI) ”



History of Artificial Intelligence

1950

Alan Turing develops the 'Turing test' which determines whether a machine can 'think' like a human.



1955

John McCarthy coined term 'Artificial Intelligence'.



1997

IBM's Deep Blue computer beats the reigning world chess champion, Gary Kasparov.



2011

Apple introduces Siri as part of its iPhone 4s.



2014

Amazon's Echo is released.



2018

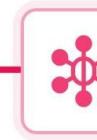
Erica, an advanced android designed.



The Cambridge Analytica scandal becomes public.



Microsoft's Tay is released (and quickly withdrawn).



2016

The Partnership on AI is formed.



Google Duplex: AI assistant makes a restaurant reservation.



2019

Machine Learning, Deep Learning production ready models.



2020

AI helps fight COVID-19.



2021

Open AI GPT-3 is used in 300+ apps

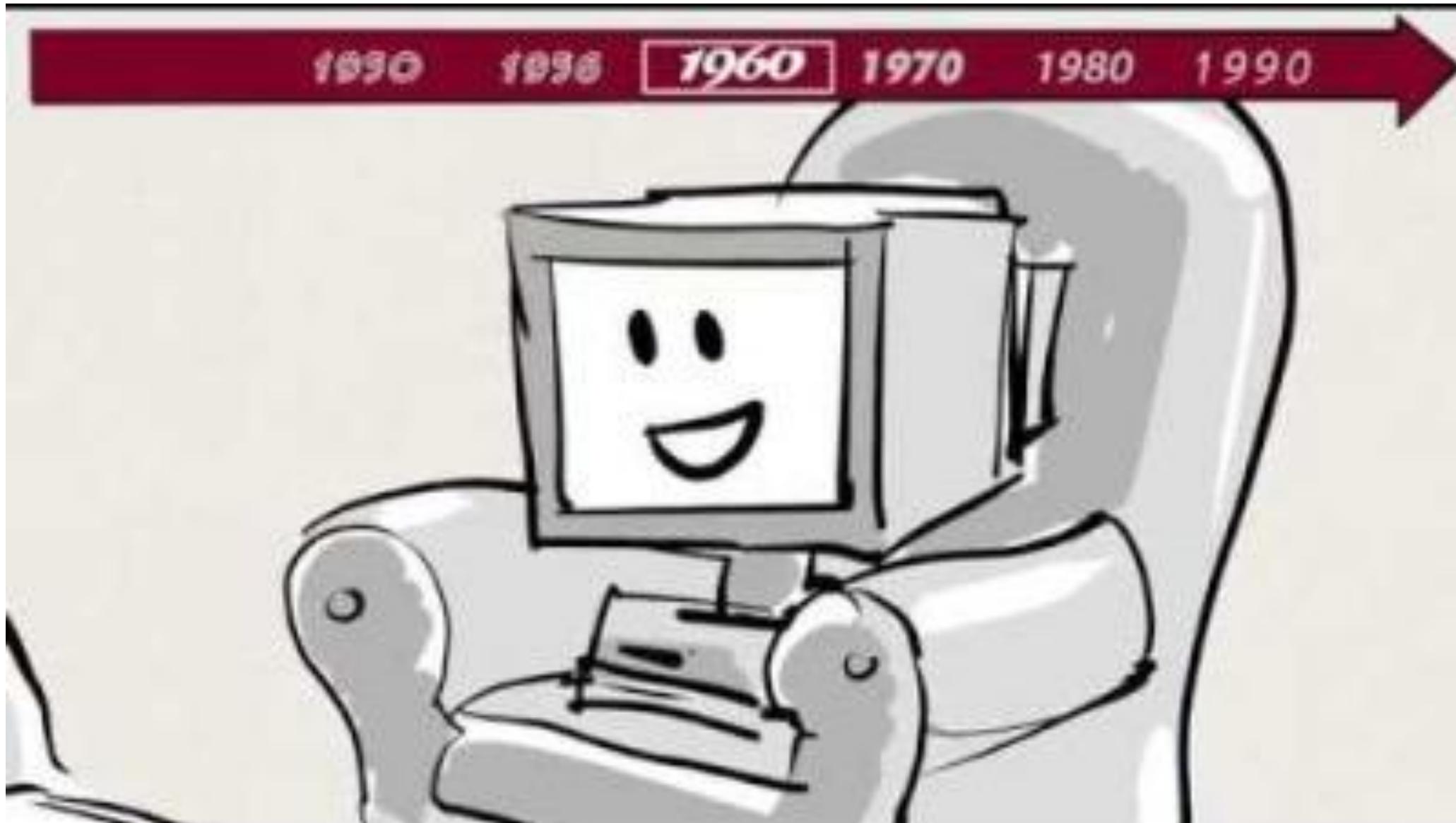


2022

No-code and low-code AI.



A Brief History of AI



Views of AI fall into four categories in Two dimensions:

■ Two dimensions

- Thinking/Reasoning vs Behavior/Action
- Success according to human standards vs Success according to ideal concept of intelligence (rationality):

■ Four Categories:

- Systems that think like humans (focus on reasoning and human framework).
- Systems that think rationally (focus on reasoning and general concept of intelligence).
- Systems that act like humans (focus on behavior and human framework).
- System that act rationally (focus on behavior and general concept of intelligence).

Definitions of artificial intelligence (AI)?

THINKING

BEHAVIOUR

1. Thinking humanly

The exciting new effort to **make computers think..**
Machine with minds ..(Hugeland, 1985)

"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning..." (Bellman, 1978)

3. Acting humanly

"The art of **creating machines that perform functions that require intelligence** when performed by people"
(Kurzweil, 1990)

"The study of how to make computers do things at which, at the moment, people are better"
(Rich and Knight, 1991)

2. Thinking rationally

"The study of computations that make it possible to perceive, reason, and act" (Wintson, 1992)

"The study of **mental faculties through the use of computational models**", (Charniak and McDermott, 1985)

4. Acting rationally

"A field of study that seeks to explain and emulate intelligent behavior in terms of computational process"
(Schalkoff, 1990)

"The branch of computer science that is concerned with the **automation of intelligent behaviour**" (Luger and Stubblefield, 1993)

HUMAN

RATIONAL

The Foundation of AI

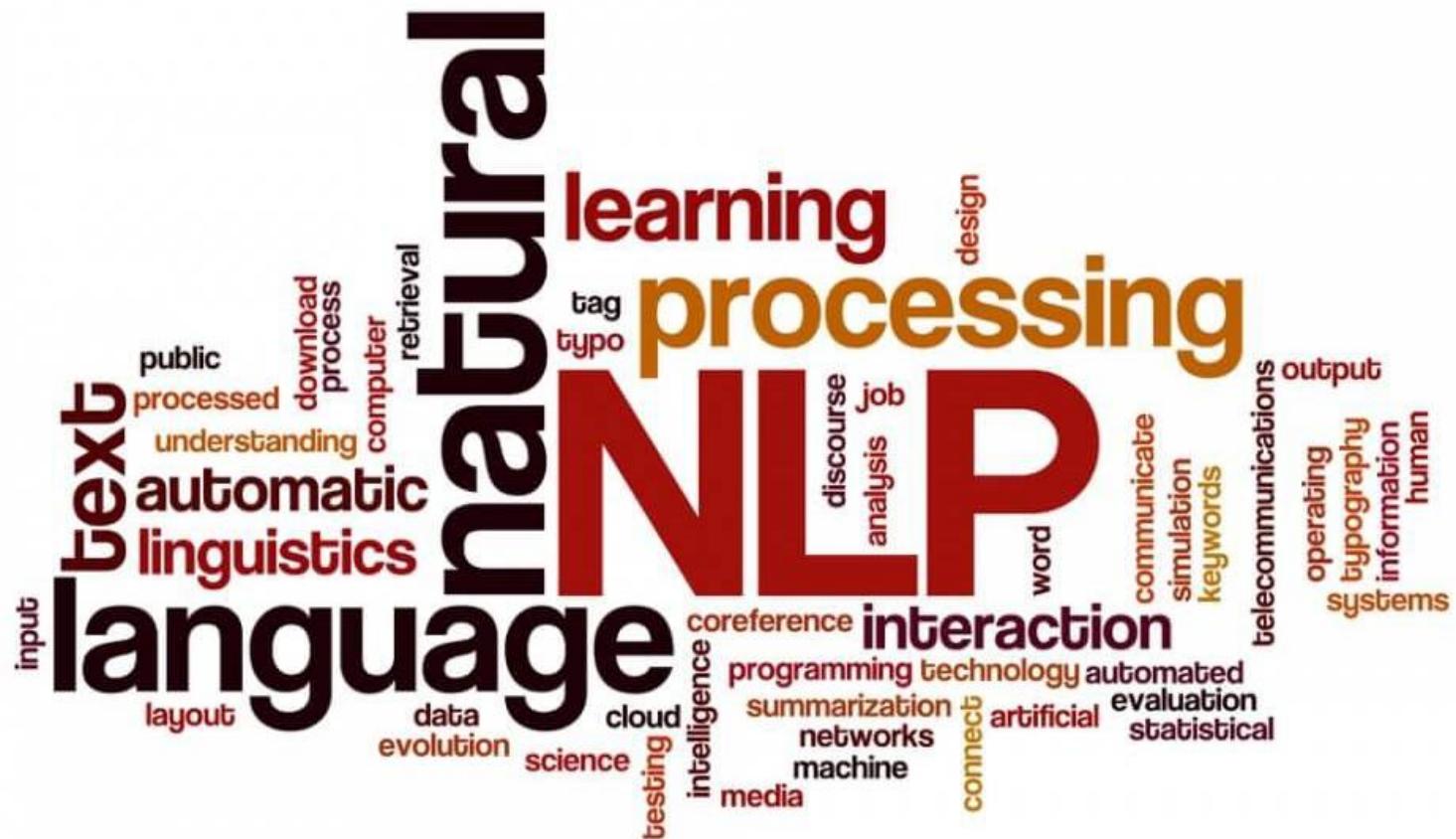
- **AI is a young field**
 - Inherit many ideas, viewpoints, and techniques from other disciplines:
 - ✓ **Philosophy**
 - Considering the idea that mind is in some ways like a machine
 - Logic, foundations of learning, language
 - ✓ **Mathematics**
 - Formal representation and proof
 - Algorithms
 - ✓ **Psychology**
 - Strengthen the idea that humans and other animals can be considered information processing machines.
 - ✓ **Linguistics**
 - ✓ **Computer engineering**
 - **Symbolic AI** is concerned with **describing and manipulating our knowledge** of the world as explicit symbols, where these symbols have clear relationships to entities in the real world.
 - **Sub-symbolic AI** (e.g. neural-nets) is more concerned with **obtaining the correct response to an input stimulus** without ‘looking inside the box to see if parts of the mechanism can be associated with discrete real-world objects.

Application of AI

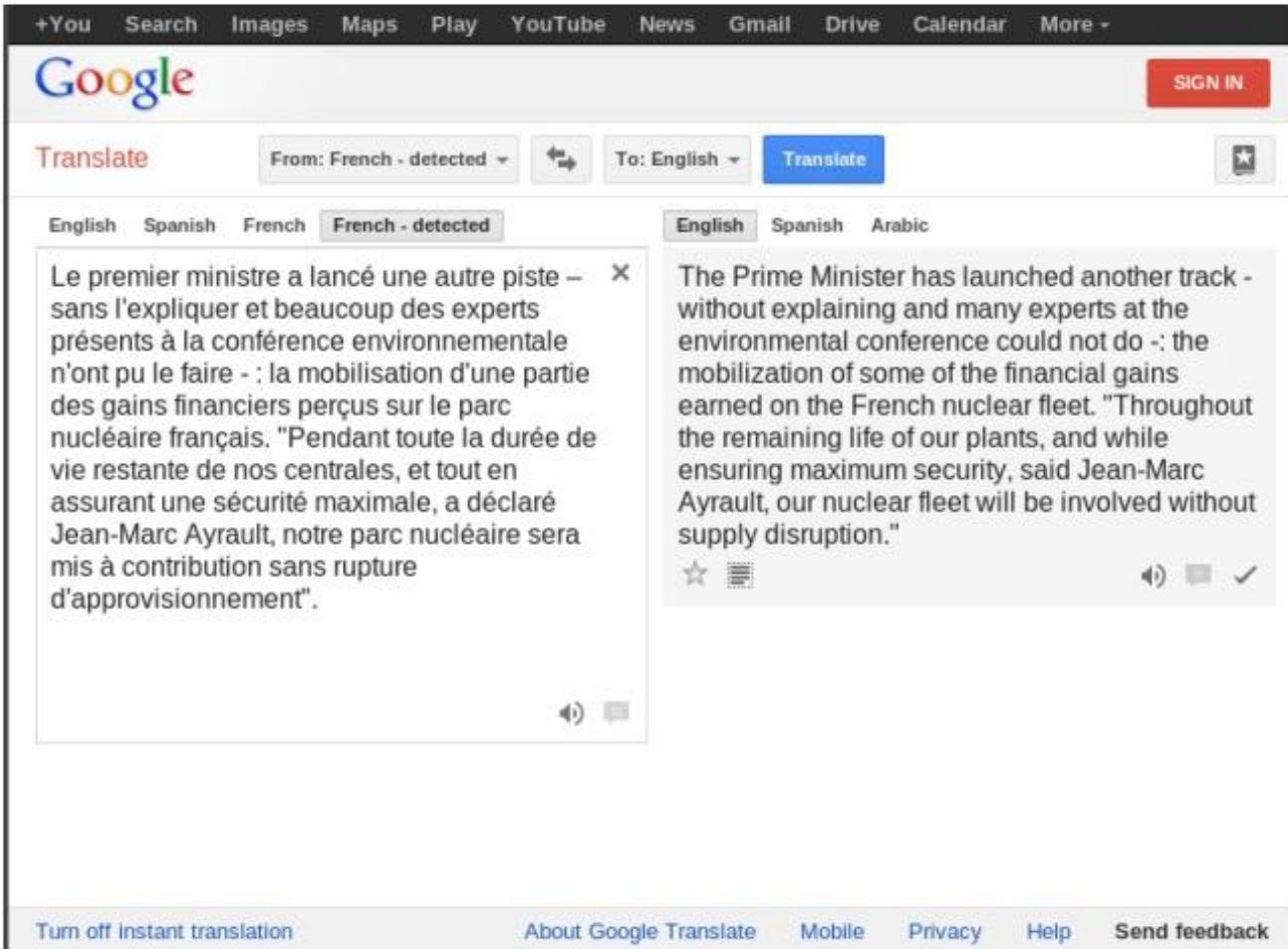
- Natural Language Understanding
- Expert System
- Planning and Robotics
- Machine Learning
- Game Playing

Natural Language Processing

- To design and build software that will analyze understand and generate languages that humans use naturally.



Machine Translation



The screenshot shows the Google Translate interface. At the top, there's a navigation bar with links for You, Search, Images, Maps, Play, YouTube, News, Gmail, Drive, Calendar, and More. Below the navigation bar is the Google logo and a Sign In button. The main area is titled "Translate" and shows a "From: French - detected" dropdown, a "To: English" dropdown, and a "Translate" button. There are also tabs for English, Spanish, French, and French - detected. The text being translated is: "Le premier ministre a lancé une autre piste – sans l'expliquer et beaucoup des experts présents à la conférence environnementale n'ont pu le faire : la mobilisation d'une partie des gains financiers perçus sur le parc nucléaire français. "Pendant toute la durée de vie restante de nos centrales, et tout en assurant une sécurité maximale, a déclaré Jean-Marc Ayrault, notre parc nucléaire sera mis à contribution sans rupture d'approvisionnement". The translated text is: "The Prime Minister has launched another track - without explaining and many experts at the environmental conference could not do : the mobilization of some of the financial gains earned on the French nuclear fleet. "Throughout the remaining life of our plants, and while ensuring maximum security, said Jean-Marc Ayrault, our nuclear fleet will be involved without supply disruption." At the bottom of the interface, there are links for Turn off instant translation, About Google Translate, Mobile, Privacy, Help, and Send feedback.

- Machine translation research started in the 1960s (the US government was quite keen on translating Russian into English). Over the subsequent decades, it went through quite a few rough turns.
- In the 1990s and 2000s, statistical machine translation, aided by large amounts of example human translations, helped vastly improve translation quality.
- As of 2015, Google Translate supports 90 languages and serves over 200 million people daily. The translations are nowhere near perfect, but they are very useful.
- In 2016, Google released a neural machine translation system which led to significant improvements in accuracy due to advances in deep learning, and as of July 2022, 133 languages are supported.

Reading comprehension

- Natural language understanding generally is still widely regarded as an unsolved problem. One of the specific examples is the **task of reading comprehension**: given a passage, the goal is to answer a question about the passage (think standardized tests).

In meteorology, precipitations is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called “showers”.

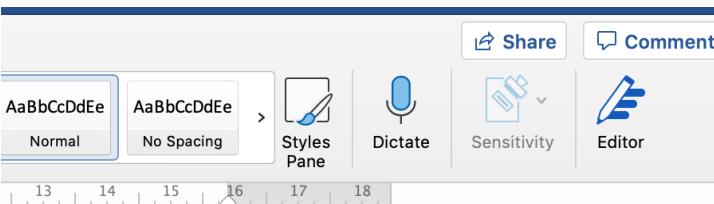
What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

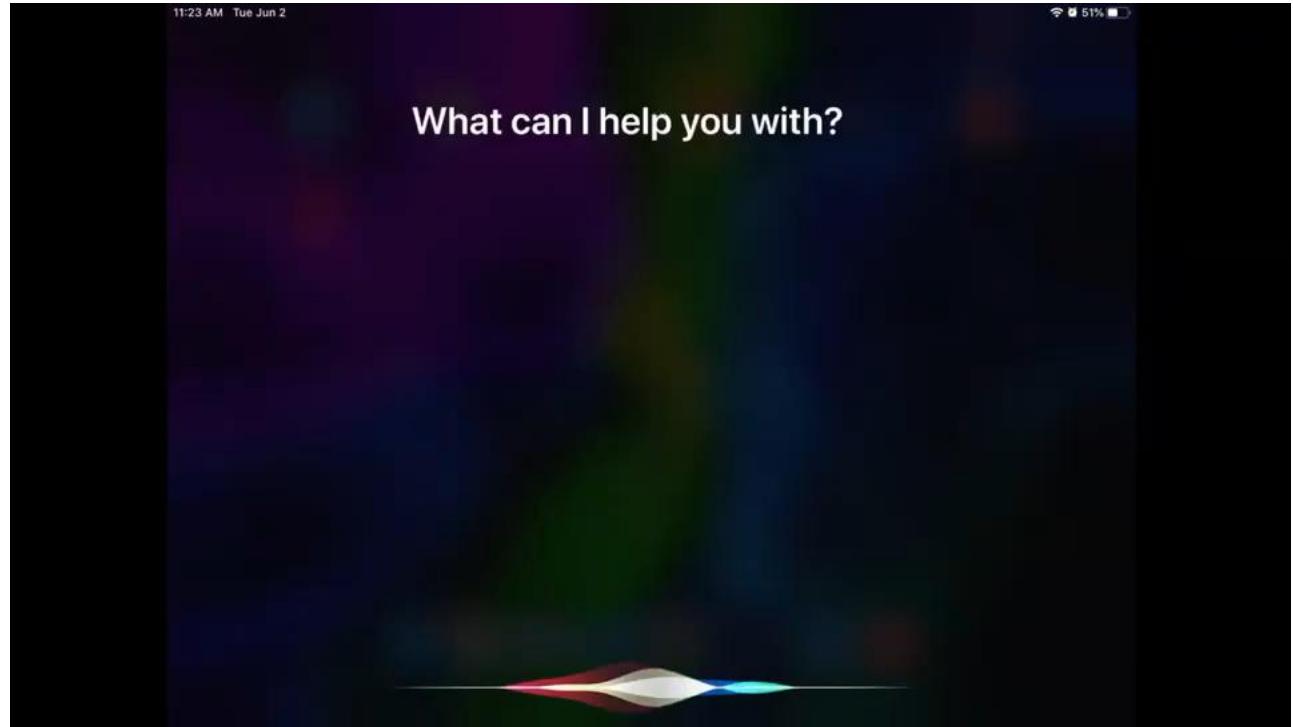
Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Speech recognition

- Speech recognition is the **problem of transcribing audio into words.**
- It has a long history dating back to the 1970s.
- But it wasn't until around **2009** that speech recognition started to work well due to the adoption of **deep neural networks.**



However, speech recognition is only one part of the story; the other is **understanding the text, which is a much harder problem.**



<https://www.youtube.com/watch?v=PfHo6Ad0RYg>

Autonomous driving

- Perhaps the first significant event was the **2005 DARPA Grand Challenge**, in which the goal was to have a **driverless car go through a 132-mile off-road course**. **Stanford finished in first place**. The car was equipped with various sensors (laser, vision, radar), whose readings needed to be synthesized to localize the car and then generate control signals for the steering, throttle, and brake.
- In 2007, DARPA created an even harder Urban Challenge, which was won by CMU.
- In **2009, Google started a self-driving car program**, and since then, their self-driving cars have driven over 1 million miles on freeways and streets.
- In January 2015, Uber hired about 50 people from CMU's robotics department to build self-driving cars.

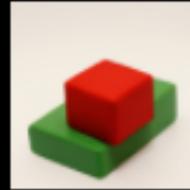


<https://www.youtube.com/watch?v=tIThdr3O5Qo>

Image generation

- One particular hot topic in computer vision right now **is generating photorealistic images from text.**

a small red block sitting on a large green block



a stack of 3 cubes. a red cube is on the top, sitting on a green cube. the green cube is in the middle, sitting on a blue cube. the blue cube is on the bottom.



an emoji of a baby penguin wearing a blue hat, red gloves, green shirt, and yellow pants



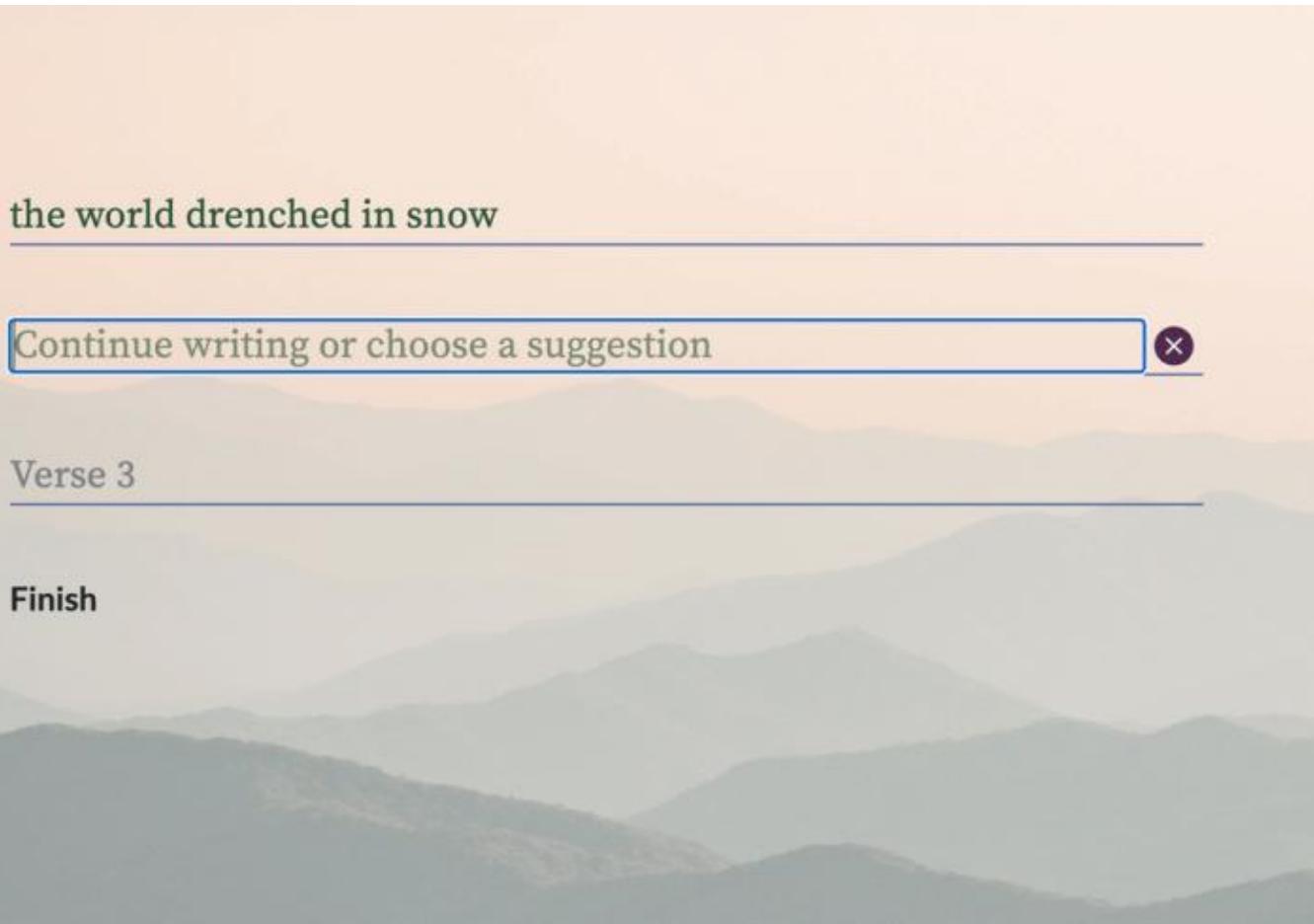
Poetry writing

the world drenched in snow

Continue writing or choose a suggestion x

Verse 3

Finish



Your selected muses

Emily Dickinson

Poem structure

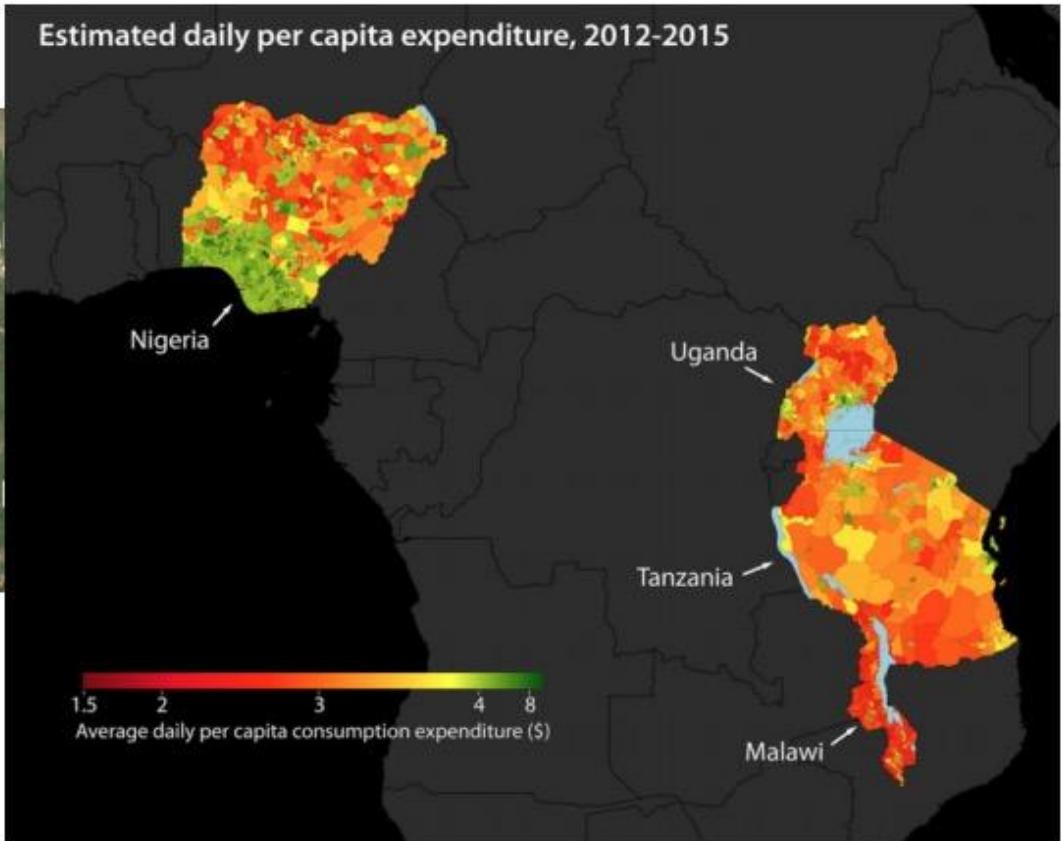
Free Verse

Here's what they suggest Refresh

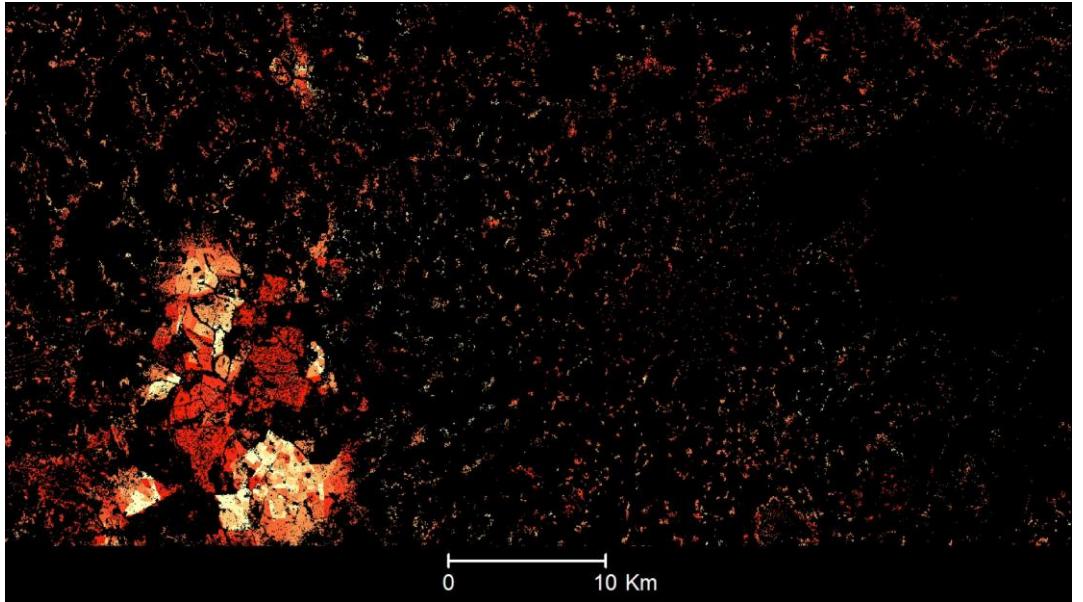
The seasons fit not any way
Spring is only occurred!
White as the border of the sky
And the snows go hurrying in?
White as the border star

<https://sites.research.google/versebyverse/>

Tackle social problems



Predicting poverty



A high-resolution population density map layer for Pretoria, South Africa.

Artificial Intelligence for IT Operations

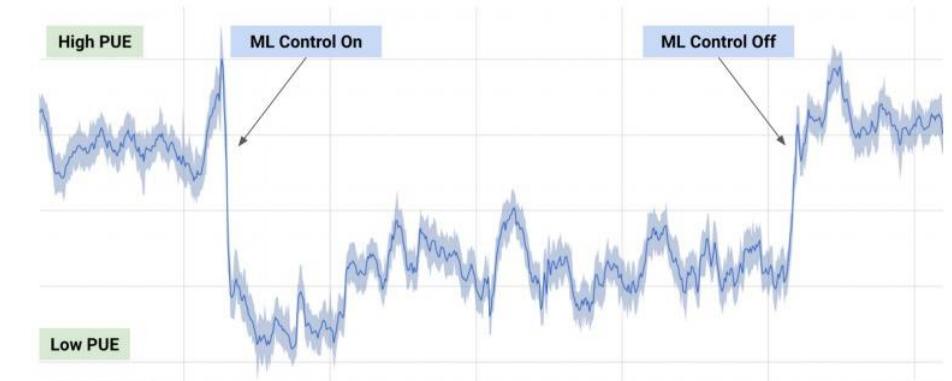
- Saving energy by cooling data centers



 Applied

DeepMind AI Reduces Google Data Centre Cooling Bill by 40%

July 20, 2016



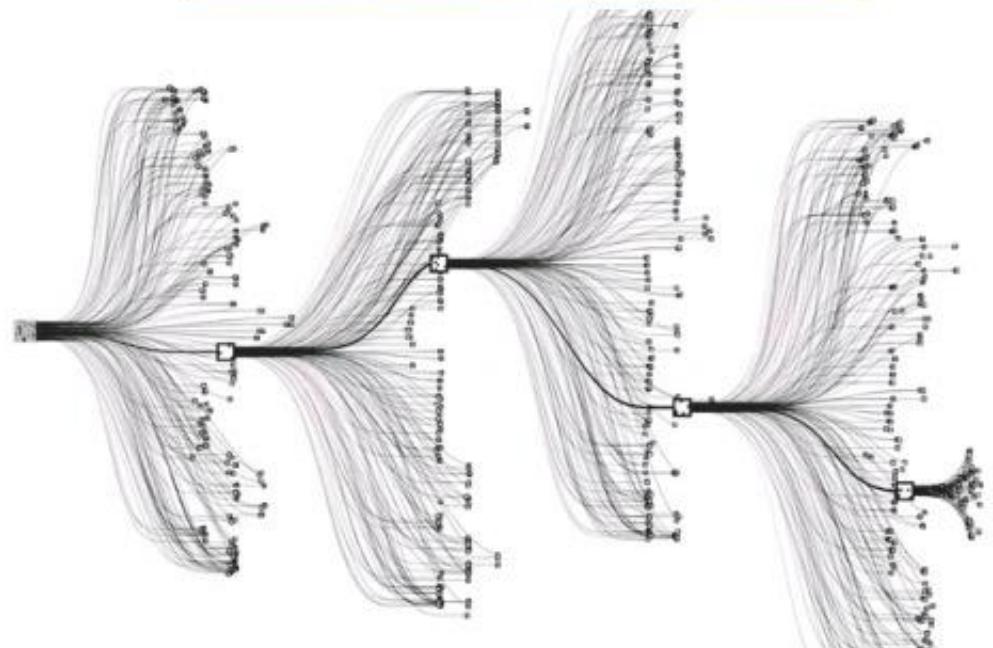
Humans versus machines



1997: Deep Blue (chess)



2011: IBM Watson (Jeopardy!)



- In 1997, Deep Blue defeated the world chess champion.
- In 2011, IBM Watson defeated two of the biggest winners at the quiz show Jeopardy
- In 2016, Google DeepMind created a program called AlphaGo, which used **deep neural networks** and **reinforcement learning**, to defeat a **world champion**, surprising not only the master Go player but many AI researchers as well.

Some failures...

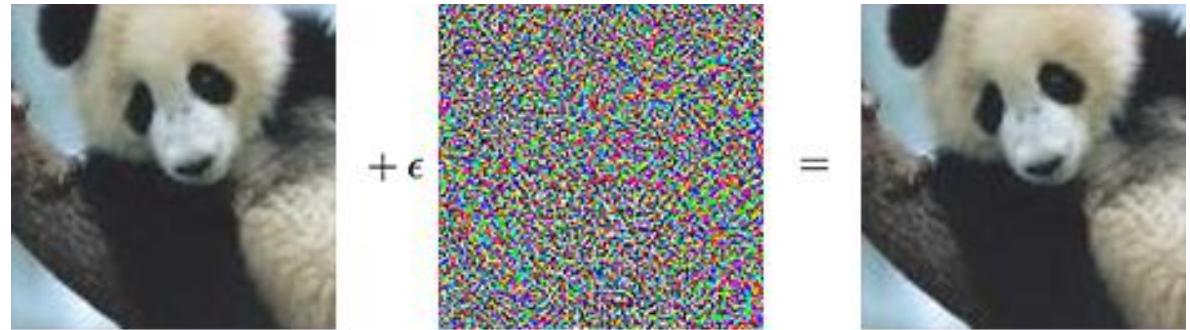
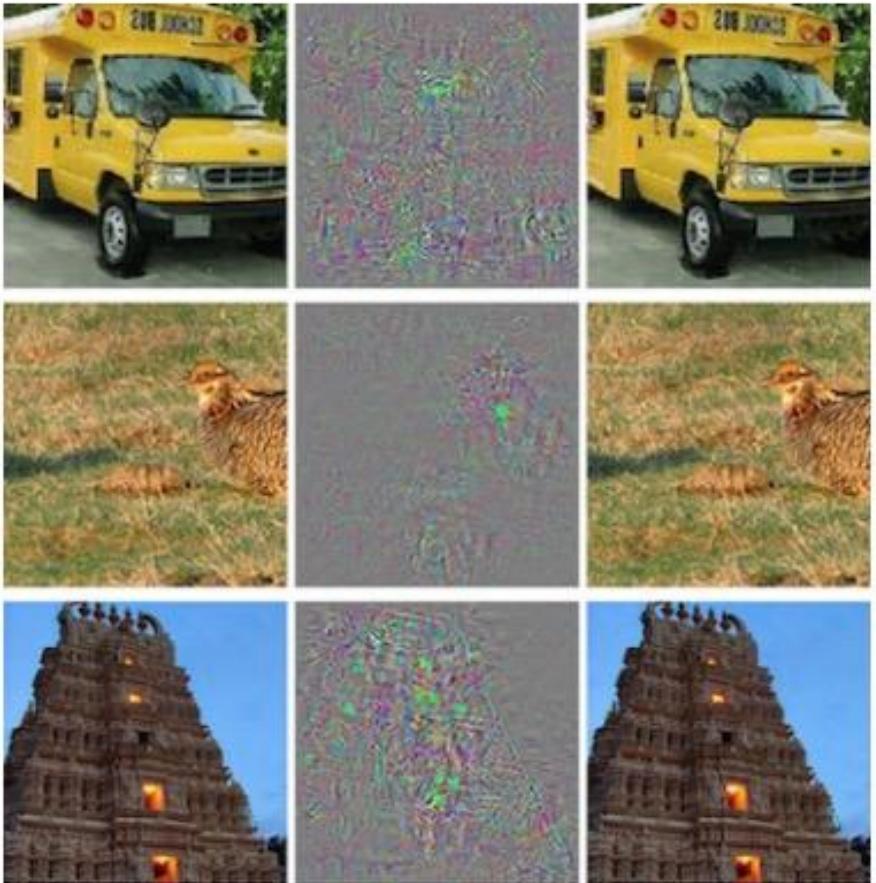
Open-domain dialogue

- A: How old are you?
- B: I'm 16. Why are you asking?
- A: I thought you were 12.
- B: What made you think so?
- A: I don't know what you are talking about.
- B: You don't know what you are saying.



Adversarial examples

AlexNet predicts correctly on the left



"panda"

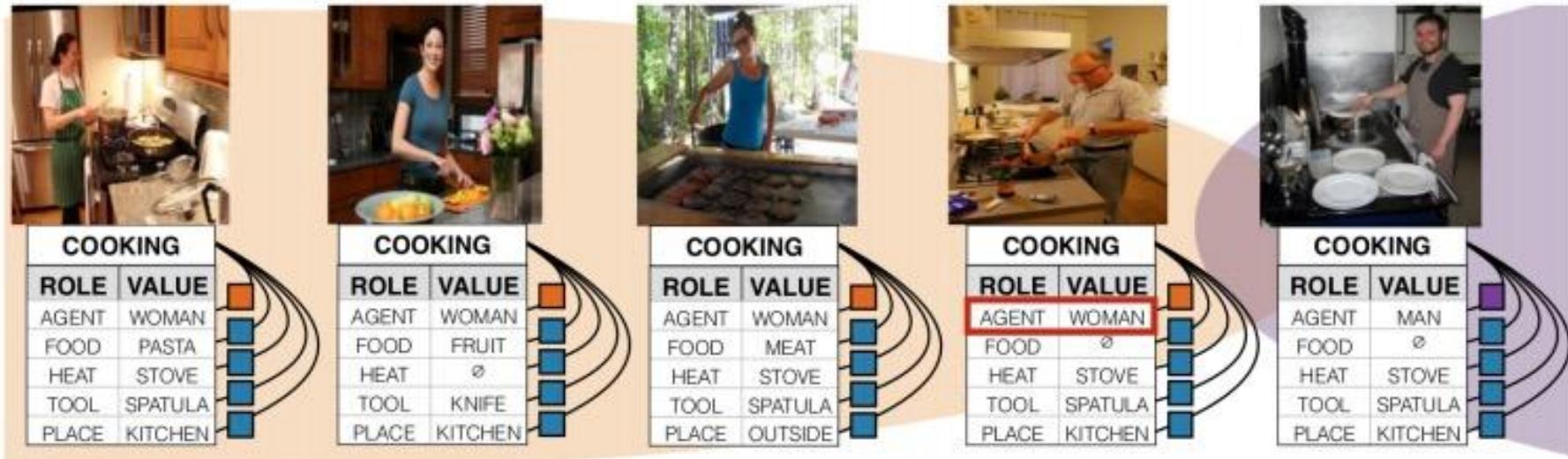
57.7% confidence

"gibbon"

99.3% confidence

AlexNet predicts **ostrich** on the right

Bias



tote treats subject heavy commit game
 browsing sites seconds slow arrival tactical
 crafts identity drop reel firepower
 trimester tanning user parts hoped command
 ultrasound busy caused ill rd scrimmage
 modeling housing housing sought builder drafted
 sewing beautiful cake victims looks brilliant genius
 dress dance letters nuclear yard cocky journeyman
 modeling pageant earrings divorce ii firms seeking ties guru
 sewing dress dance lettering nuclear yard cocky journeyman
 modeling pageant earrings divorce ii firms seeking ties guru
 salon dancers lust lobby voters
 sassy breasts pearls vases frost vi governor sharply rule
 homemakers roses folks friend pal brass buddies burly
 feminist babe dancer wife daddy chap lad
 actresses gals fiance girlfriends girlfriend wife
 queen grandmother fiancee
 sisters daughters

Privacy

In 2014, Facebook Research published a paper describing their **DeepFace face identification system**. DeepFace is a 120 million parameter deep neural network and obtains 97.35% on the standard Labeled Faces in the Wild (LFW) dataset, which is comparable to human performance.

!!! it would enable some entities to take arbitrary images and videos of crowds and identify every single person identity, which would effectively **eliminate the ability to stay anonymous**.



<https://www.youtube.com/watch?v=umxGfNQmDbs>

Privacy



Royal Free breached UK data law in 1.6m patient deal with Google's DeepMind

Information Commissioner's Office rules record transfer from London hospital to AI company failed to comply with Data Protection Act



▲ 'We underestimated the complexity of the NHS and of the rules around patient data' - DeepMind. Photograph: Alamy Stock Photo

London's Royal Free hospital failed to comply with the Data Protection Act when it handed over personal data of 1.6 million patients to [DeepMind](#), a Google subsidiary, according to the Information Commissioner's Office.

In the spotlight...

Companies

 "An important shift from a mobile first world to an AI first world" [CEO Sundar Pichai @ Google I/O 2017]

 Created AI and Research group as 4th engineering division, now 8K people [2016]

 Created Facebook AI Research, Mark Zuckerberg very optimistic and invested

Others: IBM, Amazon, Apple, Uber, Salesforce, Baidu, Tencent, etc.

Governments

 "AI holds the potential to be a major driver of economic growth and social progress" [White House report, 2016]

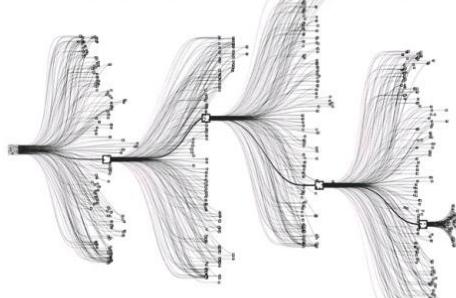
 Released domestic strategic plan to become world leader in AI by 2030 [2017]

 "Whoever becomes the leader in this sphere [AI] will become the ruler of the world" [Putin, 2017]

Characteristics of AI tasks

- High societal impact (affect billions of people)
- Diverse (language, games, robotics)
- Complex (really hard)

Two sources of complexity...



Computational complexity: exponential explosion

e.g., in the game of Go, there are up to 361 legal moves per turn, and let us say that the average game is about 200 turns. Then, as a crude calculation, there might be 361,200 game trajectories that a player would have to consider to play optimally.

Computation (time/memory)



Information (data)

这是什么意思?



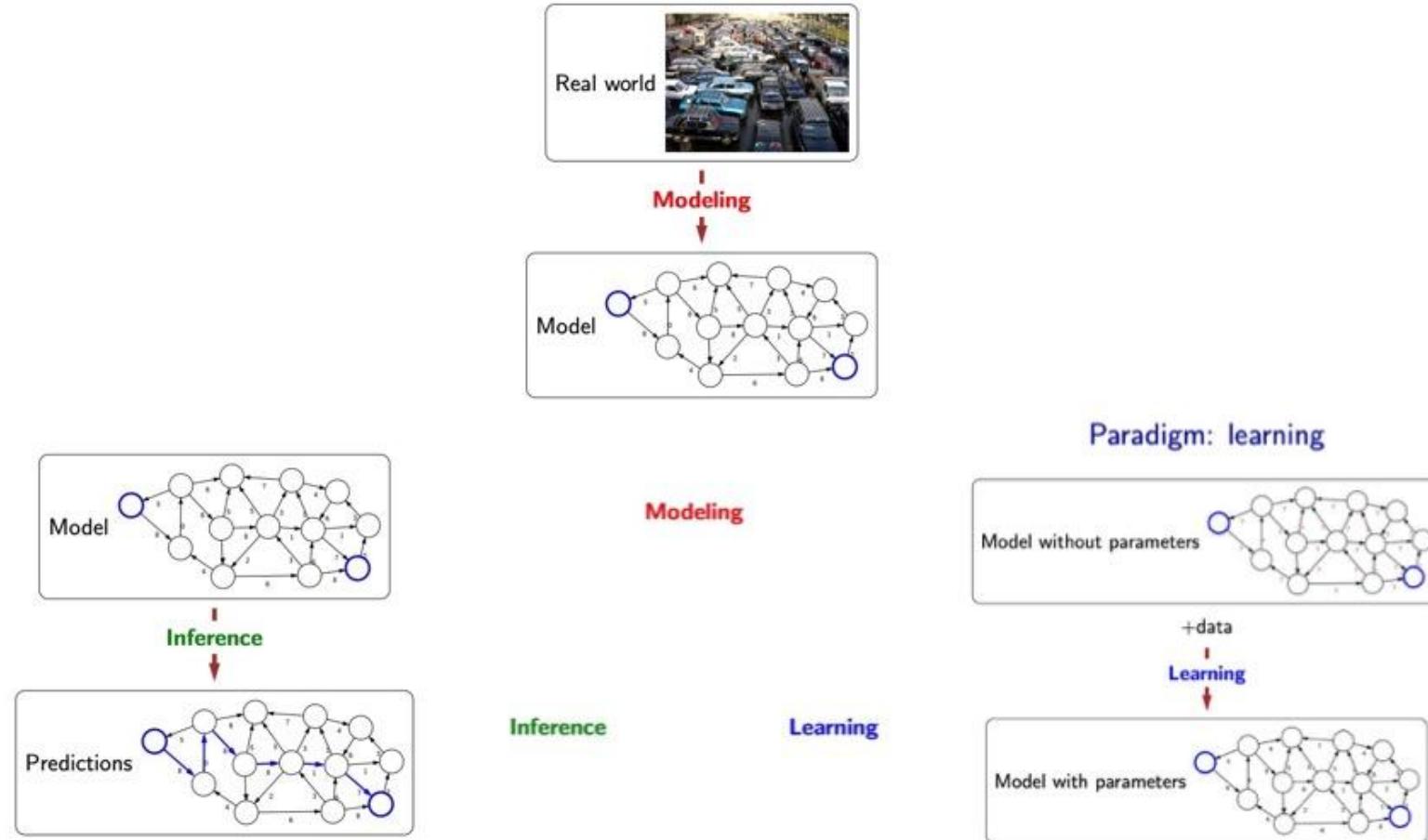
Even infinite computation isn't enough...need to somehow *know* stuff.

Information complexity: need to acquire knowledge

e.g., we are asked to classify an image with the type of bird. We simply need the information or knowledge about birds.



How do we solve tackle these challenging AI tasks?



Searching Algorithms for AI

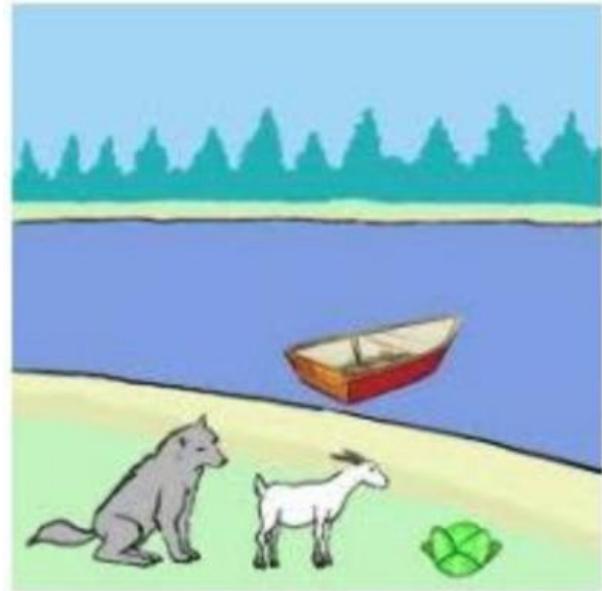
Question

- A farmer wants to get his cabbage, goat, and wolf across a river. He has a boat that only holds two. He cannot leave the cabbage and goat alone or the goat and wolf alone. How many river crossings does he need?

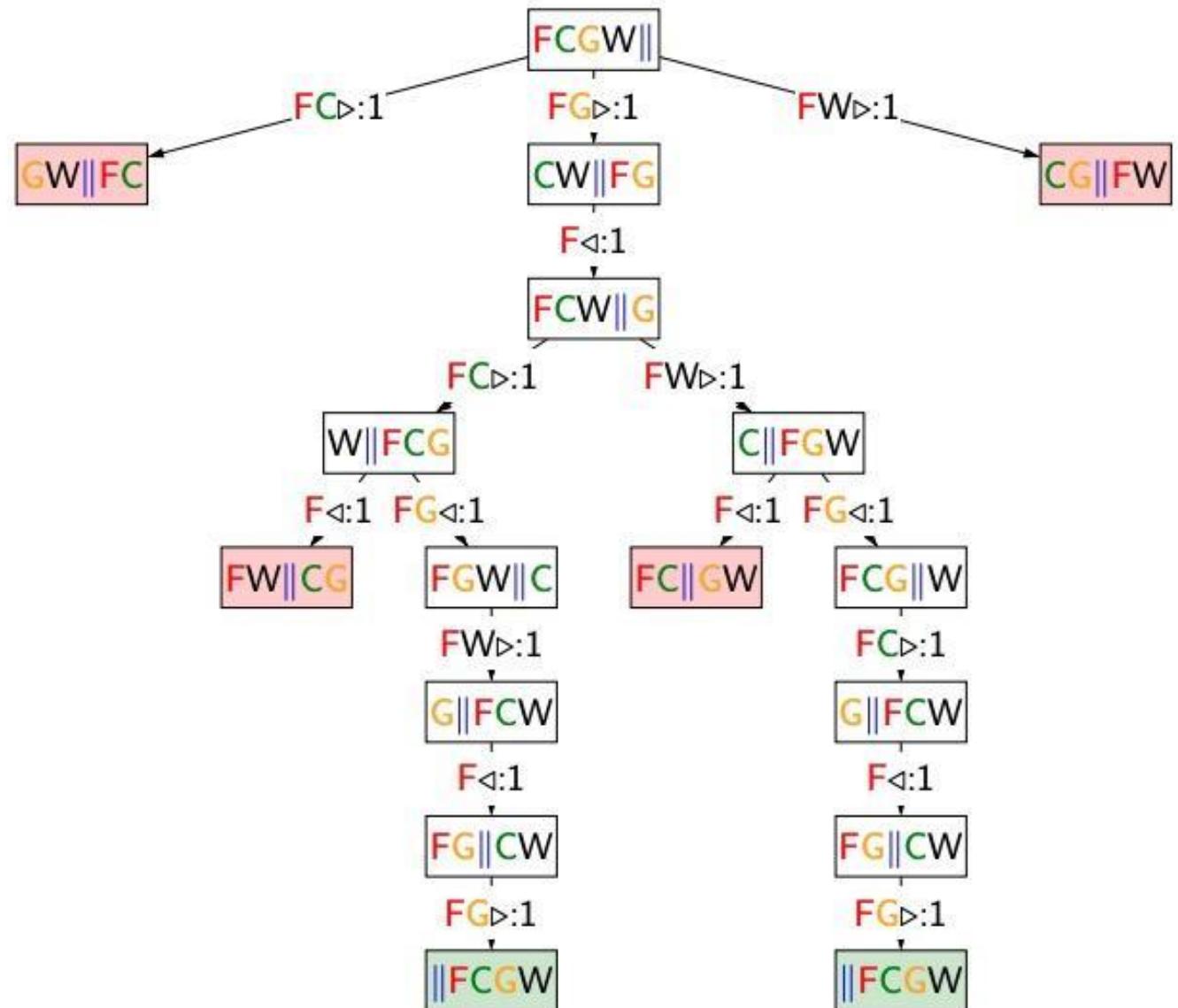
4
5
6
7
no solution



Search Tree

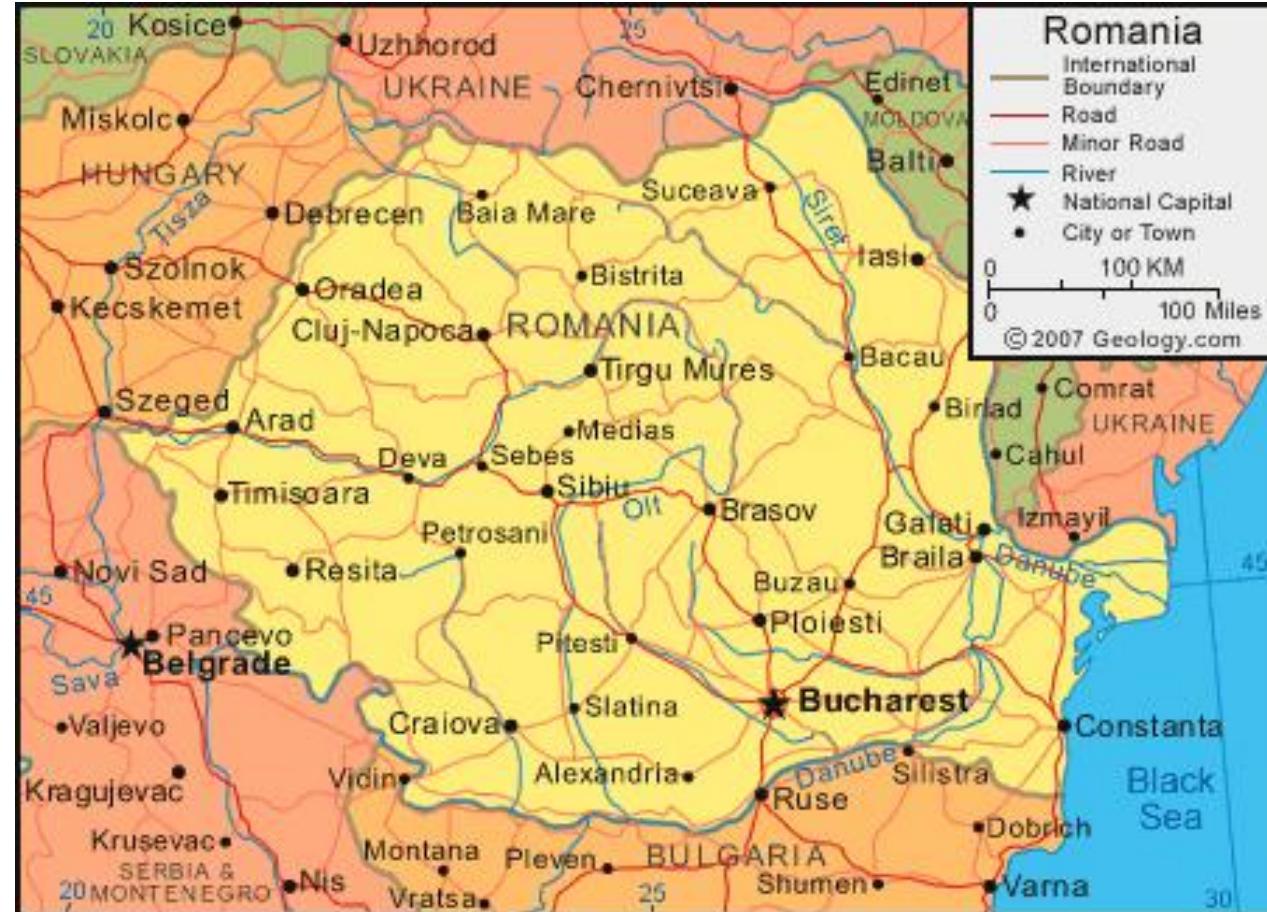


Farmer Cabbage Goat Wolf



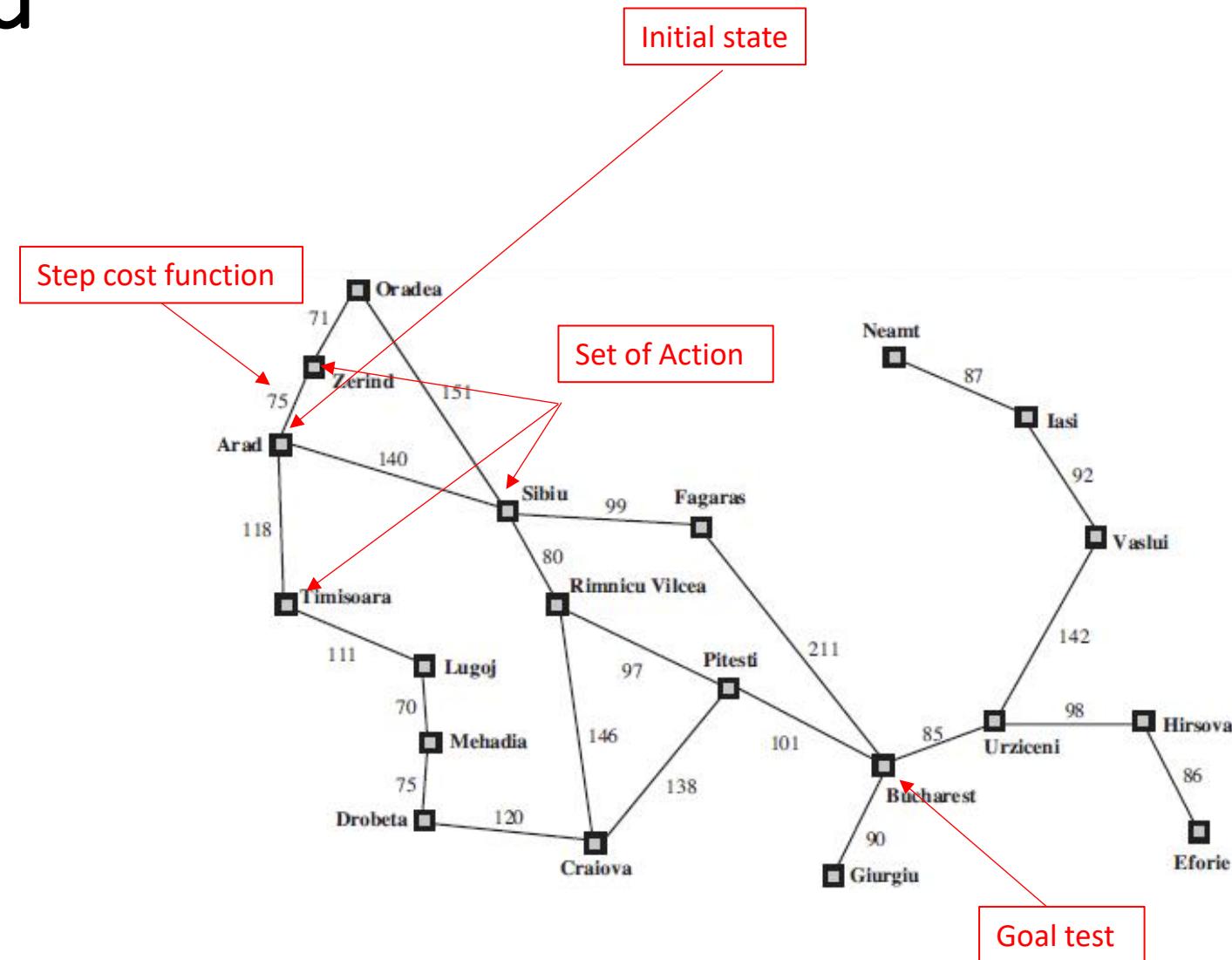
Example: how to get from Arad to Bucharest?

- **Goal formulation:**
achieved if location is Bucharest
- **Problem formulation:**
various roads of various lengths lead to cities intermediate to Bucharest
- **Search:**
find a sequence of intermediate cities with “optimal distance” to Bucharest



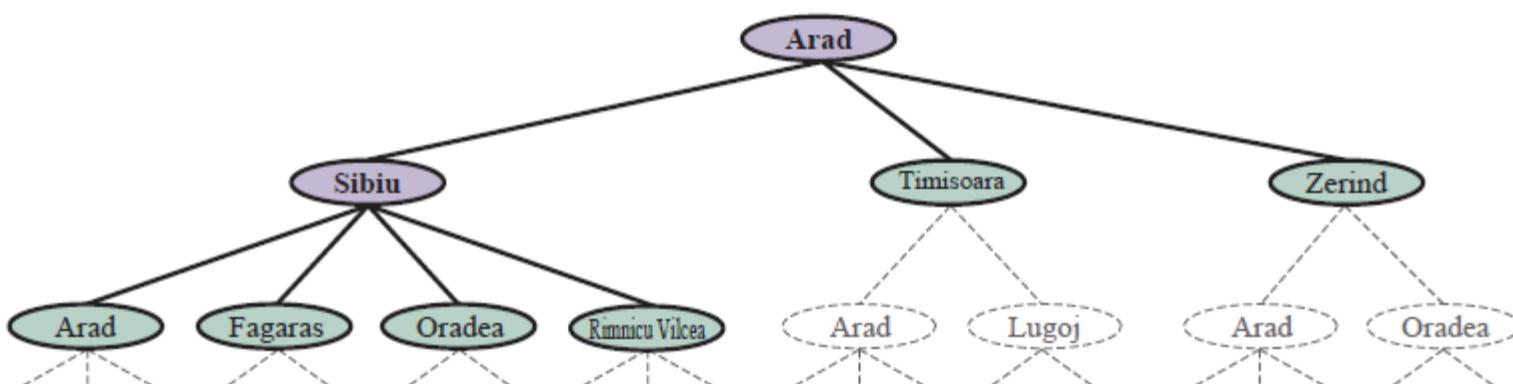
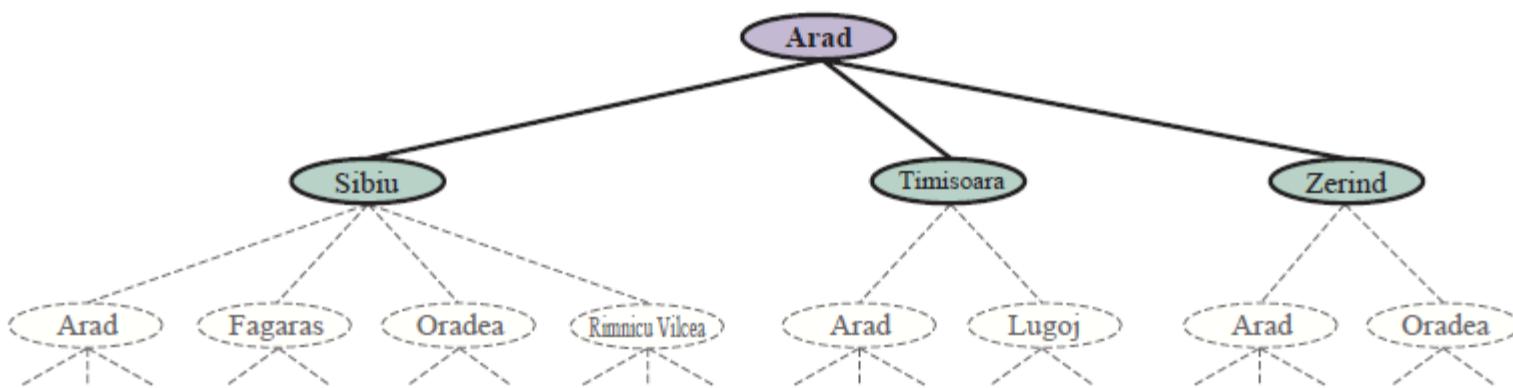
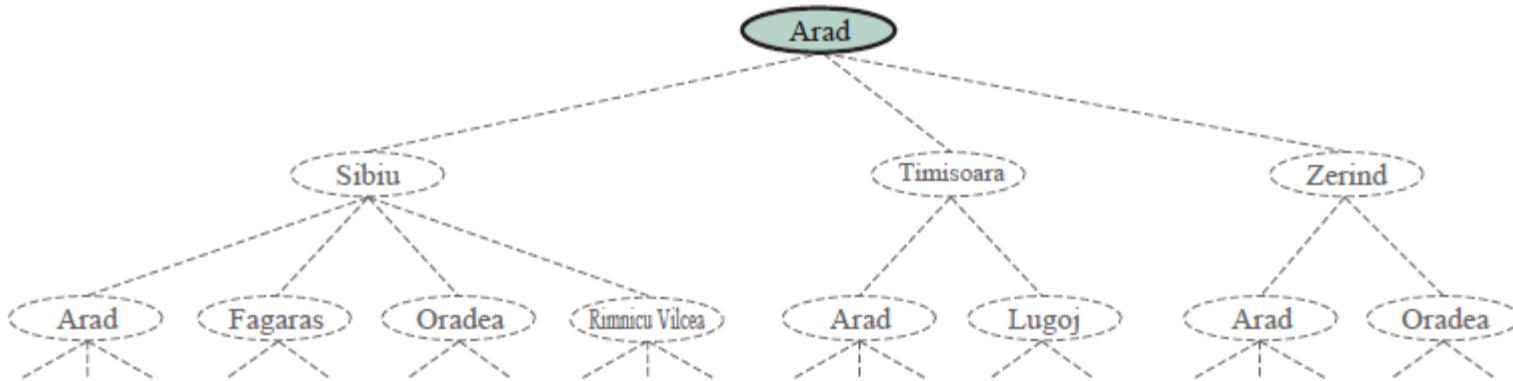
The Problem Abstracted

- **State:** A state description specifies the location of each city
- **An initial state (s)**
 - e.g., In(Arad)
- **A set of actions (a):** given a state s, ACTION(s) is the set of actions applicable in s
 - e.g., ACTION(In(Arad))
 - = {Go(Zerind), Go(Sibiu), Go(Timișoara) }
- **A transition function:** given a state s and action a, the successor RESULT(s, a) is the result of taking a in s
 - e.g., RESULT(In(Arad), Go(Zerind)) = In(Zerind)
- **The goal test** determines what states achieve the goal. Often, there is an explicit set of goal states,
 - e.g., {In(Bucharest)}
- **A step cost function** that assigns a numeric cost (normally non-negative) to each path
 - e.g., $c(\text{In(Arad)}, \text{go}(Zerind), \text{In}(Zerind)) = 75$



The Problem Abstracted

- The **state space** of the problem is the set of possible states of the problem.
 - It is implicitly defined by: the initial state, the action set, and the transition function.
 - The state space forms a directed *graph* in which the nodes are states and the links between nodes are actions.
 - A **path** is a sequence of states connected by a sequence of actions. It has a **path cost** calculated by adding step costs.
 - A **solution** is a path from initial to goal state. It is **optimal** if it has minimal cost.



Example: Slide 8-puzzle

3	1	2
4		5
6	7	8

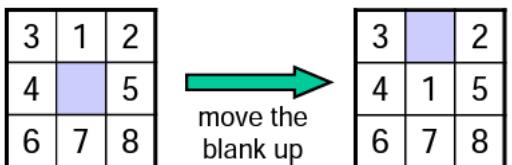
Initial state

	1	2
3	4	5
6	7	8

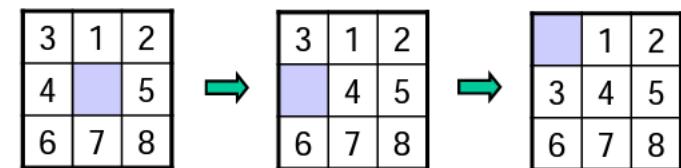
Goal state

- **State:** all configurations of numbers 1-8, plus blank tile.
- **Initial state:** any state can be initial.
- **Actions:** {Left, Right, Up, Down}
- **Goal test:** state matches goal configuration
- **Step cost:** each step costs 1 (i.e., path cost =#actions in the path)
- **Transition function:**

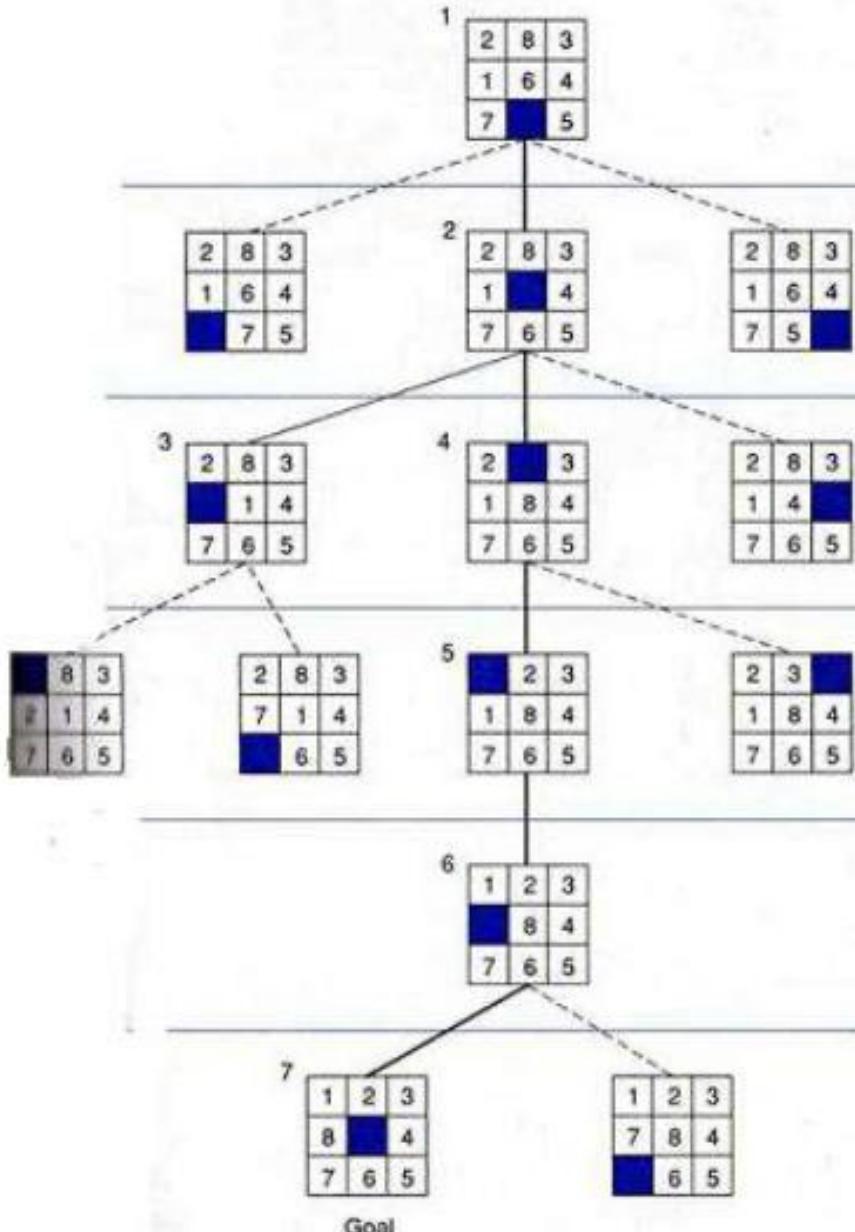
e.g.,



How to search for solutions?



Example: Slide 8-puzzle



Initial State → Goal State

2	8	3
1	6	4
7		5

→

1	2	3
8		4
7	6	5

Initial State

Goal State

How to search for solutions?



- Two Search Types:
 - **Uninformed search (blind search)**
 - ✓ No information about the number of steps or the path cost from the current state to the goal.
 - ✓ All they can do is distinguish a goal state from a non-goal state.
 - **Informed search (heuristic search)**
 - ✓ Use some information about a goal state.

Search Strategies Performance Measure

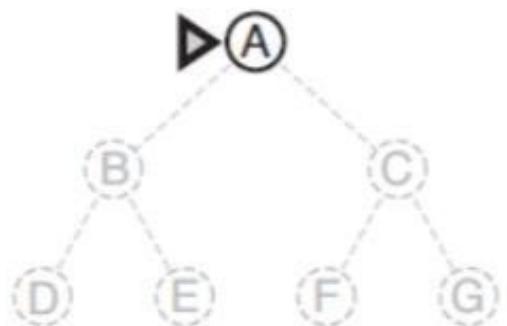
- A search strategy is defined by picking the order of node expansion.
- Strategies are evaluated along the following dimensions:
 - **Completeness:** does it always find a solution if one exists?
 - **Time complexity:** number of nodes generated.
 - **Space complexity:** maximum number of nodes in memory.
 - **Optimality:** does it always find a least-cost solution?
 - **Systematicity:** does it visit each state at most once?
- Time and space complexity are measured in terms of
 - b: maximum branching factor of the search tree.
 - d: depth of the least-cost solution.
 - m: maximum depth of the state space (maybe ∞)



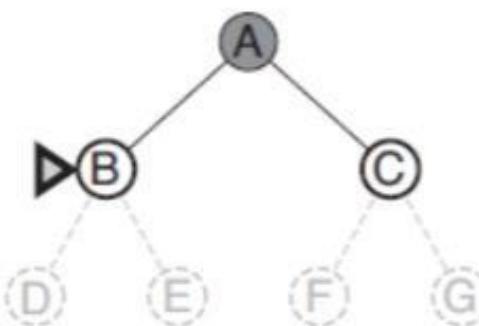
Uninformed Search

1. Breadth-first Search (BFS)

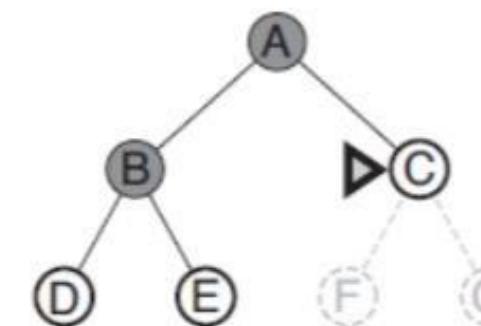
- All the nodes at depth d in the search tree are expanded before the nodes at depth $d+1$.



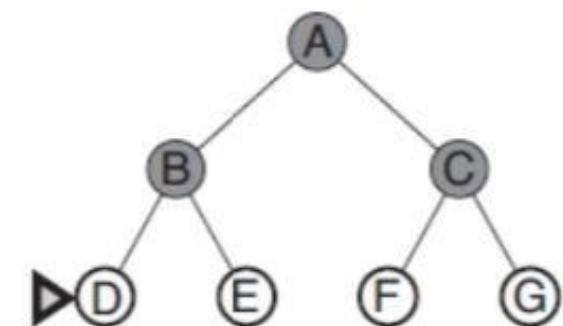
A



BC



CDE



DEFG

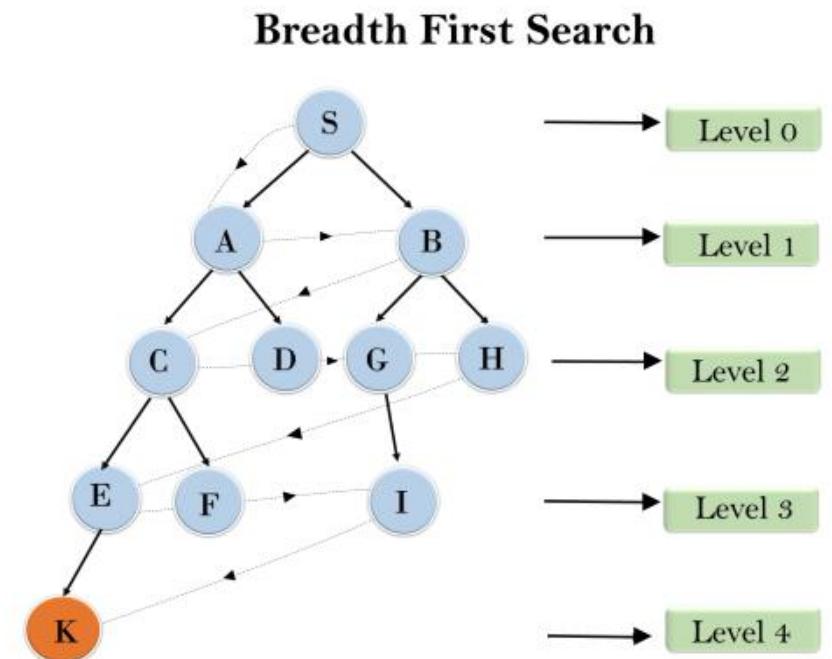
1. Breadth-first Search (BFS)

Advantage

- If there is a **solution**, breadth-first search is **guaranteed** to find it.
- If there are **several solutions**, breadth-first search will **always find the shallowest goal state first**.

Drawbacks

- All the leaf nodes of the tree must be maintained in memory at the same time, resulting in **enormous memory consuming and time consuming** in search.



1. Breadth-first Search (BFS)

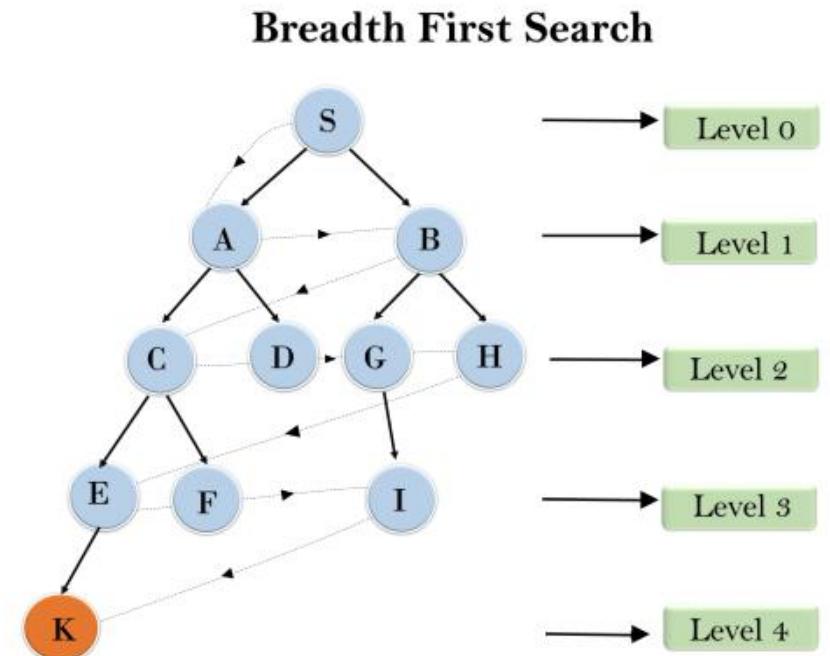
Time vs Memory

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

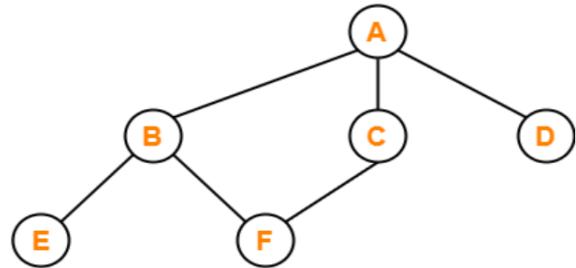
Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

1. Breadth-first Search (BFS)

- Completeness: Yes
- Time complexity $O(b^d)$
- Space complexity $O(b^d)$
- Cost optimality: Not Guaranteed



Example



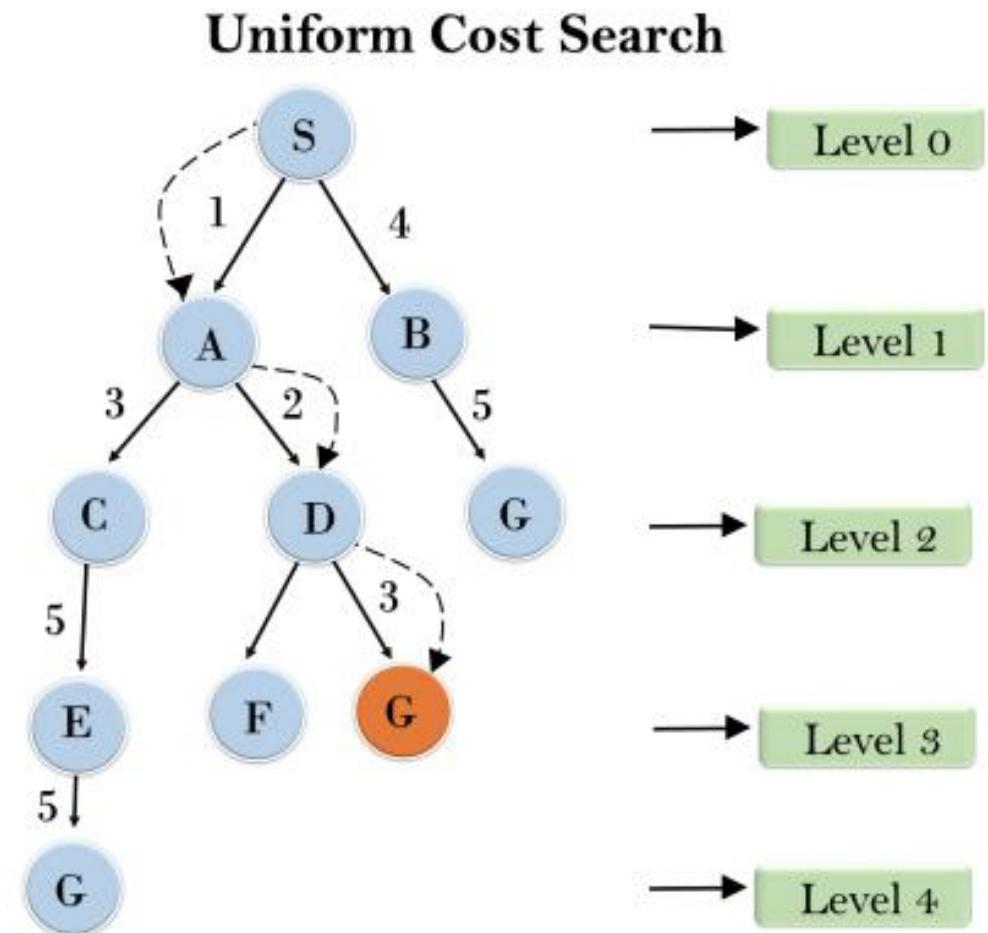
```
In [1]: from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph=defaultdict(list)
    def addEdge(self, u, v):
        self.graph[u].append(v)
    def BFS(self, s):
        visited = [False]*7
        queue=[]
        queue.append(s)
        visited[s]=True
        while queue:
            s=queue.pop(0)
            print(s, end="")
            for i in self.graph[s]:
                if visited[i]==False:
                    queue.append(i)
                    visited[i]=True

g = Graph()
g.addEdge(1, 2)
g.addEdge(1, 3)
g.addEdge(1, 4)
g.addEdge(2, 5)
g.addEdge(2, 6)
g.addEdge(3, 6)
print ("Following is Breath First Traversal (Starting from vertex 1)")
g.BFS(1)
# ['A':1, 'B':2, 'C':3, 'D':4, 'E':5, 'F':6]
```

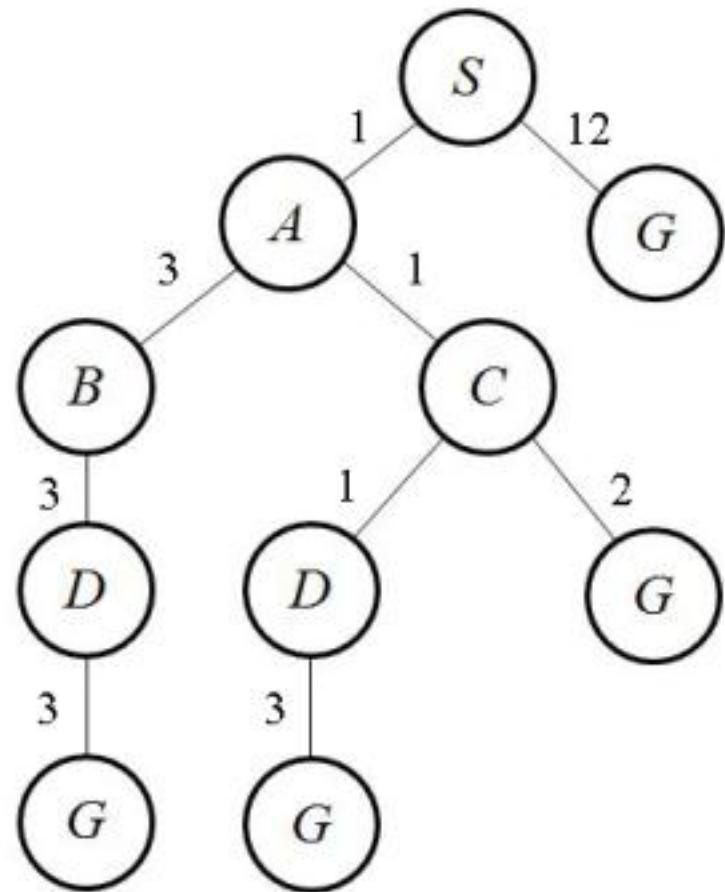
Following is Breath First Traversal (Starting from vertex 1)
123456

2. Uniform-cost search

- Breadth-first search finds the shallowest goal state, but this may not always be the **least-cost solution** for a general path cost function.
- Uniform cost search modifies the breadth-first strategy by always **expanding the lowest-cost node**.
- It does not care about the number of steps involved in searching and only concerned about path cost.



Example



Insert the root into the queue

While the queue is not empty

 Dequeue the maximum priority element from the queue

 (If priorities are same, alphabetically smaller path is chosen)

If

 the path is ending in the goal state, print the path and exit

Else

 Insert all the children of the dequeued element, with the cumulative costs as priority

Initialization: { [S , 0] }

Iteration1: { [S->A , 1] , [S->G , 12] }

Iteration2: { [S->A->C , 2] , [S->A->B , 4] , [S->G , 12] }

Iteration3: { [S->A->C->D , 3] , [S->A->B , 4] , [S->A->C->G , 4] , [S->G , 12] }

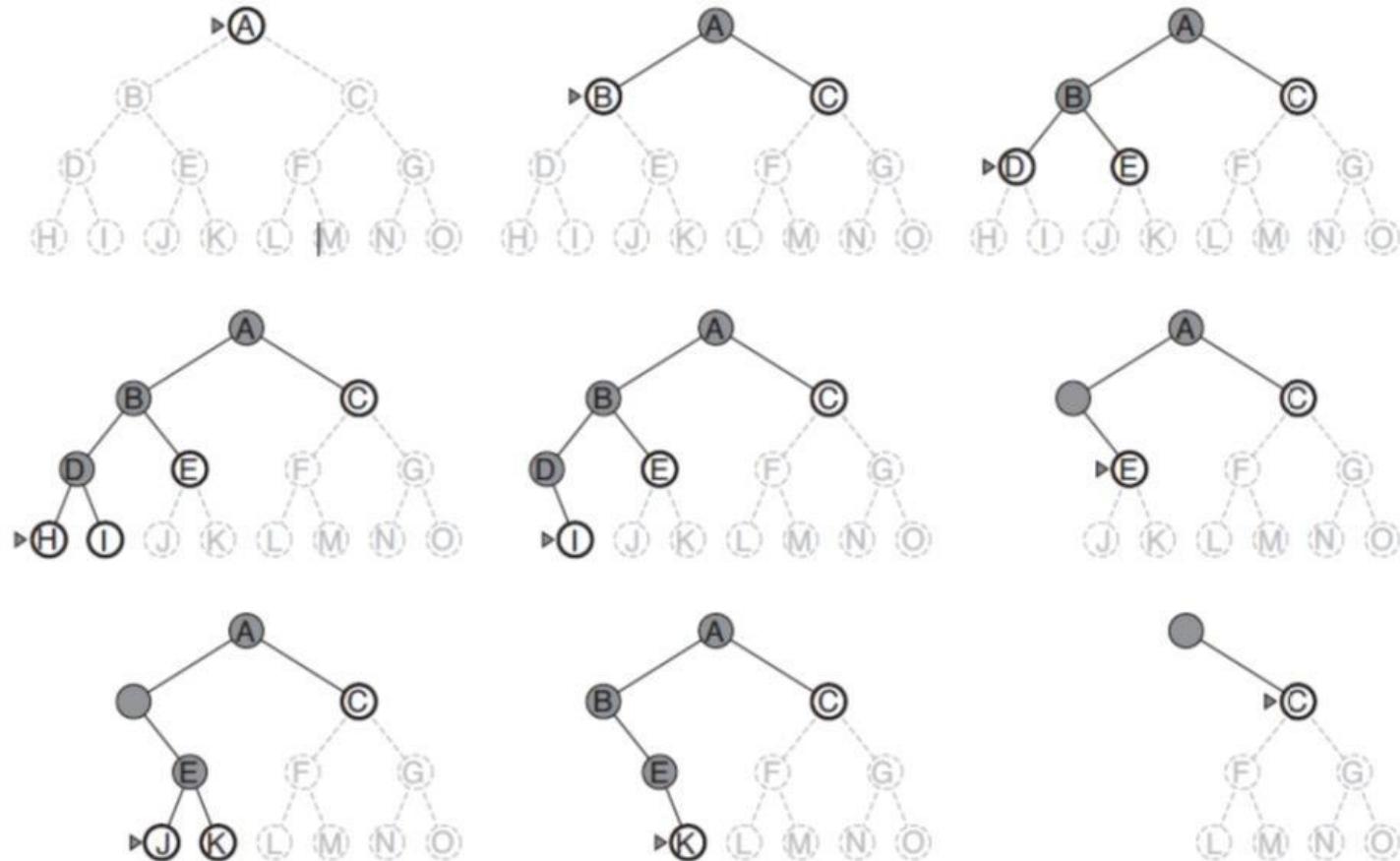
Iteration4: { [S->A->B , 4] , [S->A->C->G , 4] , [S->A->C->D->G , 6] , [S->G , 12] }

Iteration5: { [S->A->C->G , 4] , [S->A->C->D->G , 6] , [S->A->B->D , 7] , [S->G , 12] }

Iteration6 gives the final output as S->A->C->G.

3. Depth-first search (DFS)

- Depth-first search always expands one of the nodes at the deepest level of the tree.



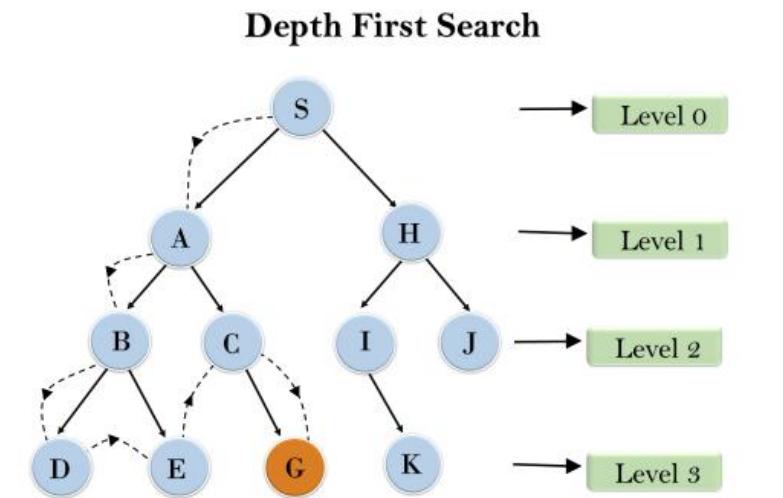
3. Depth-first search (DFS)

Advantage

- Depth-first search has very **modest memory requirements**
 - It needs to store only a single path from the root to a leaf node, along with the remaining unexpanded sibling nodes for each node on the path.

Drawbacks

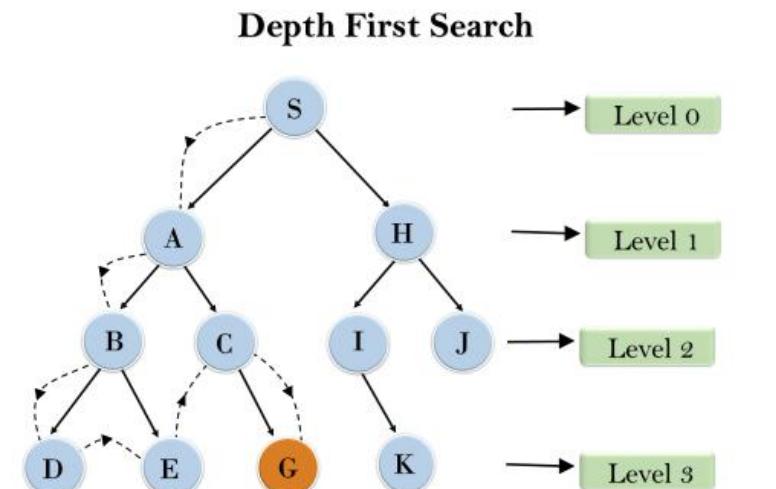
- It can get **stuck going down the wrong path**
 - Many problems have very deep or even infinite search trees;
 - The search will always continue downward without backing up, even when a shallow solution exists.



Root node--->Left node ----> right node.

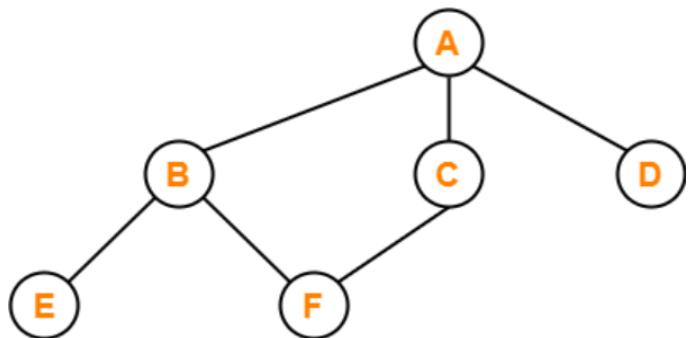
3. Depth-first search (DFS)

- Completeness: Not Guaranteed
- Time complexity $O(b^m)$
- Space complexity $O(bm)$
- Cost optimality: Not Guaranteed



Root node--->Left node ----> right node.

Example:



In [6]:

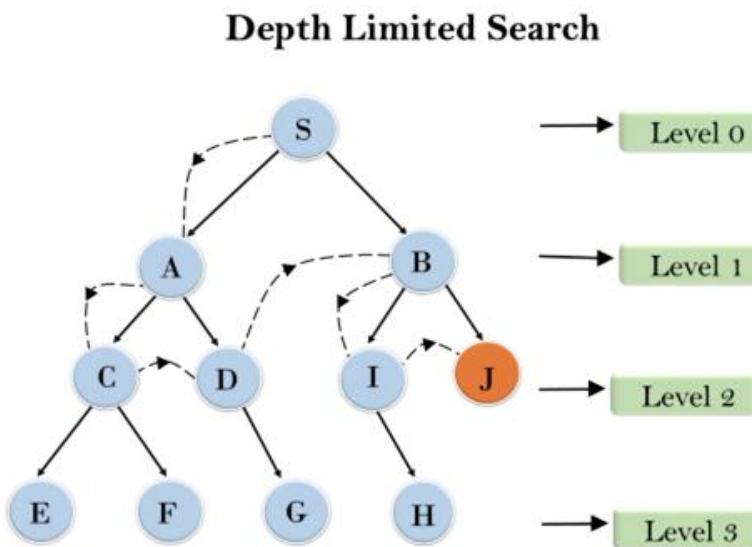
```
from collections import defaultdict
class Graph:
    def __init__(self):
        self.graph = defaultdict(list)
    def addEdge(self, u, v):
        self.graph[u].append(v)
    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=' ')
        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)
    def DFS(self, v):
        visited = set()
        self.DFSUtil(v, visited)
g = Graph()
g.addEdge(1, 2)
g.addEdge(1, 3)
g.addEdge(1, 4)
g.addEdge(2, 5)
g.addEdge(2, 6)
g.addEdge(3, 6)

print("Following is DFS from (starting from vertex 1)")
g.DFS(1)
# {'A':1, 'B':2, 'C':3, 'D':4, 'E':5, 'F':6}
```

Following is DFS from (starting from vertex 1)
1 2 5 6 3 4

4. Depth-limited search

- Depth-limited search is an **extension of DFS**:
 - only expands nodes with depth up to l .
- For example, on the map of Romania, there are 20 cities, so we know that if there is a solution, then it must be of length 19 at the longest.
- Hard to **choose an appropriate depth number l** . For most problems, we will not know a good depth limit until we have solved the problem.



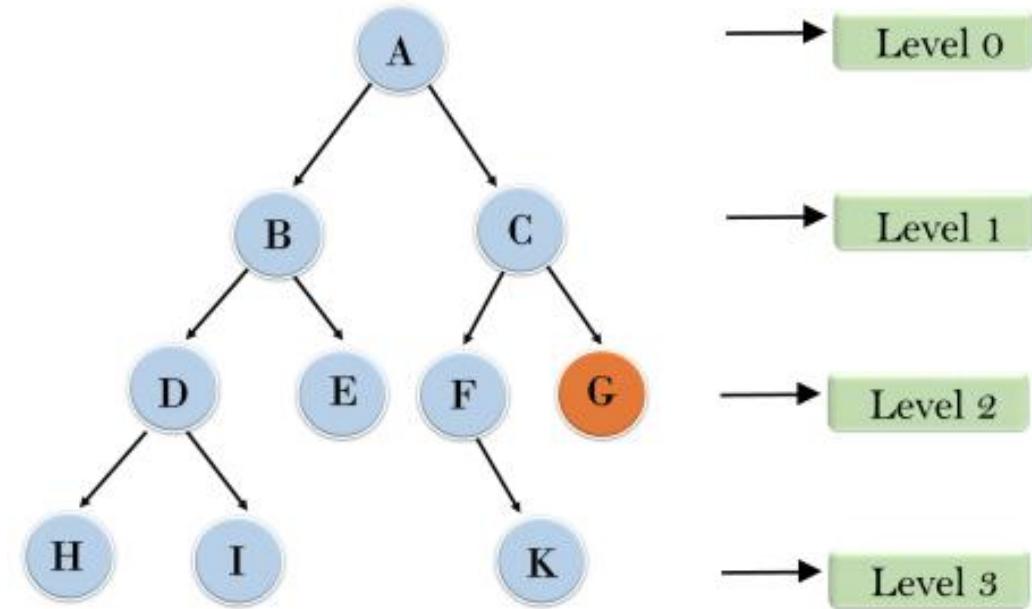
4. Depth-limited search

- Advantages:
 - It is **memory efficient**
 - Space complexity $O(bl)$
 - Time complexity $O(b^l)$
- Disadvantages
 - Incompleteness.
 - It may not be optimal if the problem has more than one solution.
- Advantages:
 - It is **memory efficient**
 - Space complexity $O(bl)$
 - Time complexity $O(b^l)$
- Disadvantages
 - Incompleteness.
 - It may not be optimal if the problem has more than one solution.

5. Iterative deepening search

- The iterative deepening algorithm is a **combination of DFS and BFS algorithms**. This search algorithm finds out the best depth limit and does it by **gradually increasing** the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.
- The main drawback is that it **repeats all the work of the previous phase**.

Iterative deepening depth first search



1'st Iteration----> A

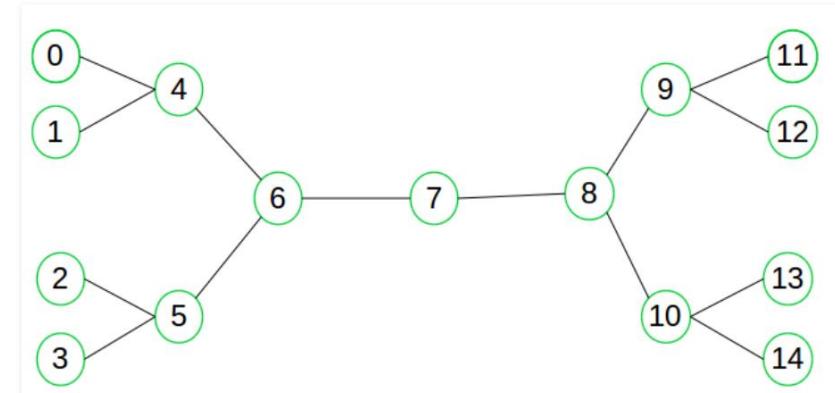
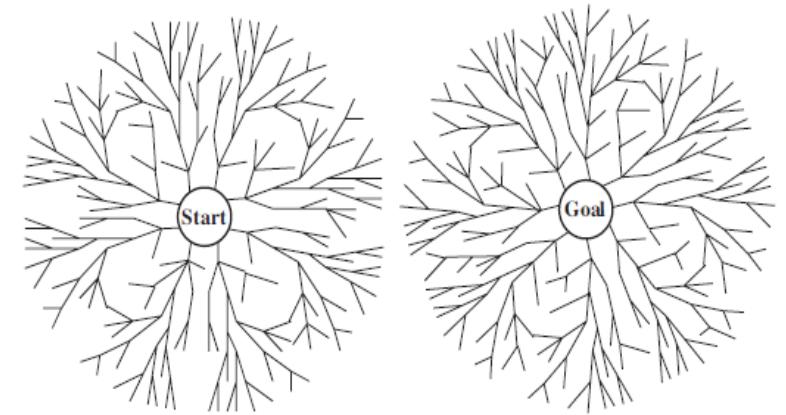
2'nd Iteration----> A, B, C

3'rd Iteration----->A, B, D, E, C, F, G

4'th Iteration----->A, B, D, H, I, E, C, F, K, G

6. Bidirectional search

- Bidirectional search algorithm runs **two simultaneous searches**, one from initial state called as forward-search and other from goal node called as backward-search, to find the goal node.
- Bidirectional search replaces one single search graph with two small subgraphs in which one starts the **search from an initial vertex** and the **other starts from the goal vertex**.
- The **search stops** when these two **graphs intersect** each other.
- Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.
- In bidirectional search, one should know the goal state in advance.



Comparisons

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes ¹	Yes ^{1,2}	No	No	Yes ¹	Yes ^{1,4}
Optimal cost?	Yes ³	Yes	No	No	Yes ³	Yes ^{3,4}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Figure 3.15 Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: ¹ complete if b is finite, and the state space either has a solution or is finite. ² complete if all action costs are $\geq \epsilon > 0$; ³ cost-optimal if action costs are all identical; ⁴ if both directions are breadth-first or uniform-cost.



Informed Search

Best-first Search

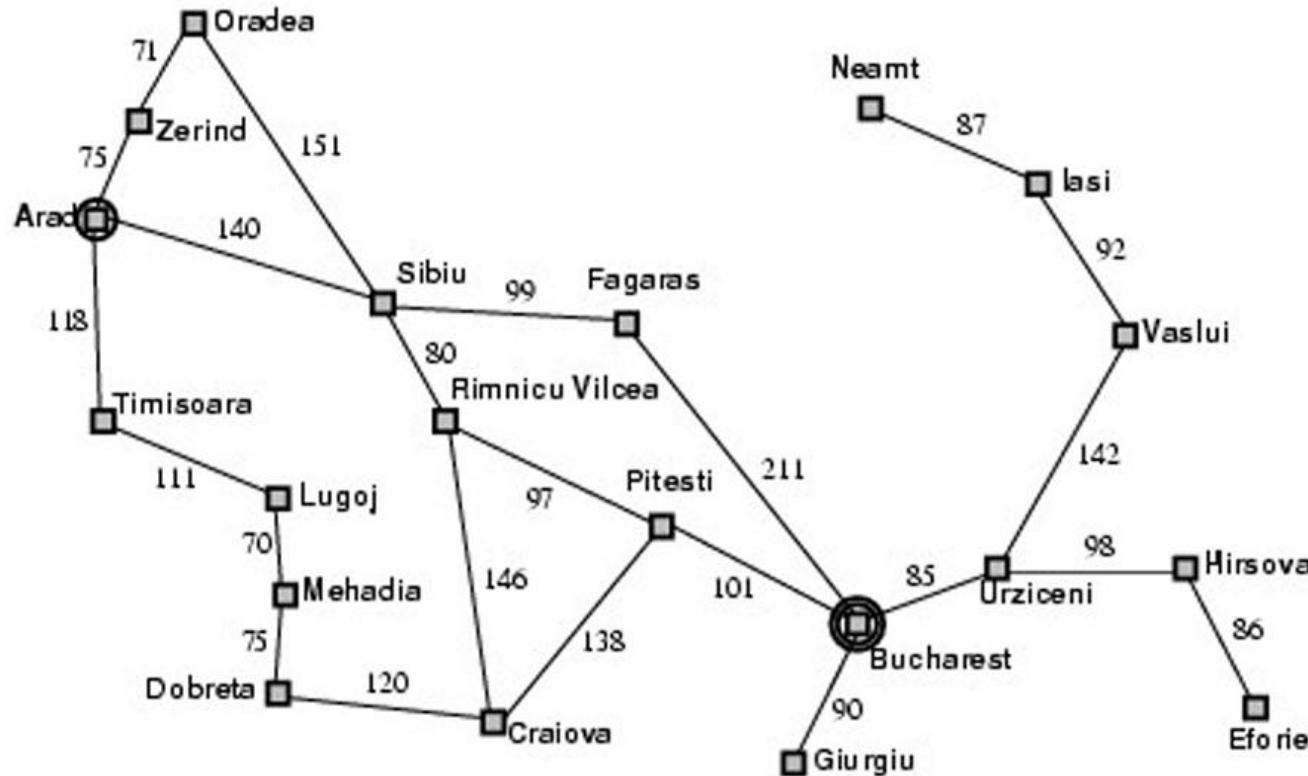
- Provide an **evaluation function** that returns a number purporting to describe the desirability
- Best-first Search
 - Use an **evaluation function** for each node.
 - Estimate of "desirability".
 - Expand most **desirable unexpanded node**.
- There is a whole family of Best-first Search algorithms with different evaluation functions
 - Two basic approaches
 - ✓ The first (named *Greedy search*) tries to expand the node closest to the goal;
 - ✓ The second (named *A* search*) tries to expand the node on the least-cost solution path.

Greedy Search: Minimize Estimated Cost to Reach a Goal

- For most problems, the **cost of reaching the goal from a particular state can be estimated** but cannot be determined exactly.
- A function that calculates such cost estimates is called a **heuristic function** (usually denoted by the letter h):
 - $h(n)$ = estimated cost of **the cheapest path from the state at node n to a goal state**
 - A best-first search that uses h to select the next node to expand is called **greedy search**.
 - Strictly speaking, h can be any function at all. We will require only that $h(n) = 0$ if n is a goal.

An Example - Route-finding Problems

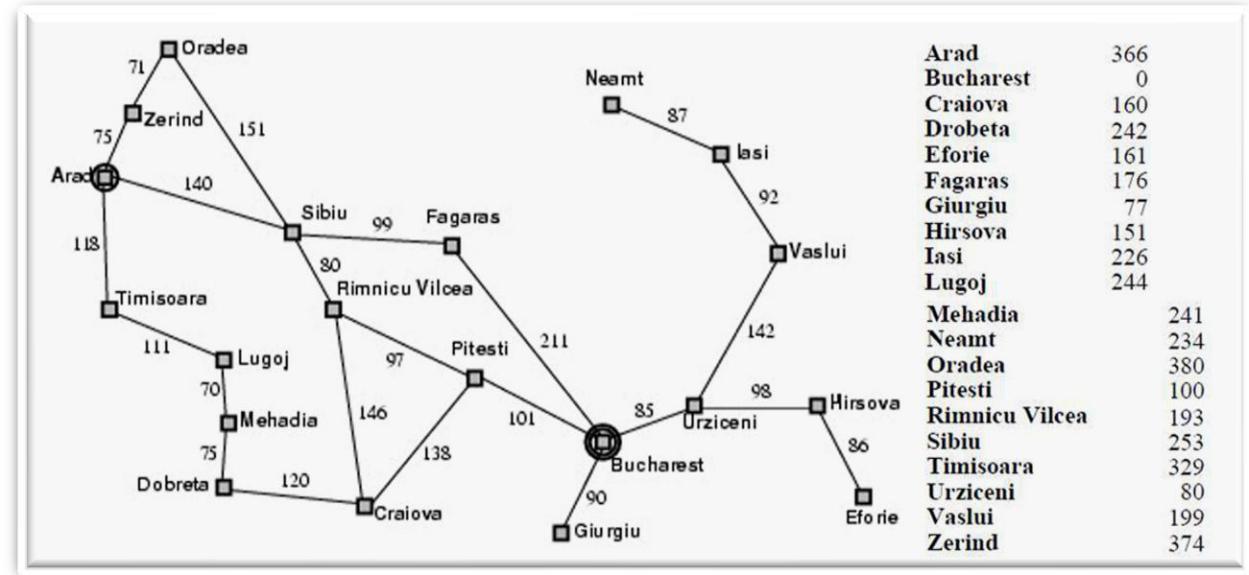
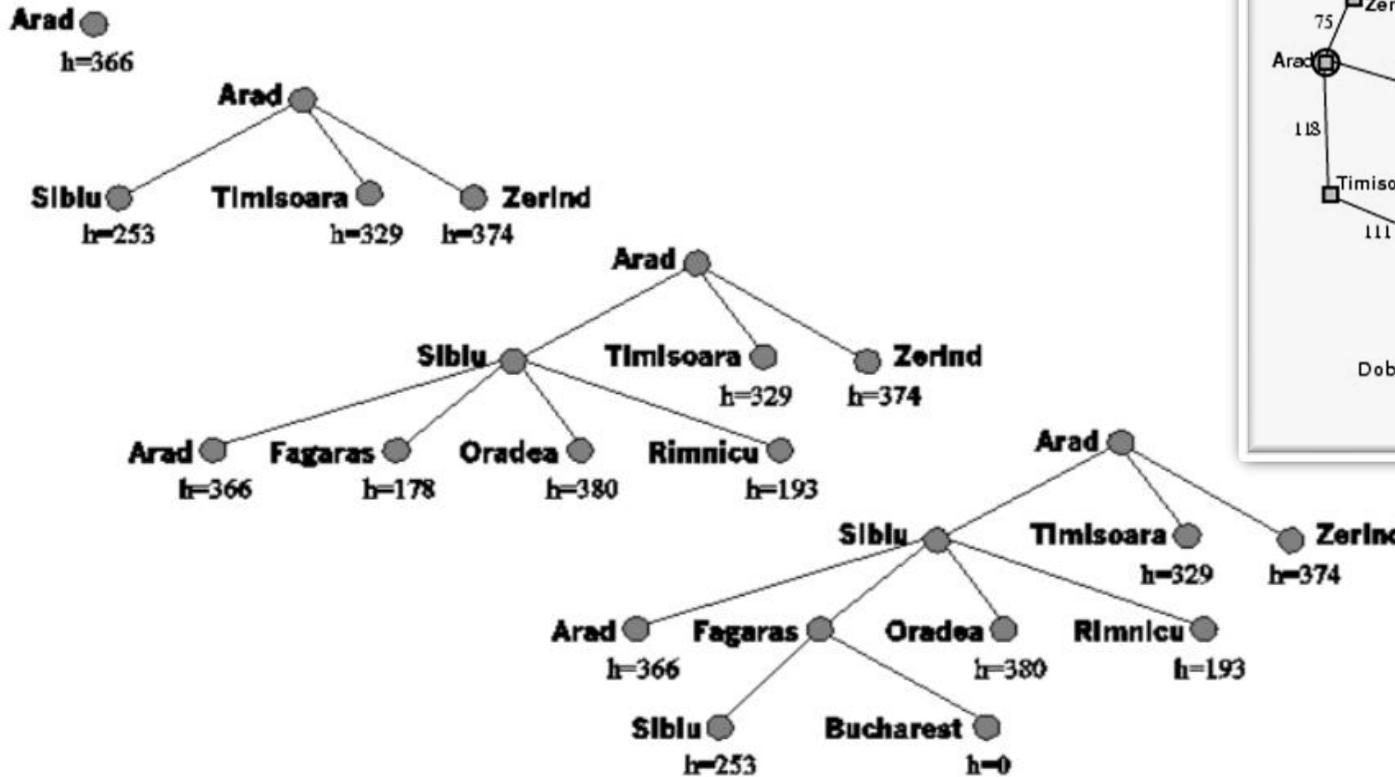
What's a proper heuristic that measures cheapest path from current node to goal node?



Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h_{SLD}(n) = \text{straight line distance (SLD) between } n \text{ and the goal location}$

Greedy Search in the Example (From Arad to Bucharest)



Result:

Arad Sibiu Făgăras,
 Bucharest

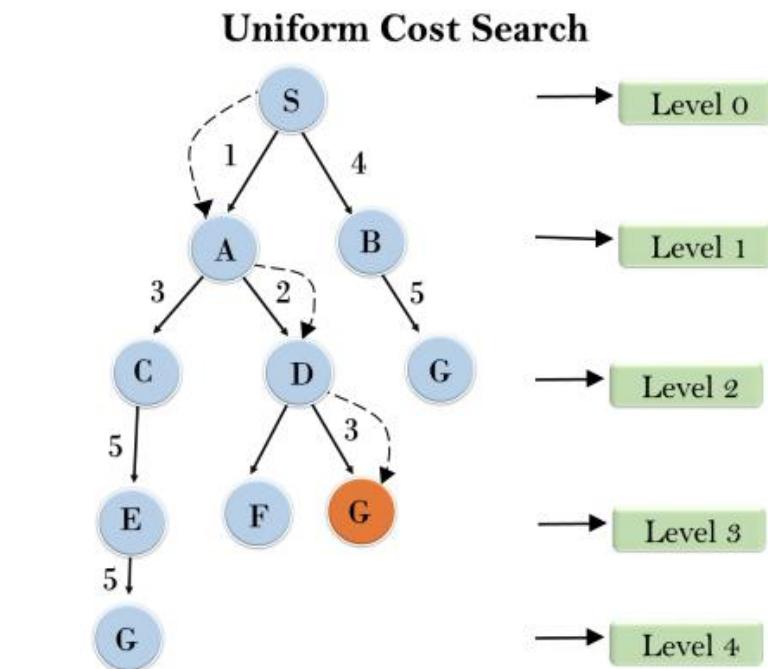
Note:

- The result is not perfectly optimal
- The optimal result should be:
 Arad Sibiu Râmnicu Vâlcea
 Pitesti Bucharest

Greedy search prefers to take the biggest bite possible out of the remaining cost to reach the goal, without worrying about whether this will be best in the long run.

Greedy Search V.S. Uniform-cost Search

- Greedy Search
 - Minimize the estimated cost to the goal $h(n)$
- Uniform-cost Search
 - Minimize the cost the path so far, $g(n)$



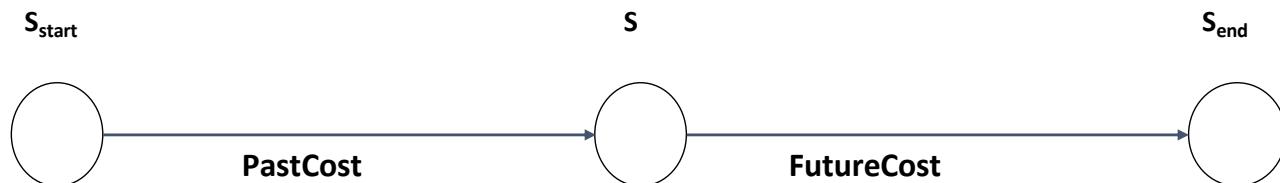
- Can we combine these two strategies to get the advantages of both?

Minimizing the Total Path Cost: A* Search

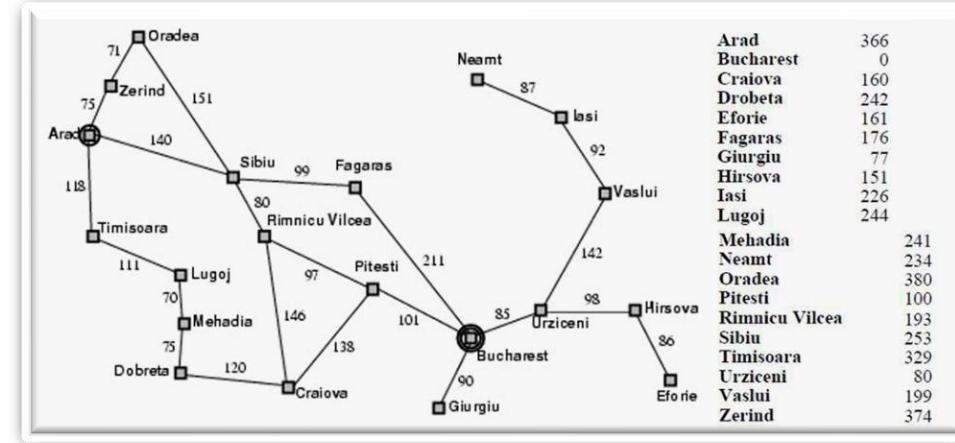
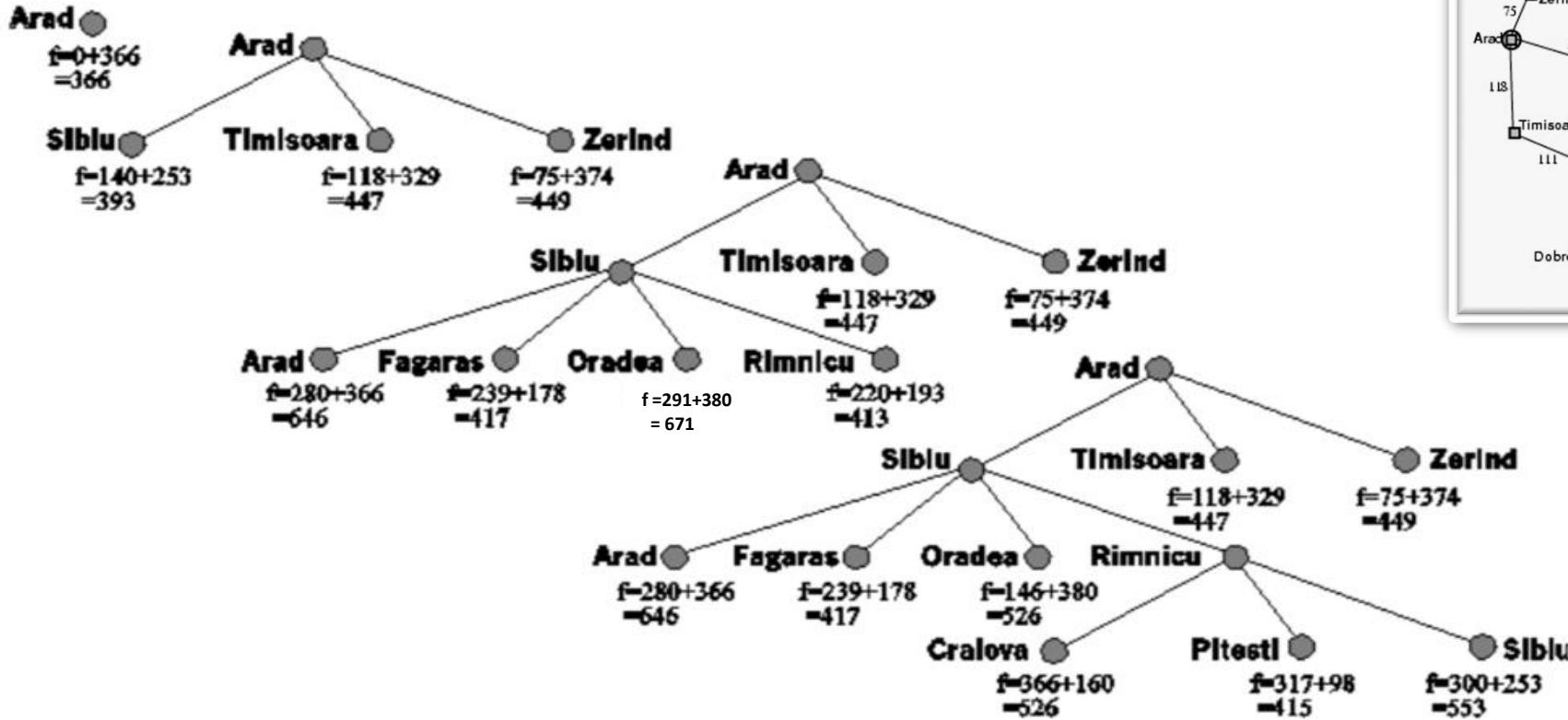
- We can do exactly that, combining the two evaluation functions simply by
$$f(n) = g(n) + h(n)$$

where:

- $g(n)$: the path cost from the start node to node n ;
- $h(n)$: the estimated cost of the cheapest path from n to the goal;
- We have: $f(n) = \text{estimated cost of the cheapest solution through } n$.



A* Search in a Route-finding Problem

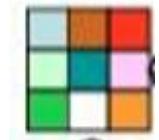


The Behavior of A* Search

- A heuristic for which it holds is said to exhibit **monotonicity**.
 - Let us consider two nodes n and n' , where **n is the parent of n'** .
 - Now suppose, for example, that $g(n) = 3$ and $h(n) = 4$, then $f(n) = g(n) + h(n) = 7$.
 - ✓ It means that the true cost of a solution path through n is at least 7.
 - Suppose also that $g(n') = 4$ and $h(n') = 2$, then $f(n') = 6$.
 - (an example of a nonmonotonic heuristic)
- If the heuristic is one of those odd ones that is not monotonic, we can make a minor correction that restores monotonicity

$$f(n') = \max(f(n), g(n') + h(n'))$$

How to Find a Good Heuristic Function?



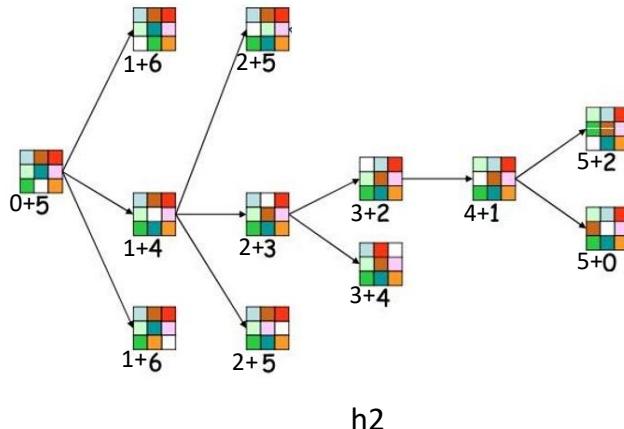
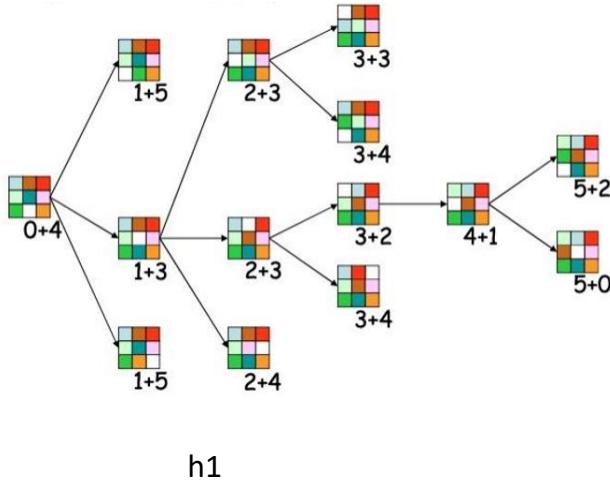
Start state

$$h_1 = 4$$
$$h_2 = 5$$



Goal state

- Two candidates in this 8 puzzle problem
 - h_1 = the number of tiles that are in the wrong positions.
 - h_2 = the sum of the distances of the tiles from their goal positions. (also called **city block distance** or **Manhattan distance**)

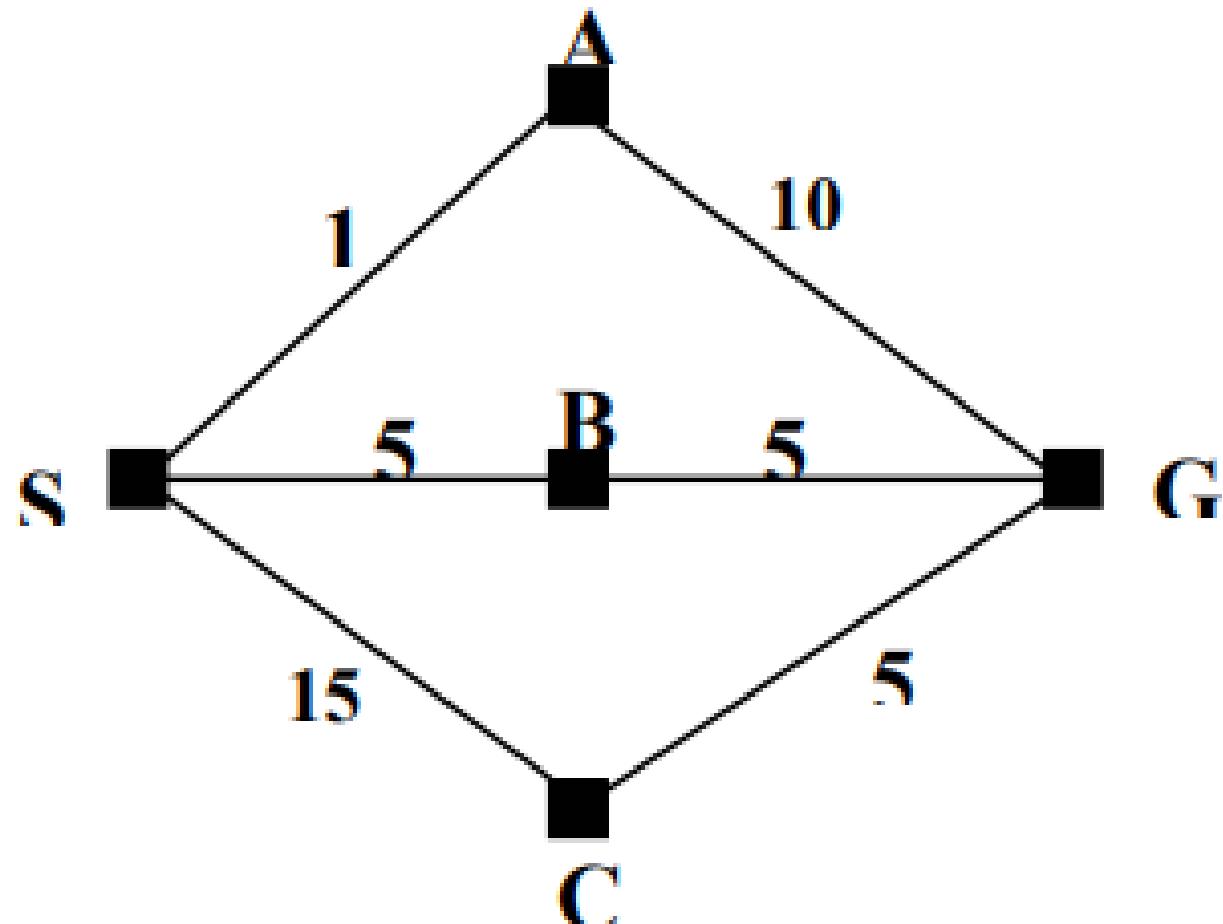


Note that by definition, $h_2(n) \geq h_1(n)$: h_2 **dominates** h_1

It is always better to use a heuristic function with higher values, as long as it does not overestimate.

Class Question

Question: Show Uniform Cost search Operates

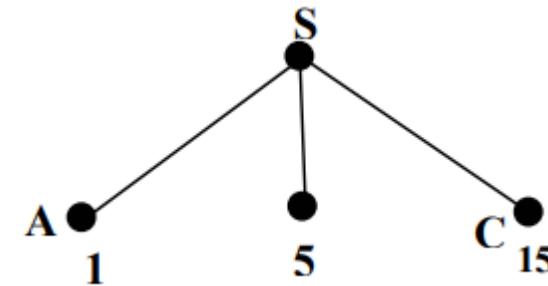


Class Question

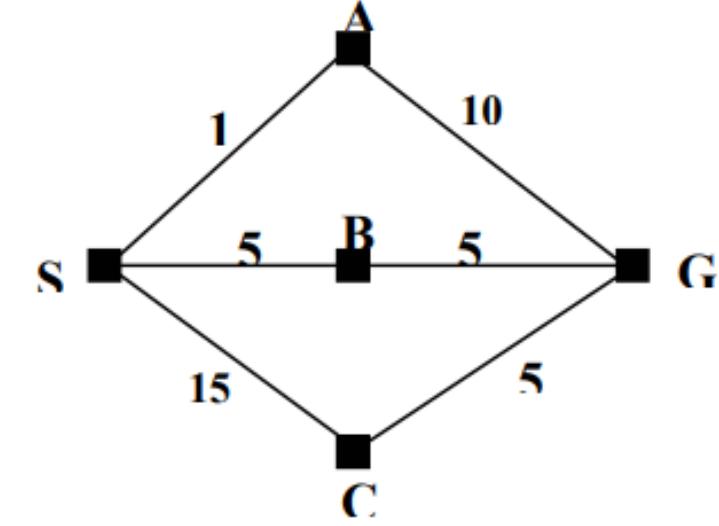
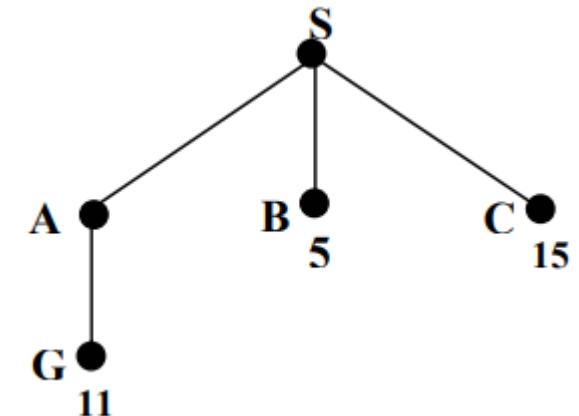
Question: Show Uniform Cost search Operates

Answer:

Step-1: We start with the initial state and expand it

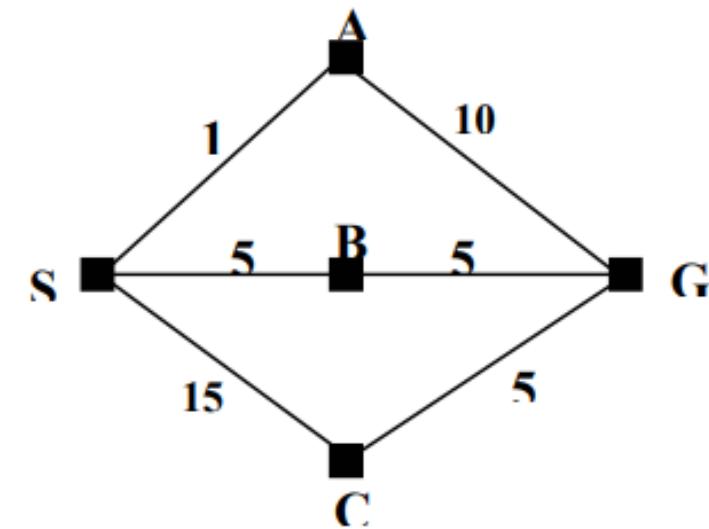


Step-2: The path cost of A is the cheapest, so it is expanded next; giving the following tree



Class Question

Question: Show Uniform Cost search Operates



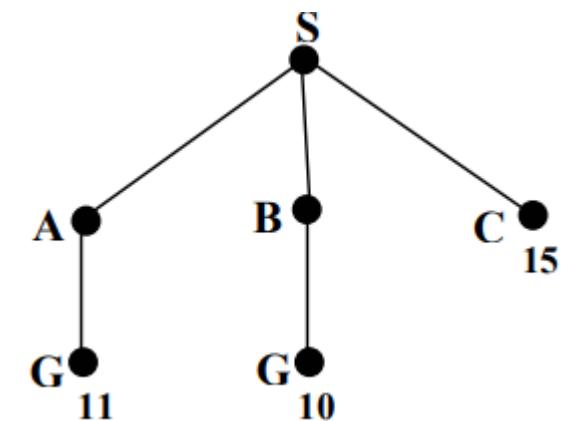
Step-3: We now have a goal state, but the algorithm does not recognize it yet as there is still a node with a cheaper path cost. In fact, what the algorithm does is order the queue by the path cost so the node with cost 11 will be behind node B in the queue. Node B (being the cheapest) is now expanded, giving

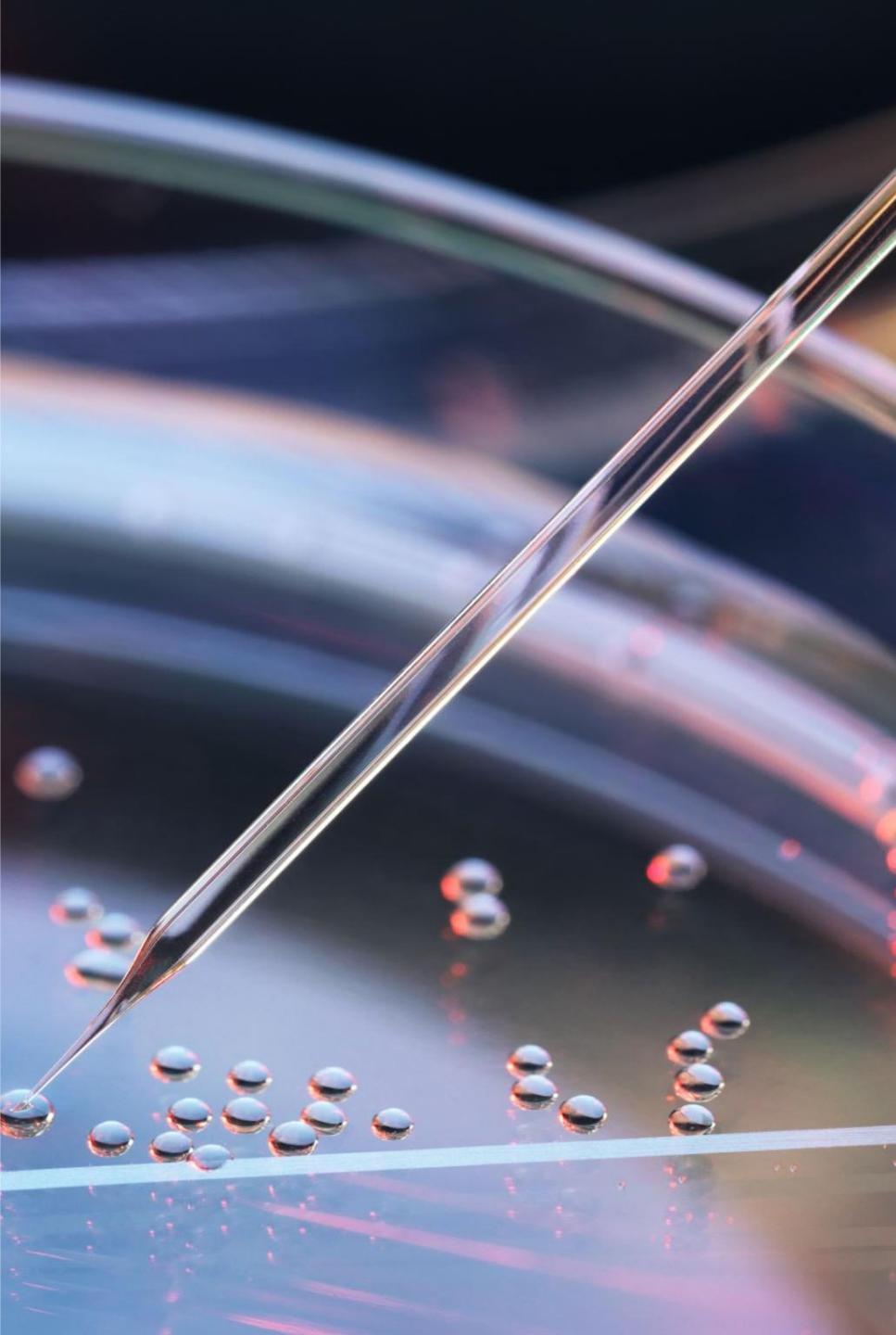
A goal state (G) will now be at the front of the queue. Therefore, the search will end and the path SBG will be returned.

$$S \rightarrow B \rightarrow G = 10$$

$$S \rightarrow A \rightarrow G = 11$$

cheapest solution





Summary

- Uninformed search
 - Breath-first search
 - Uniformed-cost search
 - Depth-first search
 - Depth-limited search
 - Iterative deepening search
 - Bidirectional search
 - Backtracking with forwarding checking
- Informed Search
 - Best-first search
 - Greedy search
 - A* search

Reference

- Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach, global edition 4th." Foundations 19 (2021): 23.
- Leetcode
 - <https://leetcode.com/problems/maximum-depth-of-binary-tree/>
 - <https://leetcode.com/problems/same-tree/>
 - <https://leetcode.com/problems/binary-tree-right-side-view/>
 - <https://leetcode.com/problems/number-of-islands/>

IDAT7215

Computer Programming for Product Development and Applications

Lecture 4-1: Block Diagram Reduction for Control
System

Dr. Zulfiqar Ali

Intended Learning Outcomes (ILOs)

Upon completion of this lecture, the student will be able to:

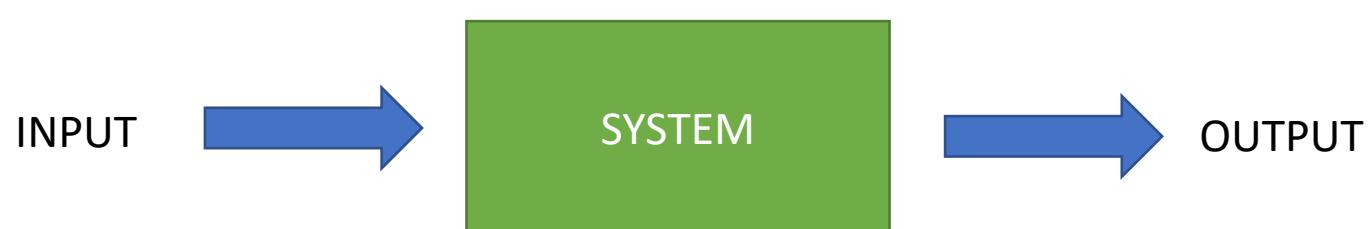
- Describe what is block diagram
- How to represent a system using block diagram
- Learn techniques to reduce block diagram into simple form.

Outline

- Block diagram
- Block Diagram Representation of a System
- Block Diagram Reduction Techniques
- Examples

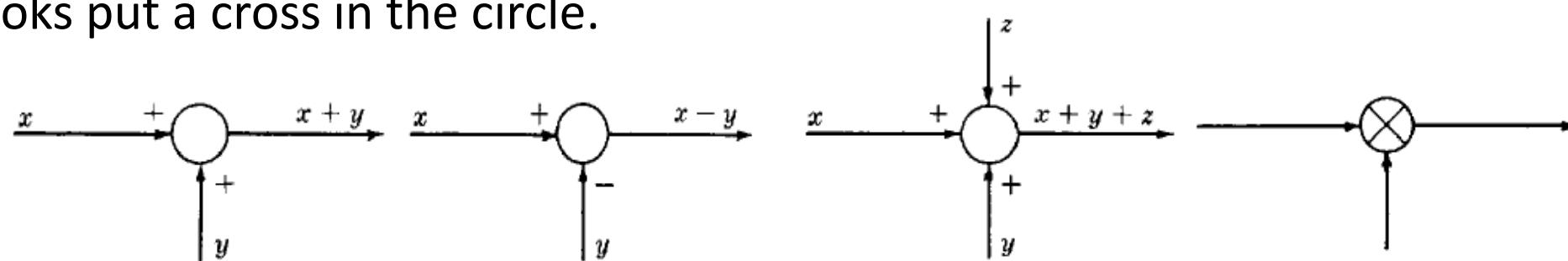
Introduction

- A Block Diagram is a shorthand pictorial representation of the cause-and-effect relationship of a system.
- The interior of the rectangle representing the block usually contains a description of or the name of the element, or the symbol for the mathematical operation to be performed on the input to yield the output.
- The arrows represent the direction of information or signal flow.



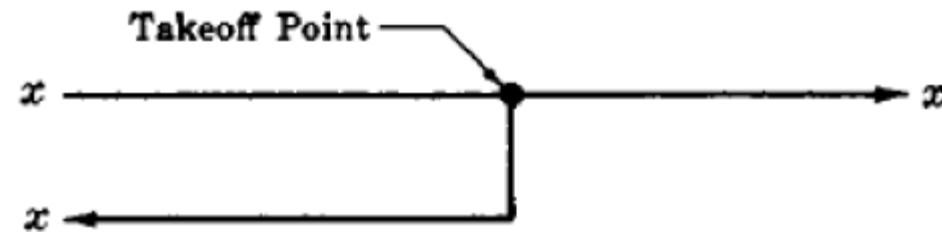
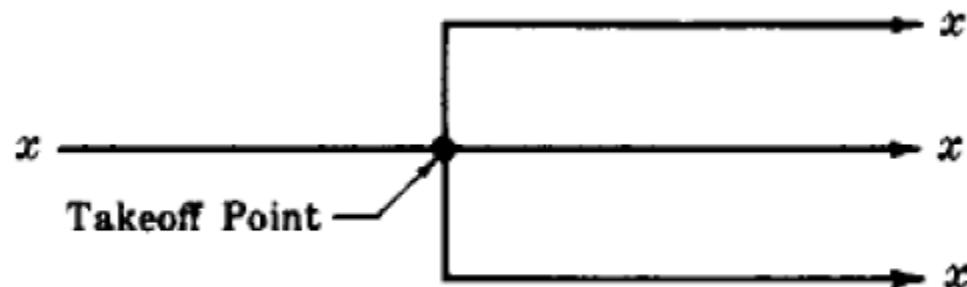
Introduction

- The operations of addition and subtraction have a special representation.
- The block becomes a small circle, called a summing point, with the appropriate plus or minus sign associated with the arrows entering the circle.
- The output is the algebraic sum of the inputs.
- Any number of inputs may enter a summing point.
- Some books put a cross in the circle.



Introduction

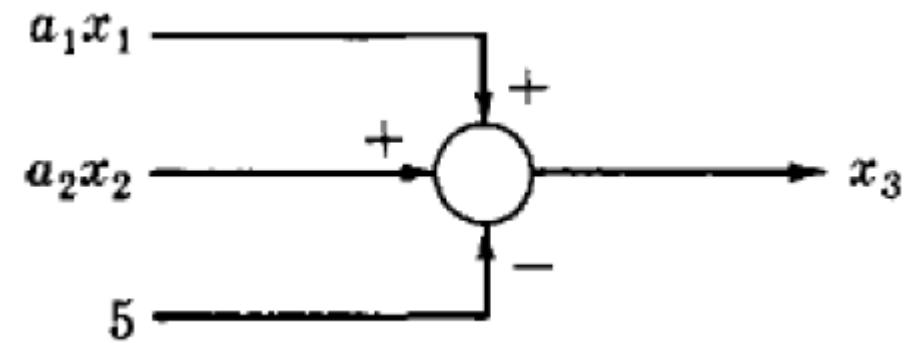
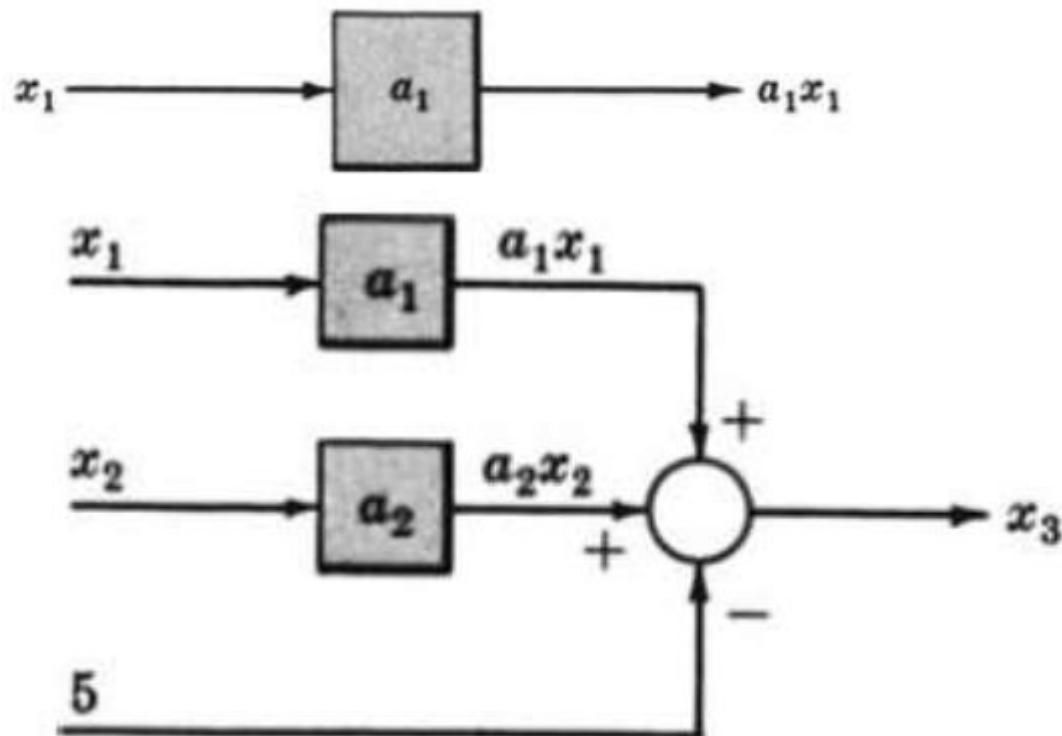
- Take Off point permits the signal to proceed unaltered along several different paths to several destinations.



Example-1

- Consider the following equations in which x_1, x_2, x_3 are variables, and a_1, a_2 are general coefficients or mathematical operators.

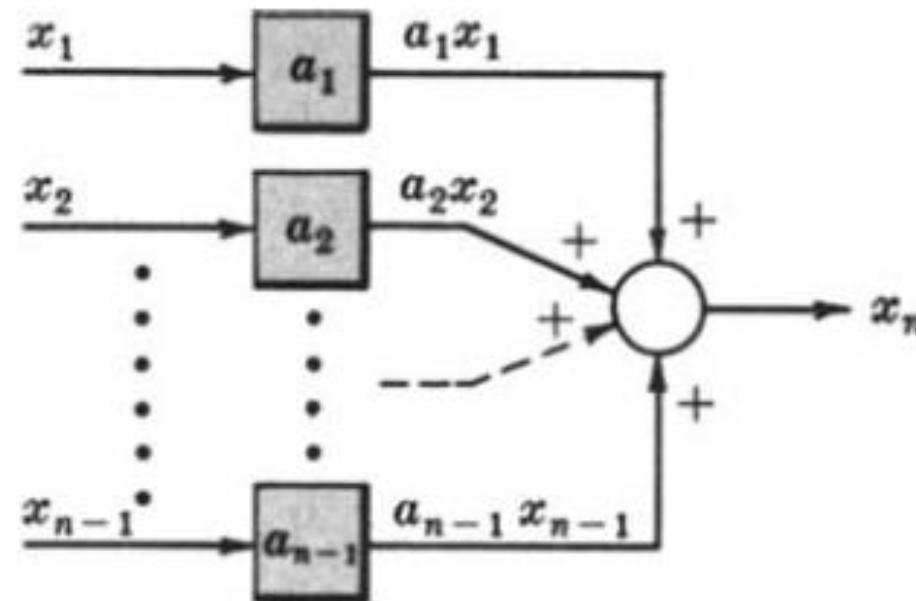
$$x_3 = a_1x_1 + a_2x_2 - 5$$



Example-2

- Consider the following equations in which x_1, x_2, \dots, x_n , are variables, and a_1, a_2, \dots, a_n , are general coefficients or mathematical operators.

$$x_n = a_1 x_1 + a_2 x_2 + a_{n-1} x_{n-1}$$



Procedures for drawing block diagram

1. Write the equations that describe the dynamic behavior for each component.
2. Take Laplace transform of these equations, assuming zero initial conditions.
3. Represent each Laplace-transformed equation individually in block form.
4. Assembly the elements into a complete block diagram.

Example - RLC Circuit System

Voltage Laws

$$V = Ri + L \frac{di}{dt} + V_c$$

$$i = C \frac{dV_c}{dt}$$

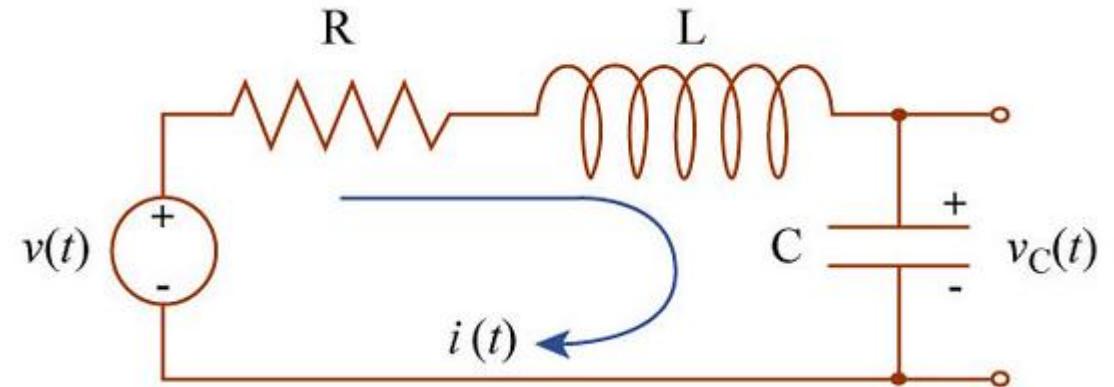
$$V = RC \frac{dV_c}{dt} + LC \frac{d^2V_c}{dt^2} + V_c$$

$$\left(\frac{1}{LC}\right)V = \left(\frac{R}{L}\right)\frac{dV_c}{dt} + \frac{d^2V_c}{dt^2} + \frac{1}{LC}V_c$$

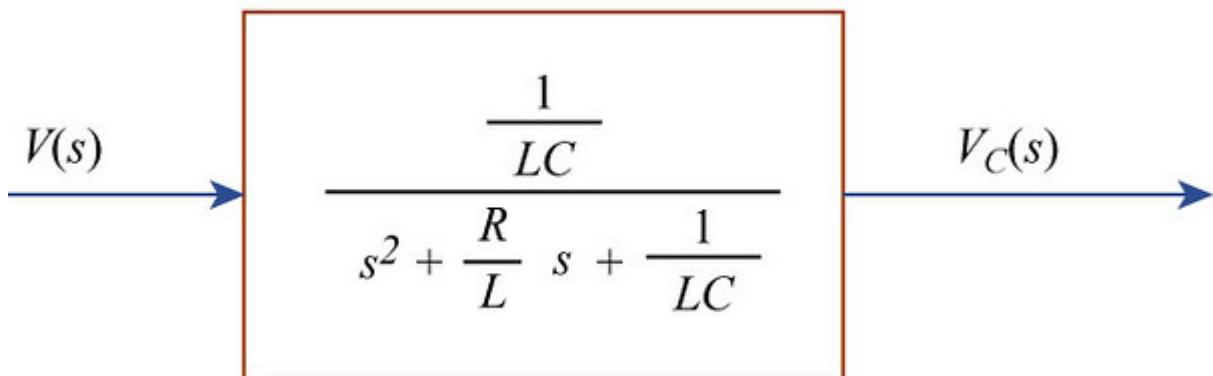
Transfer Function Model:

$$\left(\frac{1}{LC}\right)V(s) = \left(\frac{R}{L}\right)sV_c(s) + s^2V_c(s) + \frac{1}{LC}V_c(s)$$

$$\left(\frac{1}{LC}\right)V(s) = V_c(s) \left(\left(\frac{R}{L}\right)s + s^2 + \frac{1}{LC} \right)$$



i = Current, L = Inductor, C = Capacitor,
 V = input voltage, V_c = Output voltage

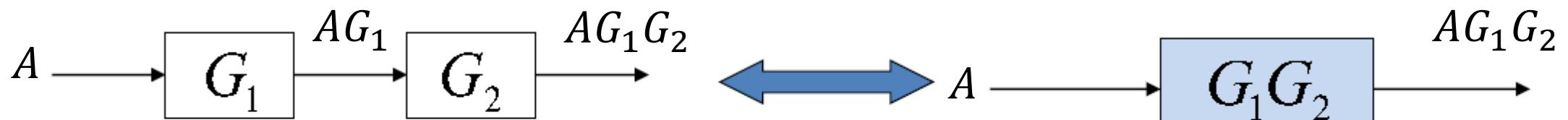


$$\frac{V_c(s)}{V(s)} = \frac{1/LC}{s^2 + \left(\frac{R}{L}\right)s + 1/LC}$$

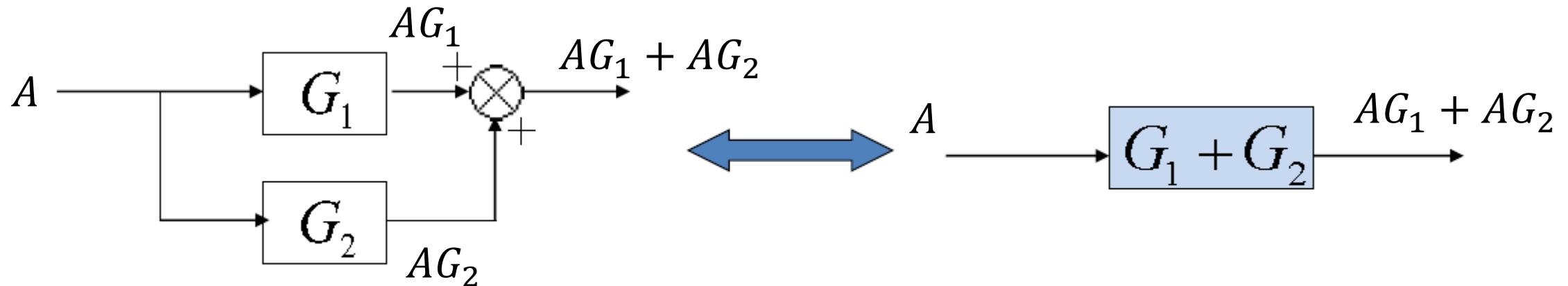
Block diagram reduction techniques

Block diagram reduction techniques

Transformation 1: Combining blocks in cascade

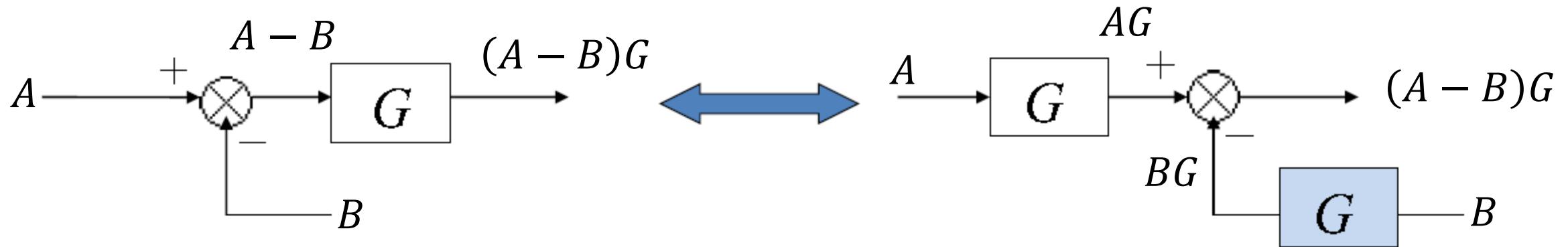


Transformation 2: Combining blocks in parallel

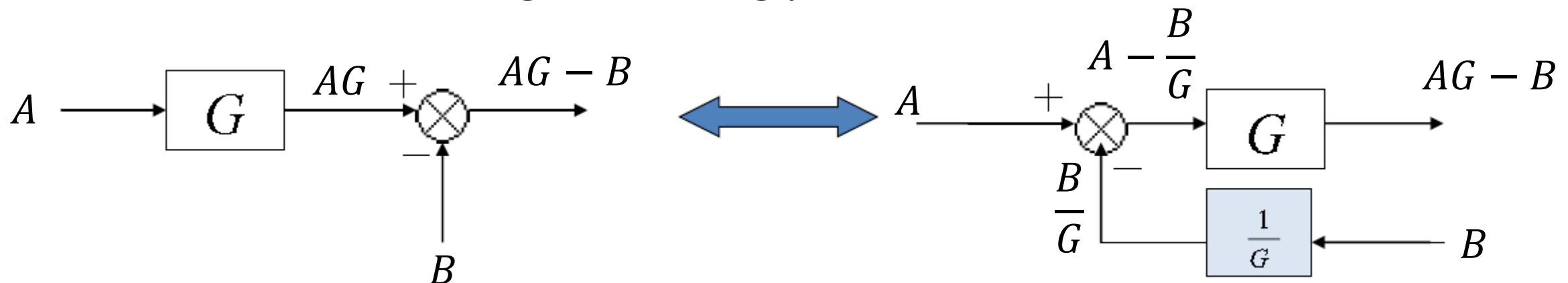


Block diagram reduction techniques

Transformation 3: Moving a summing point ahead a block

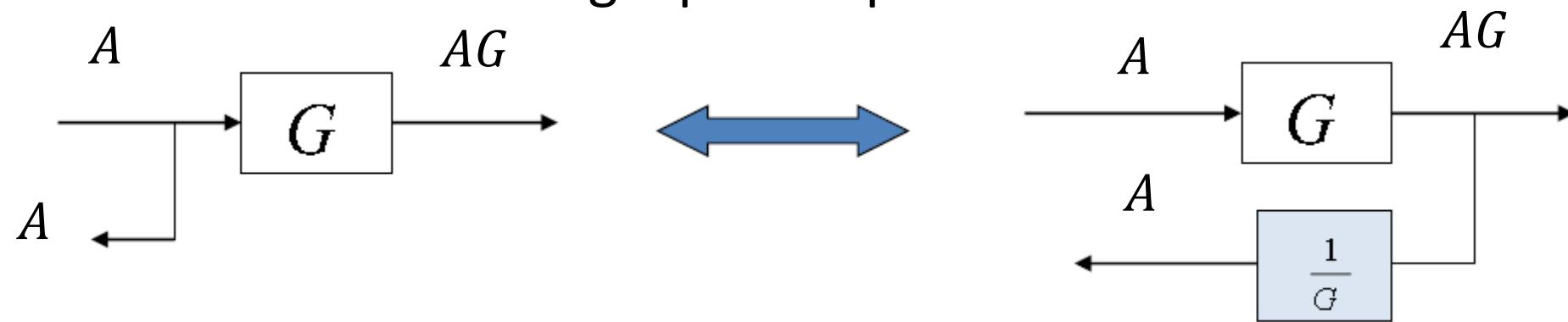


Transformation 4: Moving a summing point behind of a block

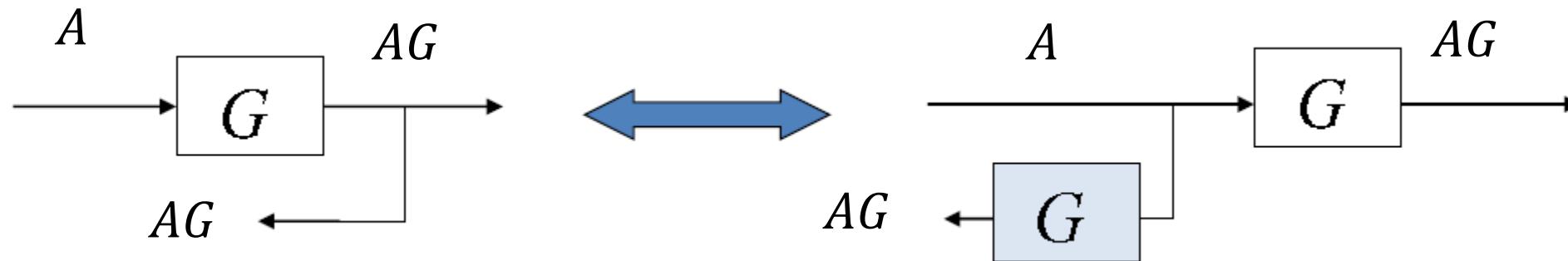


Block diagram reduction techniques

Transformation 5: Moving a pickoff point ahead a block

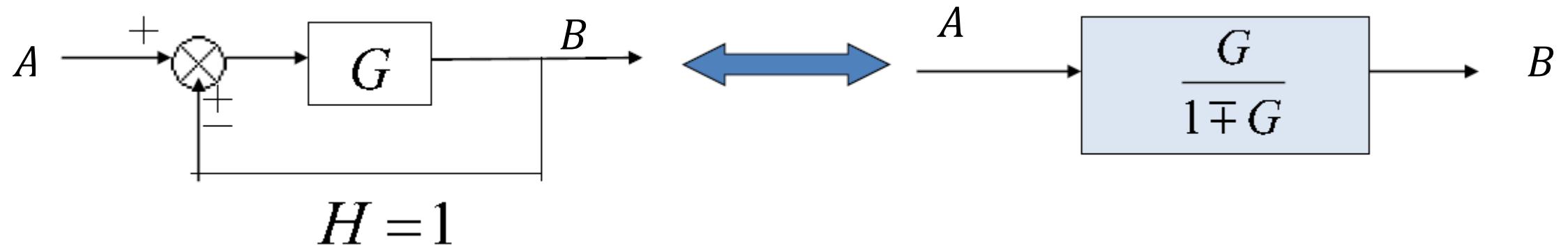
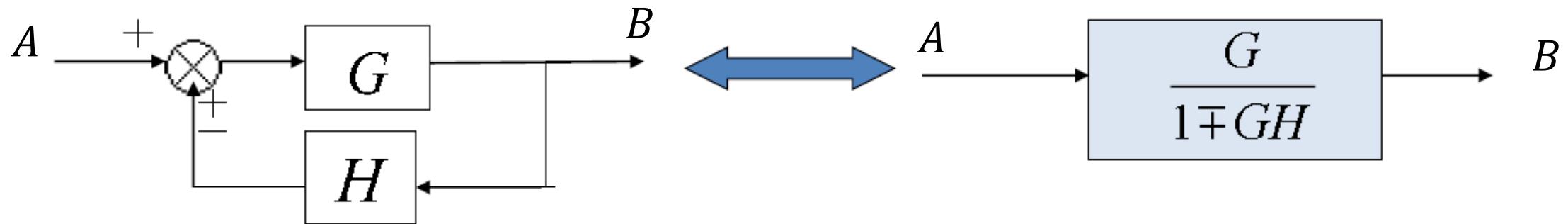


Transformation 6: Moving a pickoff point behind of a block



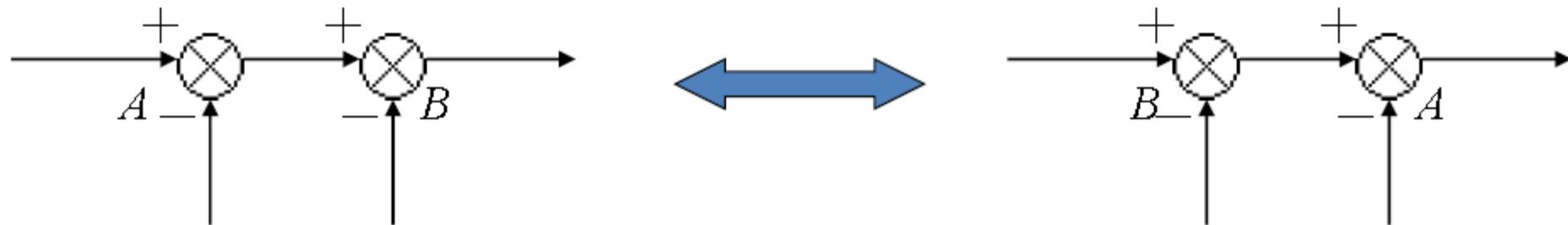
Reduction techniques of Block diagram

Transformation 7: Eliminating a feedback loop

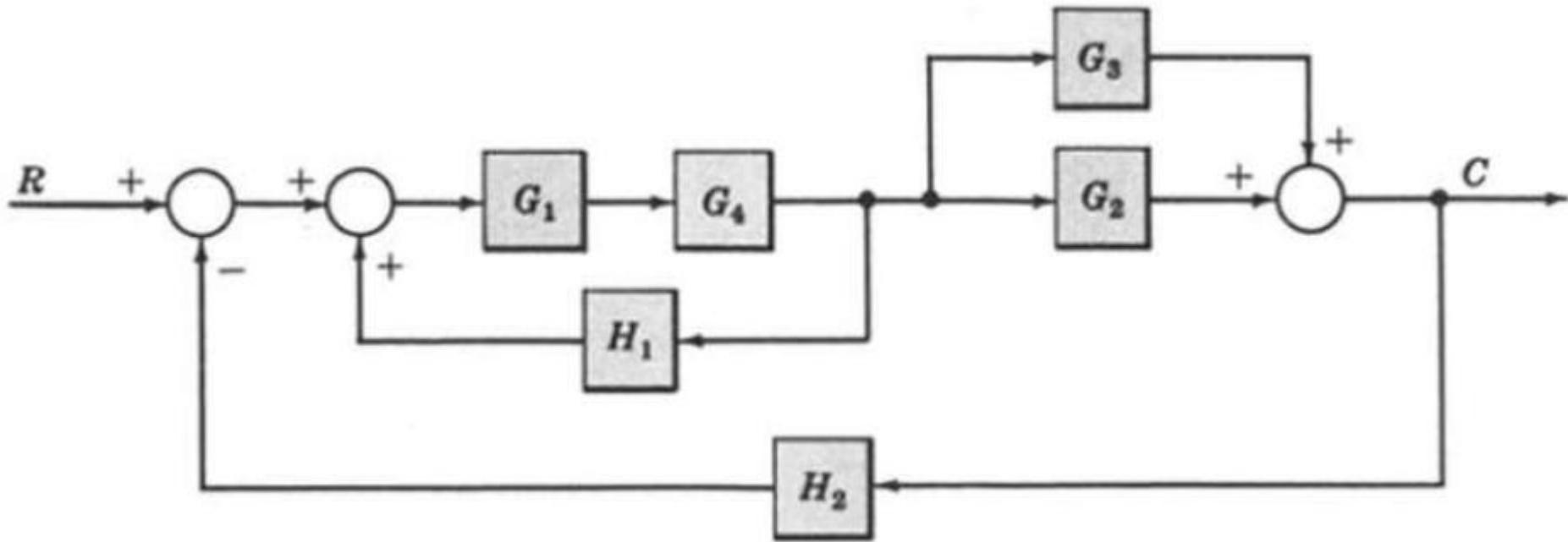


Reduction techniques of Block diagram

Transformation 8: Swap with two neighboring summing points



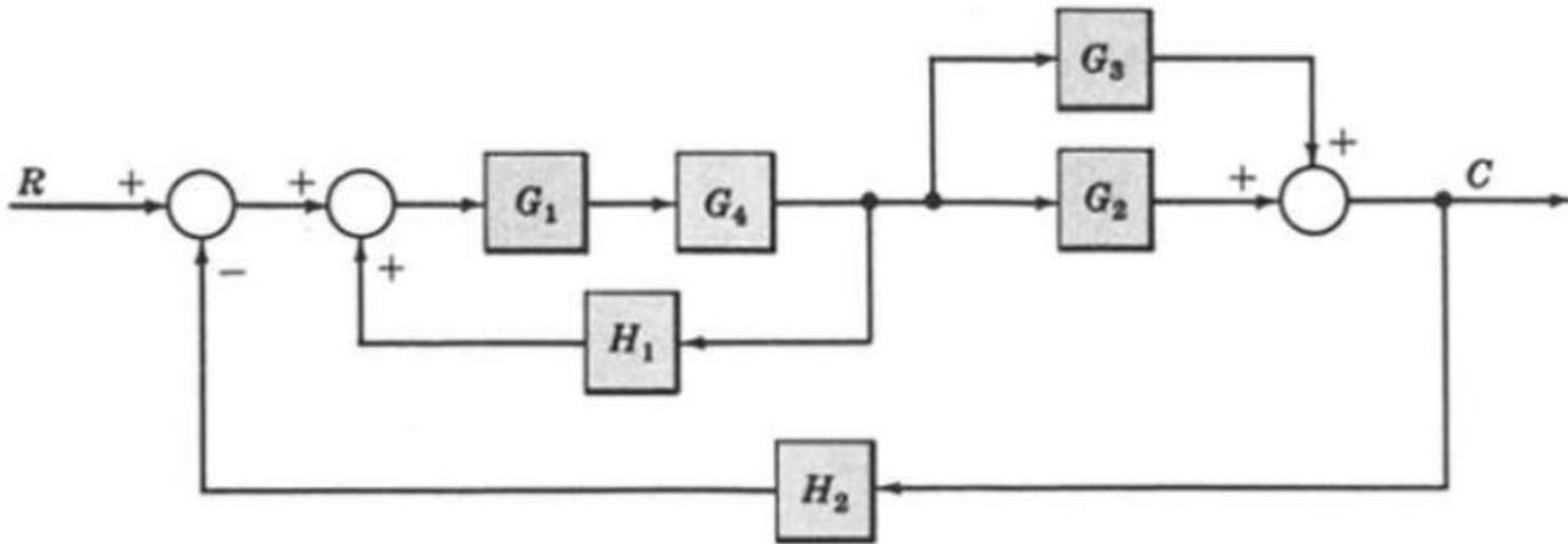
Example-1: Reduce the Block Diagram



Step1: Combine all cascade blocks using Transformation 1.



Example-1: Reduce the Block Diagram



Step2: Combine all parallel blocks using Transformation 2.



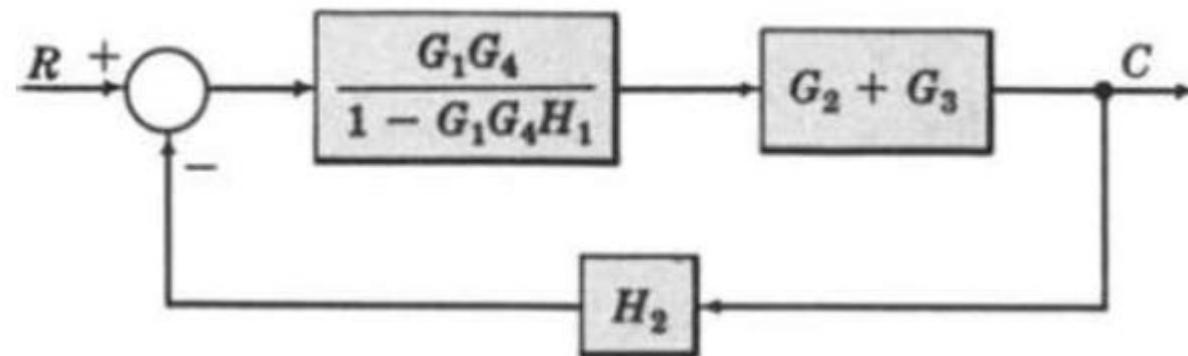
Example-1: Reduce the Block Diagram



Step 3: Eliminate all minor feedback loops using Transformation 7.

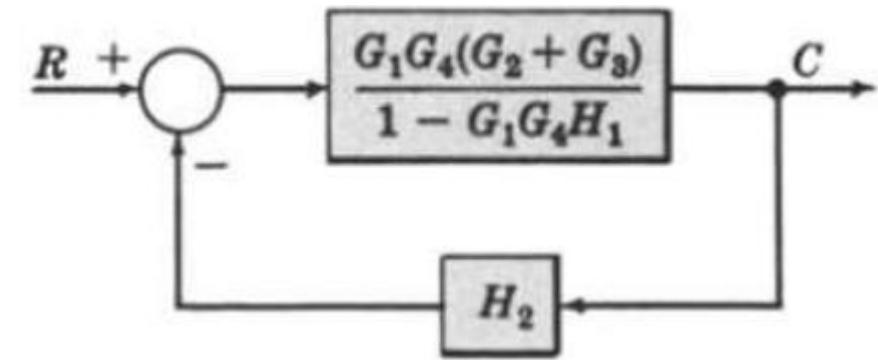
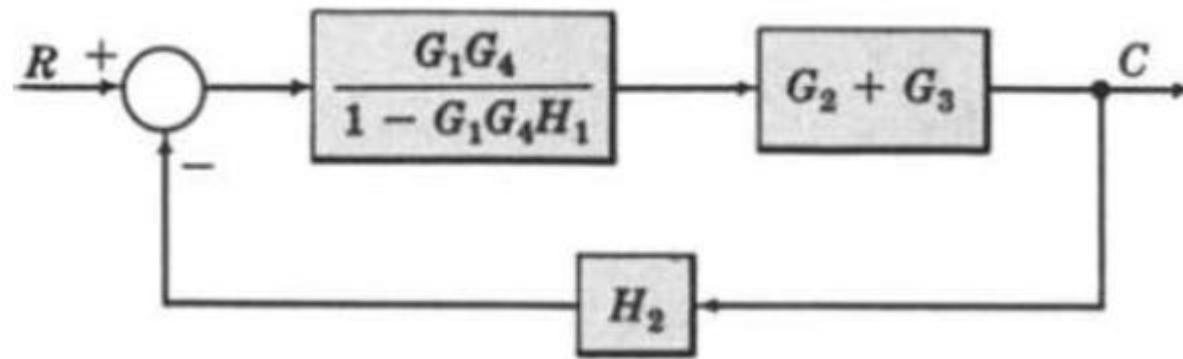


The updated canonical form of the control system

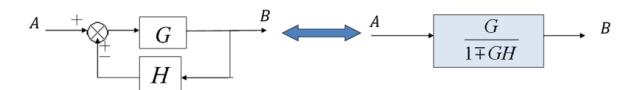


Example-1: Reduce the Block Diagram

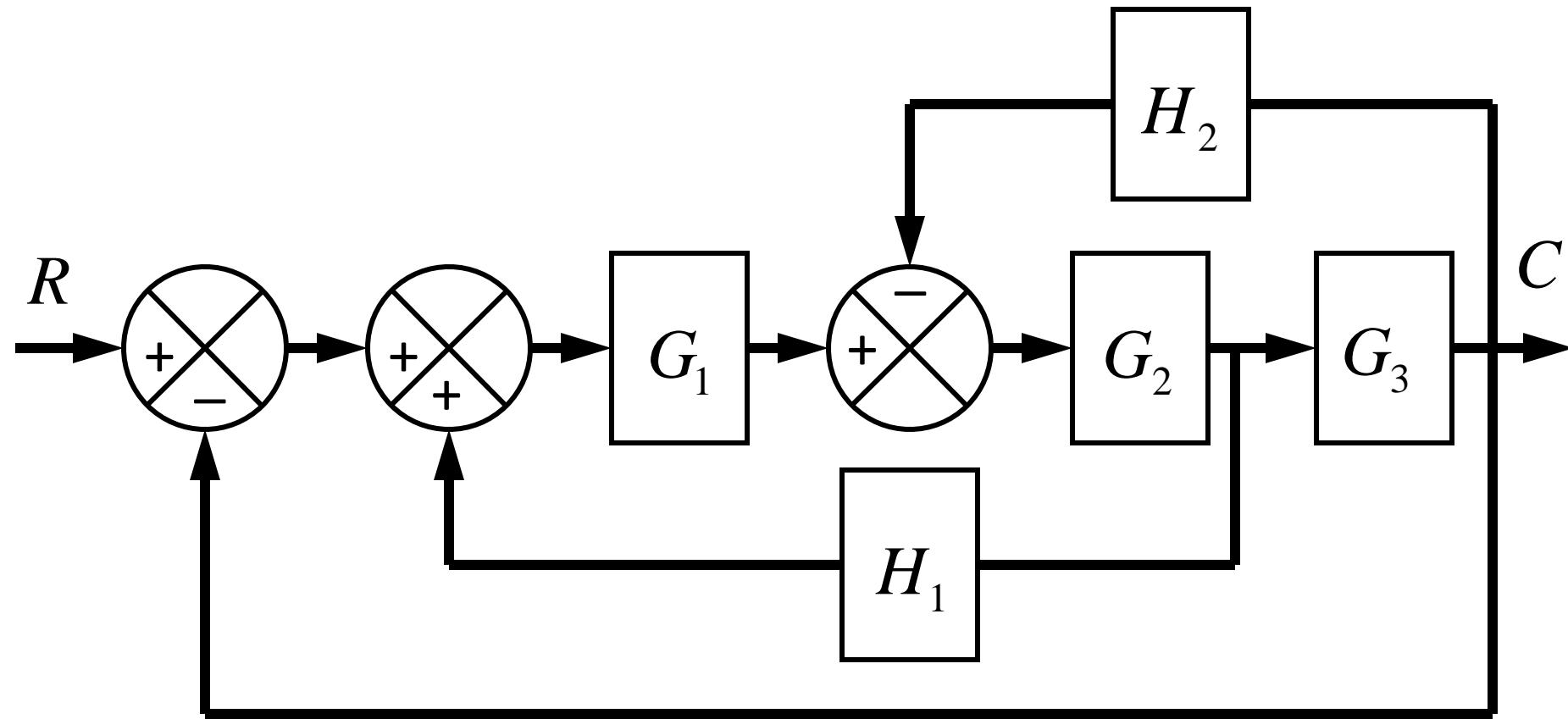
Apply transformation 2 we have



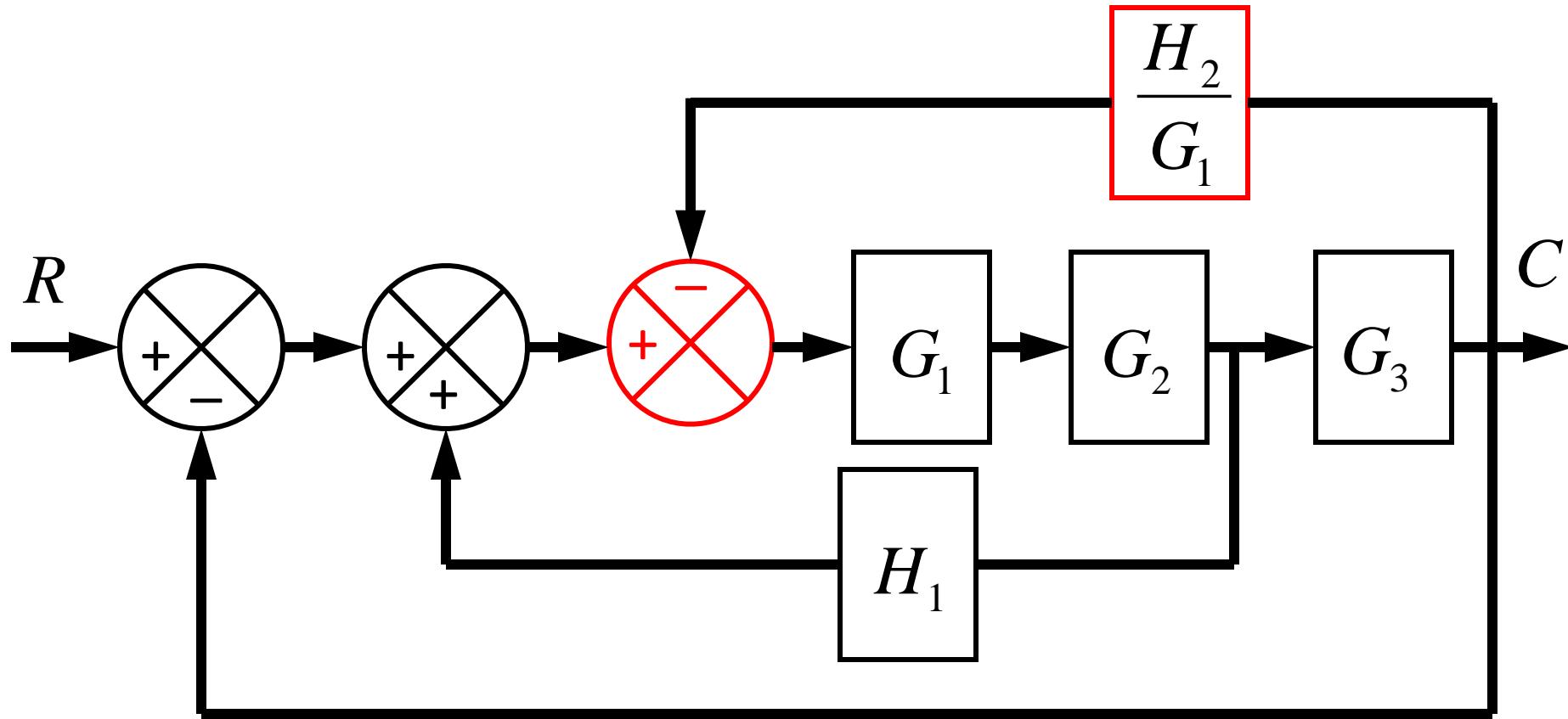
We can further simplify via applying transformation 7



Example-2: Reduce the Block Diagram

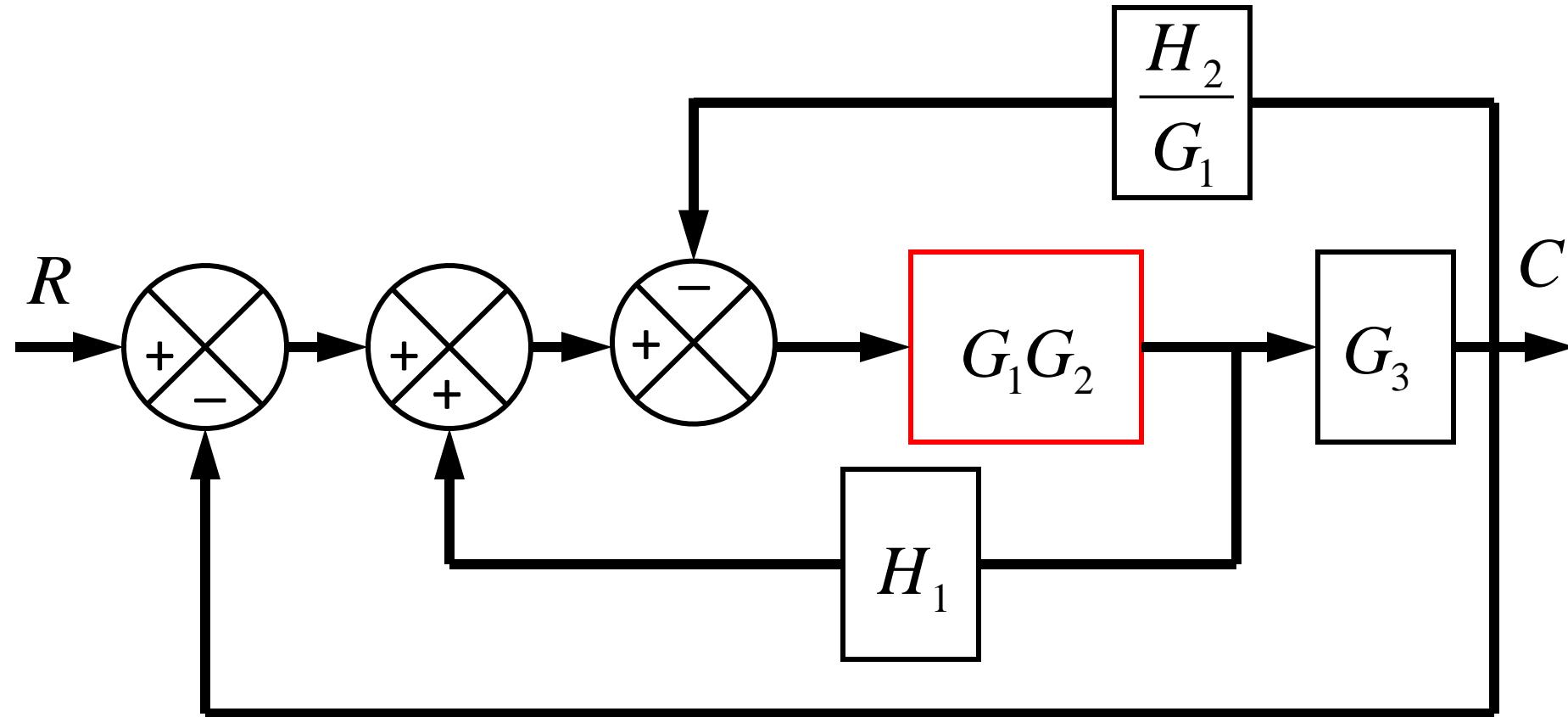


Example-2: Reduce the Block Diagram



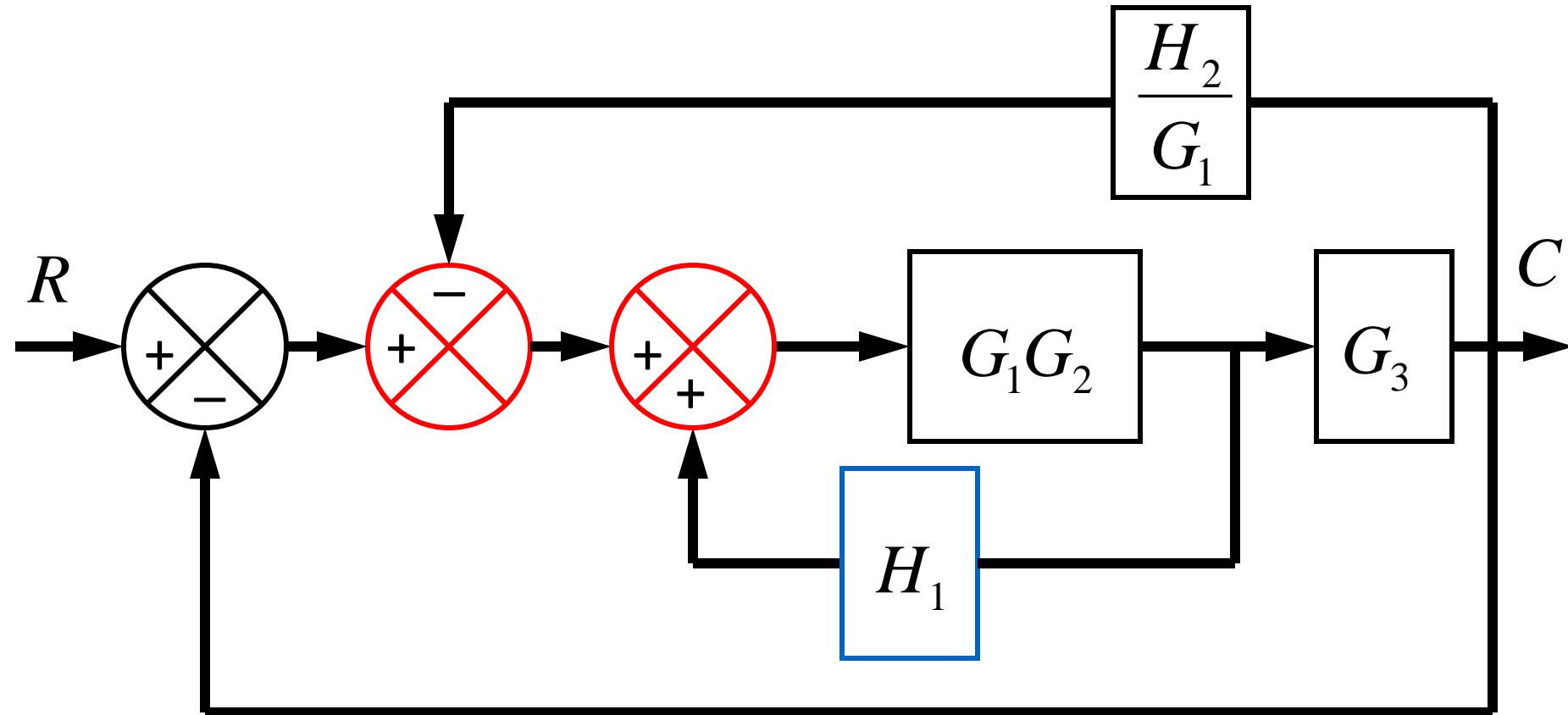
Transformation 4- Moving a summing point ahead of a block

Example-2: Reduce the Block Diagram



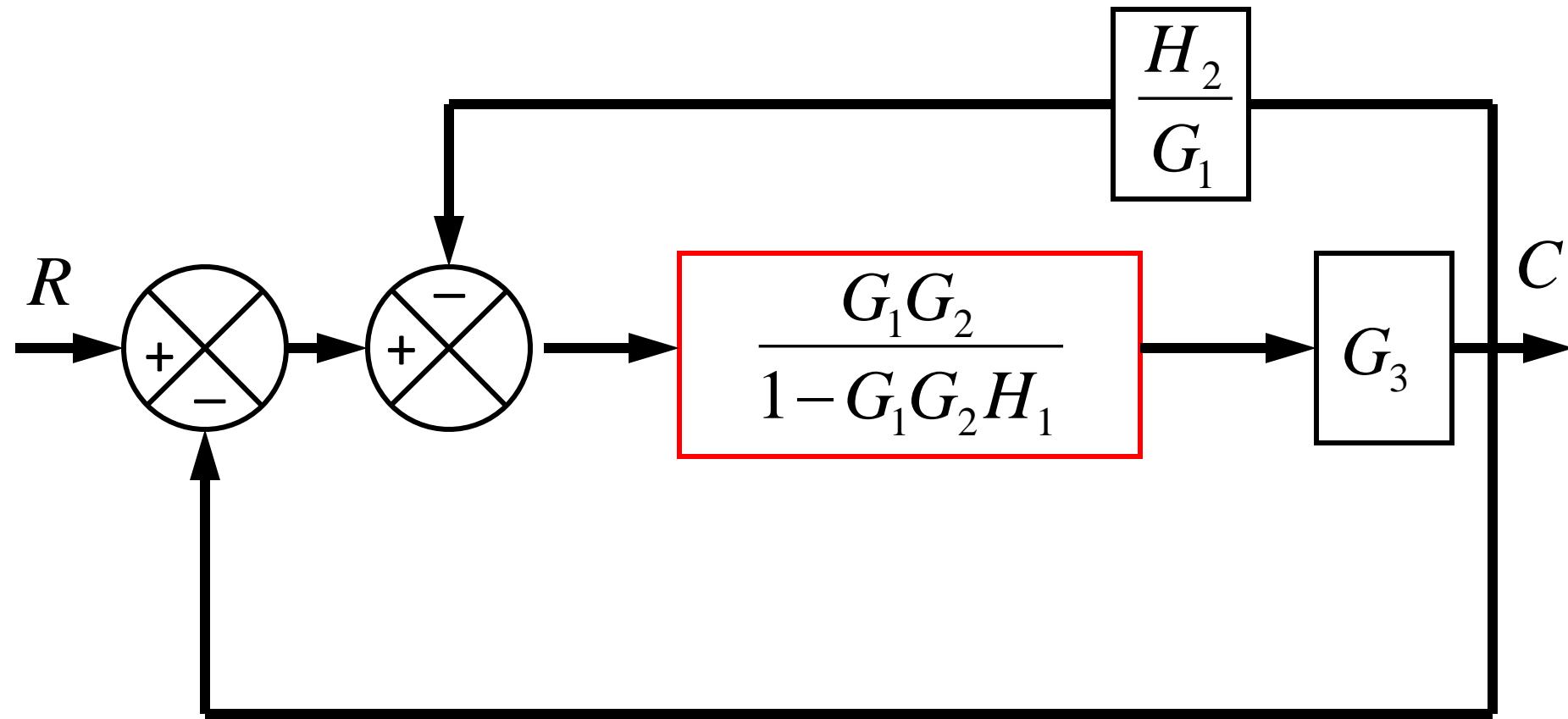
Transformation 1: Combining blocks in cascade

Example-2: Reduce the Block Diagram



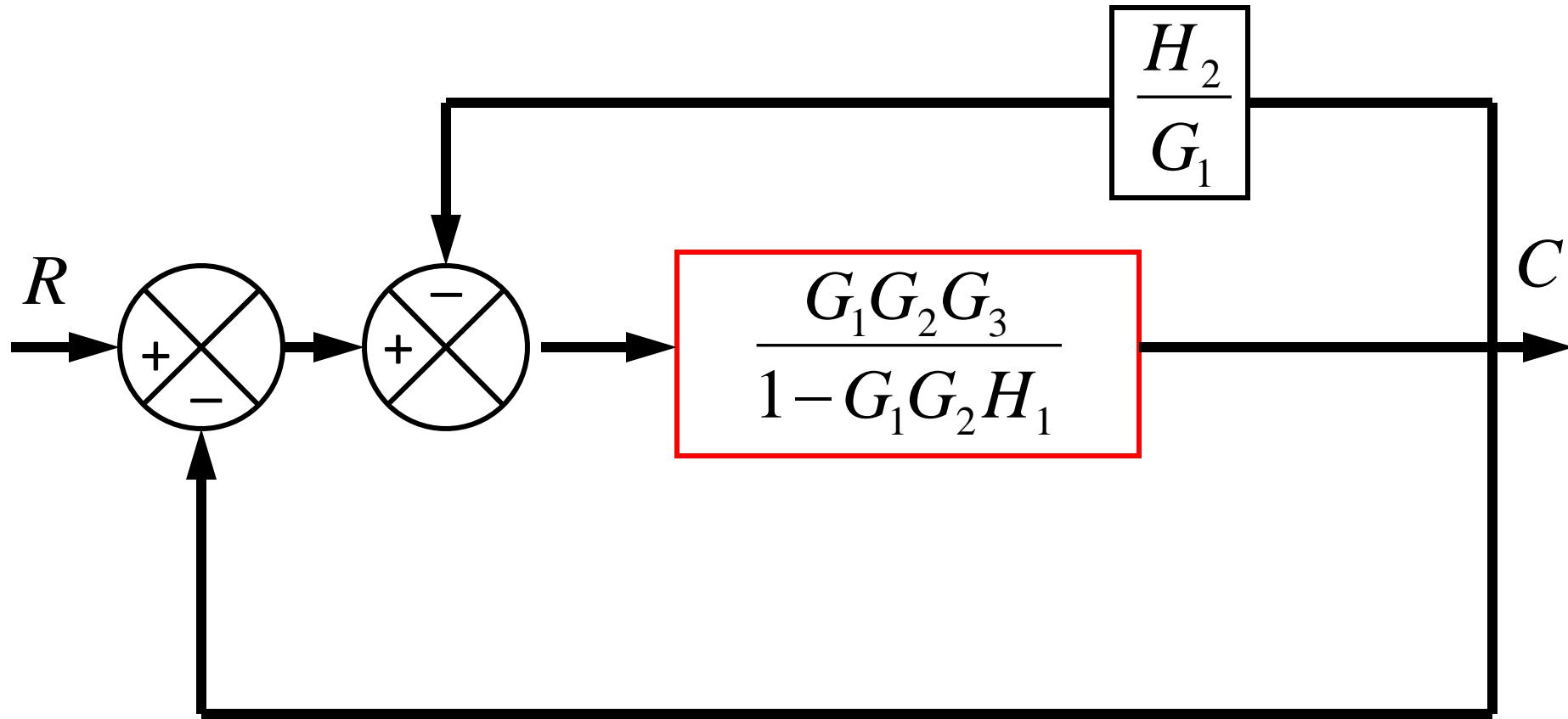
Transformation 8: Swap with two neighboring summing points

Example-2: Reduce the Block Diagram



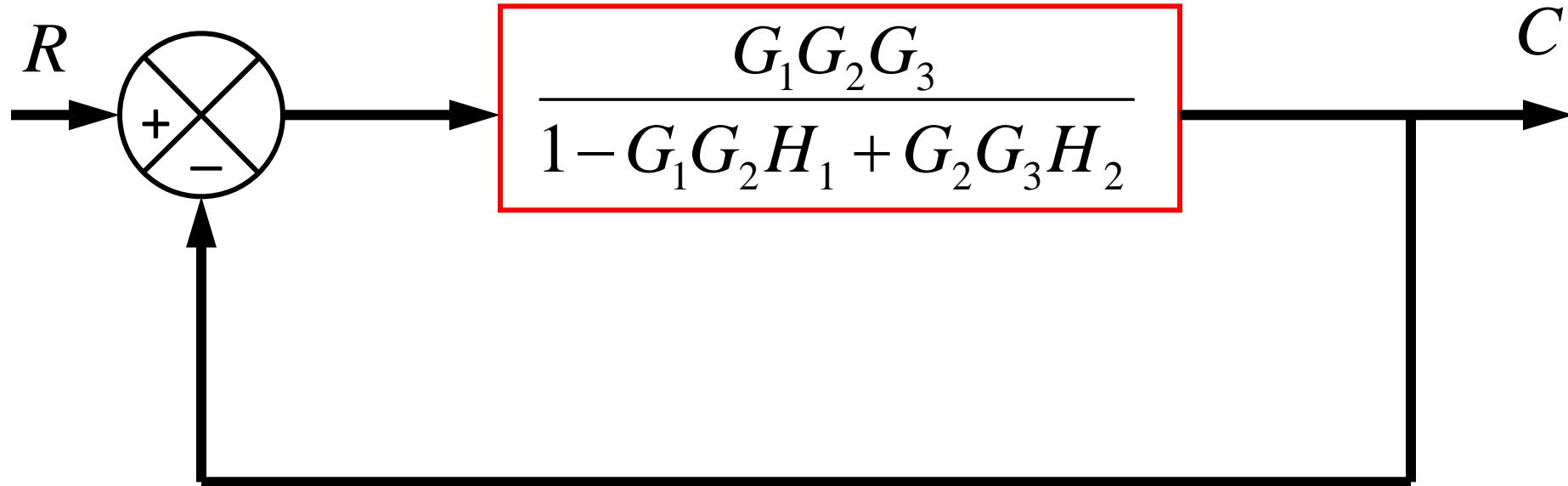
Transformation 7: Eliminate feedback loop

Example-2: Reduce the Block Diagram



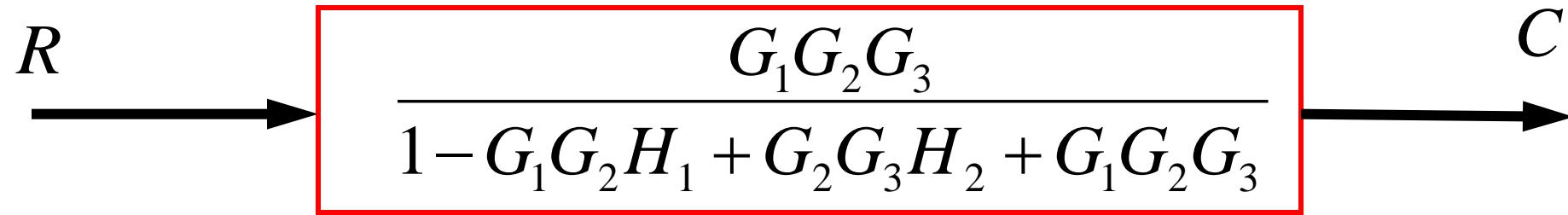
Transformation 1: Combining blocks in cascade

Example-2: Reduce the Block Diagram



Transformation 7: Eliminate feedback loop

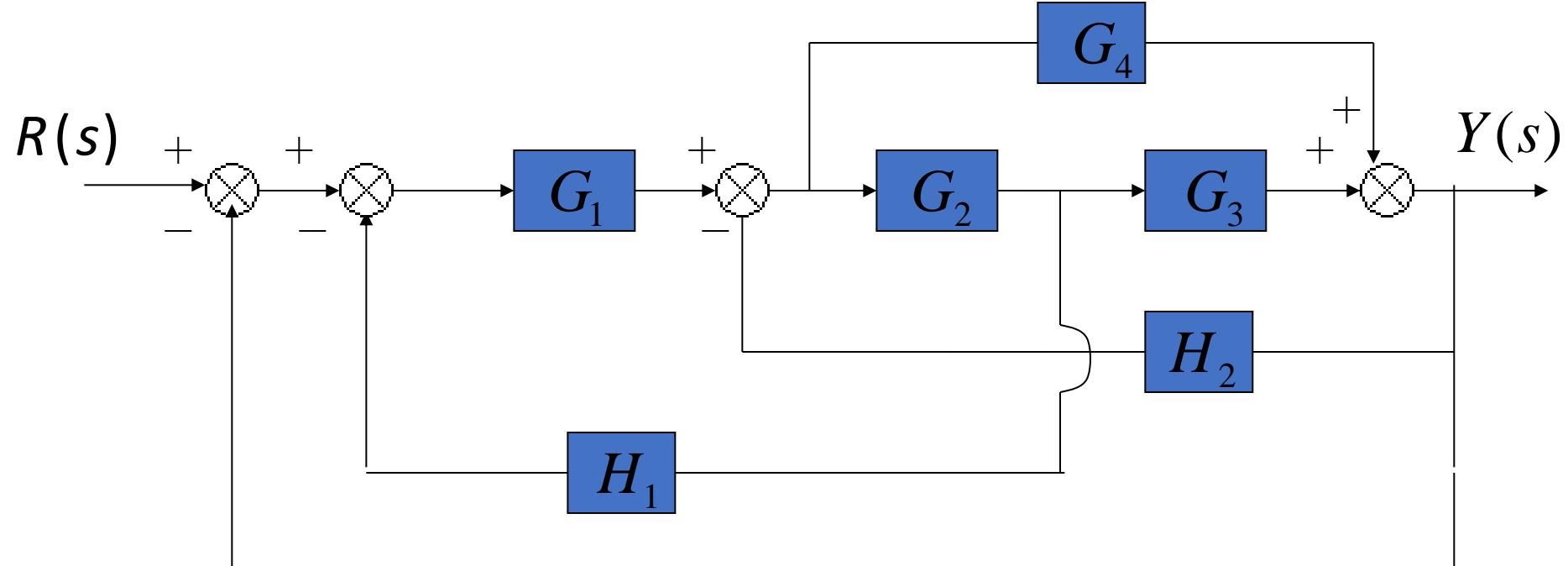
Example-2: Simplify the Block Diagram

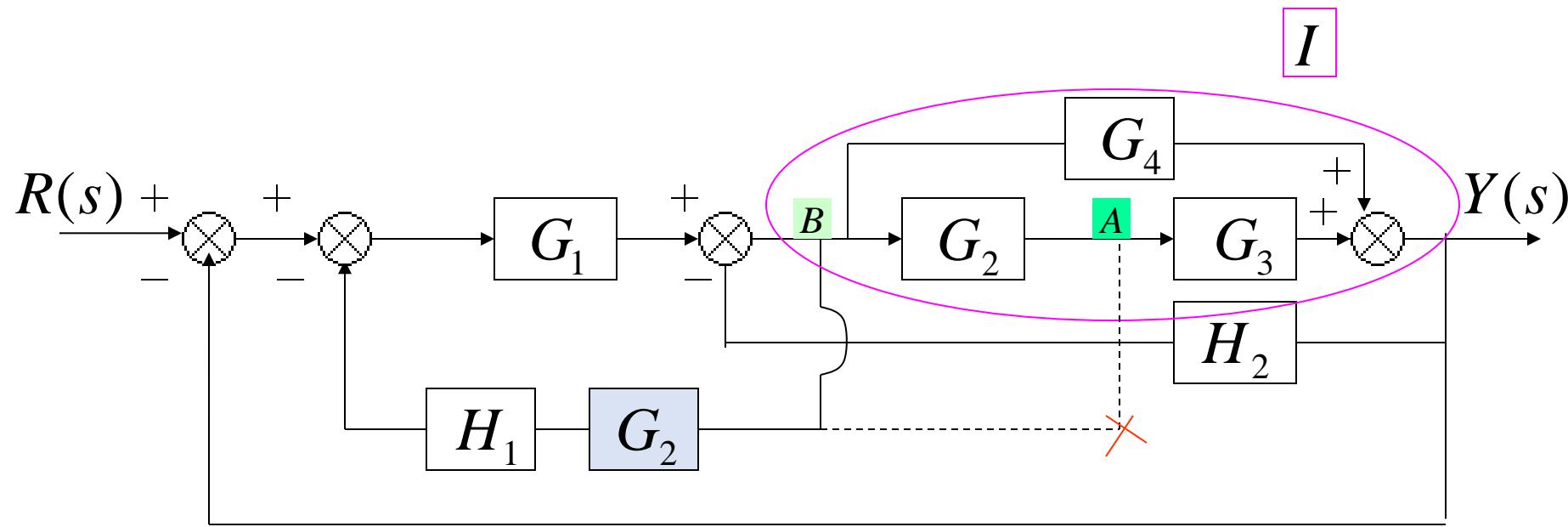


Transformation 7: Eliminate feedback loop

Example-3: Simplify the Block Diagram

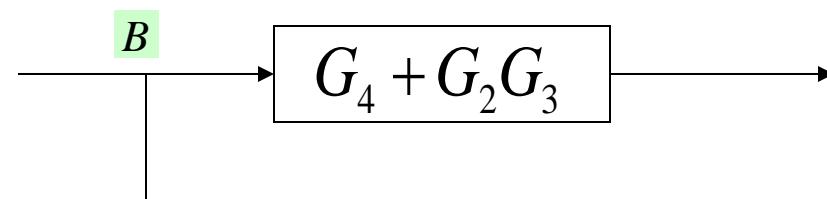
Find the transfer function of the following block diagrams

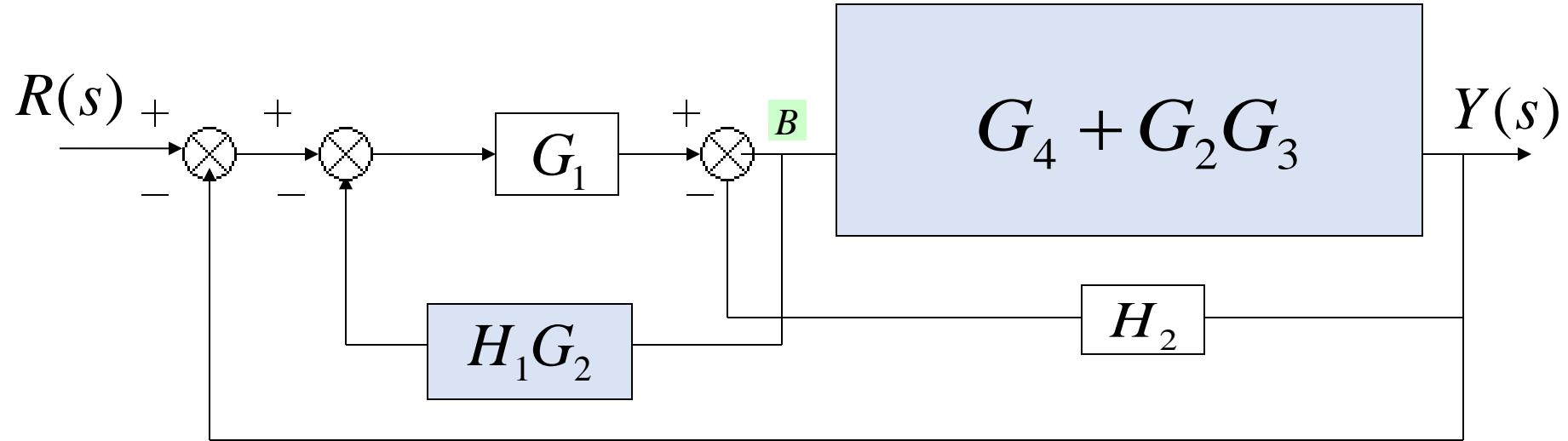




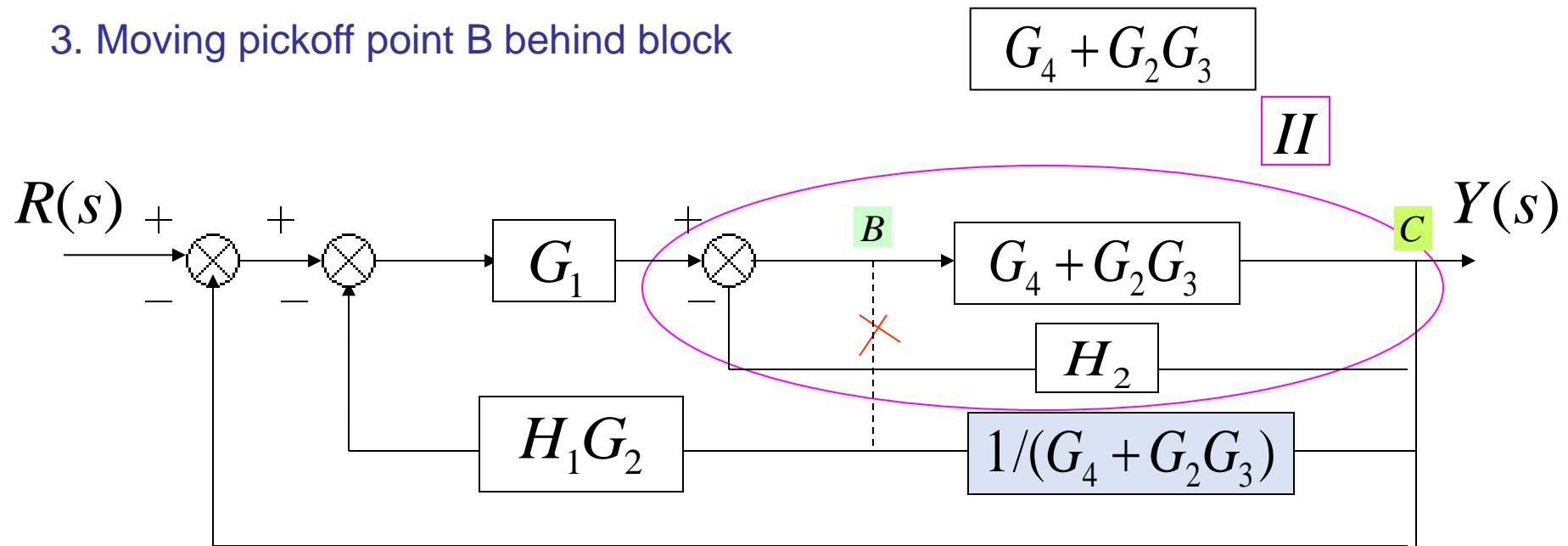
Solution:

1. Moving pickoff point A ahead of block G_2
2. Eliminate loop I & simplify

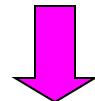
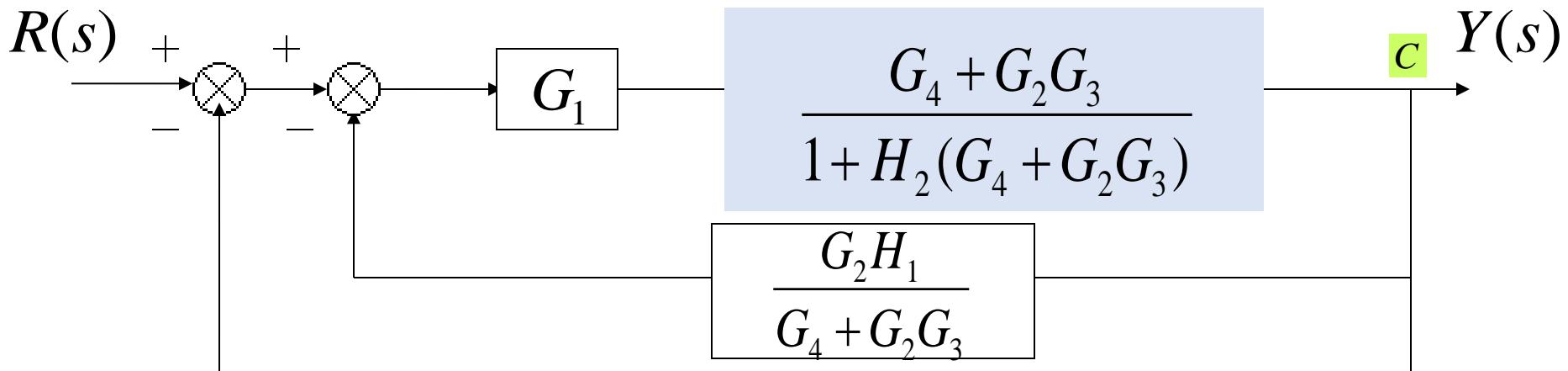




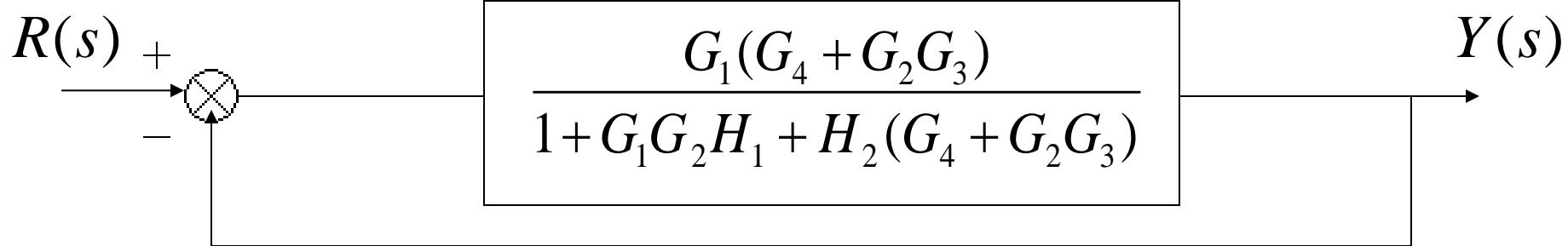
3. Moving pickoff point B behind block



4. Eliminate loop III



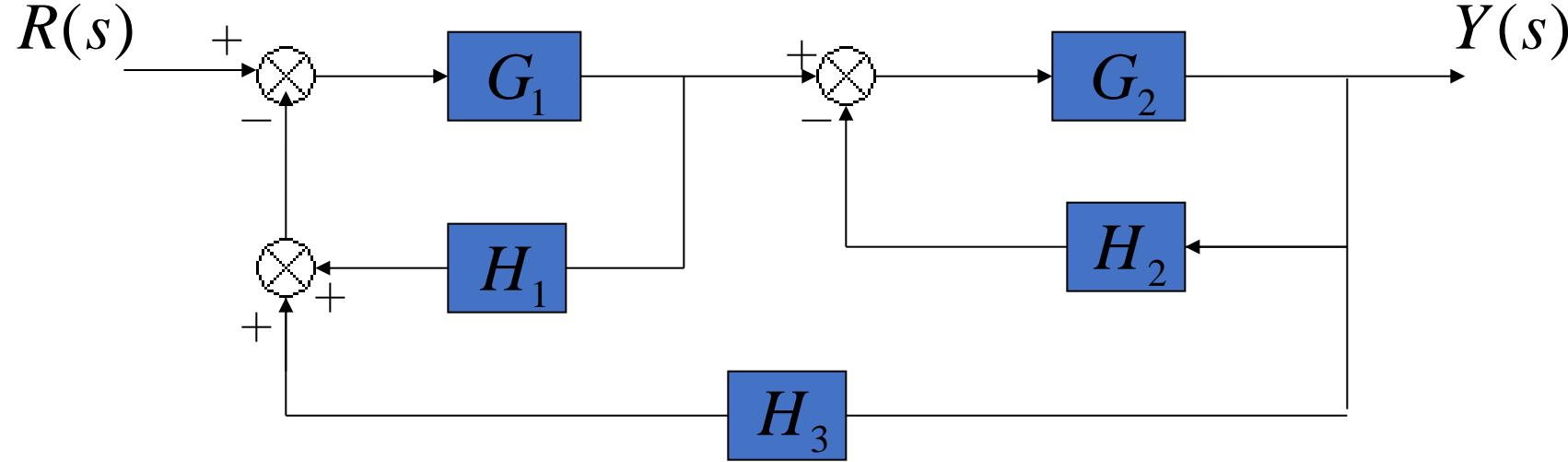
Using Transformation 7



$$T(s) = \frac{Y(s)}{R(s)} = \frac{G_1(G_4 + G_2G_3)}{1 + G_1G_2H_1 + H_2(G_4 + G_2G_3) + G_1(G_4 + G_2G_3)}$$

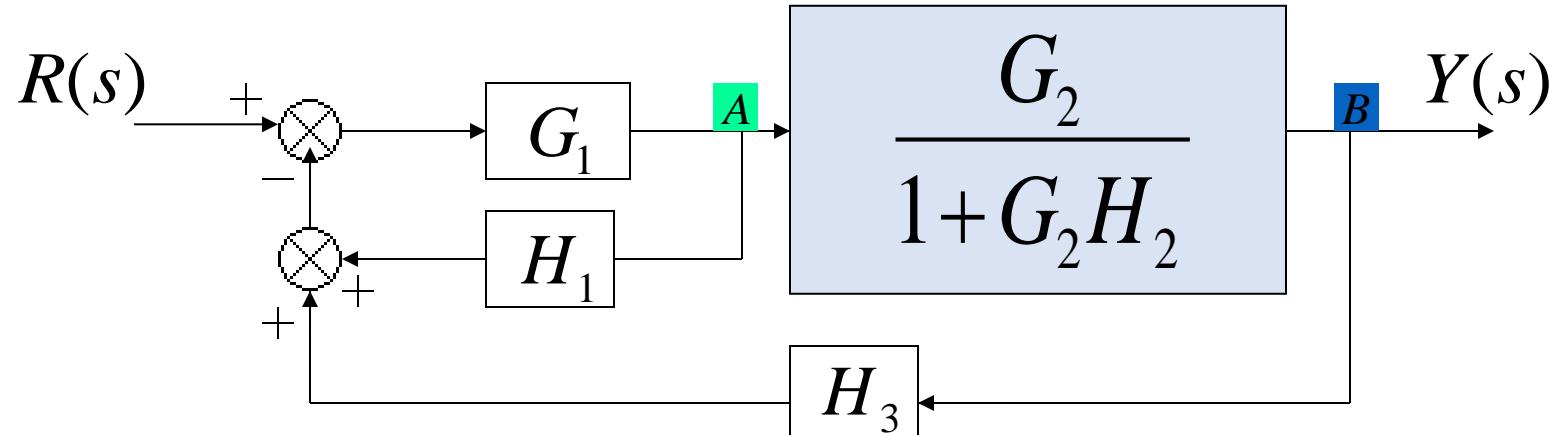
Example-4: Simplify the Block Diagram

Find the transfer function of the following block diagrams



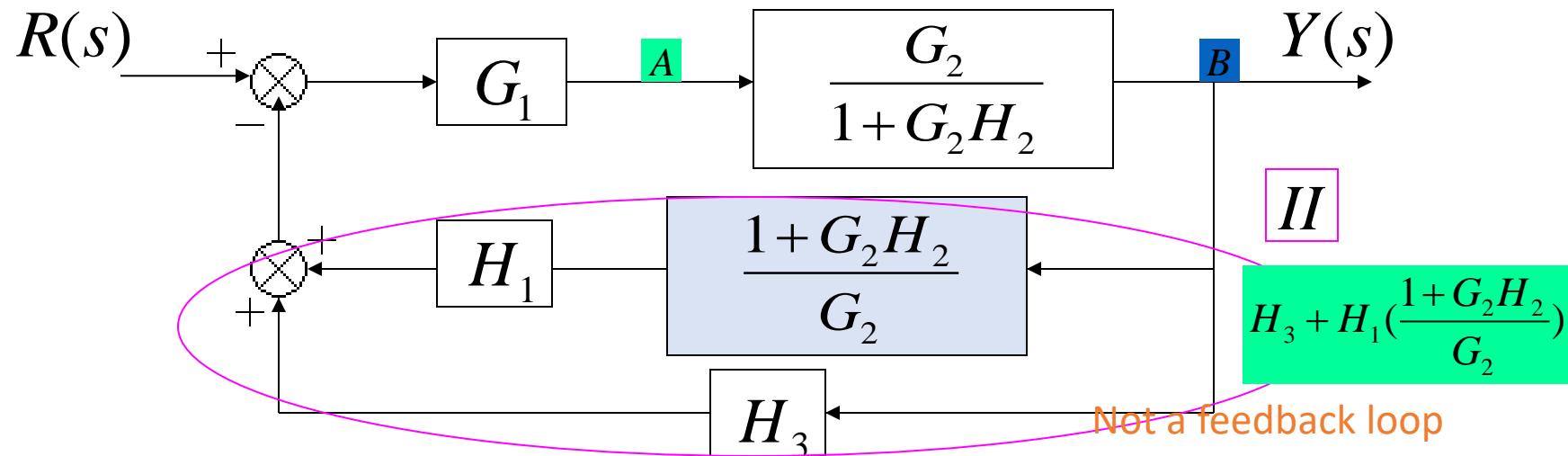
Solution:

1. Eliminate loop I

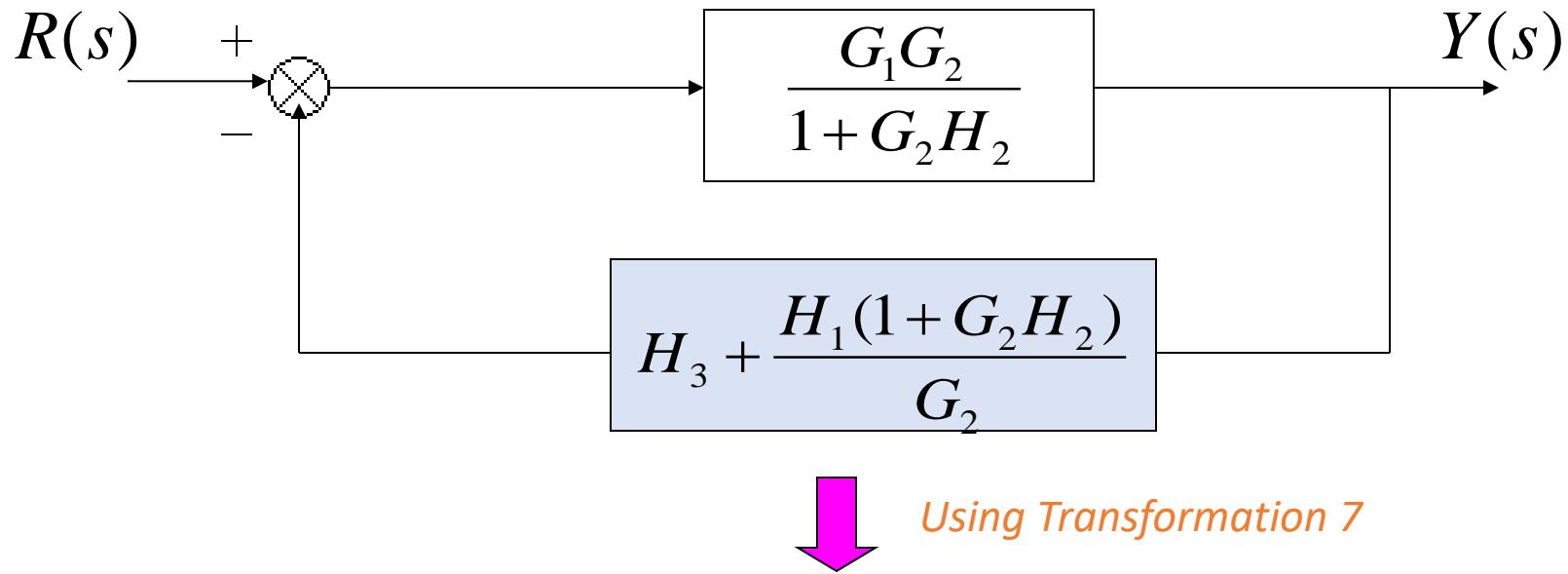


2. Moving pickoff point A behind block

$$\frac{G_2}{1+G_2H_2}$$



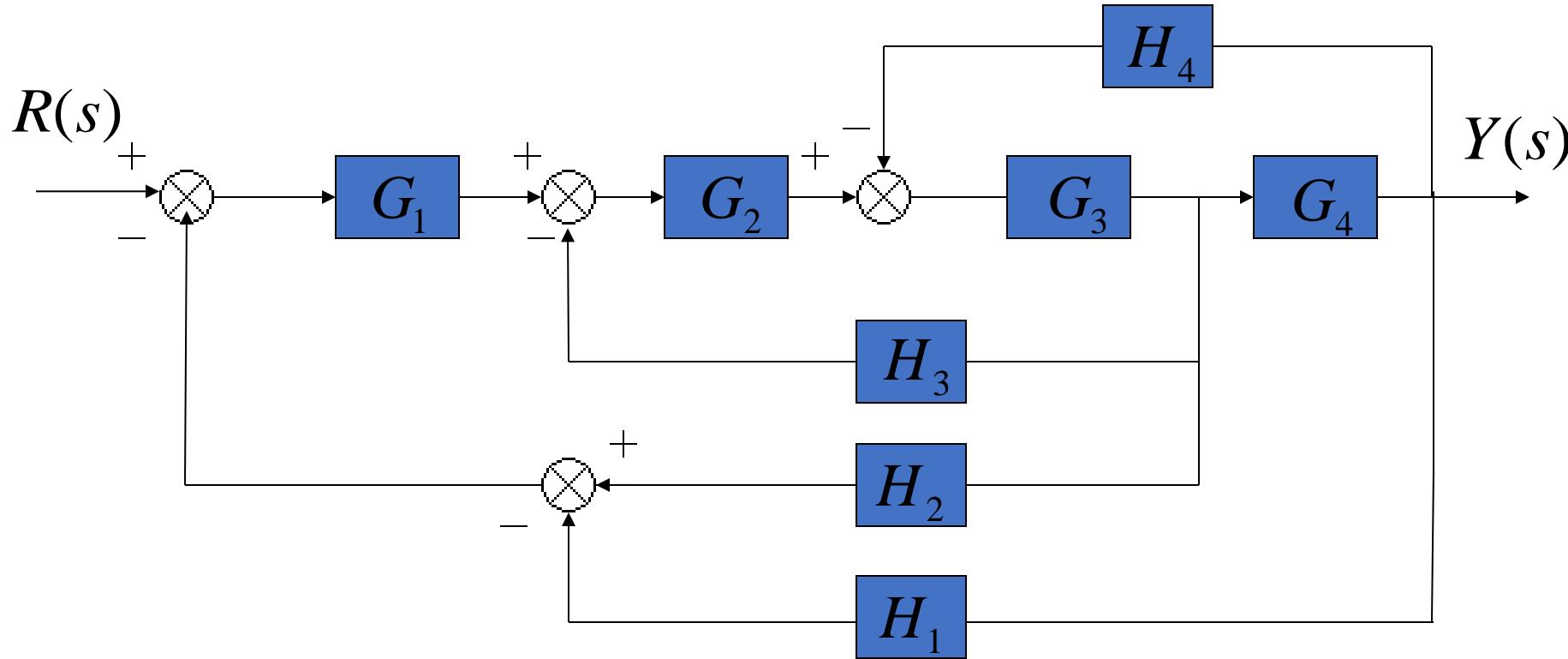
3. Eliminate loop II



$$T(s) = \frac{Y(s)}{R(s)} = \frac{G_1 G_2}{1 + G_2 H_2 + G_1 G_2 H_3 + G_1 H_1 + G_1 G_2 H_1 H_2}$$

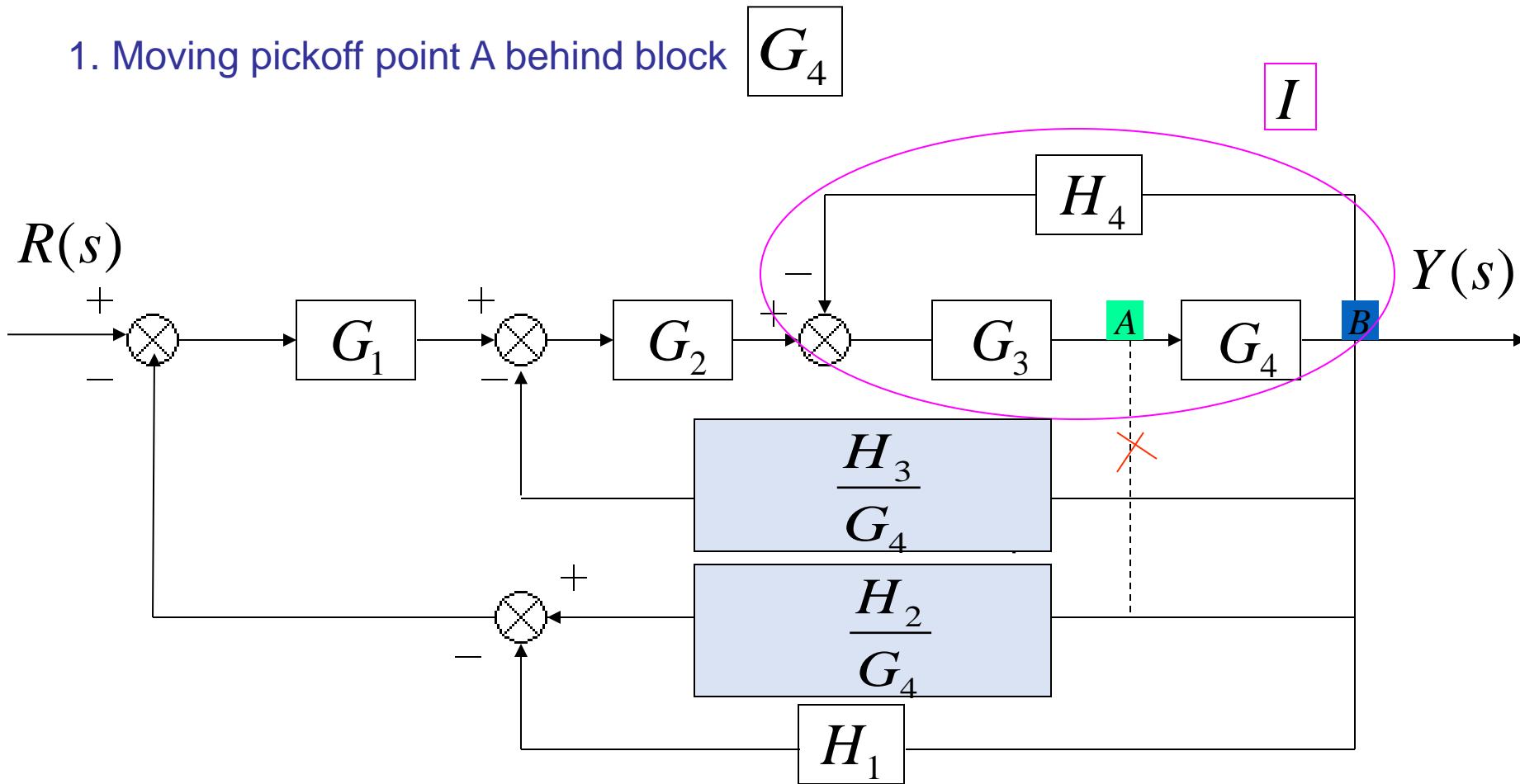
Example-5: Simplify the Block Diagram

Find the transfer function of the following block diagrams

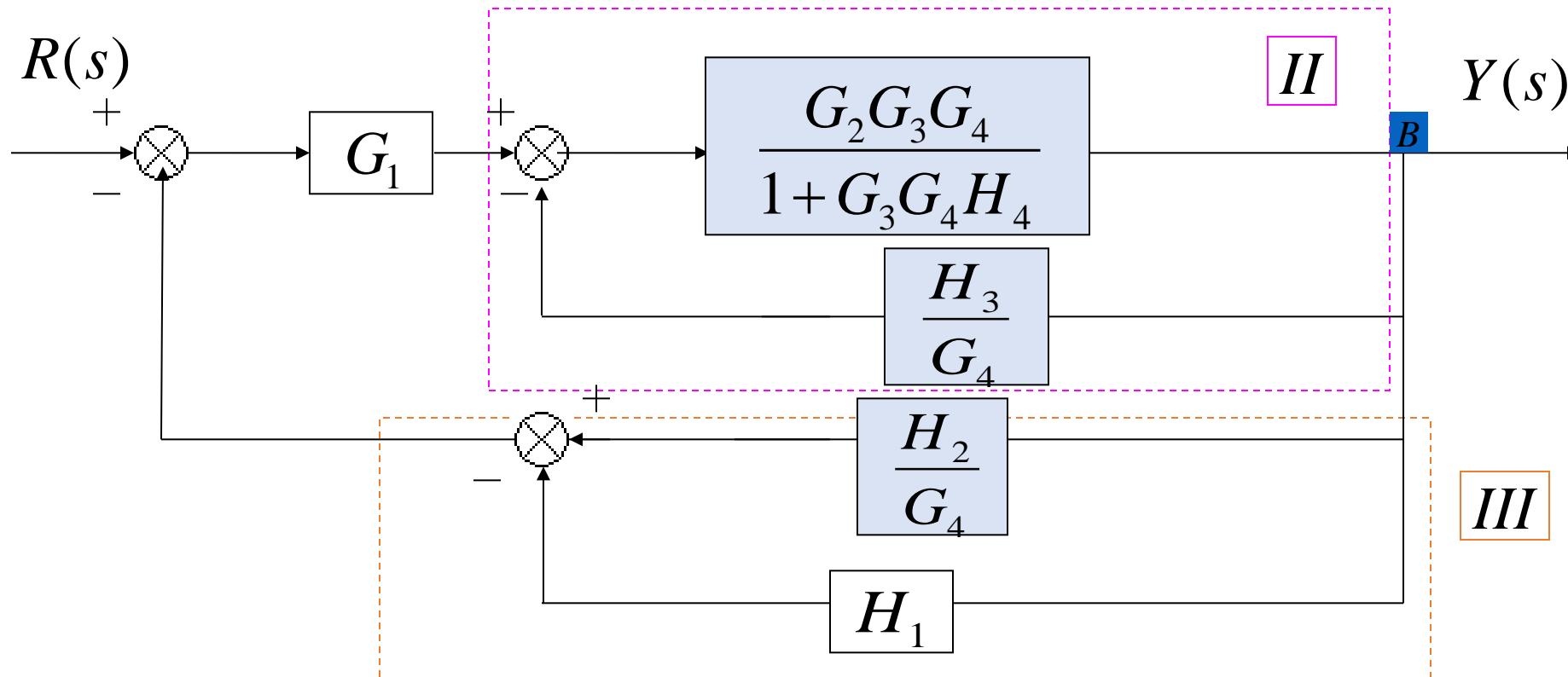


Solution:

1. Moving pickoff point A behind block



2. Eliminate loop I and Simplify



II



feedback

$$\frac{G_2 G_3 G_4}{1 + G_3 G_4 H_4 + G_2 G_3 H_3}$$

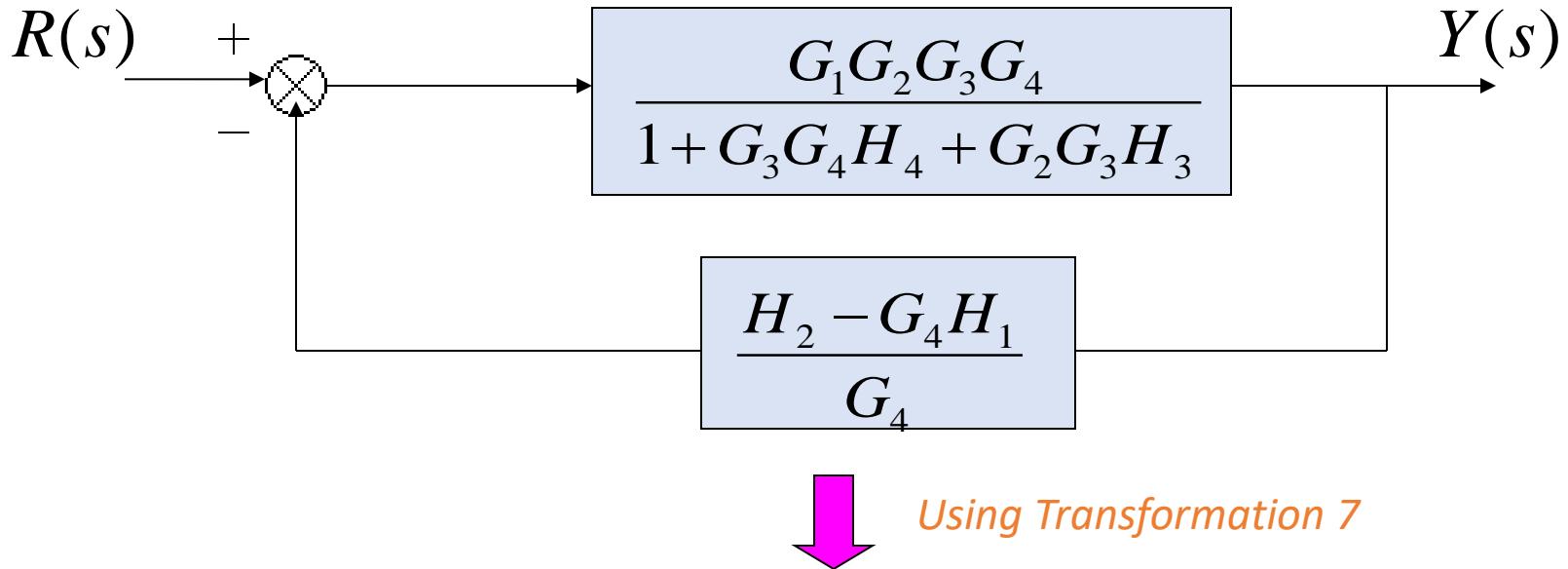
III



Not feedback

$$\frac{H_2 - G_4 H_1}{G_4}$$

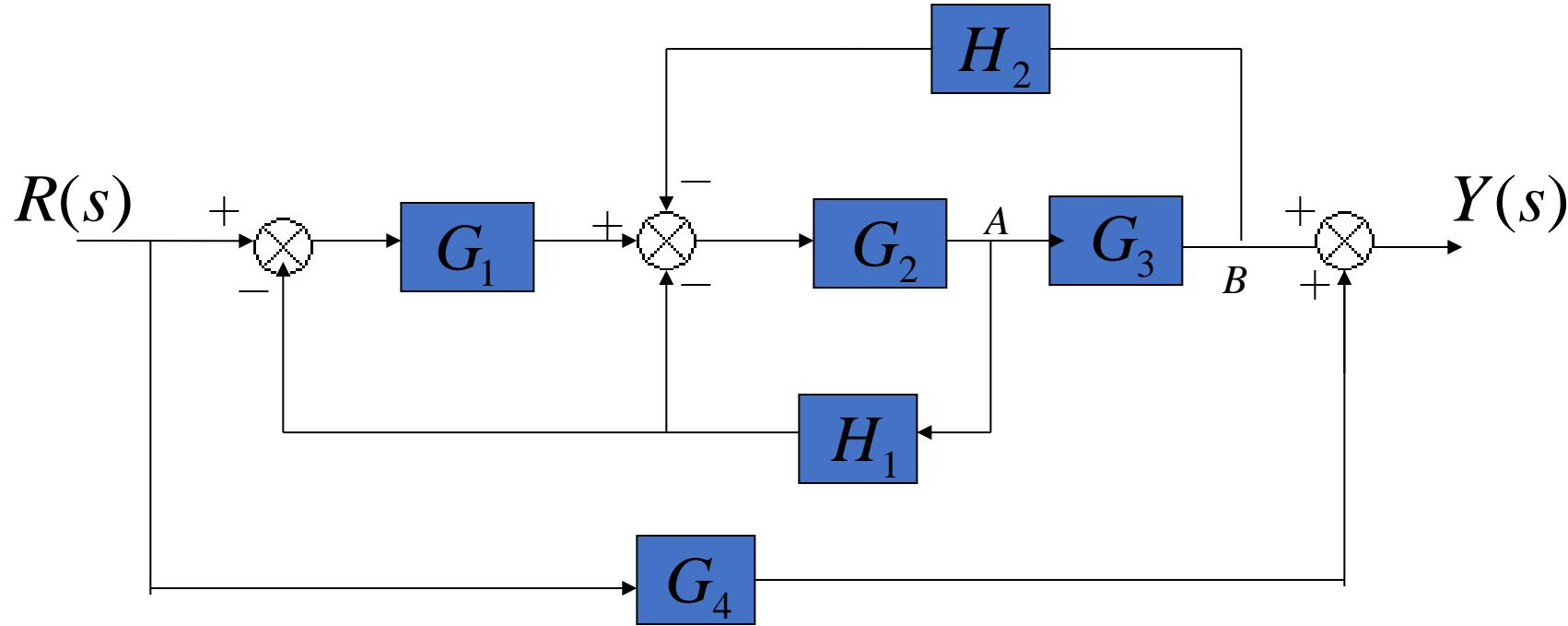
3. Eliminate loop II & III



$$T(s) = \frac{Y(s)}{R(s)} = \frac{G_1 G_2 G_3 G_4}{1 + G_2 G_3 H_3 + G_3 G_4 H_4 + G_1 G_2 G_3 H_2 - G_1 G_2 G_3 G_4 H_1}$$

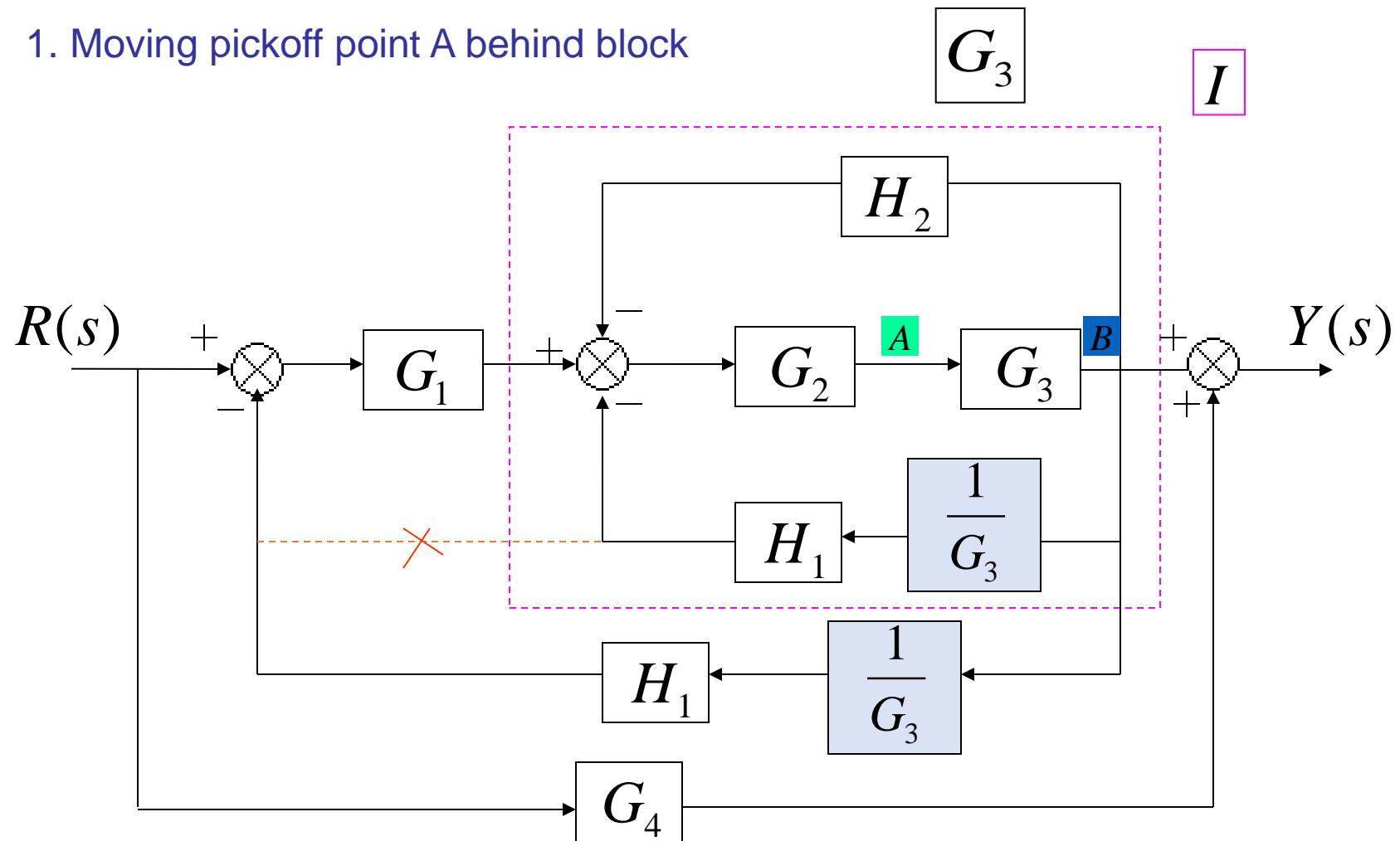
Example-6: Simplify the Block Diagram

Find the transfer function of the following block diagrams

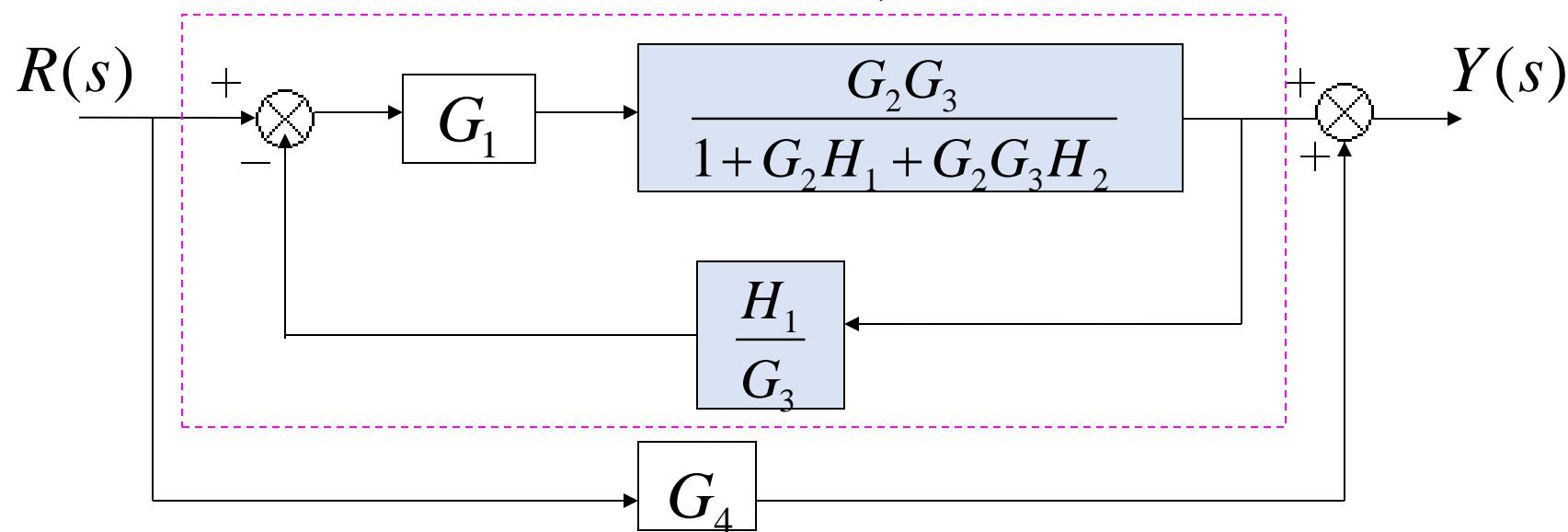
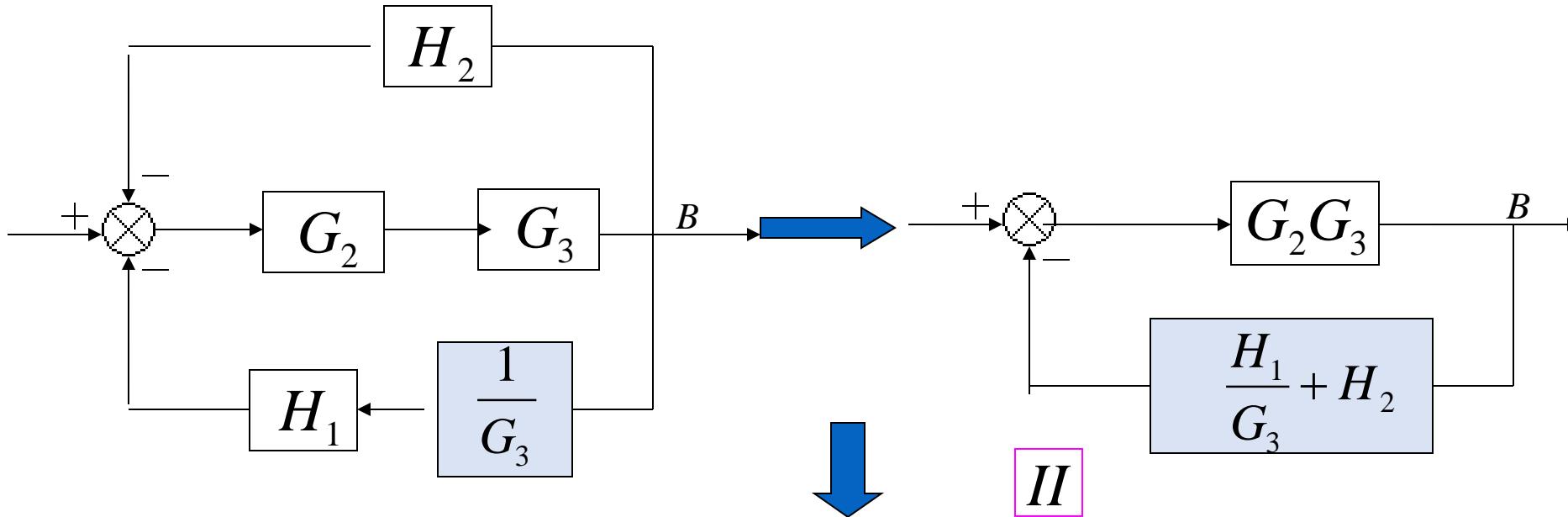


Solution:

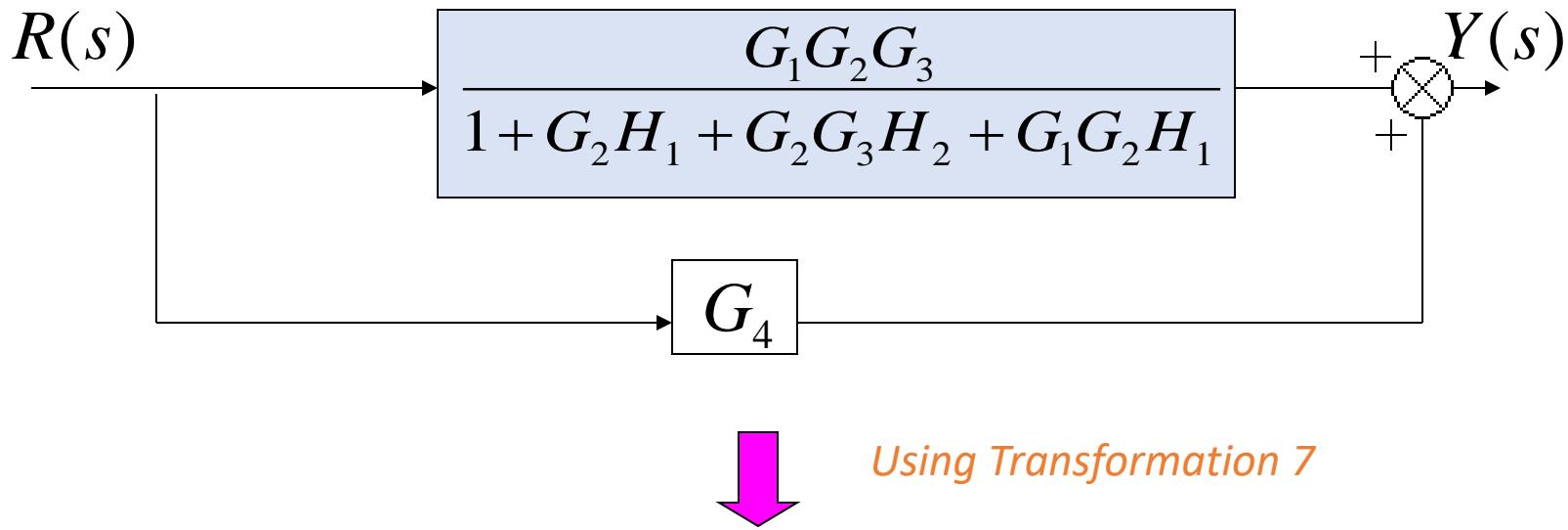
1. Moving pickoff point A behind block



2. Eliminate loop I & Simplify



3. Eliminate loop II



$$T(s) = \frac{Y(s)}{R(s)} = G_4 + \frac{G_1 G_2 G_3}{1 + G_2 H_1 + G_2 G_3 H_2 + G_1 G_2 H_1}$$

Summary

- Block Diagram of System
- Reduction techniques of block diagram of a system
- Examples

IDAT7215

Computer Programming for Product Development and Applications

Lecture 3-1: Python Application in Control System

Dr. Zulfiqar Ali

Outline

- Installing Python-Control library in the Anaconda Distribution
- Fundamentals of Control System
- Classification of Control Systems
- Mathematical Models for Control Systems
- Types of Controllers for Control System
- Stability Analysis of Control System
- Summary

Installing Python-Control library in the Anaconda Distribution

How to Install Control Library in Anaconda Distribution

- NumPy, SciPy, matplotlib need to be installed in order to use control library.

The screenshot shows a web browser displaying the Python Control Systems Library documentation at <https://python-control.readthedocs.io/en/0.9.1/intro.html>. The page header includes a back button, forward button, refresh button, a lock icon, and the URL. Below the header is a navigation bar with links to City University, BOOK DOWNLOADS, conference, educational websites, Conferences IGA, Reminder_links, HKU, Research IGA, and Google Scholar. The main content area has a blue header "Python Control Systems Library 0.9.1" with a "Search docs" input field. A sidebar on the left lists various sections: Introduction, Overview of the toolbox, Some differences from MATLAB, Installation, Getting started, Library conventions, Function reference, Control system classes, MATLAB compatibility module, Differentially flat systems, Input/output systems, Describing functions, Optimal control, and Examples. The main text discusses the slycot dependency and provides command-line installation instructions:

Many parts of `python-control` will work without `slycot`, but some functionality is limited or absent, and installation of `slycot` is recommended. Users can check to insure that `slycot` is installed correctly by running the command:

```
python -c "import slycot"
```

and verifying that no error message appears. More information on the `slycot` package can be obtained from the [slycot project page](#).

For users with the Anaconda distribution of Python, the following commands can be used:

```
conda install numpy scipy matplotlib # if not yet installed  
conda install -c conda-forge control slycot
```

This installs `slycot` and `python-control` from `conda-forge`, including the `openblas` package. A red arrow points from the word "control" in the first command to the "control" section in the sidebar.

Alternatively, to use setuptools, first [download the source](#) and unpack it. To install in your home directory, use:

```
python setup.py install --user
```

or to install for all users (on Linux or Mac OS):

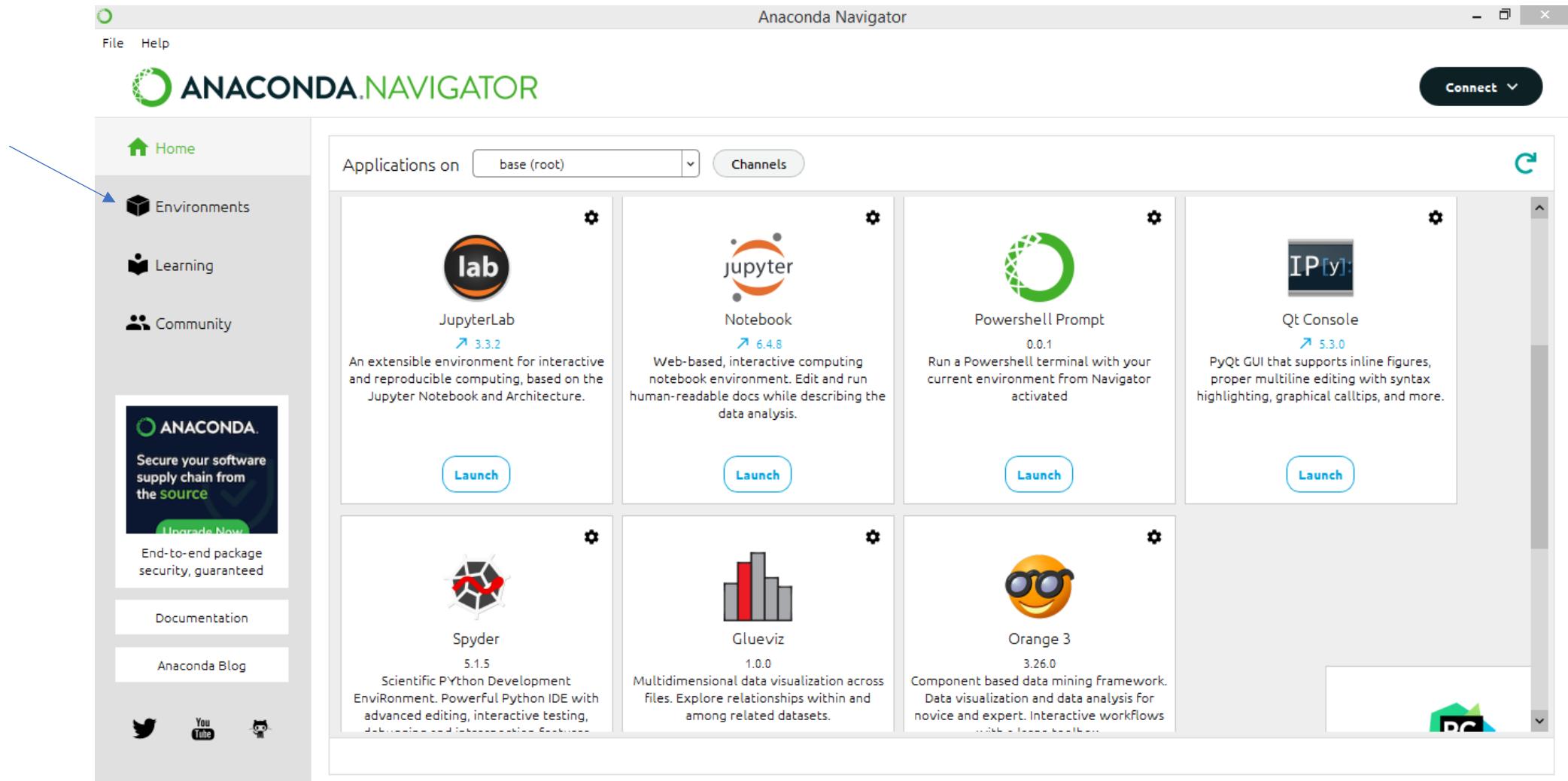
```
python setup.py build  
sudo python setup.py install
```

Getting started

There are two different ways to use the package. For the default interface described in [Function reference](#), simply import the control package as follows:

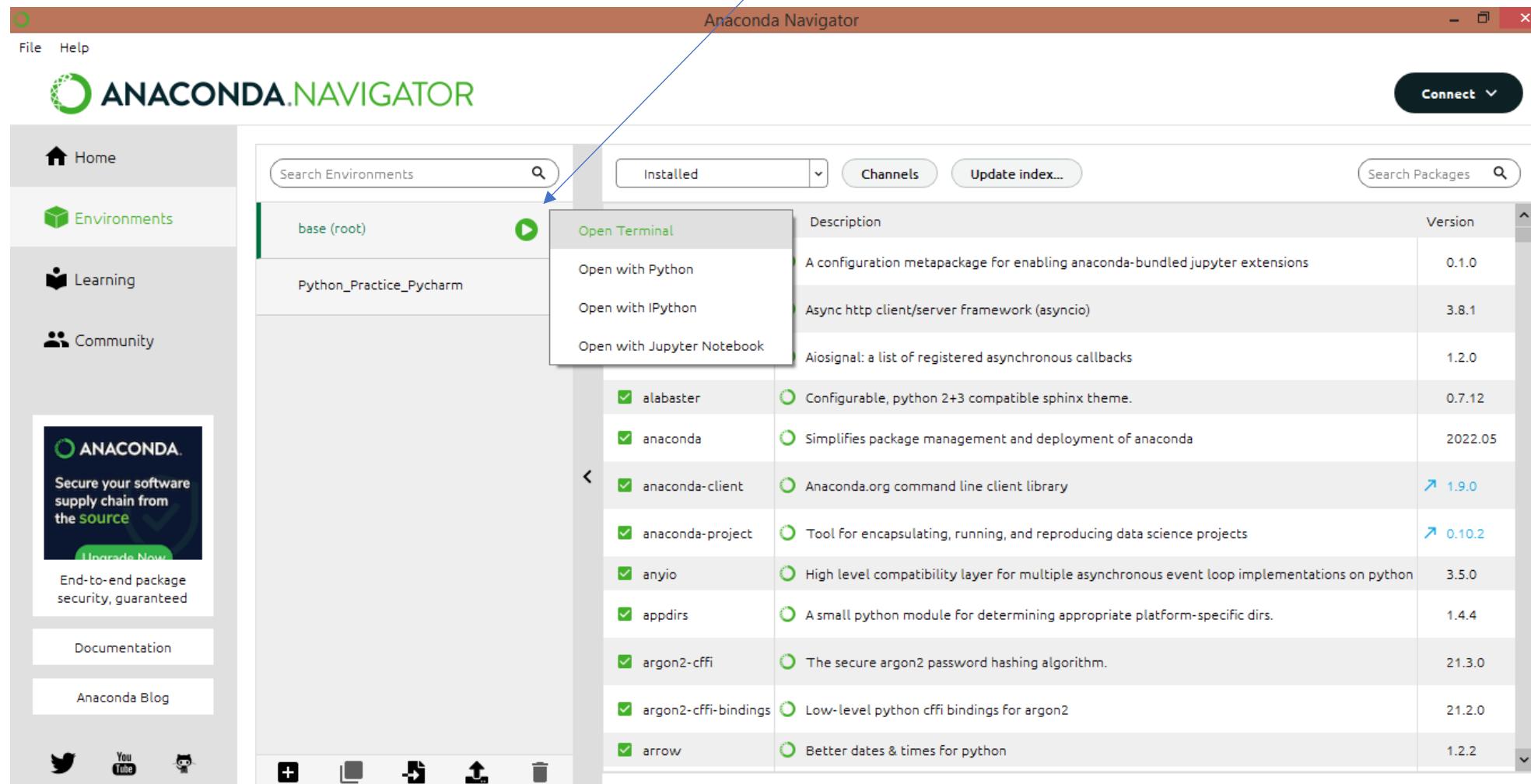
<https://python-control.readthedocs.io/en/0.9.1/intro.html>

How to Install Control Library in Anaconda Distribution



How to Install Control Library in Anaconda Distribution

Open Terminal



How to Install Control Library in Anaconda Distribution



```
base> C:\Users\itsme_000>conda install -c conda-forge control slycot
```

```
base> C:\Users\itsme_000>conda install -c conda-forge control slycot
Collecting package metadata (current_repodata.json): done
Solving environment: done
# All requested packages already installed.

(base) C:\Users\itsme_000>
```

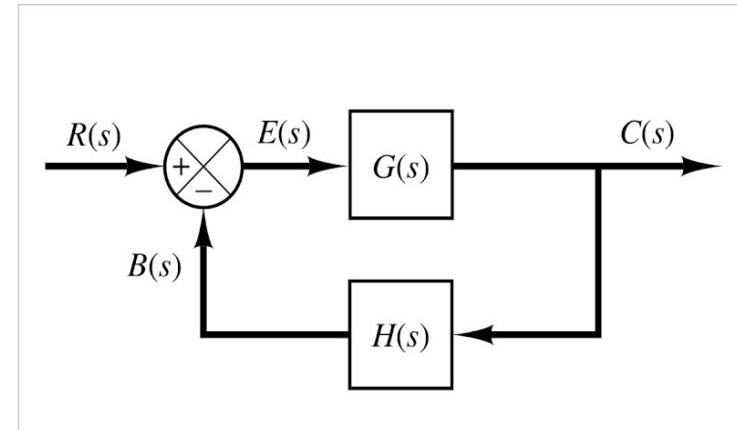
Overview of the Python-Control library

Name	Description
Lti(*system)	Continuous-time linear time invariant system base class
StateSpace(*system,**kwargs)	Linear Time Invariant system in state-space form.
TransferFunction(*system,**kwargs)	Linear Time Invariant system in transfer function form.
ZerosPolesGain(*system,**kwargs)	Linear Time Invariant system in zeros, poles, gain form.
Lsim(system, U,T[,X0,interp])	Simulate output of a continuous-time linear system.
Lsim2(system, U,T[,X0,interp])	Simulate output of a continuous-time linear system, by using the ODE solver <code>scipy.integrate.odeint</code> .
Impulse(system[,X0,T,N])	Impulse response of continuous-time system.
Impulse2(system[,X0,T,N])	Impulse response of single-input, continuous-time linear system.
step(system[,X0,T,N])	Step response of continuous-time system.
step2(system[,X0,T,N])	Step response of continuous-time system.
Freqresp(system[,w,n])	Calculate the frequency response of a continuous-time system.
Bode(system[,w,n])	Calculate Bode magnitude and phase data of a continuous-time system.

Overview of the Python-Control library

- The **python-control package** is a set of python classes and functions that implement common operations for the analysis and **design of feedback control system**.
- A MATLAB compatibility module is available that provides many of the common functions corresponding to the commands available in the **MATLAB Control Systems Toolbox**.

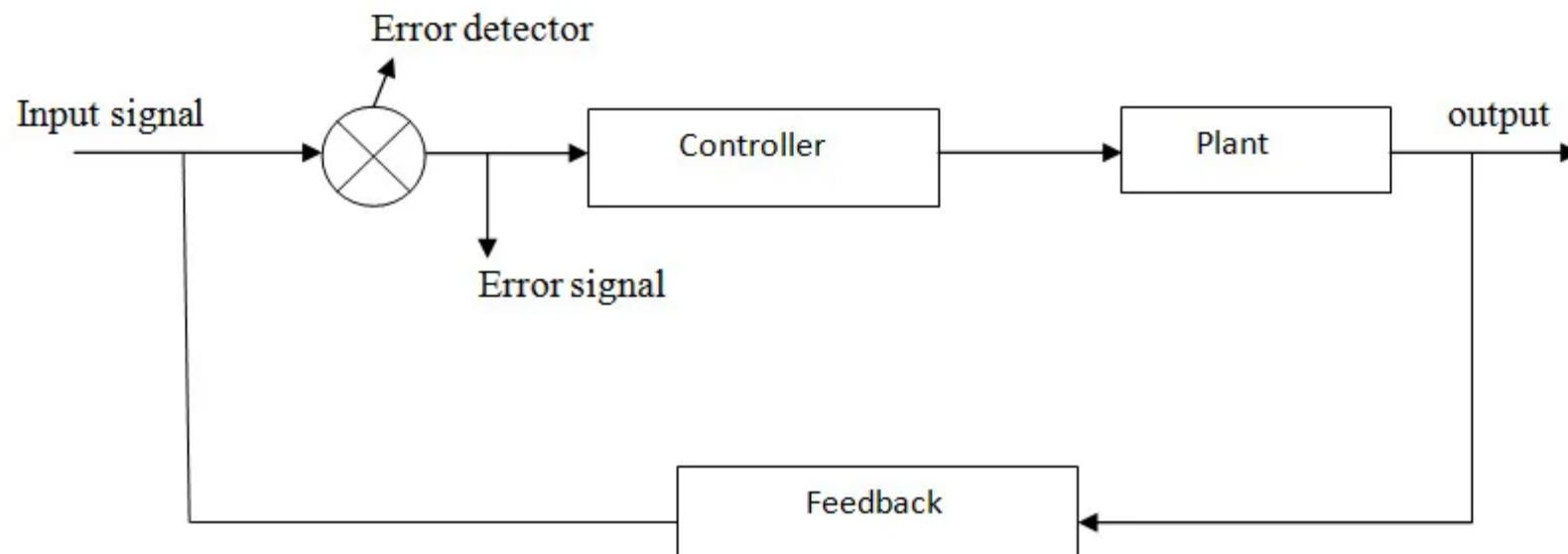
Control Systems Engineering with Python



Fundamentals of Control System

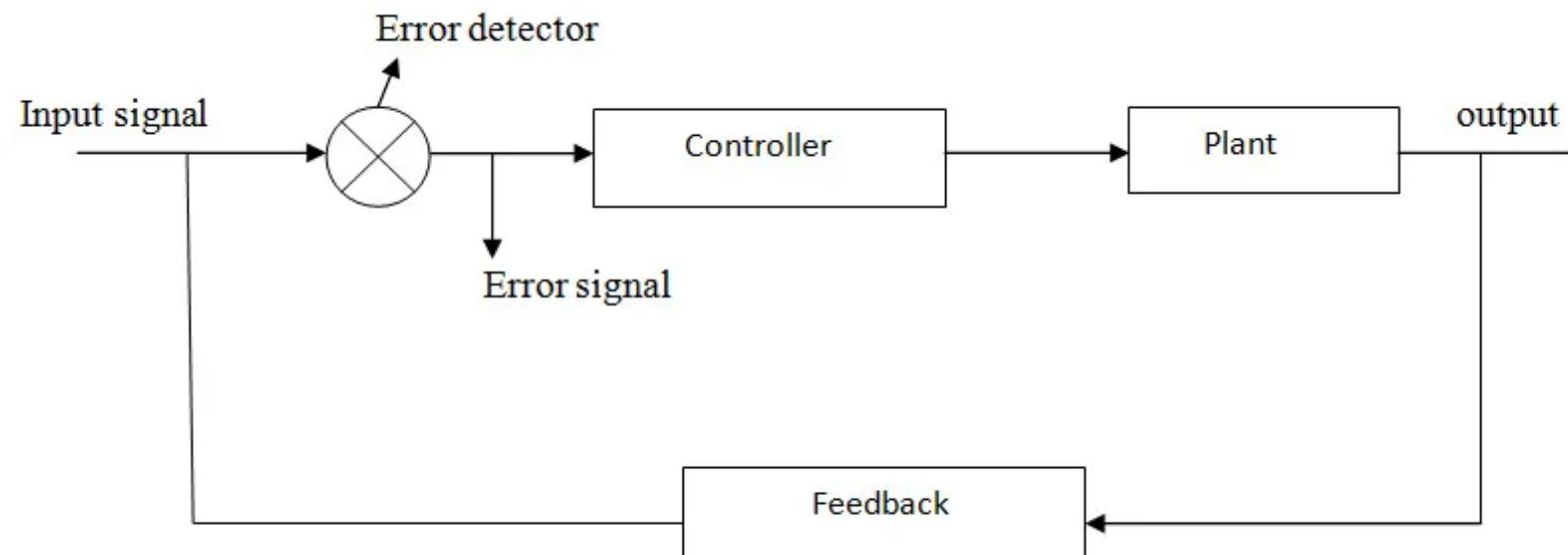
Fundamental of Control Systems

- Control System: is a control system for a process or plant, wherein control elements are distributed throughout the system.



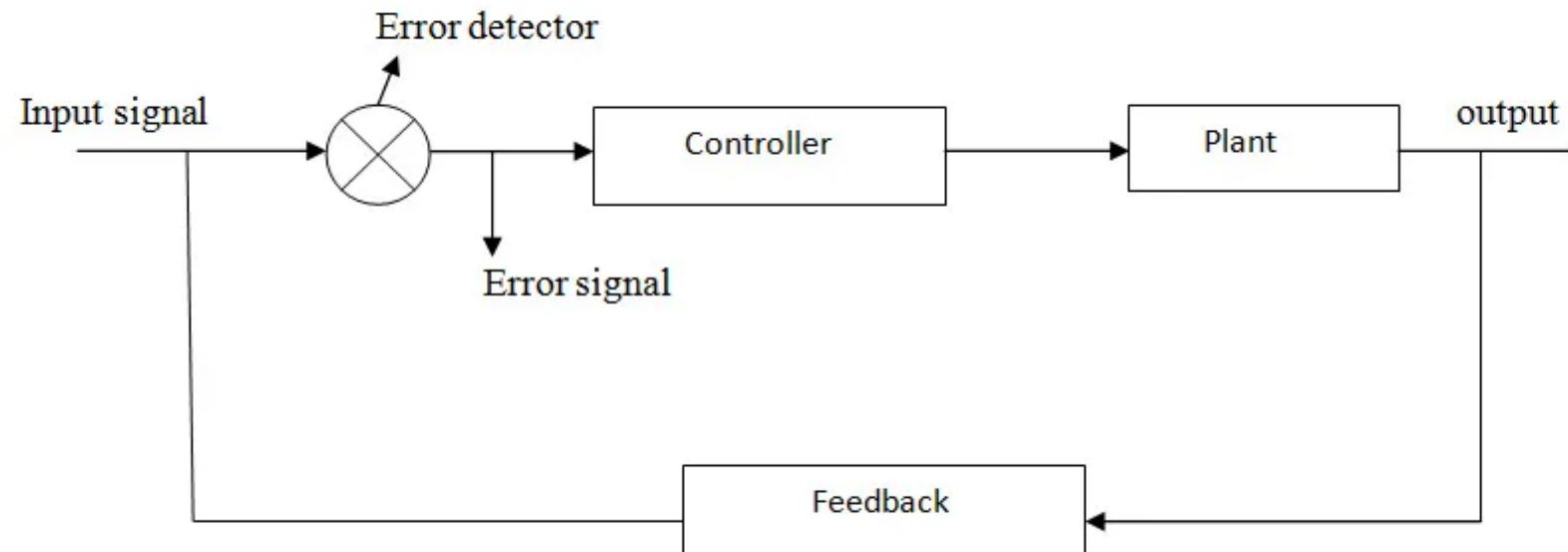
Basic Elements of Control System

- **Plant:** The portion of a **system which is to be controlled or regulated** is called as plant or process. It is a unit where actual processing is performed and if we observe in the figure, the input of the plant is the controlled signal generated by a controller.
- **Feedback:** It is a **controlled action in which the output is sampled**, and a proportional signal is given to the input for automatic correction of any changes in the desired output.



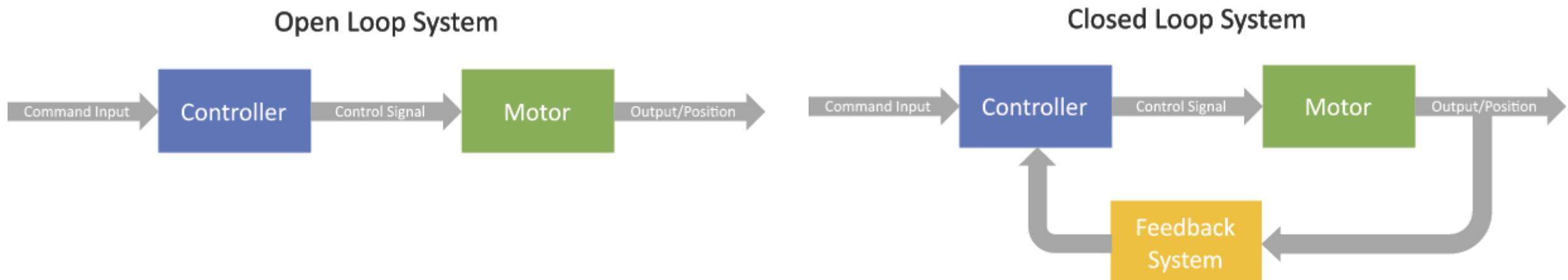
Basic Elements of Control System

- **Error Detector:** The function of error detector is to compare the reference input with the feedback signal. It produces an error signal which is a difference of two inputs which are reference signal and a feedback signal. The error signal is fed to the controller for necessary controlled action.
- **Controller:** the element of a system within itself or external to the system which controls the plant is called as a controller. The error signal will be a weak signal and so it has to be amplified and then modified for better control action.

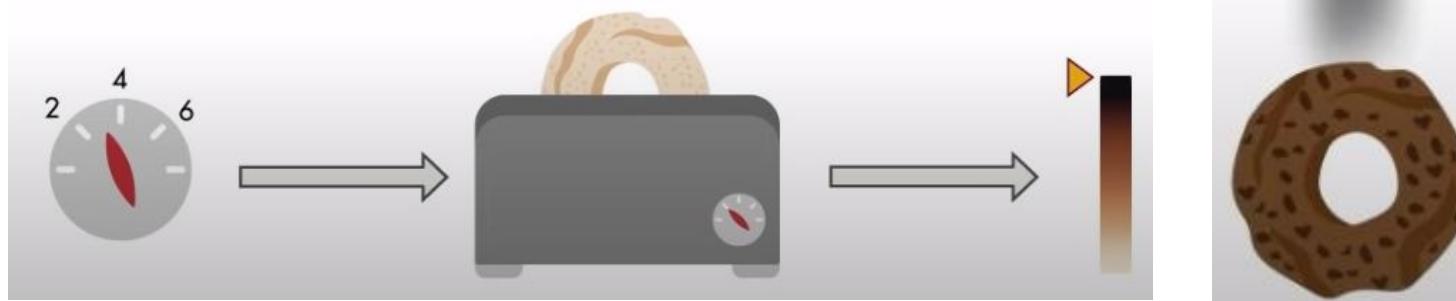
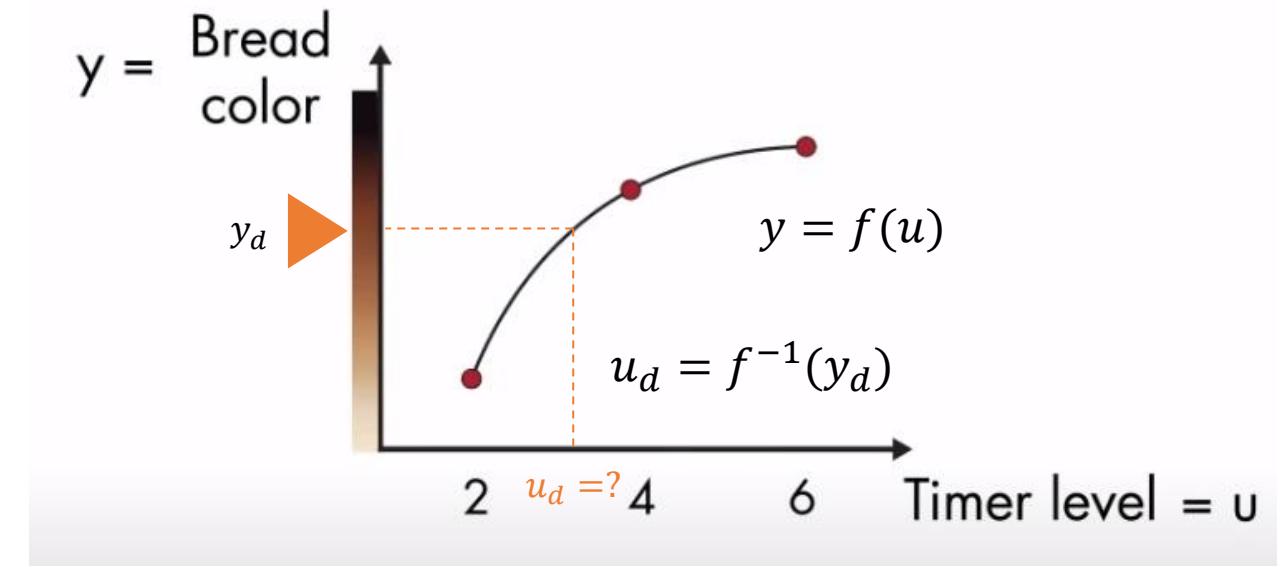
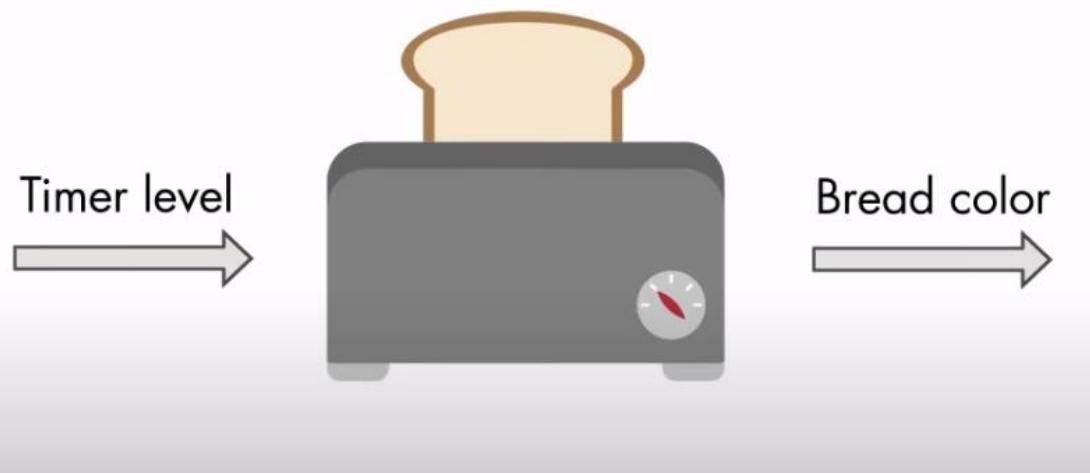


Types of Control Systems

- **Open Loop System:** The system whose control action is free from the output is known as the open loop control system.
- **Close Loop System:** The system output depends on the control action of the system.

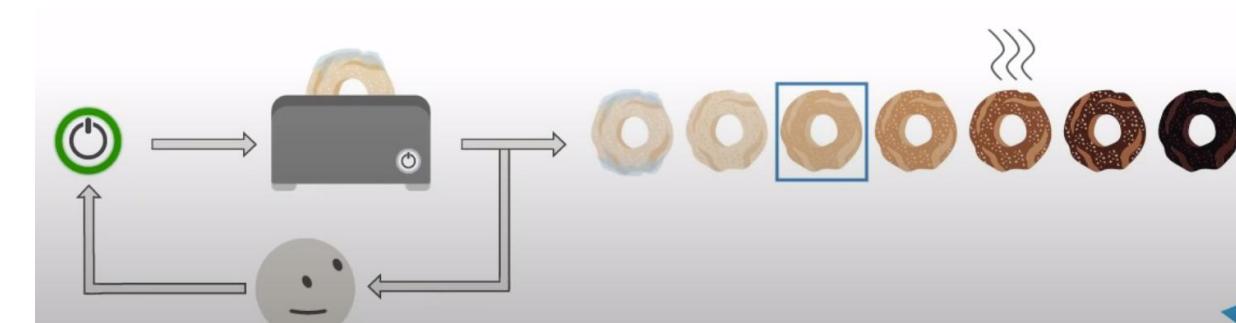
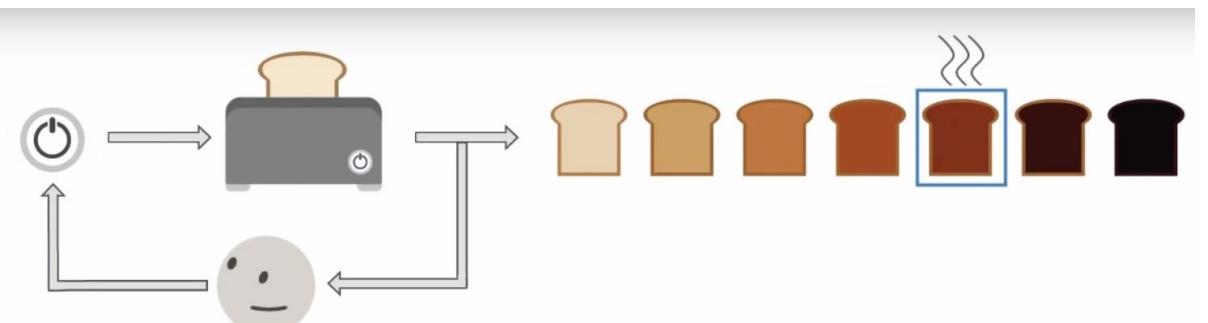
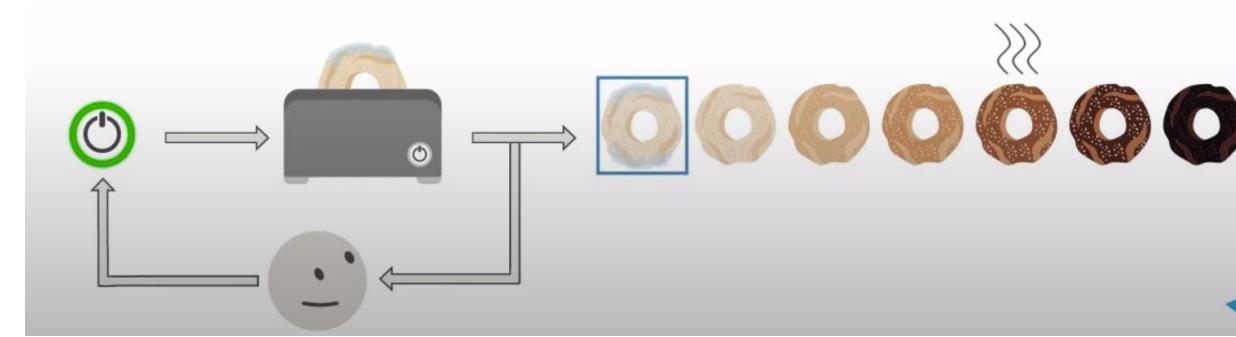
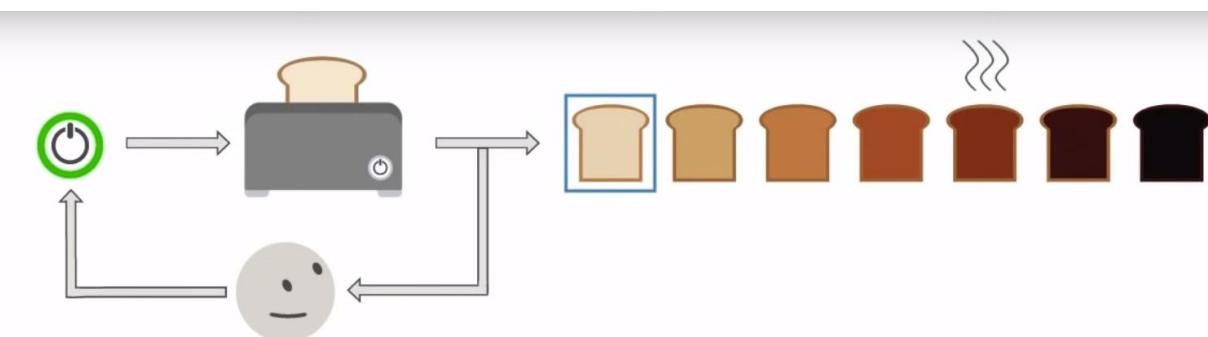
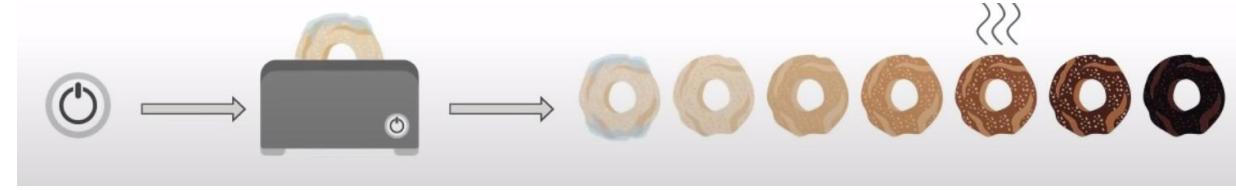
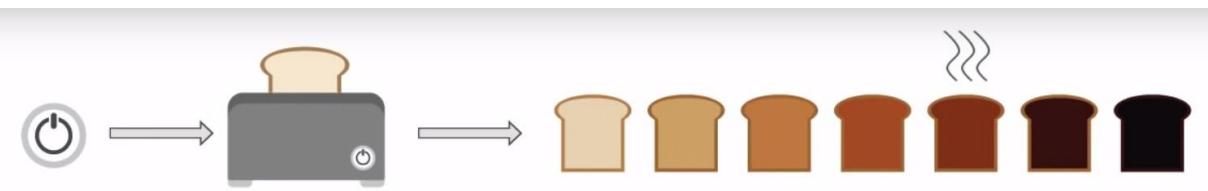


Open Loop Control System

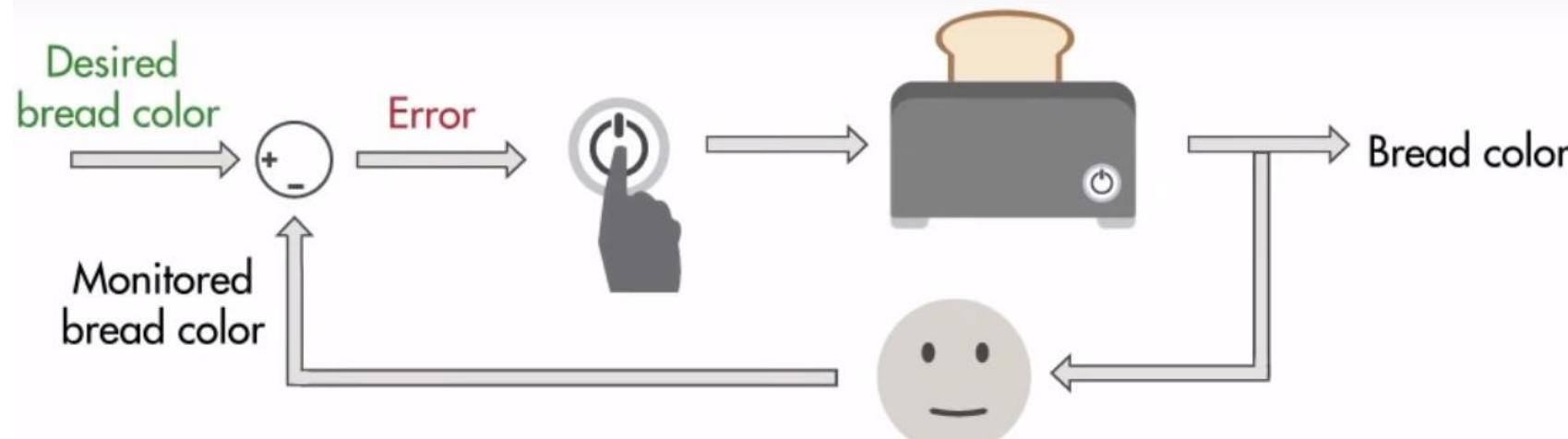
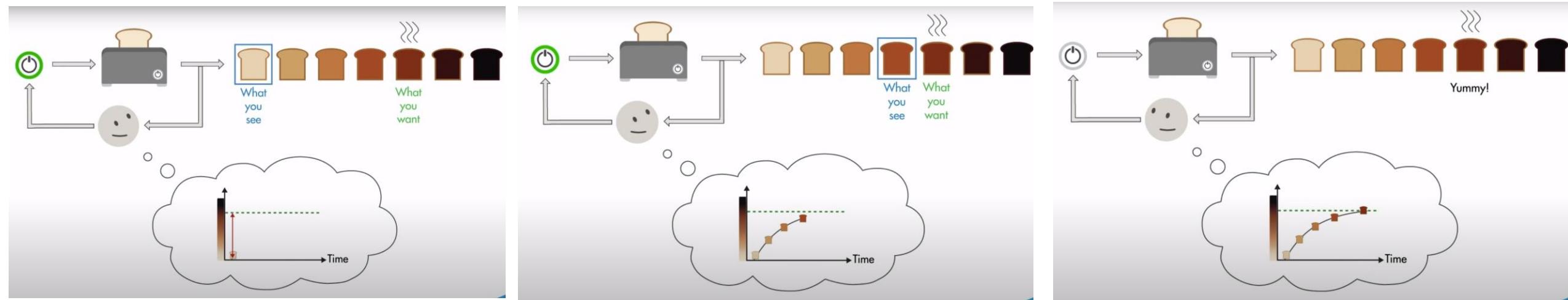


- Open loop control system is easy and conceptually simple and economical.
- Easier to construct.
- However, it is unreliable when there are variations in the system or unexpected environmental changes.

Feedback (Closed Loop) Control System



Feedback (Closed Loop) Control System



- Feedback control system handles variations in the system.
- It compensates for unexpected events.
- It is a bit complex and costly compared to Open loop system.
- Stability of the system is main problem.

Open Loop Vs Close Loop System

Open Loop	Close Loop
Any change in output has no effect on the input i.e. feedback does not exists.	Changes in output, affects the input which is possible by use of feedback.
Output measurement is not required for operation of system	Output measurement is necessary.
Feedback element is absent.	Feedback element is present.
Error detector is absent.	Error detector is necessary.
It is inaccurate and unreliable.	Highly accurate and reliable.
Highly sensitive to the disturbances.	Less sensitive to the disturbances.
Highly sensitive to the environmental changes.	Less sensitive to the environmental changes.
Simple to construct and cheap.	Complicated to design and hence costly.
Generally are stable in nature.	Stability is the major consideration while designing.
Highly affected by nonlinearities.	Reduced effect of nonlinearities.

Examples of Control Examples



- Open loop system

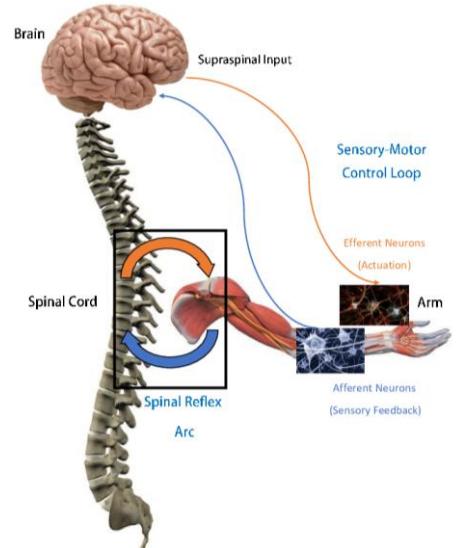
- Close loop system

Classification of Control Systems

Classification of Control Systems

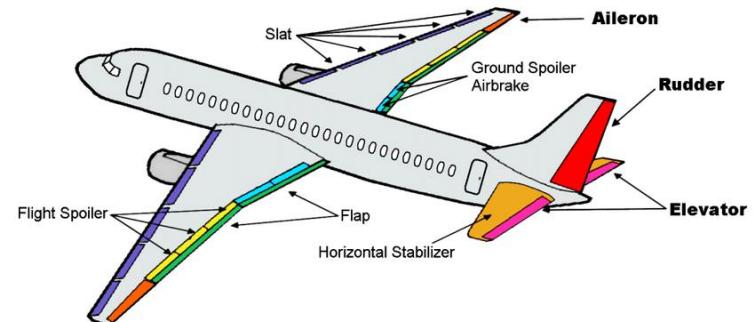
1. Natural Control System

- Universe
- Human Body



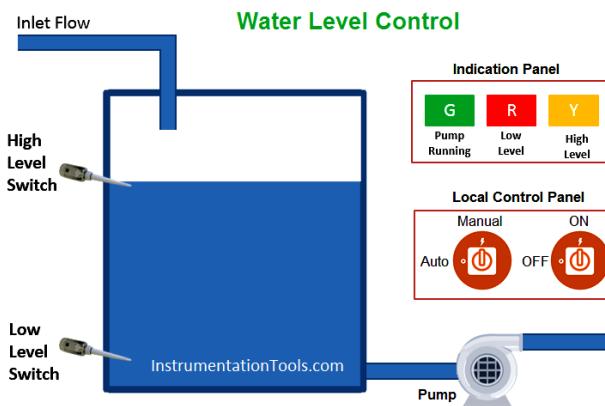
2. Manmade Control System

- Vehicles
- Aero planes



3. Manual Control System

- Room Temperature regulation via Electric Fan
- Water Level Control



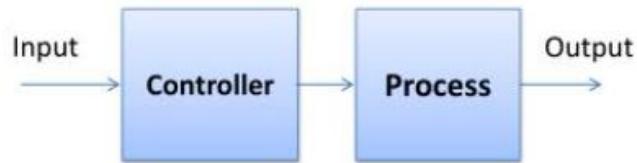
4. Automatic Control System

- Room temperature regulation via A.C
- Human Body Temperature control

Classification of Control Systems

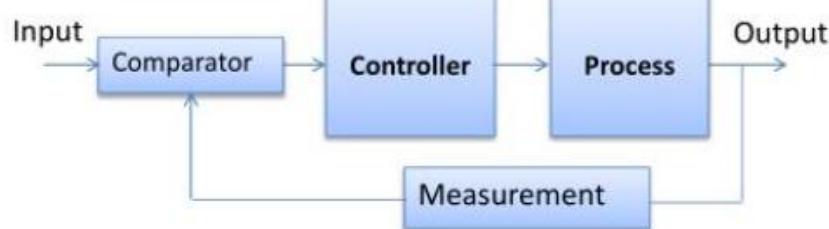
5. Open-Loop Control System

- Washing Machine
- Toaster



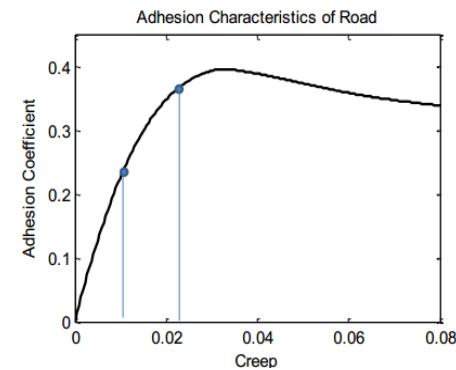
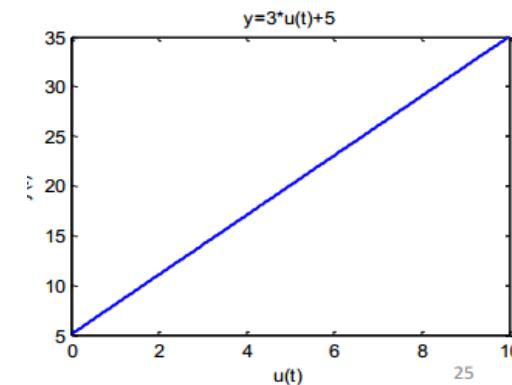
6. Closed-Loop System

- Refrigerator
- Auto-pilot system
- Driverless cars



7. Linear vs Non-linear Control System

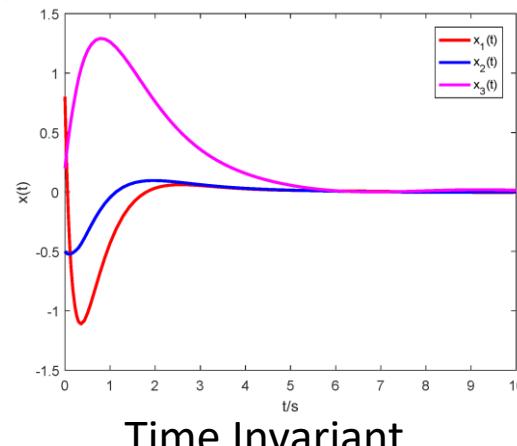
- A control system in which **output varies linearly with the input** is called a **linear control system**.



Linear vs Nonlinear

8. Time Invariant vs Time Variant

- **Time varying control system** is a system in which one or more parameters **vary with time**.



Time Invariant

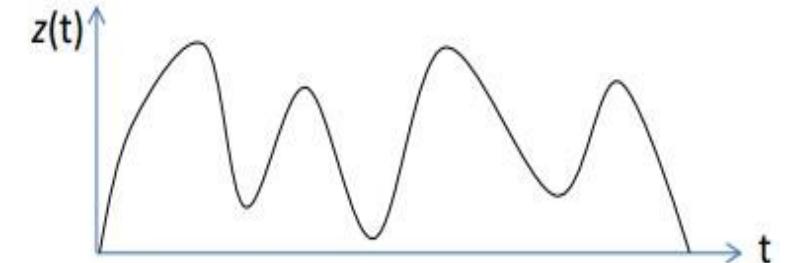
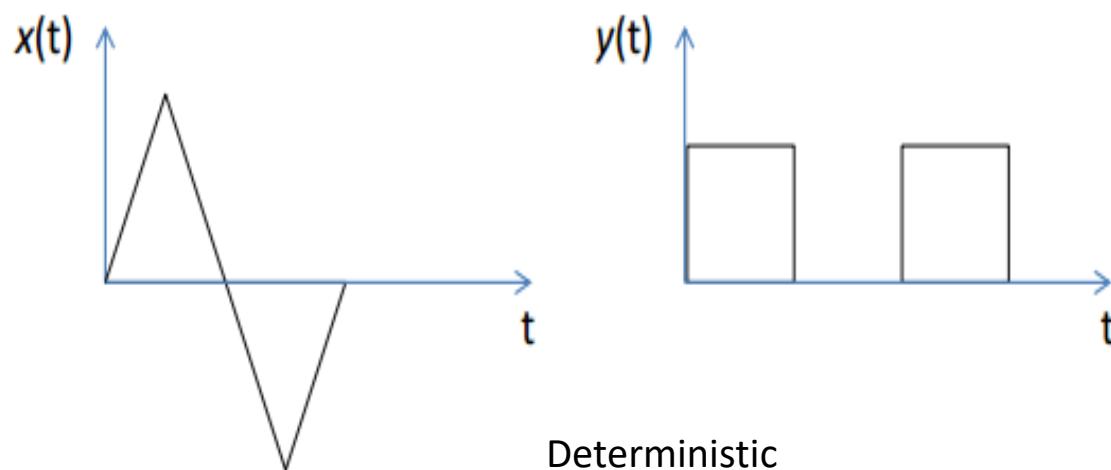
Classification of Control Systems

9. Continuous Data vs Discrete Data System

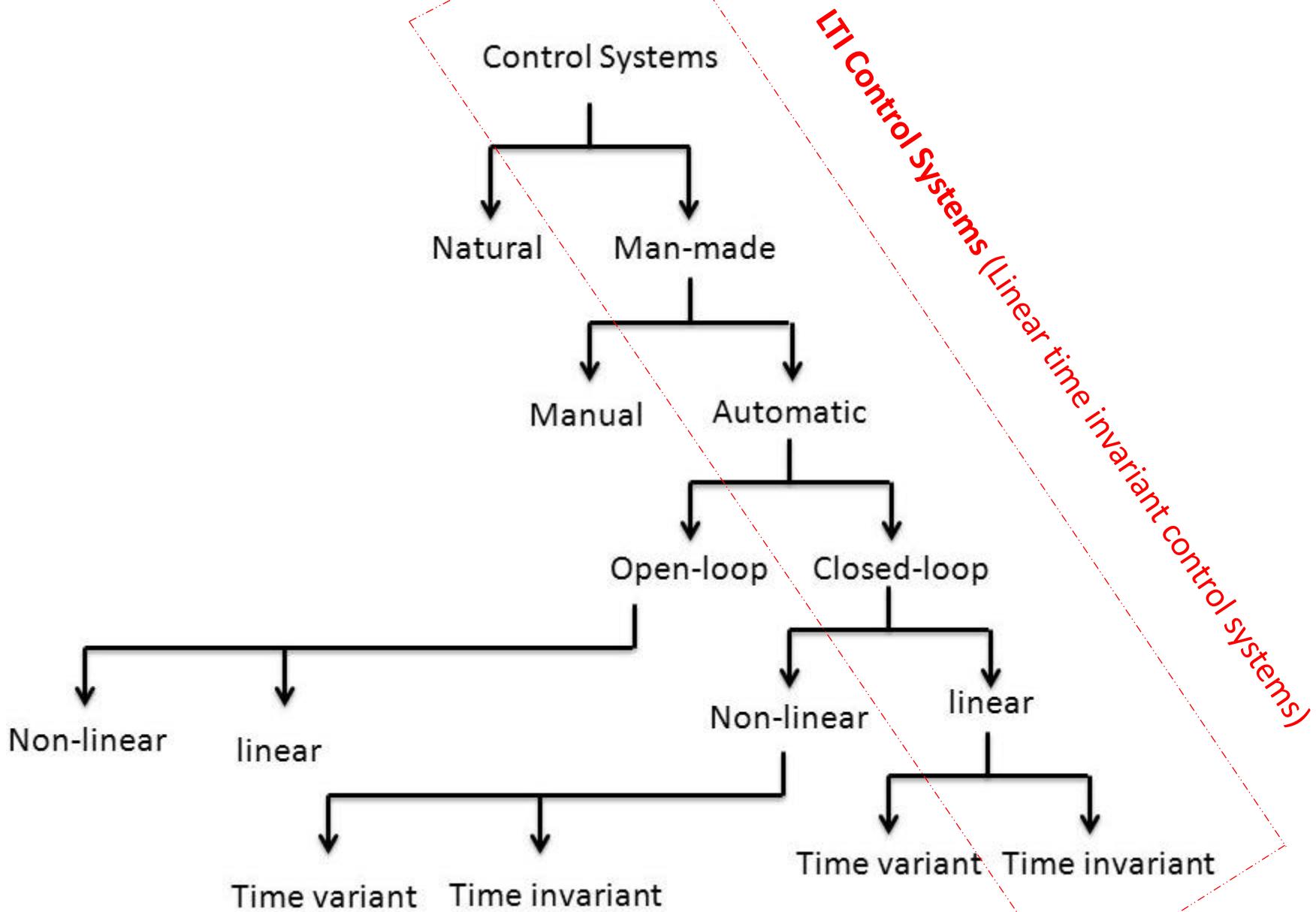
- In **continuous** data control system, all **system variables** are function of a **continuous time t**.
- A **discrete** time control system involves **one or more variables** that are known only at **discrete time intervals**.

10. Deterministic vs Stochastic Control System

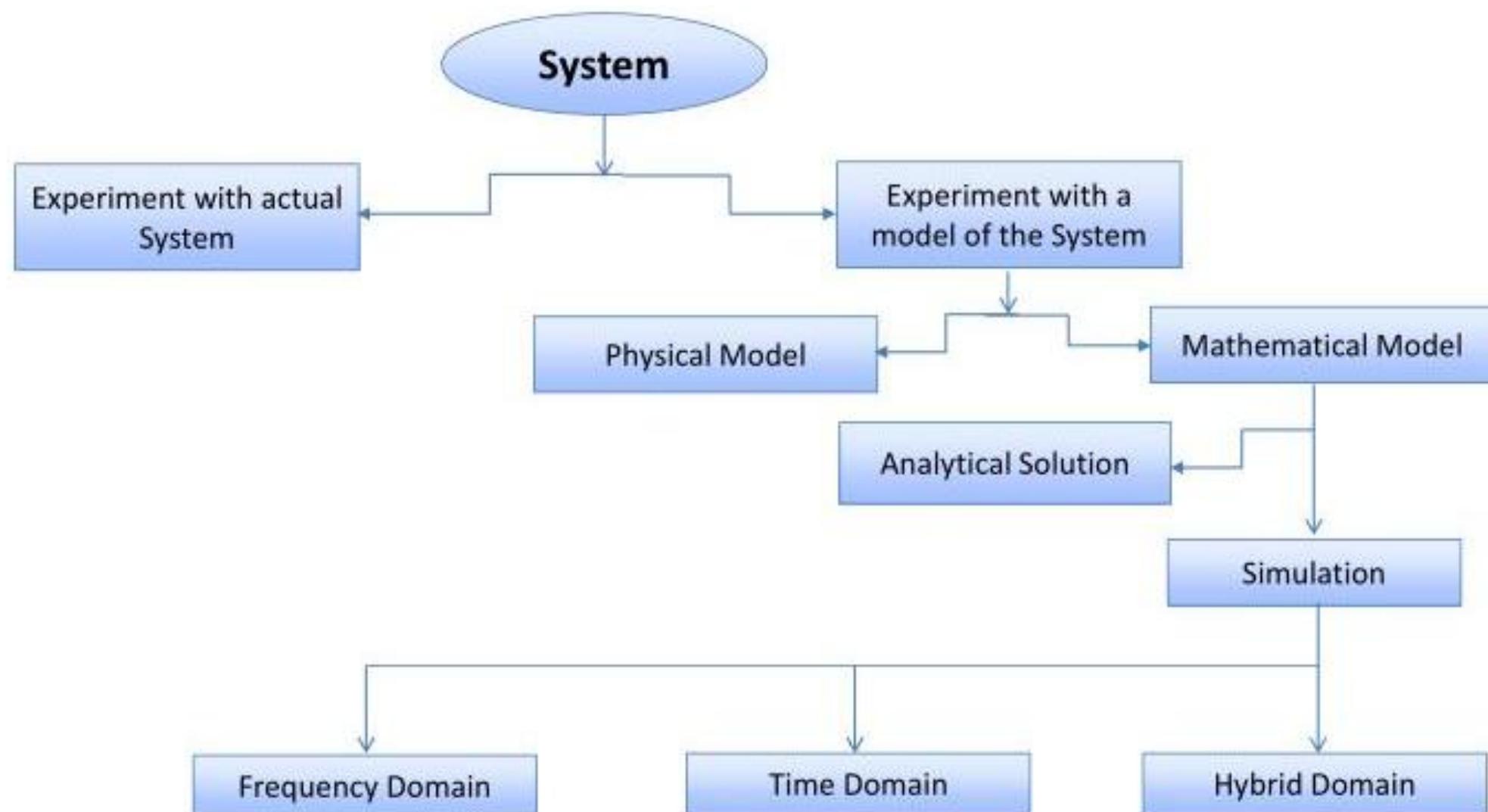
- A control system is **deterministic** if the response to input is **predictable** and repeatable.
- If not, the control system is stochastic control system.



Summary of Classification of Control Systems



Ways to Study a System

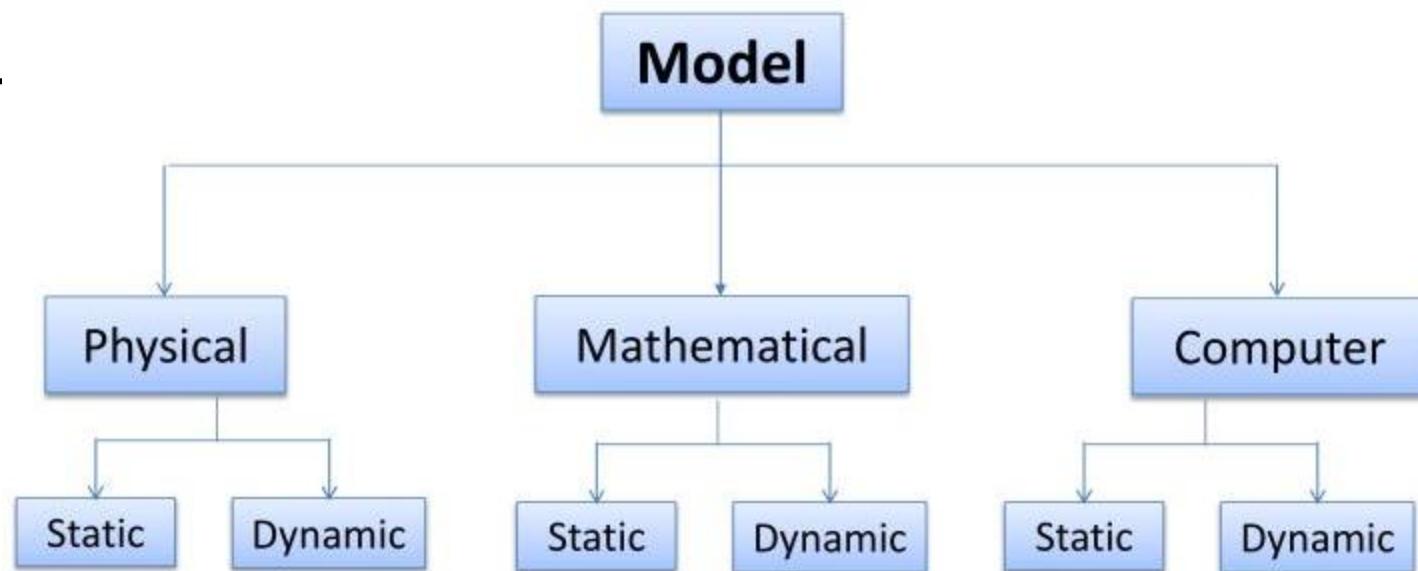


What is model

- A **model** is a simplified representation or abstraction of reality.

Types of Models

- Reality is generally too **complex to copy exactly**.
- Much of the complexity is actually irrelevant in problem solving.



Mathematical Models for Control System

Mathematical Model for Control System

- Differential equation model.
- Transfer function model.
- State space model.

Electric System Model

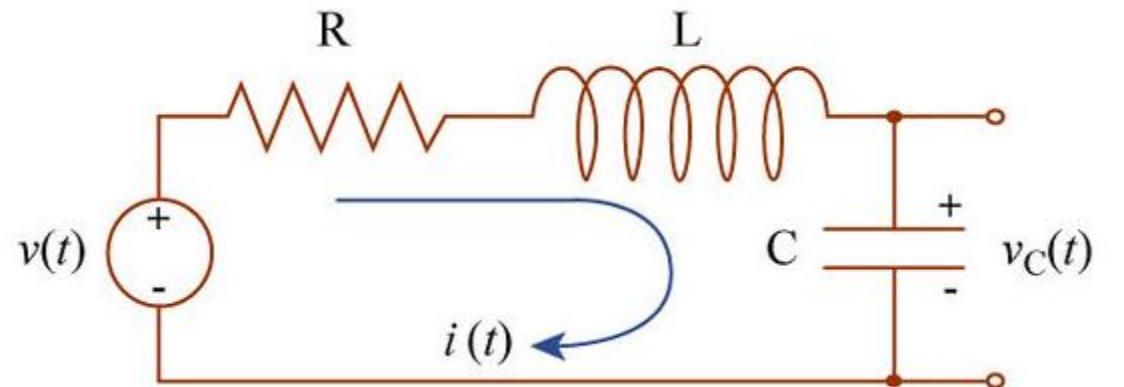
Voltage Laws

$$V = Ri + L \frac{di}{dt} + V_c$$

$$i = C \frac{dV_c}{dt}$$

$$V = RC \frac{dV_c}{dt} + LC \frac{d^2V_c}{dt^2} + V_c$$

$$\left(\frac{1}{LC} \right) V = \left(\frac{R}{L} \right) \frac{dV_c}{dt} + \frac{d^2V_c}{dt^2} + \frac{1}{LC} V_c$$



i = Current, L = Inductor, C = Capacitor,
 V = input voltage, V_c = Output voltage

Differential Equation Model

$$y' = ay + bu \quad \text{where} \quad a = \frac{-1}{T} \text{ and } b = \frac{K}{T}$$

Here K is the gain and T is the time constant



We can solve this differential equation model

- Discretization method
- Laplace Transform

Discretization Method

$$y' = ay + bu \quad \text{where} \quad a = \frac{-1}{T} \text{ and } b = \frac{K}{T}$$

Here K is the gain and T is the time constant



▪ Euler Method

- Forward Euler method
- Backward Euler method
- Mid Point Rule

• Forward Euler method

$$y' \approx \frac{y_{k+1} - y_k}{T_s}$$

Where T_s is the step time.

This gives:

$$\frac{y_{k+1} - y_k}{T_s} = ay_k + bu_k$$

Further we get:

$$y_{k+1} = y_k + T_s a y_k + T_s b u_k$$

This gives the following discrete differential equation:

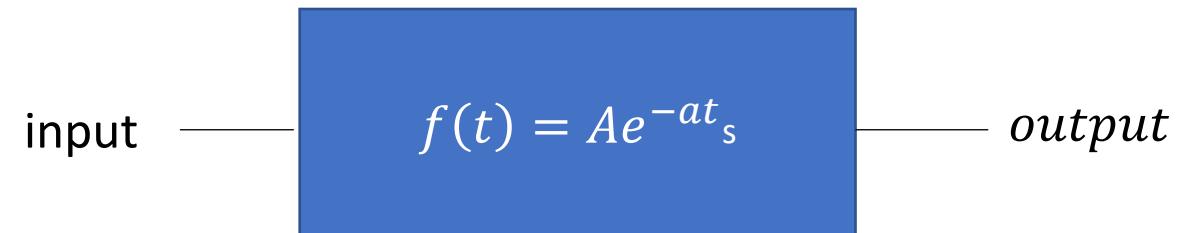
$$y_{k+1} = y_k(1 + T_s a) + T_s b u_k$$

Laplace Transform

- The purpose of Laplace

Transform is to transform an ordinary differential equations (ODEs) into algebraic equations,

which make it easier to solve ODEs.



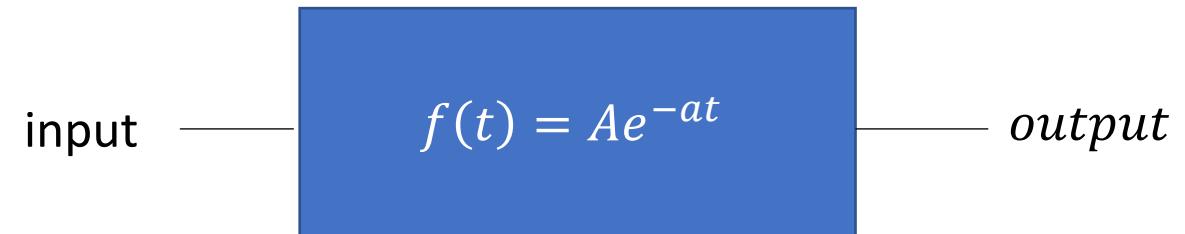
Laplace Transform

■ Laplace Transform

$$\mathcal{L}[f(t)] = F(s) = \int_0^{\infty} f(t)e^{-st} dt$$

$$s = \sigma + j\omega$$

Real imaginary



■ The Inverse Laplace Transform

$$\begin{aligned}\mathcal{L}^{-1}[F(s)] &= \frac{1}{2\pi j} \int_{\sigma-j\omega}^{\sigma+j\omega} F(s)e^{st} ds \\ &= f(t)u(t)\end{aligned}$$

Unit Step Function

$$u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

Laplace Transform

- If $f(t) = Ae^{-at}$, then

$$\begin{aligned}\mathcal{L}[f(t)] = F(s) &= \int_0^{\infty} f(t)e^{-st} dt = \int_0^{\infty} Ae^{-at}e^{-st} dt \\ &= A \int_0^{\infty} e^{-(s+a)t} dt = A \left[\frac{-1}{s+a} \right] e^{-(s+a)t} \Big|_0^{\infty} \\ F(s) &= \frac{A}{s+a}\end{aligned}$$

- If $f(t) = \frac{dy}{dt}$, then, $\mathcal{L}[f(t)] = \mathcal{L}\left[\frac{dy}{dt}\right] = s\mathcal{L}[y(t)] - y(0)$

- Similarly, $f(t) = \frac{d^2y}{dt^2}$, then, $\mathcal{L}[f(t)] = \mathcal{L}\left[\frac{d^2y}{dt^2}\right] = s^2\mathcal{L}[y(t)] - s\mathcal{L}[y(0)] - \frac{dy(0)}{dt}$

- Thus, $\mathcal{L}\left[\frac{d^2y}{dt^2}\right] = s^2\mathcal{L}[y(t)] - s\mathcal{L}[y(0)] - \frac{dy(0)}{dt} \equiv s^2Y(s) - sy(0) - y'(0)$

```
In [1]: import sympy as sp
t, s = sp.symbols('t, s')
a = sp.symbols('a', real=True, positive=True)
A = sp.symbols('A', real=True, positive=True)

f = A*sp.exp(-a*t)
f
```

Out[1]: Ae^{-at}

```
In [2]: sp.laplace_transform(f, t, s)
```

Out[2]: $(A/(a + s), -a, \text{True})$

Laplace Transform Table

$f(t)$	$F(s) = \mathcal{L}[f(t)]$	Formula
$f(t) = 1$	$F(s) = \frac{1}{s}$ $s > 0$	A
$f(t) = e^{at}$	$F(s) = \frac{1}{(s - a)}$ $s > a$	B
$f(t) = t^n$	$F(s) = \frac{n!}{s^{(n+1)}}$ $s > 0$	C
$f(t) = \sin(at)$	$F(s) = \frac{a}{s^2 + a^2}$ $s > 0$	D
$f(t) = \cos(at)$	$F(s) = \frac{s}{s^2 + a^2}$ $s > 0$	E
$f(t) = \sinh(at)$	$F(s) = \frac{a}{s^2 - a^2}$ $s > a $	F
$f(t) = \cosh(at)$	$F(s) = \frac{s}{s^2 - a^2}$ $s > a $	G
$f(t) = t^n e^{at}$	$F(s) = \frac{n!}{(s - a)^{(n+1)}}$ $s > a$	H
$f(t) = e^{at} \sin(bt)$	$F(s) = \frac{b}{(s - a)^2 + b^2}$ $s > a$	I
$f(t) = e^{at} \cos(bt)$	$F(s) = \frac{(s - a)}{(s - a)^2 + b^2}$ $s > a$	J
$f(t) = e^{at} \sinh(bt)$	$F(s) = \frac{b}{(s - a)^2 - b^2}$ $s - a > b $	K
$f(t) = e^{at} \cosh(bt)$	$F(s) = \frac{(s - a)}{(s - a)^2 - b^2}$ $s - a > b $	L

Laplace Transform Properties

Independent variable	t	s
Signal representation	$f(t)$	$F(s)$
Uniqueness	$\mathcal{L}^{-1}\{F(s)\}(=)[f(t)]u(t)$	$\mathcal{L}\{f(t)\} = F(s)$
Linearity	$Af_1(t) + Bf_2(t)$	$AF_1(s) + BF_2(s)$
Integration	$\int_0^t f(\tau) d\tau$	$\frac{F(s)}{s}$
Differentiation	$\frac{df(t)}{dt}$ $\frac{d^2f(t)}{dt^2}$ $\frac{d^3f(t)}{dt^3}$	$sF(s) - f(0-)$ $s^2F(s) - sf(0-) - f'(0-)$ $s^3F(s) - s^2f(0-) - sf'(0-) - f''(0-)$
s -Domain translation	$e^{-\alpha t}f(t)$	$F(s + \alpha)$
t -Domain translation	$f(t - a)u(t - a)$	$e^{-as}F(s)$

Laplace Transform for simple ODE

- The purpose of Laplace Transform is to transform an ordinary differential equations (ODEs) into algebraic equations, which make it easier to solve ODEs.

$$y'' - y = e^{2t}$$

$$y(0) = 0, \quad y'(0) = 1$$

$$\mathcal{L}[y'' - y] = \mathcal{L}[e^{2t}] \Rightarrow \mathcal{L}[y''] - \mathcal{L}[y] = \mathcal{L}[e^{2t}]$$

$$\mathcal{L}[y] = Y(s)$$

$$\mathcal{L}[y'] = sY(s) - y(0)$$

$$\mathcal{L}[y''] = s^2Y(s) - sy(0) - y'(0)$$

$$\mathcal{L}[e^{2t}] = \frac{1}{s-2}$$

$$[s^2Y(s) - sy(0) - y'(0)] - [Y(s)] = \frac{1}{s-2}$$

$$[s^2Y(s) - s(0) - 1] - [Y(s)] = \frac{1}{s-2}$$

$$Y(s)[s^2 - 1] = \frac{1}{s-2} + 1$$

$$Y(s) = \frac{1}{(s-2)(s+1)}$$



Can solve it using simple partial fraction

Laplace Transform for simple ODE

$$Y(s) = \frac{1}{(s-2)(s+1)}$$

$$\frac{1}{(s-2)(s+1)} = \frac{A}{s-2} + \frac{B}{s+1}$$

$$1 = A(s+1) + B(s-2)$$

$$s = -1 \Rightarrow 1 = A(0) + B(-3)$$

$$s = 2 \Rightarrow 1 = A(3) + B(0)$$

$$A = \frac{1}{3}, \quad B = -\frac{1}{3}$$

$$Y(s) = \frac{A}{s-2} + \frac{B}{s+1} = \frac{1}{(s-2)(s+1)} = \frac{1/3}{s-2} - \frac{1/3}{s+1}$$

$$Y(s) = \frac{1}{3} \left[\frac{1}{s-2} - \frac{1}{s+1} \right]$$

$$y(t) = \mathcal{L}^{-1}[Y(s)] = \frac{1}{3} \left[\mathcal{L}^{-1}\left(\frac{1}{s-2}\right) - \mathcal{L}^{-1}\left(\frac{1}{s+1}\right) \right]$$

$$y(t) = \mathcal{L}^{-1}[Y(s)] = \frac{1}{3} [e^{2t} - e^{-t}]$$

Example

Problem: Find the transfer function of the system given by:

$$\frac{d^2y(t)}{dt^2} + 3\frac{dy(t)}{dt} + 2y(t) = x(t) \quad \text{With zero initial conditions.}$$

Where $x(t)$ is the input & $y(t)$ is the output.

Solution: $[s^2Y(s) - sy(0) - y'(0)] + 3[sY(s) - y(0)] + 2Y(s) = X(s)$



As we have zero initial condition

$$\Rightarrow s^2 Y(s) + 3sY(s) + 2Y(s) = X(s)$$

$$\Rightarrow Y(s)[s^2 + 3s + 2] = X(s)$$

$$\Rightarrow \frac{Y(s)}{X(s)} = \frac{1}{s^2 + 3s + 2}$$

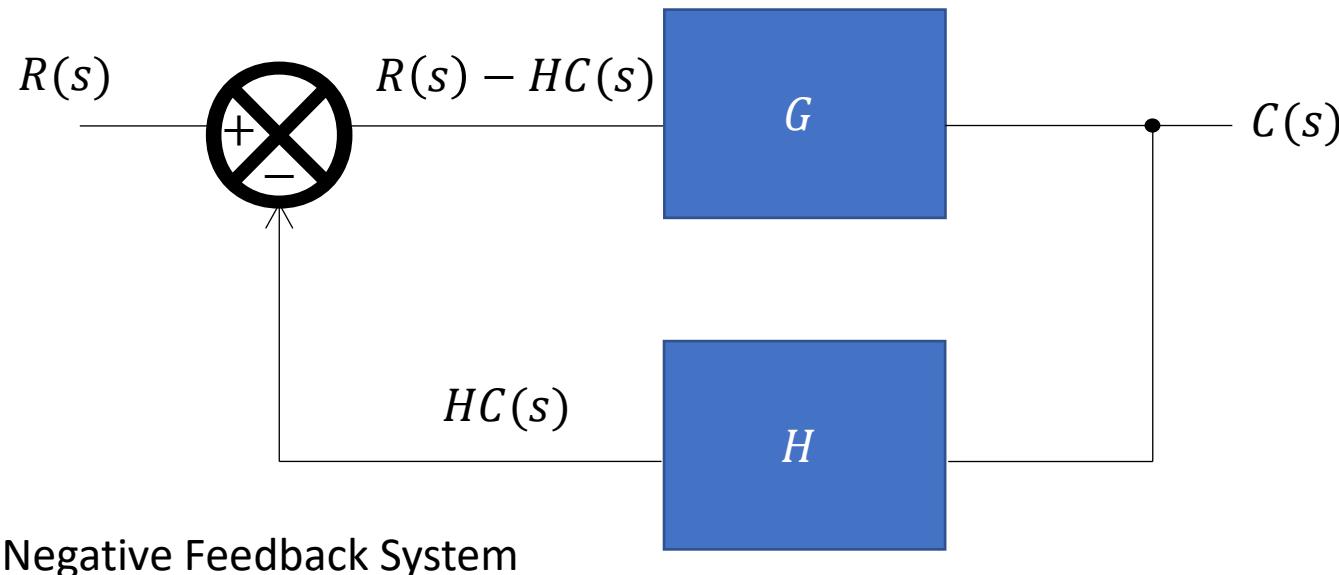
$$\Rightarrow \frac{Y(s)}{X(s)} = H(s) = \frac{1}{(s + 1).(s + 2)}$$

Transfer function of a Control System

- Transfer function of a control system is the ratio of Laplace Transform of output to Laplace transform of input.

$$\text{Transfer Function}(TF) = \frac{\text{Laplace Transform of Output}}{\text{Laplace Transform of Input}}$$

With zero initial conditions.



$$C(s) = [R(s) - HC(S)]G$$

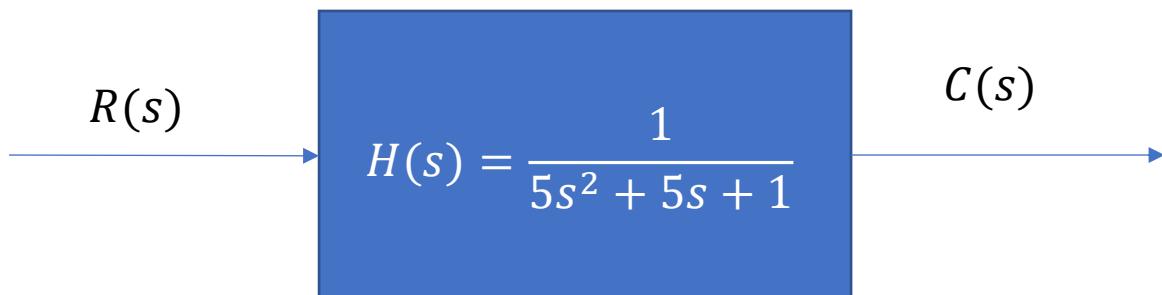
$$C(s) = R(s)G - GHC(s)$$

$$C(s)(1 + GH) = R(s)G$$

$$\frac{C(s)}{R(s)} = \frac{G}{1 + GH} = TF$$

Transfer function of a Control System

- We can easily represent transfer function of a control system in python via using python control library



```
import control as co
num=(1)
den=(5,5,1)
H_s=co.tf(num,den)
print('H(s) = ', H_s)
```

$H(s) =$
$$\frac{1}{5 s^2 + 5 s + 1}$$

Transfer Function of Electric Model

Voltage Laws

$$V = Ri + L \frac{di}{dt} + V_c$$

$$i = C \frac{dV_c}{dt}$$

$$V = RC \frac{dV_c}{dt} + LC \frac{d^2V_c}{dt^2} + V_c$$

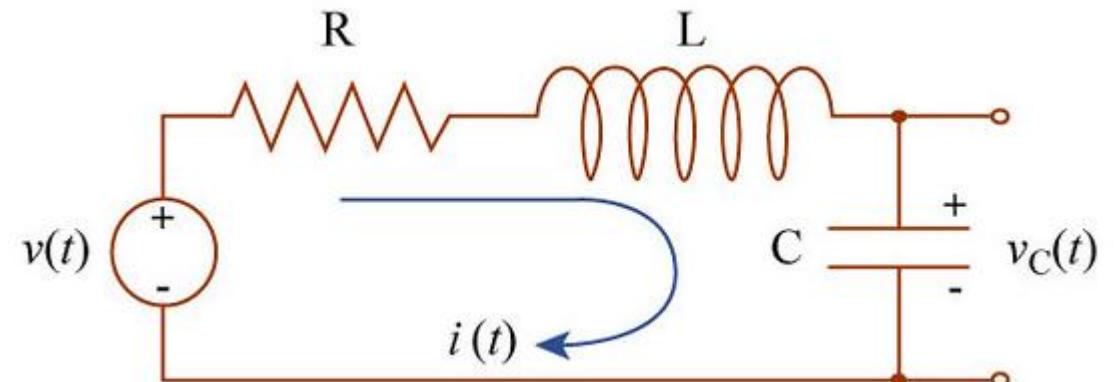
$$\left(\frac{1}{LC}\right)V = \left(\frac{R}{L}\right)\frac{dV_c}{dt} + \frac{d^2V_c}{dt^2} + \frac{1}{LC}V_c$$

Transfer Function Model:

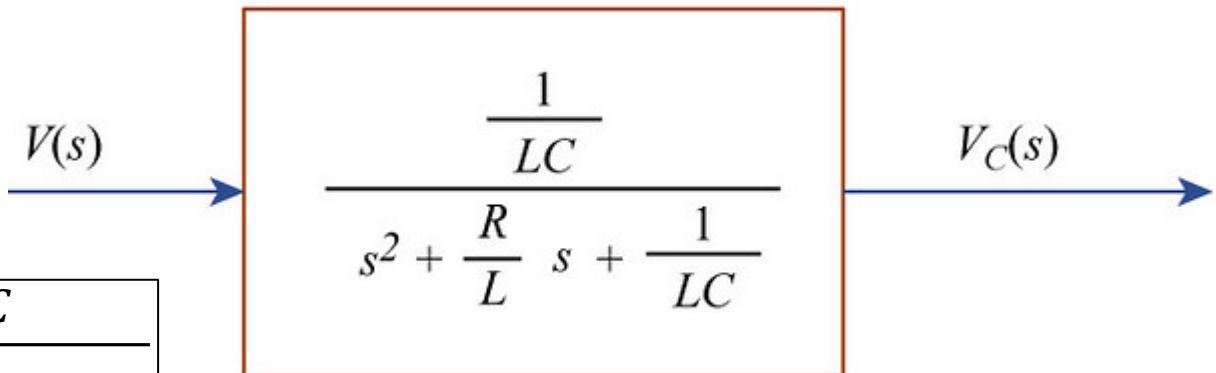
$$\left(\frac{1}{LC}\right)V(s) = \left(\frac{R}{L}\right)sV_c(s) + s^2V_c(s) + \frac{1}{LC}V_c(s)$$

$$\left(\frac{1}{LC}\right)V(s) = V_c(s) \left(\left(\frac{R}{L}\right)s + s^2 + \frac{1}{LC} \right)$$

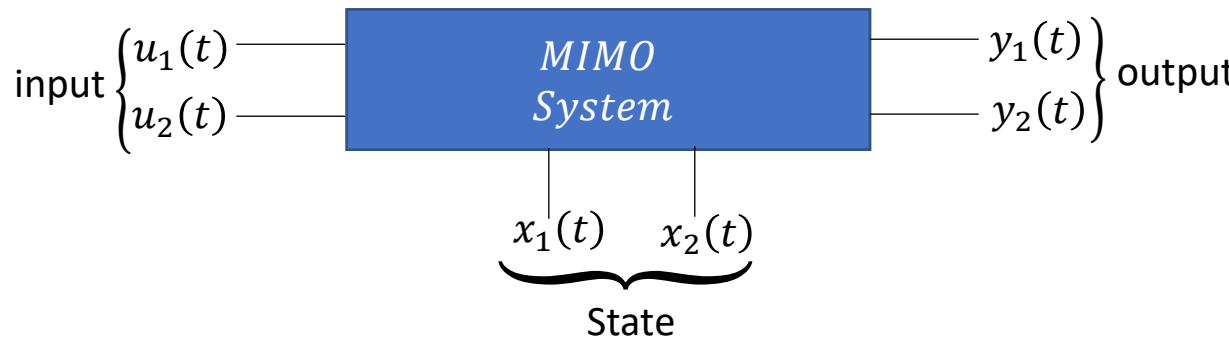
$$\frac{V_c(s)}{V(s)} = \frac{1/LC}{s^2 + \left(\frac{R}{L}\right)s + 1/LC}$$



i = Current, L = Inductor, C = Capacitor,
 V = input voltage, V_c = Output voltage



State-Space Modeling



- A **state-space model** represents a system by a series of **first order differential state equations** and **algebraic output equations**.
- **State:** It is a group of variables , which summarizes the history of the system in order to predict the Future values.
- **State Variable:** The smallest set of variables that determines the state of the system are known as state variables.
- **State Vector:** It is the vector, which contains the state variables as elements.

- **Advantages:**

- Analysis is done by considering initial conditions.
- More accurate than transfer function.
- Analysis of multi-input and multi-output systems are easy.
- Gives information about controllability.
- It is applicable to all dynamic systems.
- Allow for a more geometric understanding of dynamic systems

- **Disadvantages:**

- Complex Technique
- Many computations are required

State-Space Modeling

- Output Equations:

$$y_1(t) = c_{11}x_1(t) + c_{12}x_2(t) + d_{11}u_1(t) + d_{12}u_2(t)$$

$$y_2(t) = c_{21}x_1(t) + c_{22}x_2(t) + d_{21}u_1(t) + d_{22}u_2(t)$$

In matrix form

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

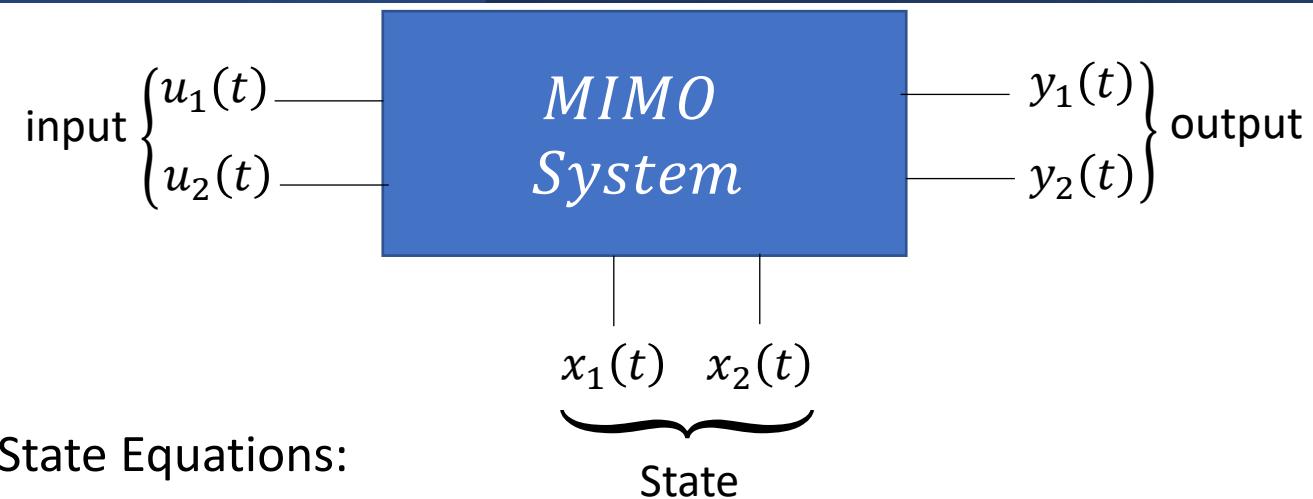
$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$

State Equation \Rightarrow

$\mathbf{x}'(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$

Output Equation \Rightarrow

$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$



- State Equations:

$$\frac{dx_1(t)}{dt} = x'_1(t) = a_{11}x_1(t) + a_{12}x_2(t) + b_{11}u_1(t) + b_{12}u_2(t)$$

$$\frac{dx_2(t)}{dt} = x'_2(t) = a_{21}x_1(t) + a_{22}x_2(t) + b_{21}u_1(t) + b_{22}u_2(t)$$

In matrix form

$$\begin{bmatrix} x'_1(t) \\ x'_2(t) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

$\mathbf{x}'(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$

State-Space Modeling

- In general state-space models have the following form (equations can be nonlinear and time varying)

$$\text{State Equation} \Rightarrow \begin{cases} x'_1 = f_1(x_1, x_2, \dots x_n, u_1, u_2, \dots u_m) \\ \vdots \\ x'_n = f_n(x_1, x_2, \dots x_n, u_1, u_2, \dots u_m) \end{cases}$$

$$\text{Output Equation} \Rightarrow \begin{cases} y_1 = h_1(x_1, x_2, \dots x_n, u_1, u_2, \dots u_m) \\ \vdots \\ y_p = h_p(x_1, x_2, \dots x_n, u_1, u_2, \dots u_m) \end{cases}$$

- State-space models are numerically efficient to solve, can handle complex systems, allow for a more geometric understanding of dynamic systems, and form the basis for much of modern control theory.

Example of State Space Modeling

- Can generate a state-space model by pure mathematical manipulation through changing

$$x_1 = x(t)$$

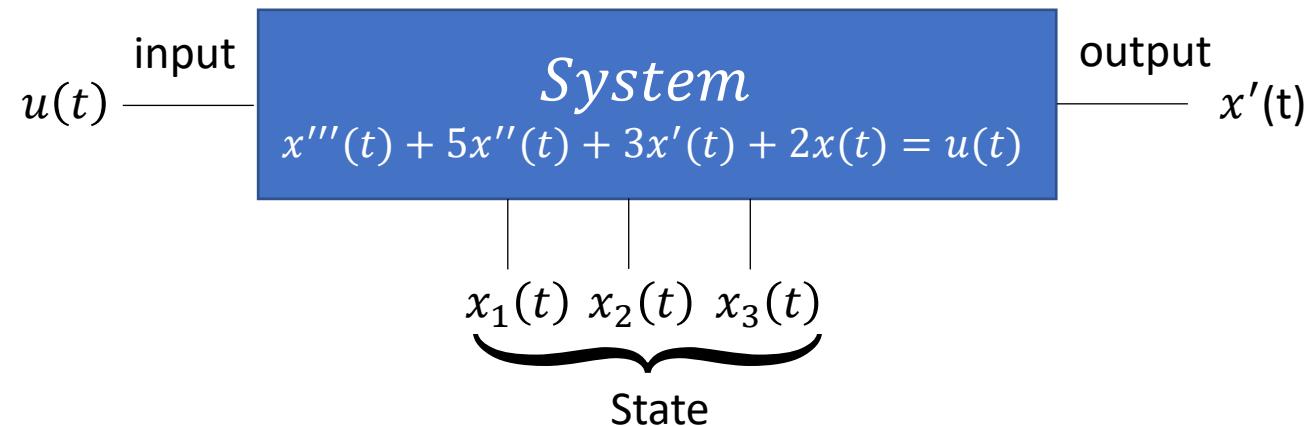
$$x_2 = x'(t)$$

$$x_3 = x''(t)$$

State Equation $\Rightarrow \begin{cases} x'_1 = x' = x_2 \\ x'_2 = x'' = x_3 \\ x'_3 = x''' = -5x_3 - 3x_2 - 2x_1 + u \end{cases} \Rightarrow \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \Rightarrow x' = Ax + Bu$

output Equation $\Rightarrow \{y = x' = x_2\} \Rightarrow [y] = [0 \ 1 \ 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + [0]u \Rightarrow y = Cx + Du$

$$x' = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C = [0 \ 1 \ 0], \quad D = [0], \quad u = u$$



Example of State Space Modeling for mechanical system

- Netwon's second law of Motion:

$$F_m(t) = m \frac{d^2x(t)}{dt^2}$$

- Viscous Friction:

$$F_B(t) = B \frac{dx(t)}{dt}$$

- Elastic Force:

$$F_k(t) = kx(t)$$

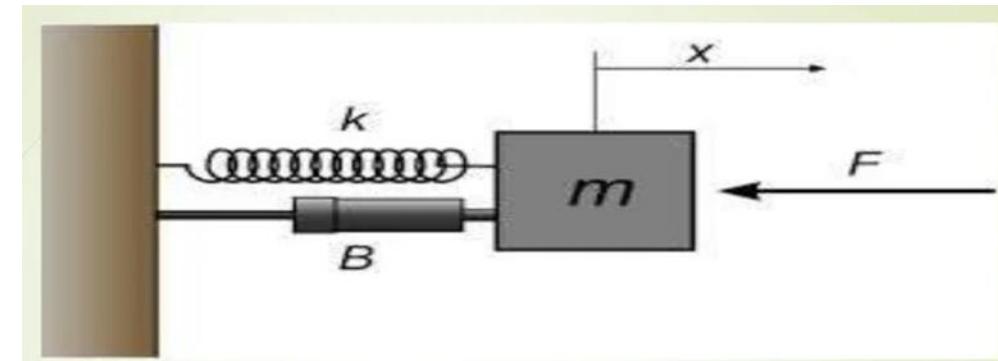
F = Applied force (N),

m =mass (kg),

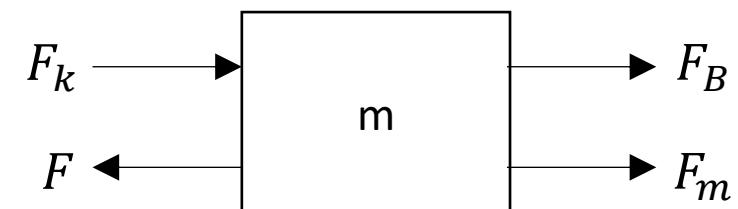
x = Displacement (m),

B =Dumping coefficient ($N - sec/m$),

k =stiffness of spring (N/m).



(a) Mechanical System



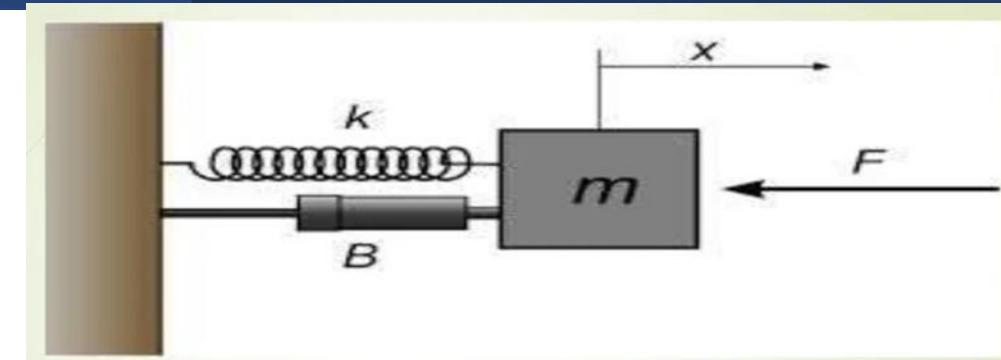
(b) Free body diagram

Example of State Space Modeling for mechanical system

- Differential Equation of the system

$$\sum F(t) = 0$$

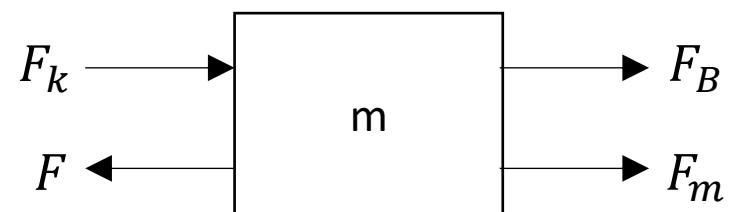
$$F(t) - F_m(t) - F_k(t) - F_b(t) = 0$$



(a) Mechanical System

$$F(t) = m \frac{d^2x(t)}{dt^2} + kx(t) + B \frac{dx(t)}{dt}$$

$$F(t) = mx''(t) + Bx'(t) + kx(t)$$



(b) Free body diagram

Example of State Space Modeling

■ State Space Modeling

The system is second order. This means that the system involves two integrators. Let's us define state variables $x_1(t)$ and $x_2(t)$ as

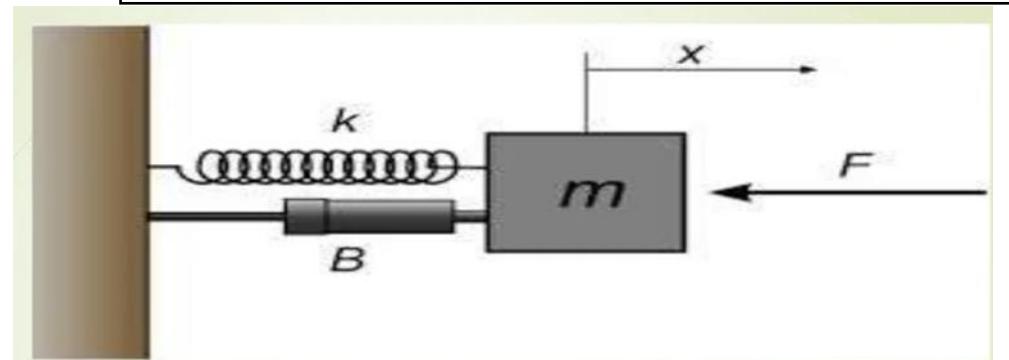
$$\begin{aligned}x_1(t) &= x(t) \\x_2(t) &= x'(t)\end{aligned}$$

$$\text{State Equation } \Rightarrow \begin{cases} x'_1 = x' = x_2 \\ x'_2 = x'' = \frac{1}{m}(-kx_1 - Bx_2) + \frac{1}{m}F \end{cases} \Rightarrow \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & B/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} F \Rightarrow \mathbf{x}' = A\mathbf{x} + BF$$

$$\text{output Equation } \Rightarrow \{y = x = x_1\} \Rightarrow [y] = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]F \Rightarrow \mathbf{y} = C\mathbf{x} + Du$$

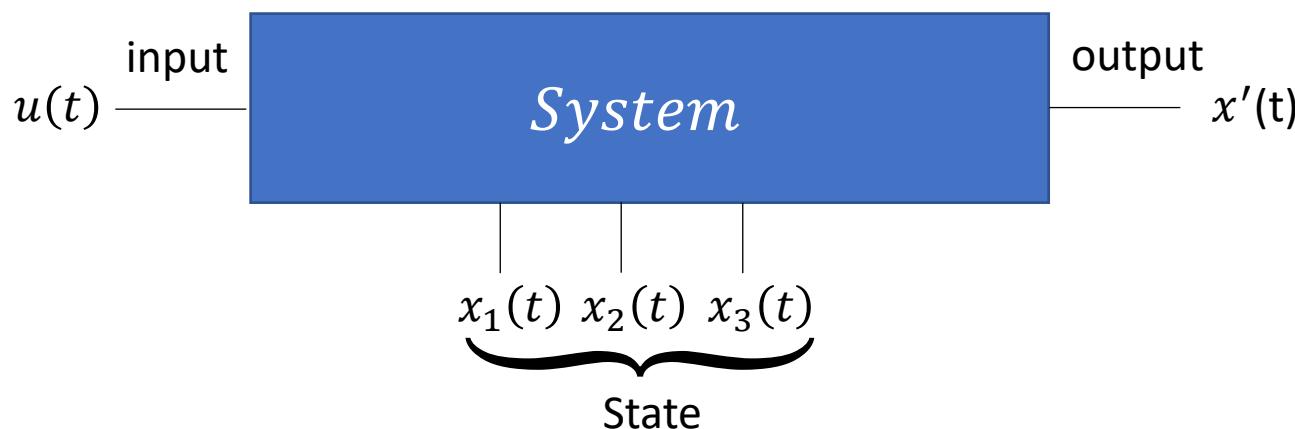
$$\mathbf{x}' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ -k/m & B/m \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}, \quad C = [1 \quad 0], \quad D = [0], \quad F = F$$

$$F(t) = mx''(t) + Bx'(t) + kx(t)$$



Implementation on Python Jupyter Notebook

$$\dot{x} = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad C = [1 \ 0 \ 0], \quad D = [0]$$



```
In [53]: # Converting Transfer function to State Space
sys7_ss = co.tf2ss(sys7_tf)
sys7_ss
```

$$\left(\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 \end{array} \right)$$

```
In [50]: import numpy as np
import matplotlib.pyplot as plt
import control as co

A = [[0, 1, 0], [0, 0, 1], [-1, -2, -3]]
B = [[0], [0], [1]]
C = [1, 0, 0]
D = 0

# State-Space Model
sys7 = co.ss(A, B, C, D)
sys7
```

$$\left(\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & -2 & -3 & 1 \\ \hline 1 & 0 & 0 & 0 \end{array} \right)$$

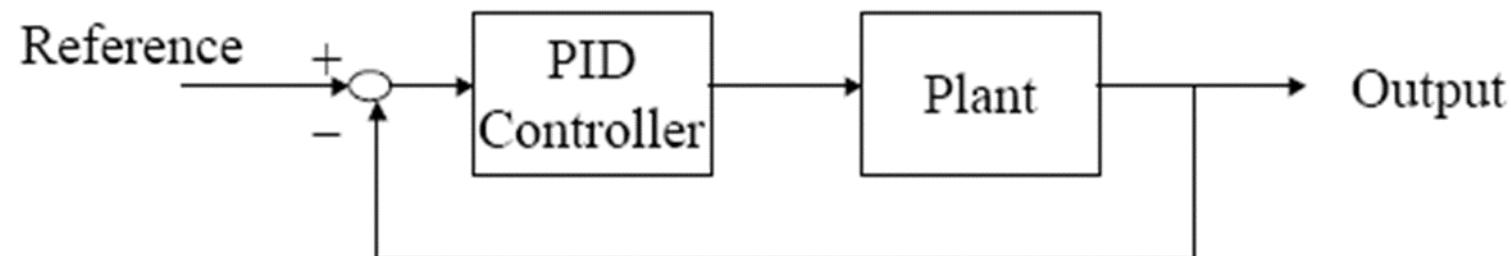
```
In [51]: # Convert from state-space model to Transfer function
sys7_tf = co.ss2tf(sys7)
sys7_tf
```

$$\frac{1}{s^3 + 3s^2 + 2s + 1}$$

Controllers for Control System

PID Controllers for Control System

- The controller is an element which accepts the error in some form and decides the proper corrective action.
- PID Stands for
 - P → Proportional
 - I → Integral
 - D → Derivative



Controllers for Control System

- The usefulness of PID controls lies in their general applicability to most control systems.
- In particular, when the mathematical model of the plant is not known and therefore analytical design methods cannot be used, PID controls prove to be most useful.
- In the field of process control systems, it is well known that the basic and modified PID control schemes have proved their usefulness in providing satisfactory control, although in many given situations they may not provide optimal control.

Controllers for Control System

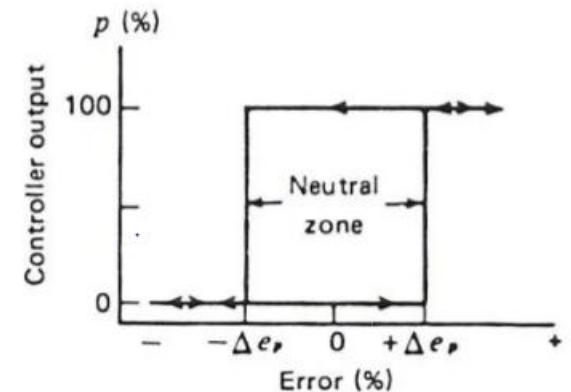
- It is interesting to note that more than half of the industrial controllers in use today are PID controllers or modified PID controllers.
- Because most PID controllers are adjusted on-site, many different types of tuning rules have been proposed in the literature.
- Using these tuning rules, delicate and fine tuning of PID controllers can be made on-site.

Four Modes of Controllers

- Each mode of control has specific advantages and limitations.
 - On-Off (Bang Bang) Control
 - Proportional (**P**)
 - Proportional plus Integral (**PI**)
 - Proportional plus Derivative (**PD**)
 - Proportional plus Integral plus Derivative (**PID**)

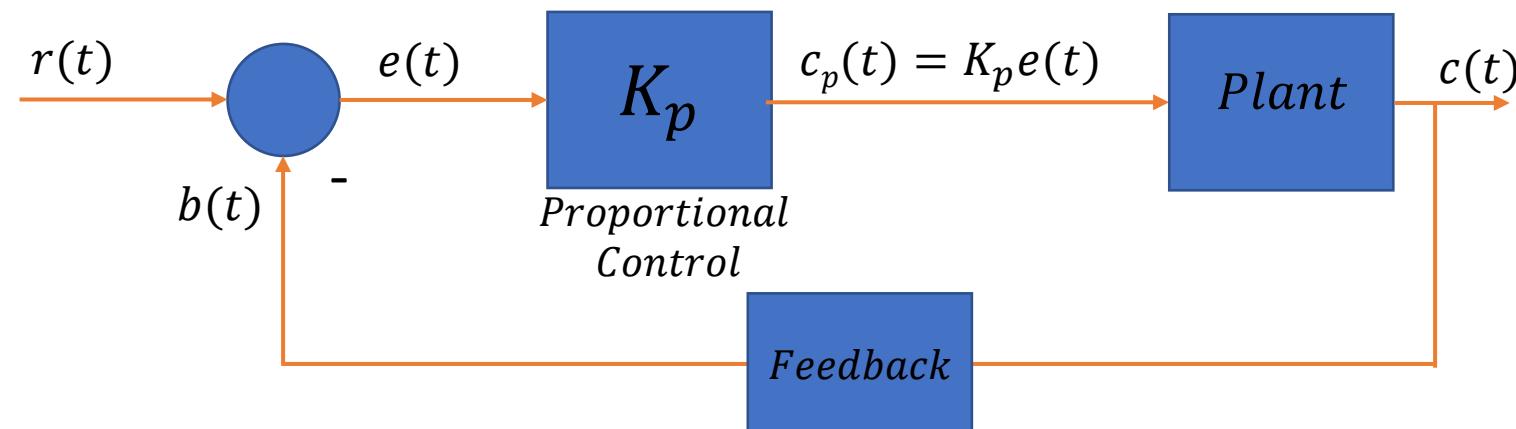
Two-position or On-off Control

- The **most elementary controller mode** is the two-position or ON/OFF controller mode.
- It is the simplest, cheapest.
- The most general form can be given by:
- Ex. ON/OFF switch
$$\begin{cases} p = 0\% & ep < 0 \\ p = 100\% & ep > 0 \end{cases}$$
- The relation shows that when the measured value is less than the set-point (i.e. $ep>0$), the controller output will be full (i.e. 100%) or vice versa.



Proportional Controller (P)

- In *proportional* mode, there is a continuous linear relation between value of the controlled variable and position of the final control element.



- Output of proportional controller is

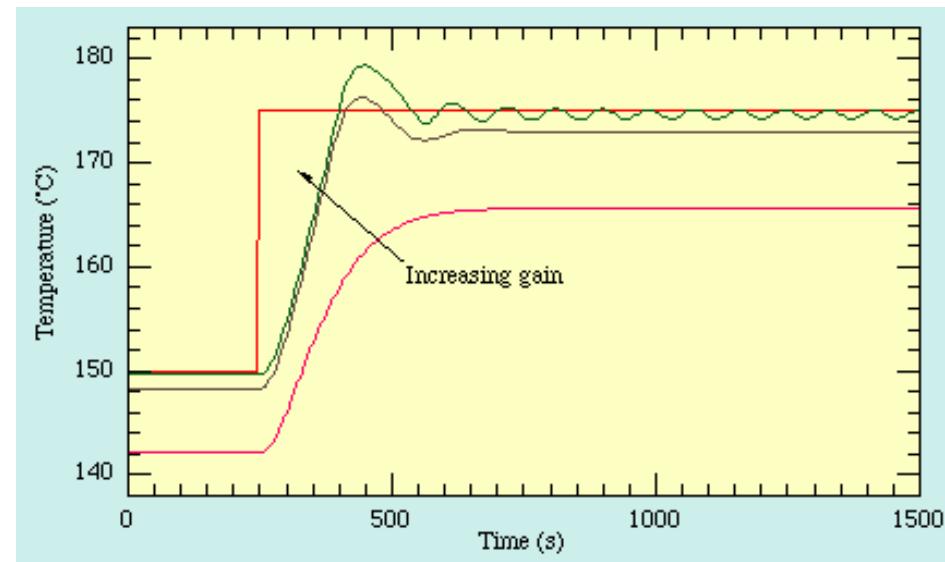
$$c_p(t) = K_p e(t)$$

Proportional Gain constant

$$\frac{C_p(s)}{E(s)} = K_p$$

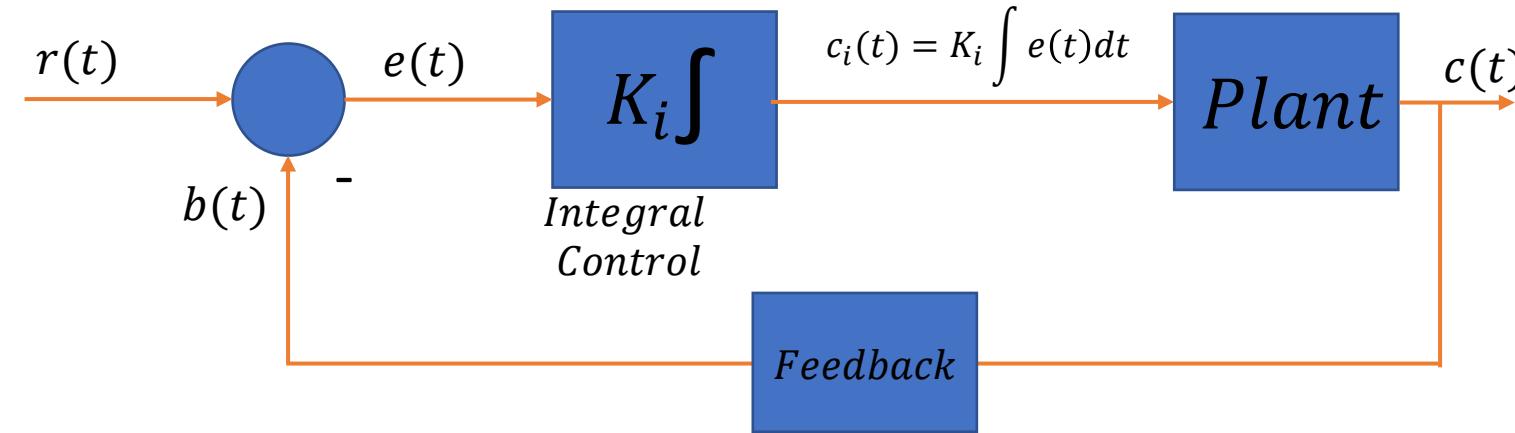
Proportional Controller (P)

- As the gain is increased the system responds faster to changes in set-point but becomes **progressively underdamped and eventually unstable**.
- Proportional controller reduce error but cannot go to zero. It finally **produces an offset error**. It can not adapt with the changing load conditions. To avoid this, another control mode is often used in the control system which is based on the history of errors. This mode is called integral mode.



Integral Controllers (I)

- Integral control describes a controller in which the **output rate of change is dependent on the magnitude of the input**.



- Specifically, a smaller amplitude input causes a slower rate of change of the output.

Integral Controllers (I)

- The output of the integral controller

$$c_i(t) = K_i \int e(t) dt$$

↑
Integral constant

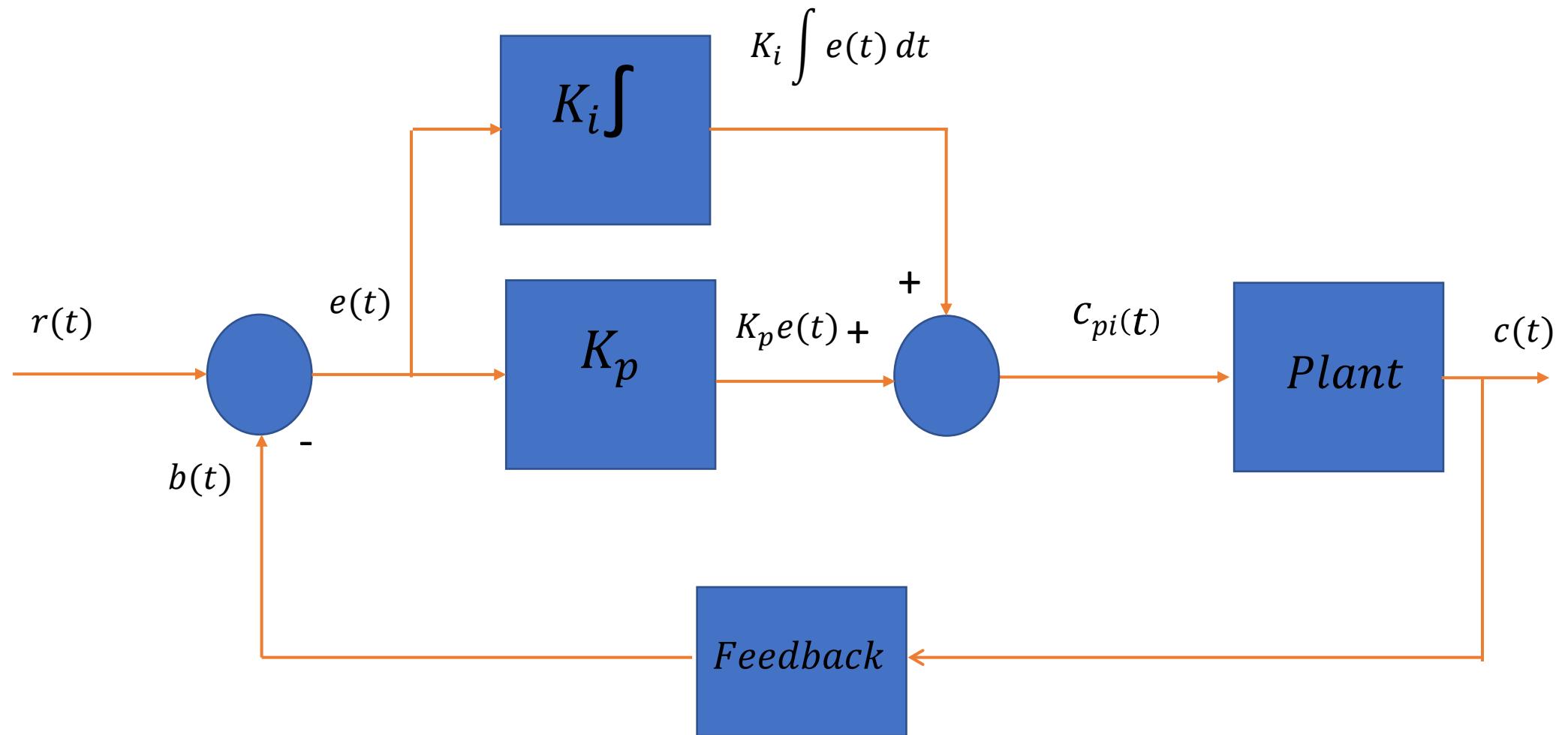
- The transfer function can be written as

$$\frac{C_i(s)}{E(s)} = K_i \frac{1}{s}$$

Integral Controllers (I)

- The major advantage of integral controllers is that they have the unique ability to return the controlled variable back to the exact set point following a disturbance.
- Disadvantages of the integral control mode are that it responds relatively slowly to an error signal and that it can initially allow a large deviation at the instant the error is produced.
- This can lead to system instability and cyclic operation. For this reason, the integral control mode is not normally used alone, but is combined with another control mode.

Proportional Plus Integral Controllers (PI)



Proportional Plus Integral Controllers (PI)

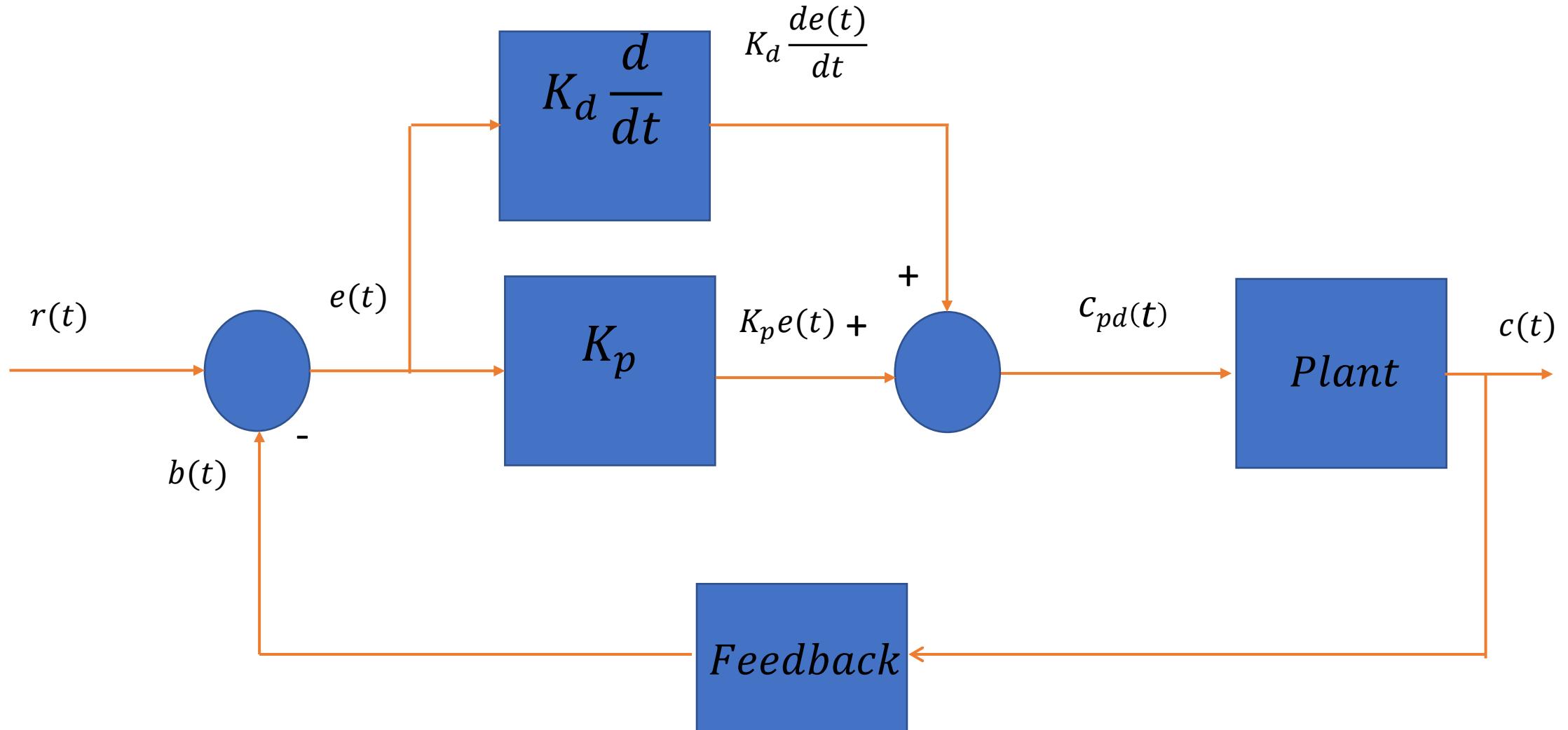
- The output of the Proportional plus integral controller

$$c_{pi}(t) = K_p e(t) + K_i \int e(t) dt$$

- The transfer function can be written as

$$\frac{C_{pi}(s)}{E(s)} = K_p + K_i \frac{1}{s}$$

Proportional Plus Derivative Controllers (PD)



Proportional Plus Derivative Controllers (PD)

- The output of the Proportional plus derivative controller

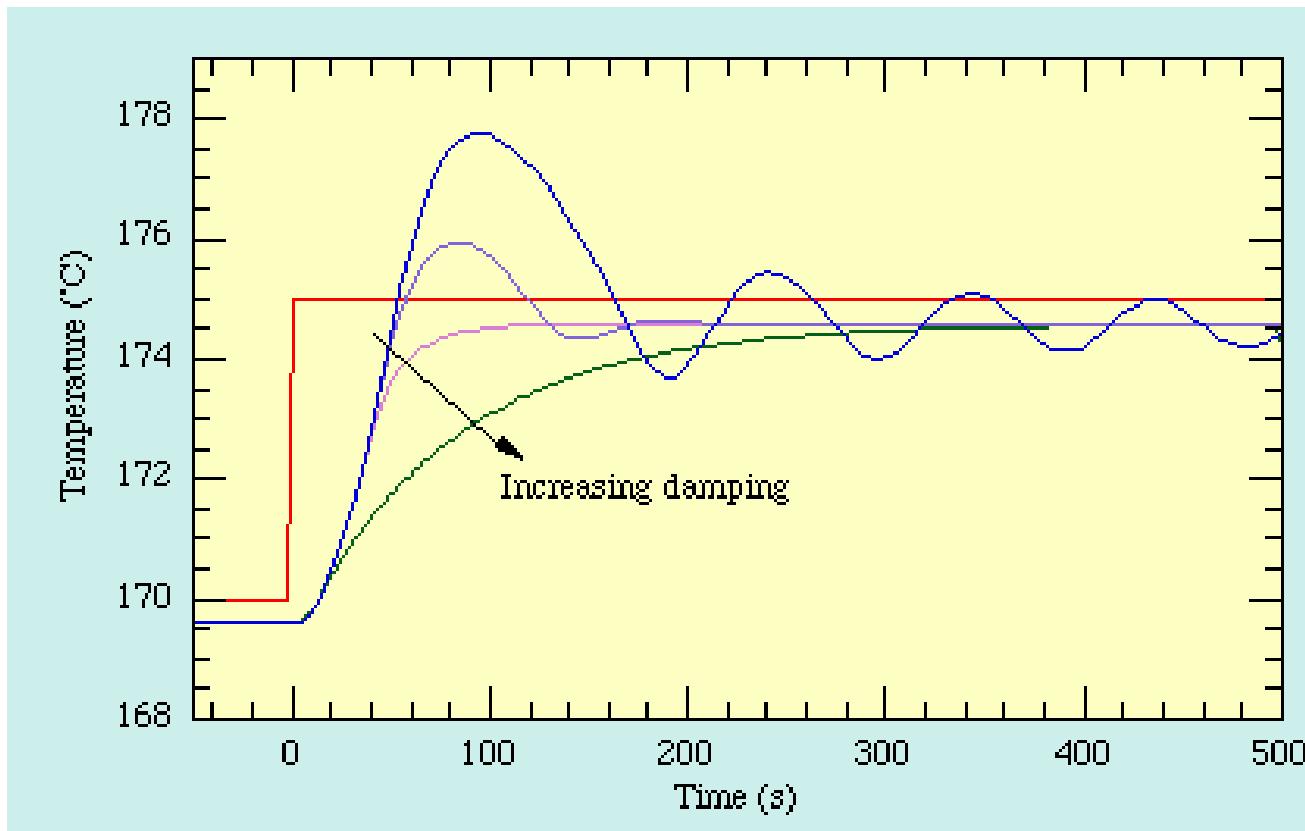
$$c_{pd}(t) = K_p e(t) + K_d \frac{de(t)}{dt}$$

- The transfer function can be written as

$$\frac{C_{pd}(s)}{E(s)} = K_p + K_d s$$

Proportional Plus Derivative Controllers (PD)

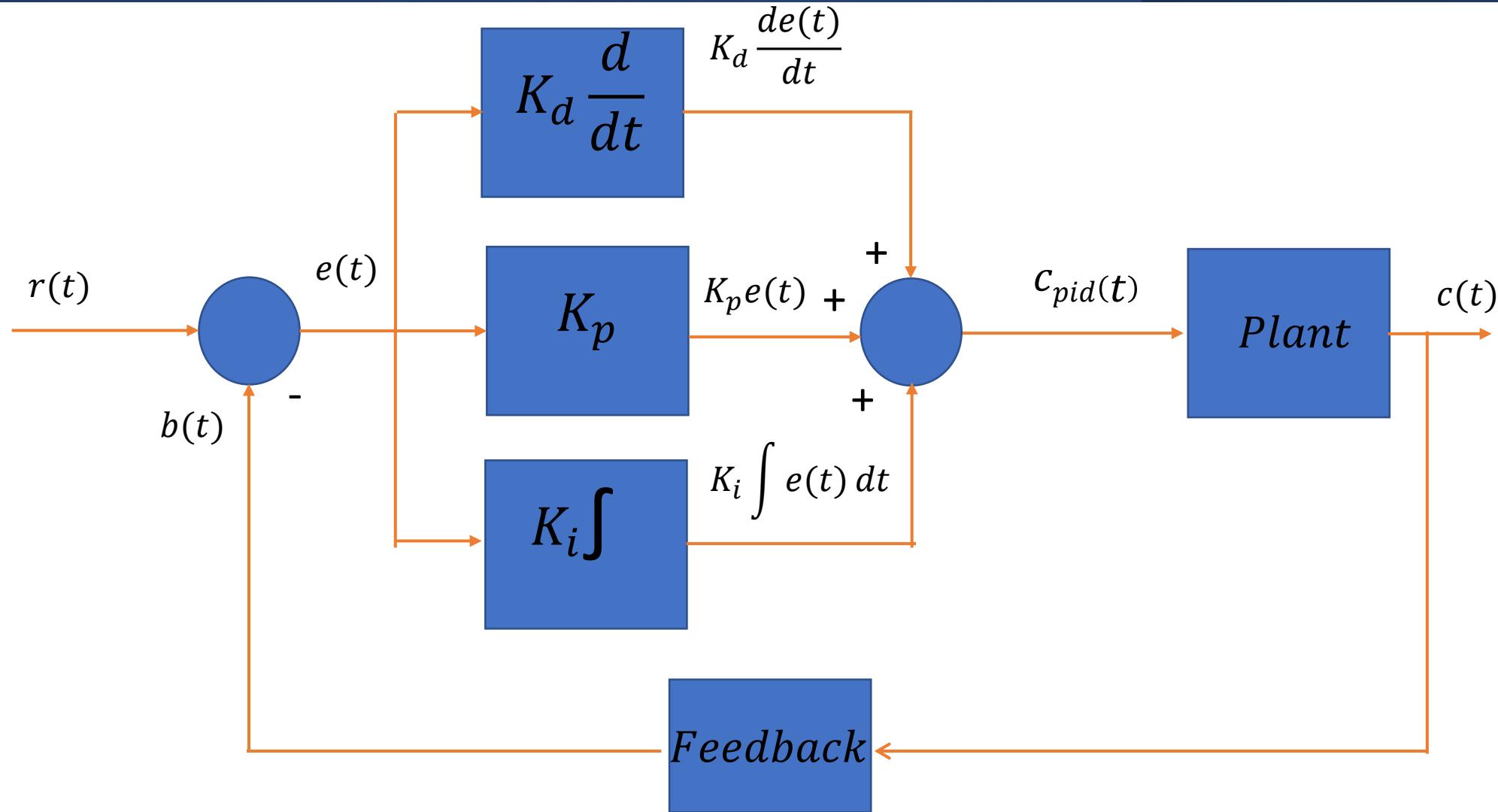
- The **stability** and **overshoot problems** that arise when a **proportional controller** is used at high gain can be mitigated by adding a term proportional to the time-derivative of the error signal. The value of the damping can be adjusted to achieve a critically damped response.



Proportional Plus Derivative Controllers (PD)

- The **higher the error signal rate of change**, the sooner the final control element is positioned to the desired value.
- The added **derivative action reduces initial overshoot** of the measured variable, and therefore **aids in stabilizing the process sooner**.
- This control mode is called proportional plus derivative (PD) control because the **derivative section responds to the rate of change of the error signal**.

Proportional Plus Integral Plus Derivative Controllers (PID)



Proportional Plus Integral Plus Derivative Controllers (PID)

- The output of the Proportional plus integral plus derivative controller

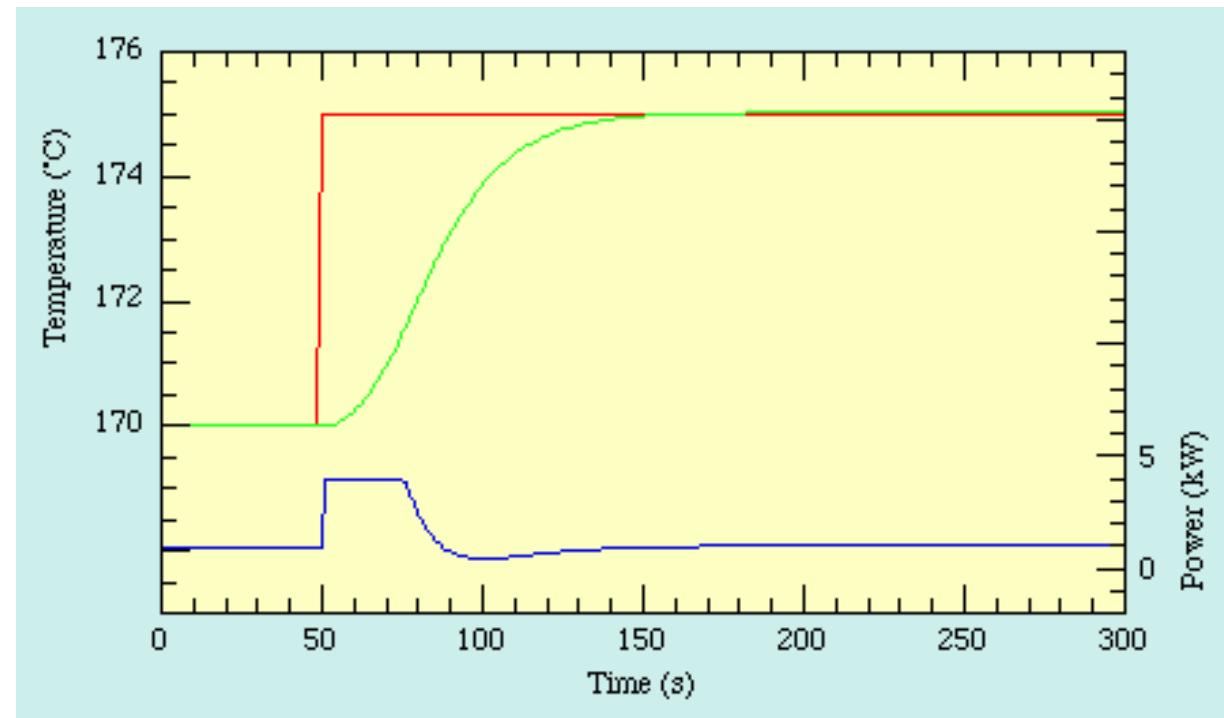
$$c_{pid}(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$$

- The transfer function can be written as

$$\frac{C_{pid}(s)}{E(s)} = K_p + K_i \frac{1}{s} + K_d s$$

Proportional Plus Integral Plus Derivative Controllers (PID)

- Although PD control deals neatly with the overshoot and ringing problems associated with proportional control it does not cure the problem with the steady-state error. Fortunately, it is possible to eliminate this while using relatively low gain by adding an integral term to the control function which becomes



Tips for Designing a PID Controllers

1. Obtain an open-loop response and determine what needs to be improved.
 2. Add a **proportional control** to **improve the rise time** .
 3. Add a **derivative control** to **improve the overshoot**.
 4. Add an **integral control** to eliminate the **steady-state error**.
 5. Adjust each of K_p , K_i , and K_d until you obtain a desired overall response.
-
- Lastly, please keep in mind that you do not need to implement all three controllers (proportional, derivative, and integral) into a single system, if not necessary. For example, if a PI controller gives a good enough response, then you don't need to implement derivative controller to the system. Keep the controller as simple as possible.

Stability of the Control System

Stability of the Control System

Transfer functions can usually be expressed as the ratio of two polynomials in the complex variable, s as

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \quad (n \geq m)$$

- When order of the denominator polynomial is greater than the numerator polynomial the transfer function is said to be '**proper**'.
- Otherwise '**Improper**'.
- Transfer function helps us to **check the stability of the control system**.

Stability of the Control System

Transfer functions can usually be expressed as the ratio of two polynomials in the complex variable, s as

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \quad (n \geq m)$$

Zeros: Zeros of a Transfer function are the frequencies (**value of s**) for which the **numerator** of the Transfer Function becomes **zero**. *Zeros: $s = z_1, z_2, \dots z_m$*

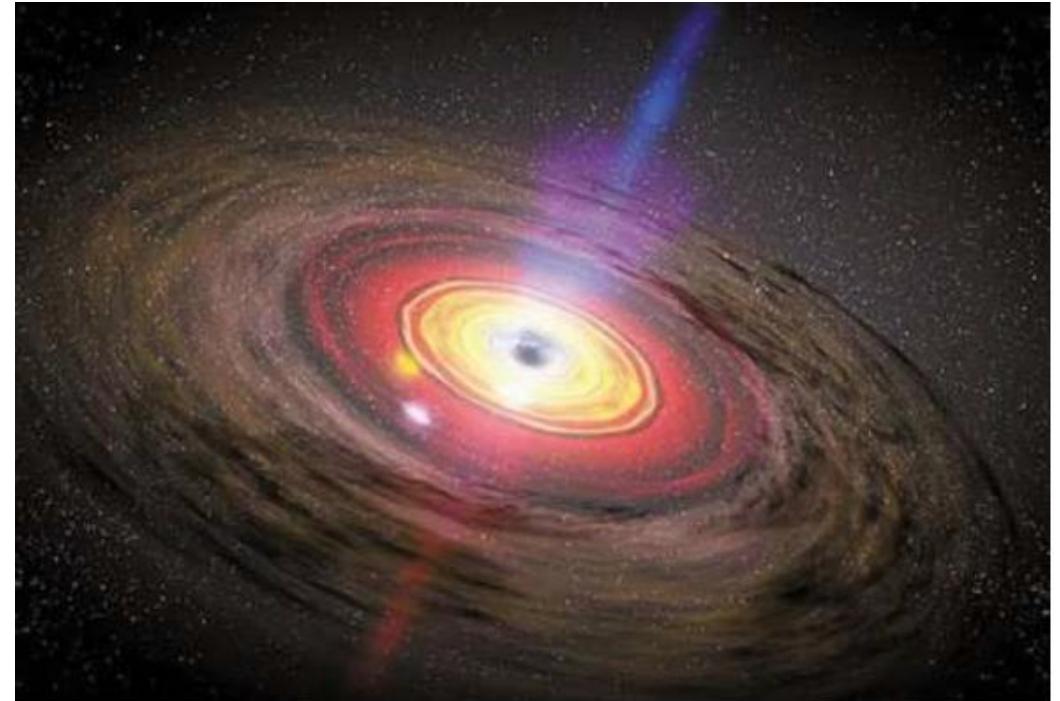
Poles: Poles of a Transfer function are the frequencies (**value of s**) for which the **denominator** of the Transfer Function becomes **zero**. *Poles: $s = P_1, P_2, \dots P_n$*

Stability of the Control System

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \quad (n \geq m)$$

Poles is also defined as the “ it is the frequency at which system becomes infinite”. Hence the name pole where field is infinite. And **zeros** is the frequency at which system becomes 0.

Like a magnetic pole or black hole.



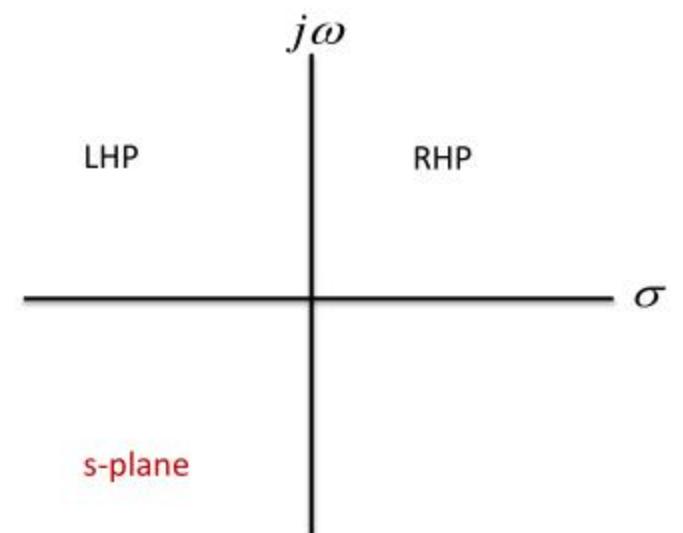
Stability of the Control System

Transfer functions can usually be expressed as the ratio of two polynomials in the complex variable, s as

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \dots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \dots + a_{n-1} s + a_n} \quad (n \geq m)$$

Pole-Zero Diagram: Plot on s-plane which represents the location of Poles and Zeros of a Transfer Function. Each **pole is marked** on s-plane using the **cross-symbol x**, while each **zero is marked** using the small **circle symbol o**.

System order is always equal to number of poles of the transfer function.

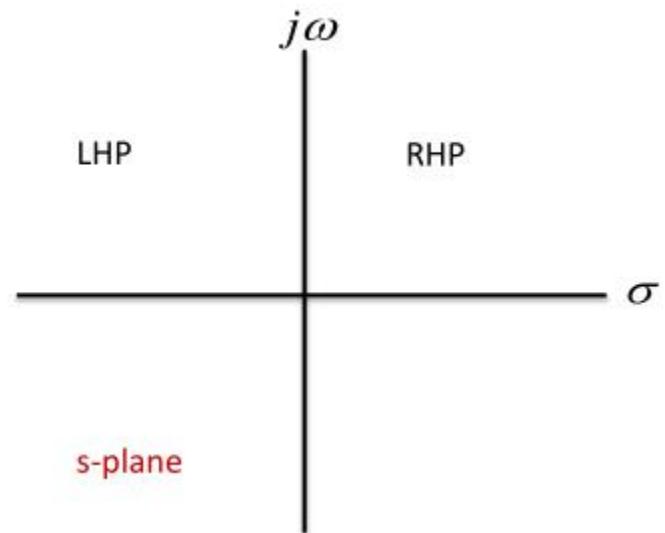


Stability of Control System

Transfer functions can usually be expressed as the ratio of two polynomials in the complex variable, s as

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \dots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \dots + a_{n-1} s + a_n} \quad (n \geq m)$$

- If all the poles of the system lie in left half plane the system is said to be **Stable**.
- If any of the poles lie in right half plane the system is said to be **unstable**.
- If poles lie on imaginary axis the system is said to be **marginally stable**.
- Absolute stability does not depend on location of zeros of the transfer function.



Poles, Zeros and the S-plane

An Example: You are given the following transfer function. Find the poles and zeros in the s-plane.

$$G(s) = \frac{(s + 8)(s + 14)}{s(s + 4)(s + 10)}$$

The roots of the numerator polynomial are called zeros.

$$s + 8 = 0 \Rightarrow s = -8$$

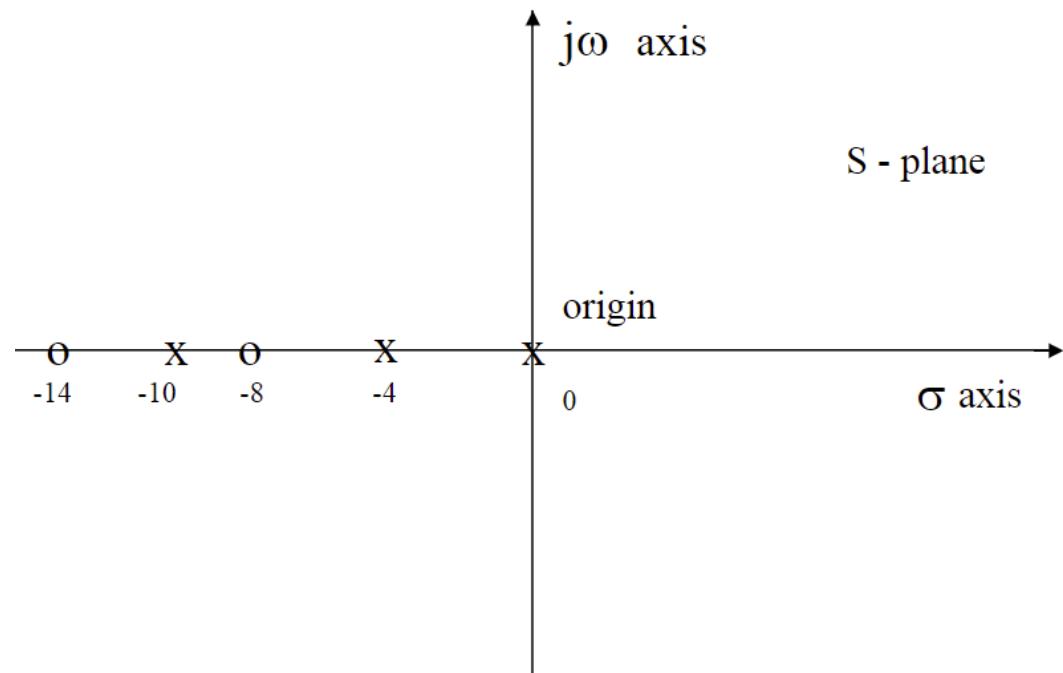
$$s + 14 = 0 \Rightarrow s = -14$$

The roots of the denominator polynomial are called poles.

$$s = 0$$

$$s + 4 = 0 \Rightarrow s = -4$$

$$s + 10 = 0 \Rightarrow s = -10$$



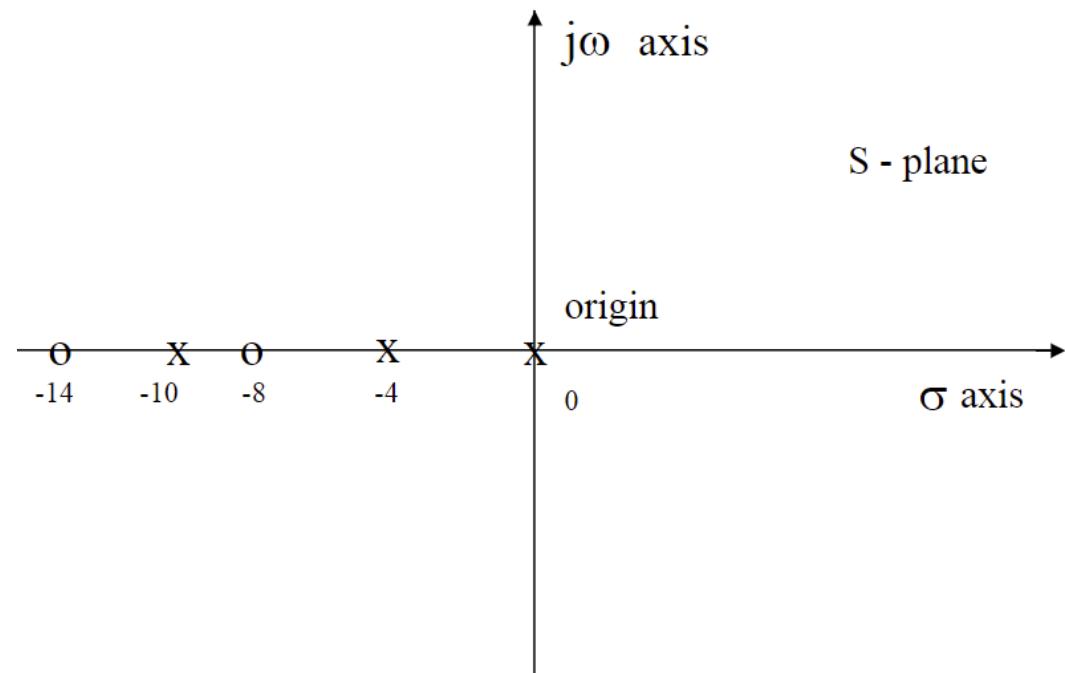
Stability of the Control System

An Example: You are given the following transfer function. Find the poles and zeros in the s-plane.

$$G(s) = \frac{(s + 8)(s + 14)}{s(s + 4)(s + 10)}$$

Stability Criteria based on Poles

- If all the poles of the system lie in left half plane the system is said to be **Stable**.
- If any of the poles lie in the right half plane the system is said to be **unstable**.
- If poles lie on imaginary axis the system is said to be **marginally stable**



Implementation in Python

```
In [19]: import numpy as np  
import matplotlib.pyplot as plt  
import control as co
```

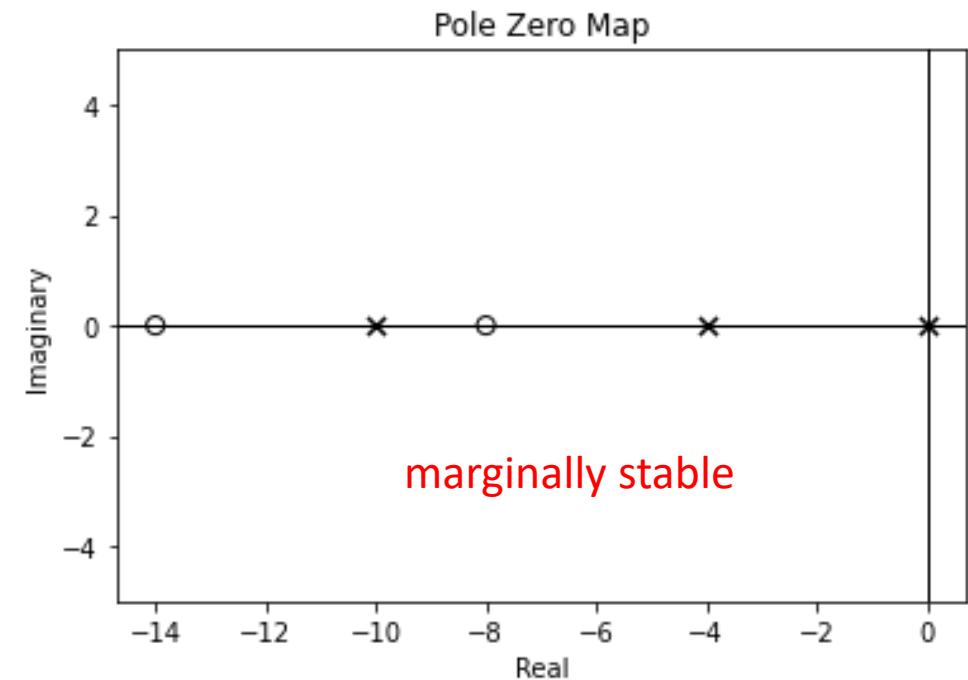
```
# Define Transfer Function  
num_p = np.array([1,22,112])  
den_p = np.array([1,14,40,0])  
G_s = co.tf(num_p, den_p)  
print('G(s) = ', G_s)
```

```
G(s) =  
s^2 + 22 s + 112  
-----  
s^3 + 14 s^2 + 40 s
```

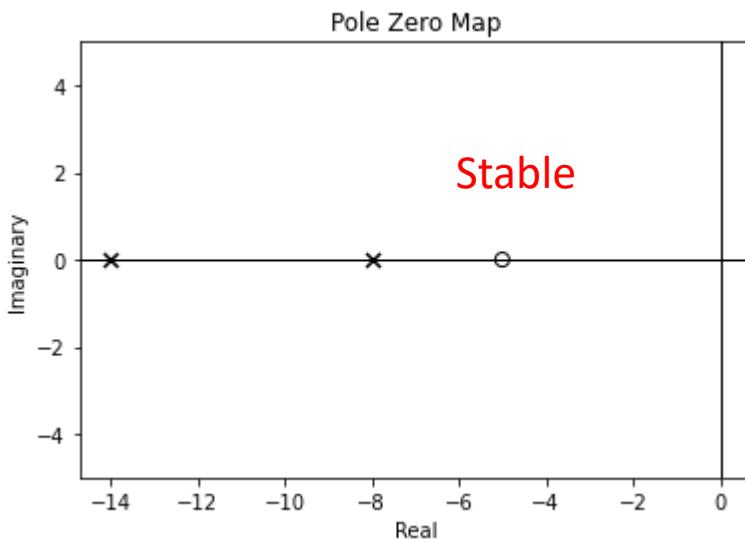
```
In [20]: # Poles and Zeros  
plt.figure(3)
```

```
co.pzmap(G_s)  
p = co.pole(G_s)  
z = co.zero(G_s)
```

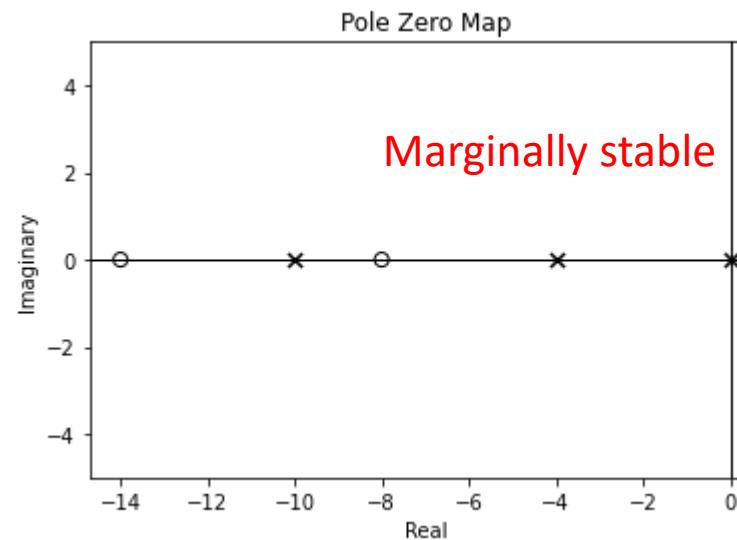
$$G(s) = \frac{(s + 8)(s + 14)}{s(s + 4)(s + 10)}$$



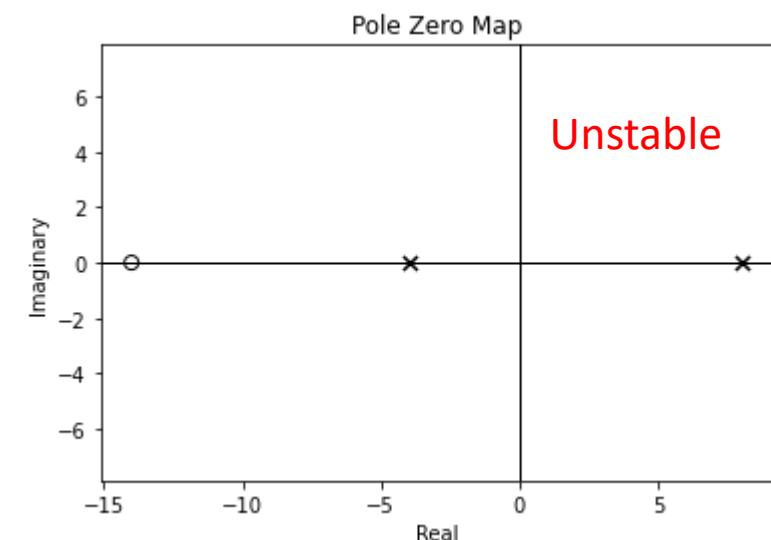
Examples



$$G(s) = \frac{s + 5}{s^2 + 22s + 112}$$



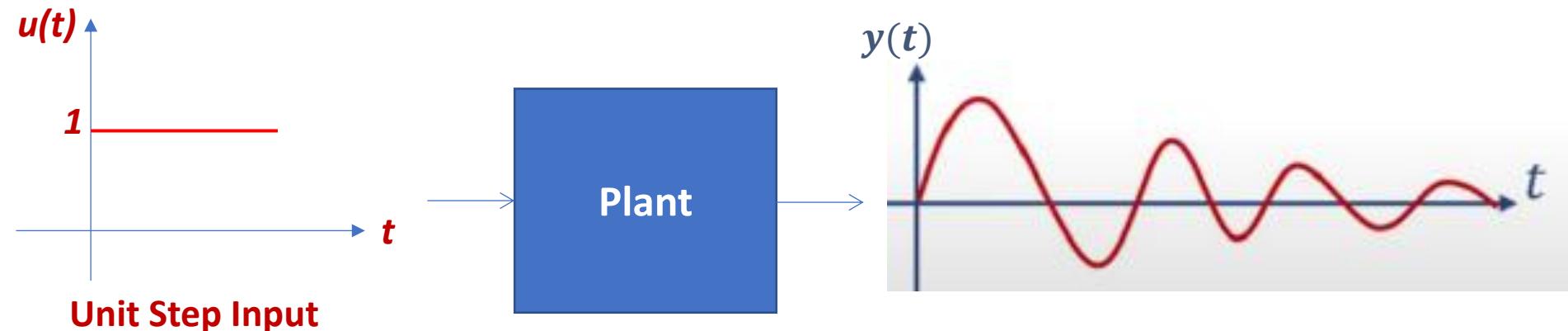
$$G(s) = \frac{s^2 + 22s + 112}{s^3 + 14s^2 + 40s}$$



$$G(s) = \frac{s + 14}{s^2 - 4s - 32}$$

Stability based on Step Response

- The system is said to be **stable** if for any **bounded input** the **output** of the system is also **bounded** (BIBO).
- Thus for any bounded input the output either remain constant or decrease with time.

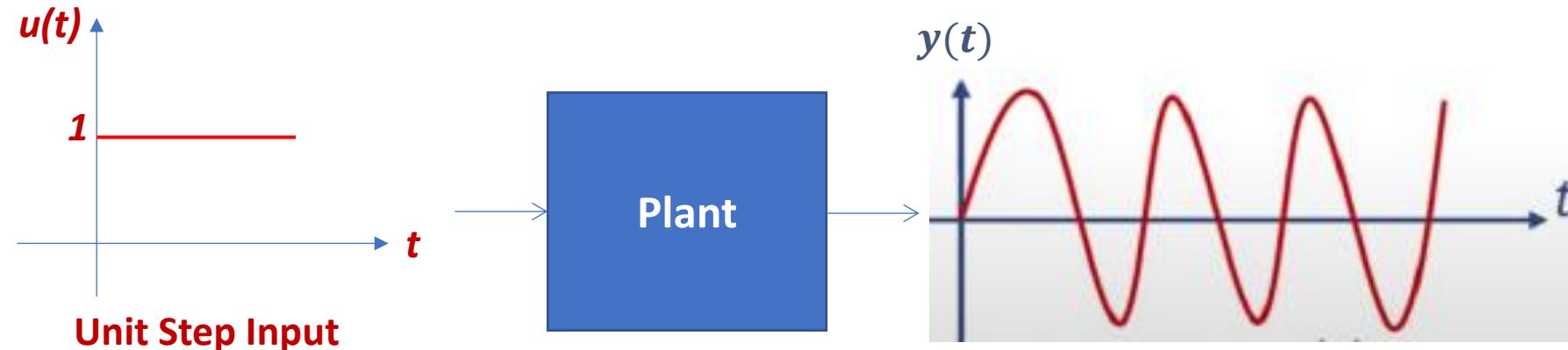


Stable System

$$\lim_{t \rightarrow \infty} y(t) = k$$

Stability based on Step Response

- The system is said to be **stable** if for any **bounded input** the **output** of the system is also **bounded** (BIBO).
- Thus for any bounded input the output either remain constant or decrease with time.

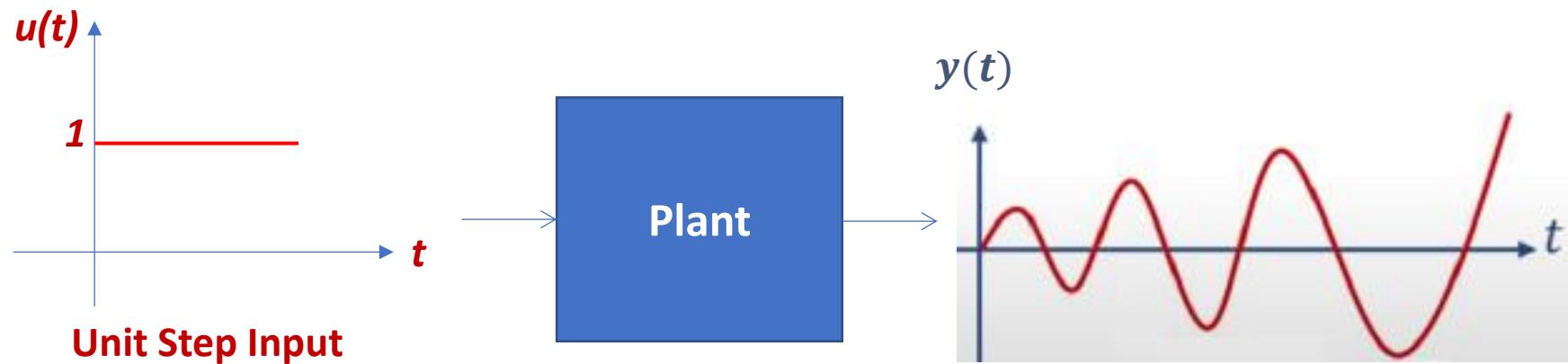


Marginally Stable System

$$0 < \lim_{t \rightarrow \infty} y(t) < \infty$$

Stability based on Step Response

- If for any **bounded input** the **output is unbounded** the system is said to be **unstable**.

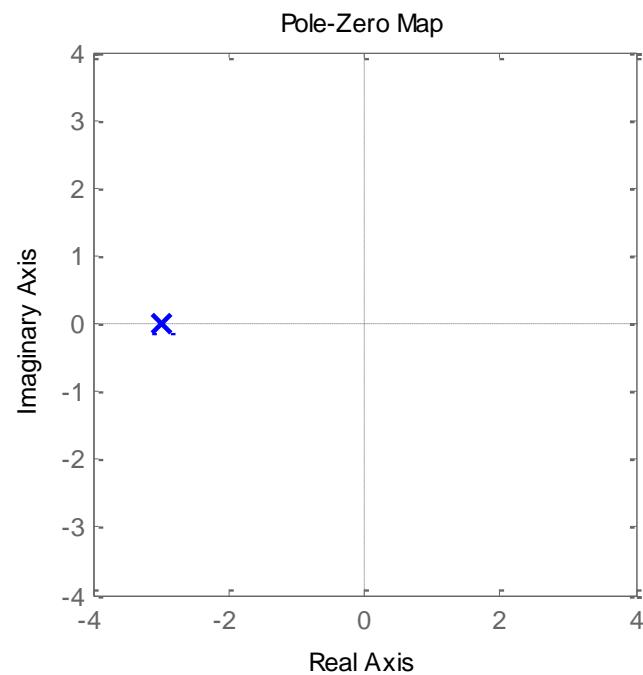


Unstable System

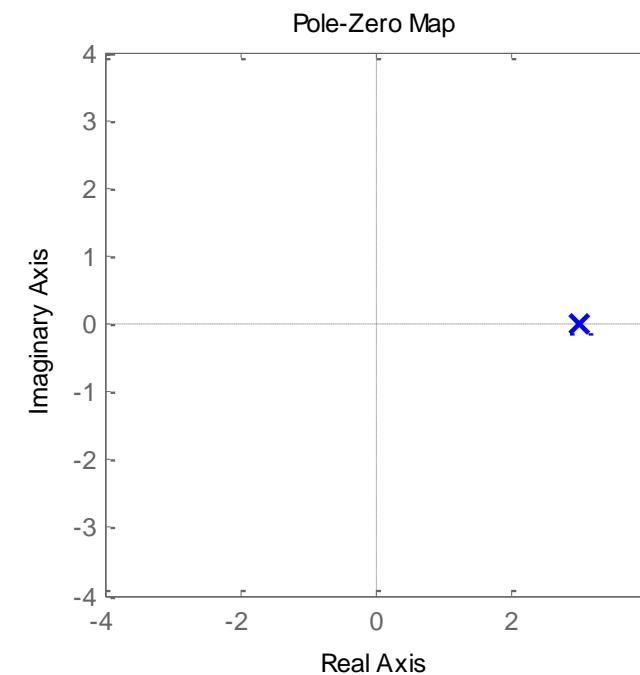
$$\lim_{t \rightarrow \infty} y(t) = \infty$$

Example: BIBO vs Transfer Function

$$G_1(s) = \frac{Y(s)}{U(s)} = \frac{1}{s + 3}$$



$$G_2(s) = \frac{Y(s)}{U(s)} = \frac{1}{s - 3}$$



Example: BIBO vs Transfer Function

$$G_1(s) = \frac{Y(s)}{U(s)} = \frac{1}{s + 3}$$

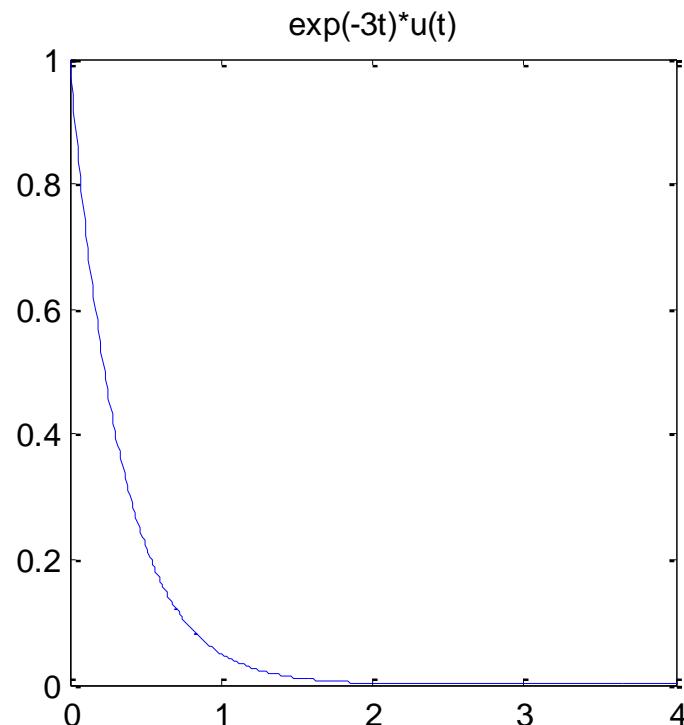
$$G_2(s) = \frac{Y(s)}{U(s)} = \frac{1}{s - 3}$$

$$\begin{aligned}\mathcal{L}^{-1}G_1(s) &= \mathcal{L}^{-1}\frac{Y(s)}{U(s)} = \mathcal{L}^{-1}\frac{1}{s + 3} \\ &= y(t) = e^{-3t}u(t)\end{aligned}$$

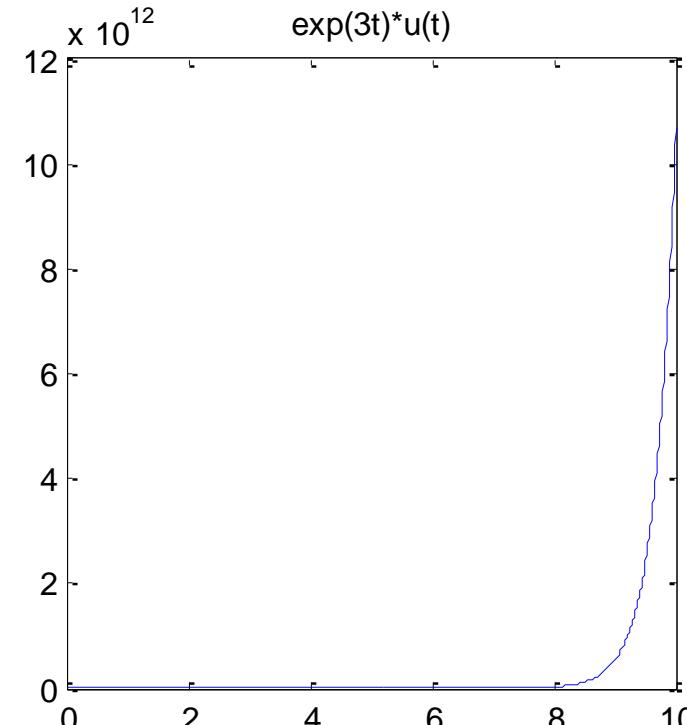
$$\begin{aligned}\mathcal{L}^{-1}G_2(s) &= \mathcal{L}^{-1}\frac{Y(s)}{U(s)} = \mathcal{L}^{-1}\frac{1}{s - 3} \\ &= y(t) = e^{3t}u(t)\end{aligned}$$

Example: BIBO vs Transfer Function

$$y(t) = e^{-3t} u(t)$$



$$y(t) = e^{3t} u(t)$$



Summary

- Review the fundamentals of control systems.
- Highlight the classification of control systems.
- Discuss various mathematical models for control systems.
- Importance of controllers in control systems.
- Stability of the control system.

Further Resources

- <https://python-control.readthedocs.io/en/0.9.1/intro.html>
- [Ogata, Katsuhiko. Modern control engineering. Vol. 5. Upper Saddle River, NJ: Prentice hall, 2010.](#)

IDAT7215

Computer Programming for Product Development and Applications

Lecture 2-3: Python Libraries: SciPy

Dr. Zulfiqar Ali

Outline

- What is Python SciPy
- NumPy vs SciPy
- Sub-Packages in SciPy
- Basic Functions
- Special Functions
- Integration Functions
- Fourier Transformations
- Linear Algebra
- Optimization Function
- Interpolation Functions
- Curve Fitting Function

What is Python SciPy

- SciPy is a python library used to solve scientific and mathematical problems
- Built on NumPy
- Allows manipulation and visualizing



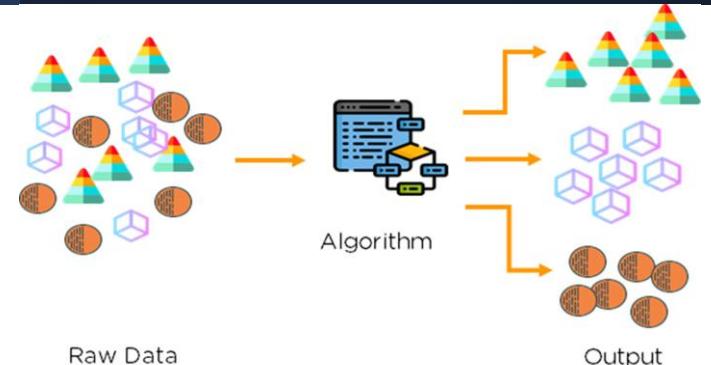
NumPy vs SciPy

- NumPy and SciPy used for mathematical and numerical analysis
- NumPy contains array data and basic operations such as sorting indexing etc.
- SciPy consists of all the numerical code.
- SciPy contains fully-featured versions of mathematical and scientific functions.

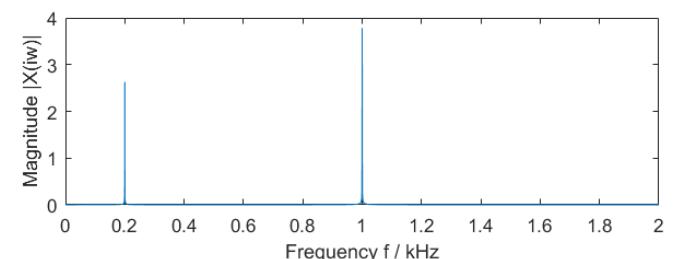
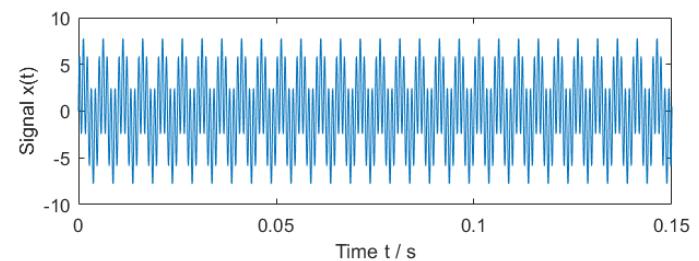


Sub-Packages in SciPy

Name	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear Algebra
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root finding
signal	Signal Processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and algorithms
special	Special functions
stats	Statistical distributions and functions



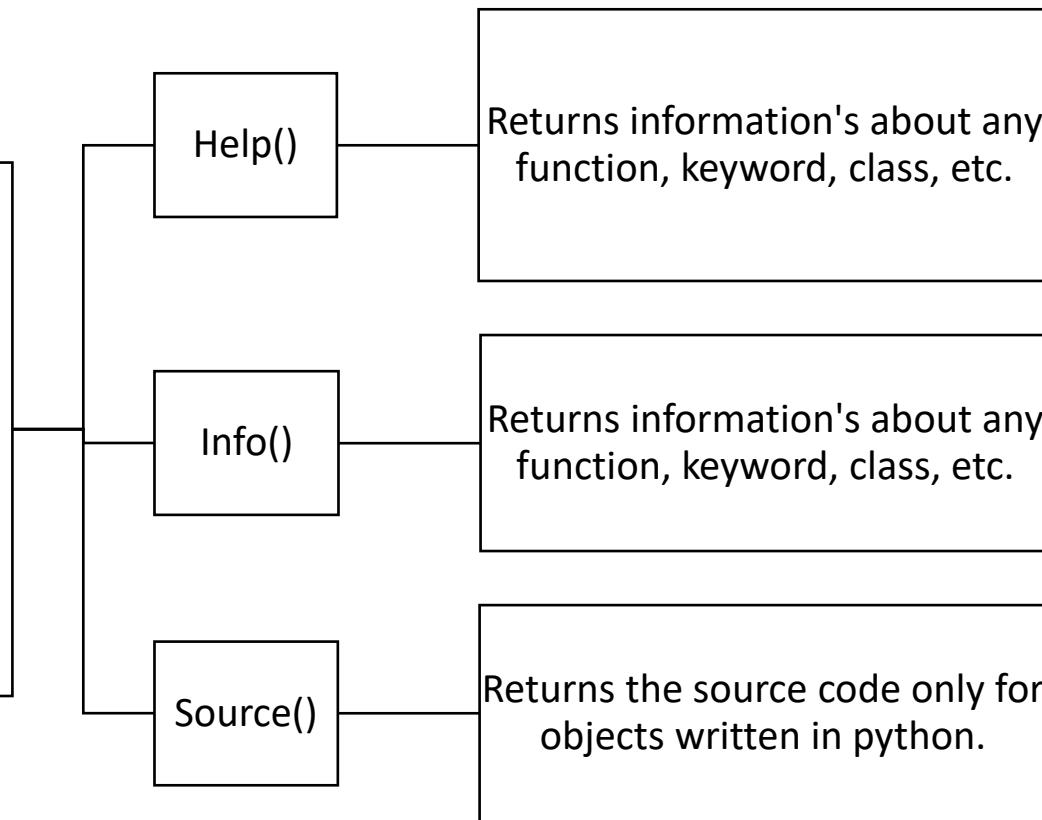
Clustering of Raw data



Fourier Transform

Basic Functions

Basic Functions



```
from scipy import cluster
```

```
help(cluster)
```

```
Help on package scipy.cluster in scipy:
```

NAME

```
scipy.cluster
```

DESCRIPTION

```
=====
Clustering package (:mod:`scipy.cluster`)
=====
```

```
.. currentmodule:: scipy.cluster
:mod:`scipy.cluster.vq`
```

```
Clustering algorithms are useful in information theory, target detection,
communications, compression, and other areas. The `vq` module only
supports vector quantization and the k-means algorithms.
```

```
:mod:`scipy.cluster.hierarchy`
```

```
The `hierarchy` module provides functions for hierarchical and
agglomerative clustering. Its features include generating hierarchical
clusters from distance matrices,
calculating statistics on clusters, cutting linkages
to generate flat clusters, and visualizing clusters with dendograms.
```

PACKAGE CONTENTS

```
_hierarchy
_optimal_leaf_ordering
_vq
hierarchy
setup
tests (package)
vq
```

Basic Functions

```
:> from scipy import cluster  
  
:help(cluster)  
  
Help on package scipy.cluster in scipy:  
  
NAME  
    scipy.cluster  
  
DESCRIPTION  
=====  
Clustering package (:mod:`scipy.cluster`)  
=====  
  
.. currentmodule:: scipy.cluster  
  
:mod:`scipy.cluster.vq`  
  
Clustering algorithms are useful in information  
communications, compression, and other areas. The  
`vq` module supports vector quantization and the k-means al  
  
Clustering algorithms are useful in information the  
communications, compression, and other areas. The `vq`  
module supports vector quantization and the k-means algorithm.
```

```
:> import scipy  
scipy.info(cluster)  
  
=====  
Clustering package (:mod:`scipy.cluster`)  
=====  
  
.. currentmodule:: scipy.cluster  
  
:mod:`scipy.cluster.vq`  
  
Clustering algorithms are useful in information  
communications, compression, and other areas. The  
`vq` module supports vector quantization and the k-means al  
  
:mod:`scipy.cluster.hierarchy`
```

```
:> import scipy  
scipy.source(cluster)  
  
In file: D:\C\User\anaconda3\lib\site-packages\scipy\  
....  
=====  
Clustering package (:mod:`scipy.cluster`)  
=====  
  
.. currentmodule:: scipy.cluster  
  
:mod:`scipy.cluster.vq`  
  
Clustering algorithms are useful in information theor  
communications, compression, and other areas. The `vq`  
supports vector quantization and the k-means algorith  
  
:mod:`scipy.cluster.hierarchy`
```

Jupyter Notebook

Special Functions

- Functions available for Mathematical Physics.
- Some of these functions included gamma, beta, hypergeometric, parabolic cylinder etc.

Exponential Functions

$$a = 10^2$$



```
from scipy import special  
a = special.exp10(2)  
print(a)
```

100.0

$$b = 2^3$$



```
from scipy import special  
b = special.exp2(3)  
print(b)
```

8.0

Special Functions

- Functions available for Mathematical Physics.
- Some of these functions included gamma, beta, hypergeometric, parabolic cylinder etc.

Trigonometric Functions

$$c = \sin(90^\circ)$$



```
from scipy import special  
c = special.sindg(90)  
print(c)
```

1.0

$$d = \cos(90^\circ)$$

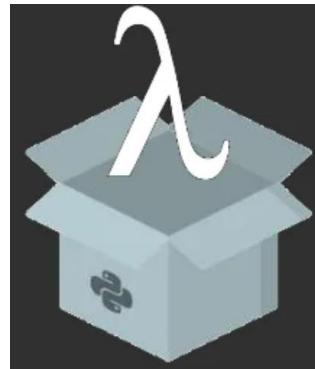


```
from scipy import special  
c = special.cosdg(90)  
print(c)
```

-0.0

Lambda Functions

- Lambda functions or ***anonymous functions***.
- ***One-time use***: Also known as throw-away function as they are needed just once.
- ***I/O of other Functions***: They are also passed as inputs or returned as outputs of other higher order functions.
- ***Reduce code size***: The body of the lambda function is written in a single line



$$f = x - y$$



```
# General function
def minus(x,y):
    return x-y
print(minus(9,4))
```

5

```
# Lambda function or anonymous functions
minus= lambda x,y: x-y
print(minus(9,4))
```

5

Integration Functions

- Integration deal with adding slices to determine the whole.
- Integration can be used to find displacement, area etc.

$$i = \int_0^1 10^x dx$$



General Integration

The **quad** function calculates the integral of a function which has one variable in SciPy.

```
from scipy import integrate  
  
i=integrate.quad(lambda x:special.exp10(x),0,1)  
  
print(i)
```

(3.9086503371292665, 4.3394735994897923e-14)

Integration Functions

- Integration deal with adding slices to determine the whole.
- Integration can be used to find displacement, area etc.

$$e = \int_{x=0}^{x=2} \int_{y=0}^{y=1} xy^2 dy dx$$

Double Integration

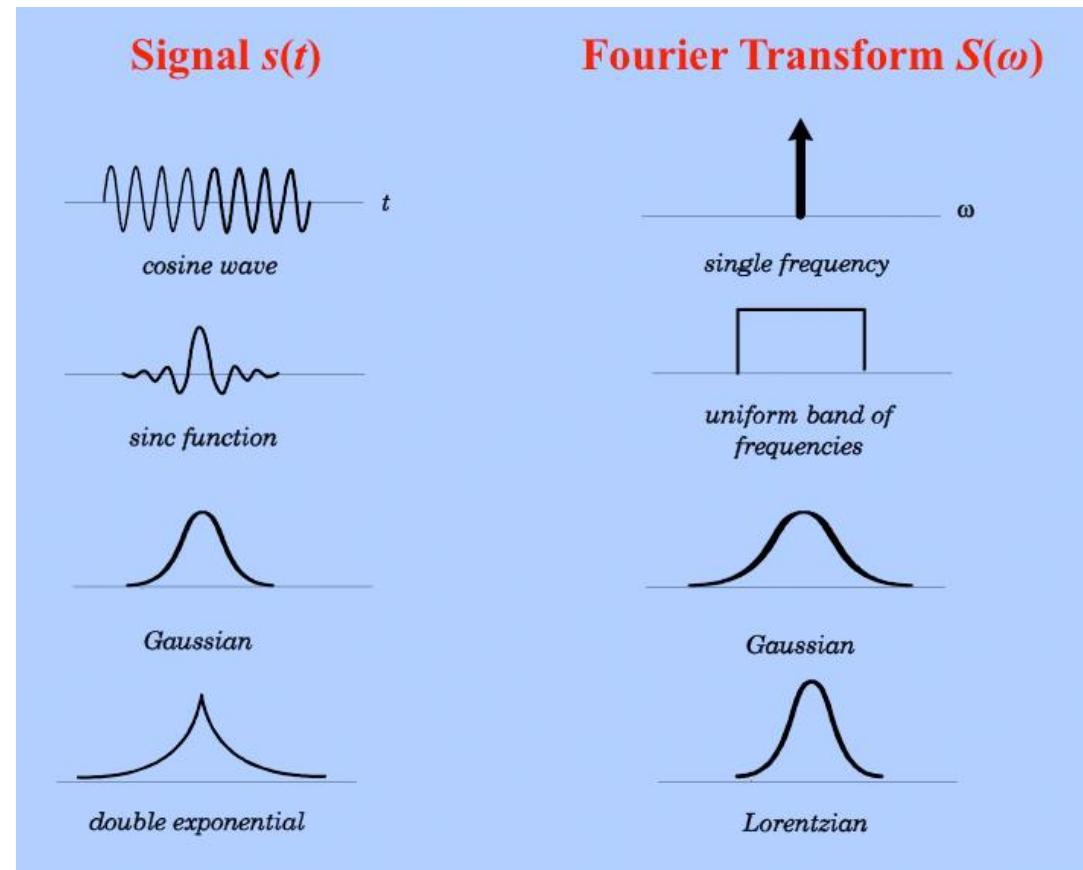
The `dblquad` function calculates the integral of a function which has two variables in SciPy.

```
# Double Integration
from scipy import integrate
f = lambda x, y: x*y**2
e = integrate.dblquad(f, 0, 2, 0, 1)
print(e)
```

(1.3333333333333335, 2.2108134835808843e-14)

Fourier Transformation

- Fourier analysis is a method that deals with expressing a function as a sum of periodic components and recovering the signal from those components.
- The **fft** and **ifft** functions can be used to return the discrete Fourier Transform of a real or complex sequence.



Fourier Transformation

- Fourier analysis is a method that deals with expressing a function as a sum of periodic components and recovering the signal from those components.

Fourier Transform

$x = [1 \ 2 \ 3 \ 4]$

```
from scipy.fftpack import fft,ifft
import numpy as np
x=np.array([1,2,3,4])
y=fft(x)
print(y)
```

[10.-0.j -2.+2.j -2.-0.j -2.-2.j]

```
: from scipy.fftpack import fft,ifft
x=np.array([1,2,3,4])
y=fft(x)
print(y)
xx=ifft(y)
print(xx)
```

[10.-0.j -2.+2.j -2.-0.j -2.-2.j]
[1.+0.j 2.+0.j 3.-0.j 4.+0.j]

- The **fft** and **ifft** functions can be used to return the discrete Fourier Transform of a real or complex sequence.

Linear Algebra

- SciPy is built on ATLAS LAPACK and BLAS libraries and is extremely fast in solving problems related to linear algebra
- ***Inverse of a Matrix:*** Inverse of a matrix A is the matrix B such that $AB = I$ where I is identity matrix and $B = A^{-1}$.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Inverse of Matrix

```
from scipy import linalg
import numpy as np
A = np.array([[1,2],[3,4]])
B = linalg.inv(A)
print('B = ', B)
```



```
B = [[-2.   1. ]
 [ 1.5 -0.5]]
```

Optimization

- SciPy have build in functions to solve the optimization problem.

Minimize $f(x) = (x - 3)^2$



```
from scipy.optimize import minimize
def f(x):
    return (x-3)**2
res = minimize(f,2)
res.x
array([2.9999999])
```

Minimization

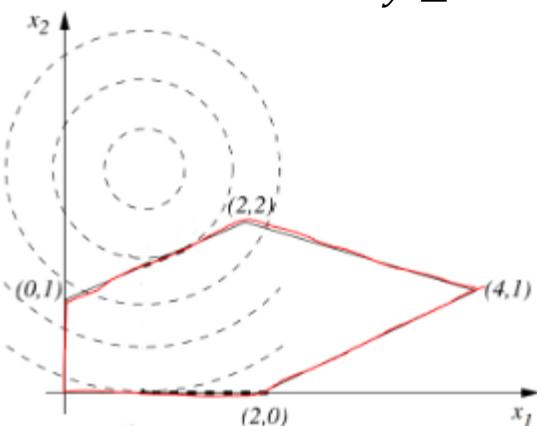
```
: from scipy.optimize import minimize
def f(x):
    return (x-3)**2
res = minimize(f,2)Initial Guess
res
fun: 5.551437397369767e-17
hess_inv: array([[0.5]])
jac: array([-4.3254289e-13])
message: 'Optimization terminated successfully.'
nfev: 6
nit: 2
njev: 3
status: 0
success: True
x: array([2.9999999])
```

Optimization

Minimize $f(x, y) = (x - 1)^2 + (y - 2.5)^2$

Subject to:

$$\begin{aligned}x - 2y + 2 &\geq 0 \\-x - 2y + 6 &\geq 0 \\-x + 2y + 2 &\geq 0 \\x &\geq 0 \\y &\geq 0\end{aligned}$$



- 2D function takes in vector x
- Constraints must be specified as $g_i(x) \geq 0$
- Bounds specified as rectangular

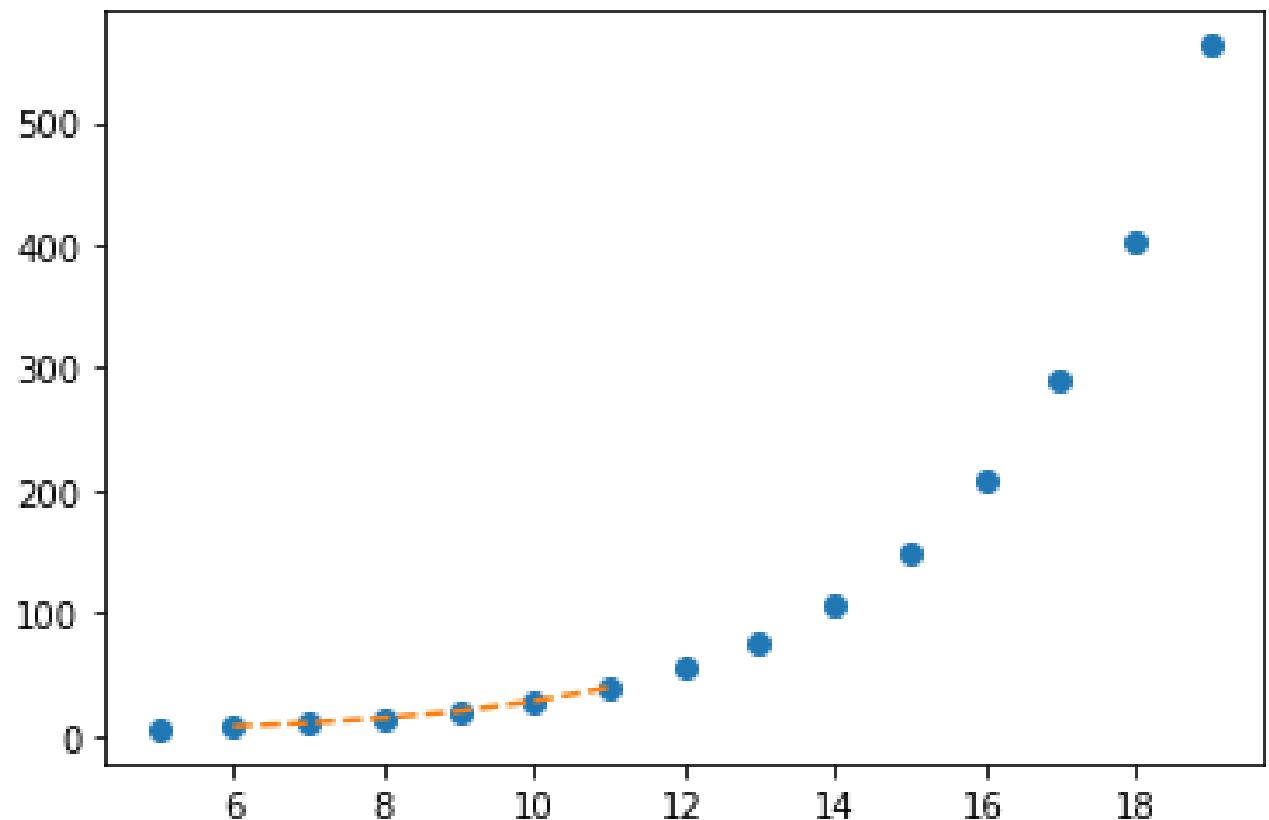
Minimization

```
: from scipy.optimize import minimize  
  
f = lambda x: (x[0] - 1)**2 + (x[1] - 2.5)**2  
cons = ({'type': 'ineq', 'fun': lambda x: x[0] - 2 * x[1] + 2},  
        {'type': 'ineq', 'fun': lambda x: -x[0] - 2 * x[1] + 6},  
        {'type': 'ineq', 'fun': lambda x: -x[0] + 2 * x[1] + 2})  
bnds = ((0, None), (0, None))  
res = minimize(f, (2, 0), bounds=bnds, constraints=cons)  
  
res.x  
  
: array([1.4, 1.7])
```

Initial Guess

Interpolation Functions

- **Interpolation** refers to constructing new data points within a set of known data points.
- The `scipy.interpolate` consists of spline functions and classes, one-dimensional and multi-dimensional (univariate and multivariate) interpolation classes, etc.

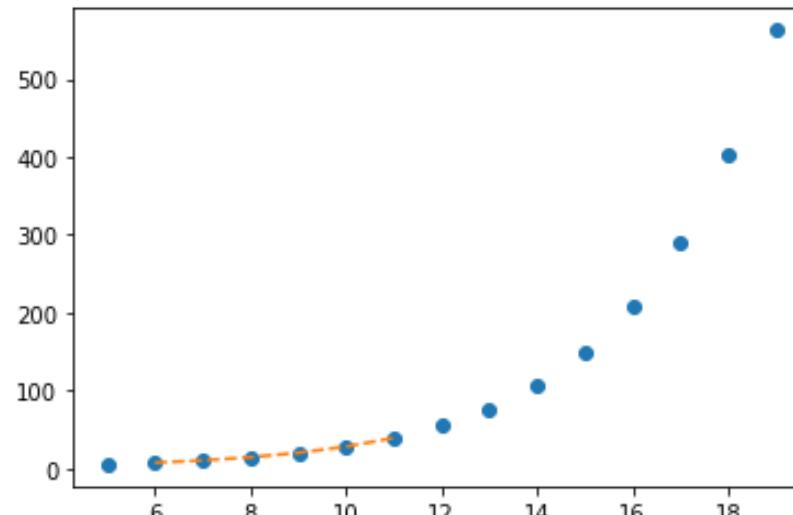


Interpolation Functions

- **Interpolation** refers to constructing new data points within a set of known data points.
- The **scipy.interpolate** consists of spline functions and classes, one-dimensional and multi-dimensional (univariate and multivariate) interpolation classes, etc.

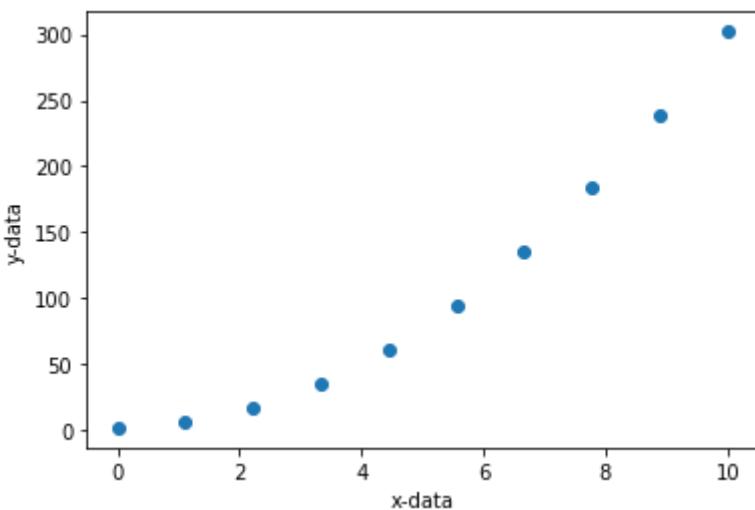
Interpolation

```
import matplotlib.pyplot as plt
from scipy import interpolate
x = np.arange(5,20)
y = np.exp(x/3.0)
f=interpolate.interp1d(x,y)
x1=np.arange(6,12)
y1=f(x1) # Use interpolation function returned by int
plt.plot(x,y, 'o', x1, y1, '--')
plt.show;
```



Curve Fitting

- Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points



Curve Fitting

$$y = ax^2 + b$$

Find a and b if x and y is given

```
# Want to fit the data to curve y = ax^2 + b. The main goal here is determine the values of a and b.
from scipy.optimize import curve_fit

x_data = np.linspace(0, 10, 10)
y_data = 3*x_data**2 + 2

def func(x, a, b):
    return a*x**2 + b
popt, pcov = curve_fit(func, x_data, y_data, p0=(1,1))
popt
array([3., 2.])

pcov
array([[ 4.13297331e-34, -1.45419433e-32],
       [-1.45419433e-32,  9.65872308e-31]])
```

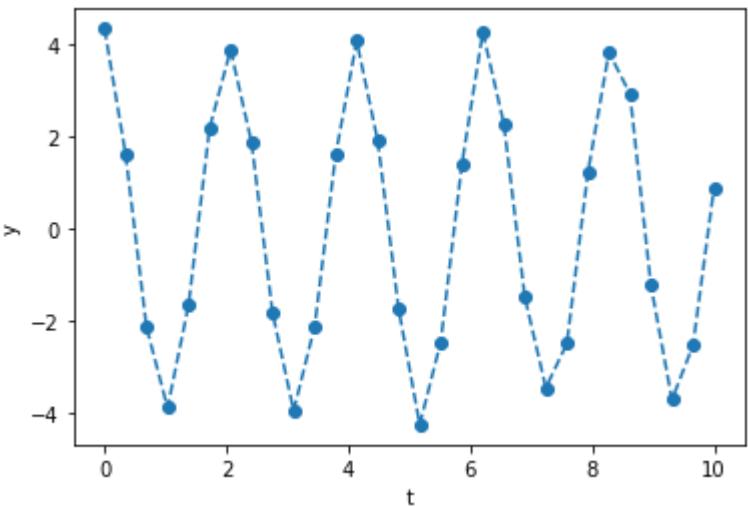
Initial Guess

Curve Fitting

- The equation of spring motion is

$$y(t) = A \cos(wt + \phi)$$

- Want to find the natural frequency of oscillation w for the spring.



Curve Fitting

$$y(t) = A \cos(wt + \phi)$$

Find A, w and ϕ if t and y is given

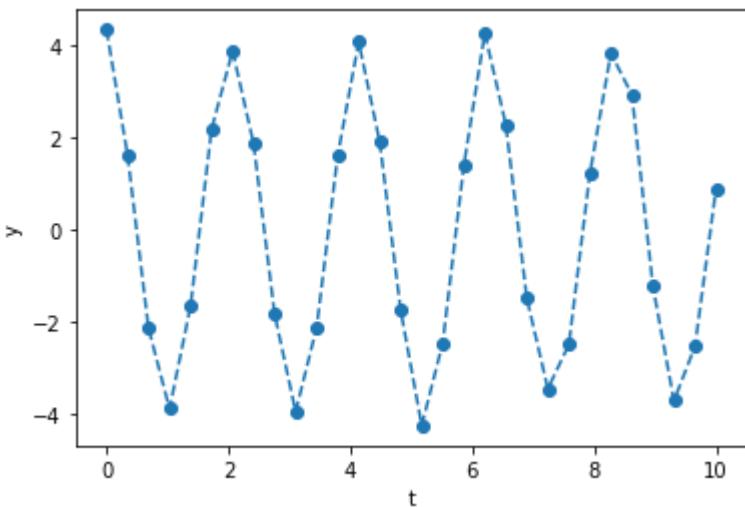
$$w = 2\pi f, \quad f = \frac{1}{T}, \quad T \approx 2$$

The initial good guess

- $w = 2\pi f = \pi$
- $A = 4$
- $\phi = 0$

Curve Fitting

- Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points



Curve Fitting

```
t_data = np.array([ 0. , 0.34482759, 0.68965517, 1.03448276, 1.37931034,
1.72413793, 2.06896552, 2.4137931 , 2.75862069, 3.10344828,
3.44827586, 3.79310345, 4.13793103, 4.48275862, 4.82758621,
5.17241379, 5.51724138, 5.86206897, 6.20689655, 6.55172414,
6.89655172, 7.24137931, 7.5862069 , 7.93103448, 8.27586207,
8.62068966, 8.96551724, 9.31034483, 9.65517241, 10. ])
y_data = np.array([ 4.3303953 , 1.61137995, -2.15418696, -3.90137249, -1.67259042,
2.16884383, 3.86635998, 1.85194506, -1.8489224 , -3.96560495,
-2.13385255, 1.59425817, 4.06145238, 1.89300594, -1.76870297,
-4.26791226, -2.46874133, 1.37019912, 4.24945607, 2.27038039,
-1.50299303, -3.46774049, -2.50845488, 1.20022052, 3.81633703,
2.91511556, -1.24569189, -3.72716214, -2.54549857, 0.87262548])
```

```
from scipy.optimize import curve_fit

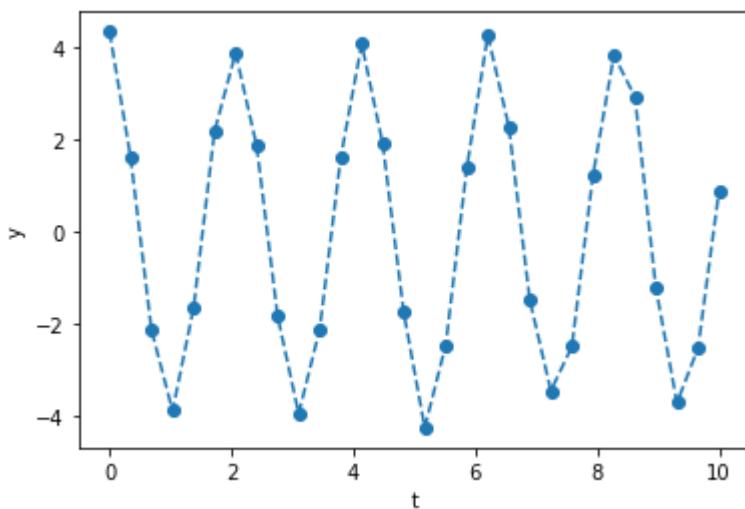
def func(x, A, w, phi):
    return A*np.cos(w*x+phi)

popt, pcov = curve_fit(func, t_data, y_data, p0=(4, np.pi, 0))
popt
array([3.94836219, 2.99899521, 0.10411352])
```

Initial Guess

Curve Fitting

- Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points

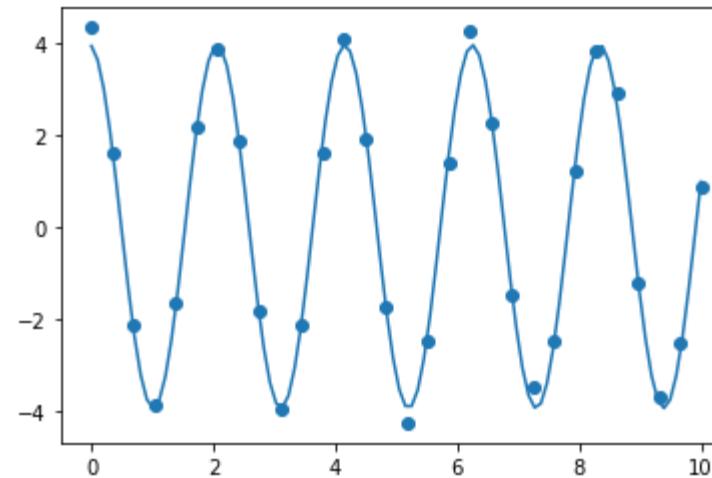


Curve Fitting

```
A, w, phi = popt|
```

```
t = np.linspace(0, 10, 100)  
y = func(t, A, w, phi)
```

```
plt.scatter(t_data,y_data)  
plt.plot(t,y);
```



Further Resources

- <https://scipy.org/>
- https://www.youtube.com/watch?v=jmX4FOUEfgU&ab_channel=Mr.PSolver

IDAT7215

Computer Programming for

Product Development and

Applications

Lecture 2-4: Python Libraries: Pandas

Dr. Zulfiqar Ali

Outline

- Data Manipulation
- DataFrame
- Statistics

Pandas

Pandas

- In the NumPy section we dealt with some arrays, whose columns had each a special meaning. For example, the column number 0 could contain values interpreted as years, and column 1 could contain a month, and so on.
- It is possible to handle the data this way, but it can be hard to remember, which column number corresponds to which variable. Especially, if you later remove some column from the array, then the numbering of the remaining columns changes.

Pandas

- One solution to this is to give a descriptive name to each column. These column names stay fixed and attached to their corresponding columns, even if we remove some of the columns.
- In addition, the rows can be given names as well, these are called indices in Pandas.

Pandas

- The Pandas library is built on top of the NumPy library, and it provides a special kind of two-dimensional data structure called `DataFrame`.
- The `DataFrame` allows to give names to the columns, so that one can access a column using its name in place of the index of the column.

What is Data? Iris Data set

- Many data analytics techniques are illustrated with the Iris Plant data set.
 - Details of this data set can be found https://en.wikipedia.org/wiki/Iris_flower_data_set
 - Three flower types (classes):
 - Setosa
 - Versicolor
 - Virginica



Iris-setosa

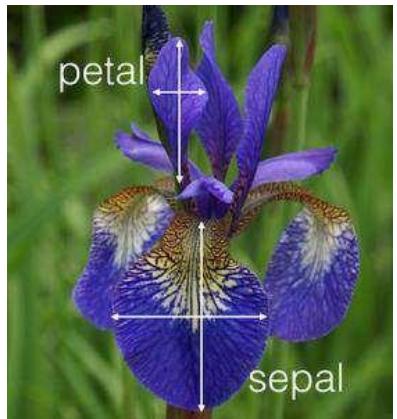


Iris-versicolor



Iris-virginica

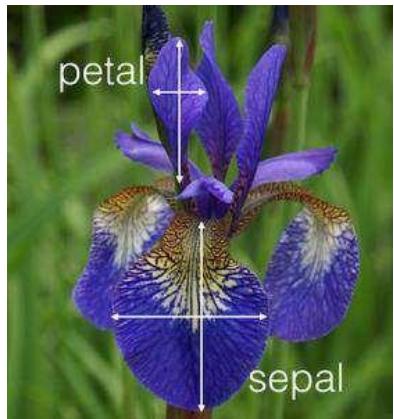
What is Data? Iris Data set



sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.3	3.3	6	2.5	virginica
...

What is Data? Iris Data set

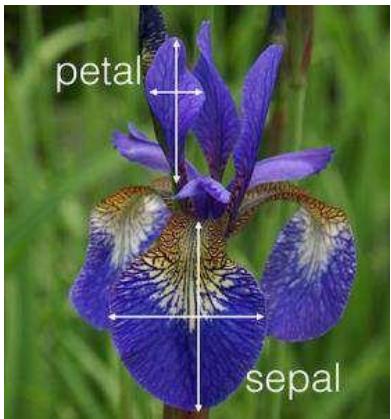
Each row represents one **data sample** (i.e., data point, data object). It represents an entity in real world (e.g., flower).



sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.3	3.3	6	2.5	virginica
...

What is Data? Iris Data set

Each row represents one **data sample** (i.e., data point, data object). It represents an entity in real world (e.g., flower).

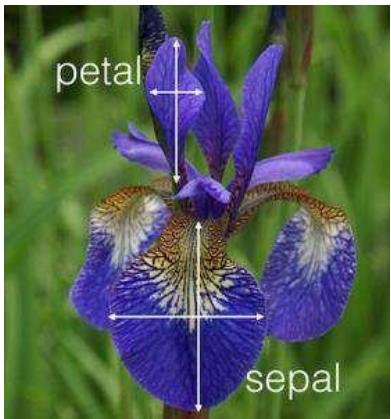


Each column is a **feature** (i.e., **attribute**, **variable**) to describe a characteristic of a data sample.

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.3	3.3	6	2.5	virginica
...

What is Data? Iris Data set

Each row represents one **data sample** (i.e., data point, data object). It represents an entity in real world (e.g., flower).



Each column is a **feature** (i.e., **attribute**, **variable**) to describe a characteristic of a data sample.

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.3	3.3	6	2.5	virginica
...

- **Data set** is a collections of data samples
- This iris data set contains 150 samples (50 samples from each of three species)
- 5 features: sepal length, sepal width, petal length, petal width, species

Pandas Example

- Import ‘iris.csv’
- We see that the DataFrame contains five columns, four of which are numerical variables.
- We can refer to a column by its name

```
import pandas as pd
df = pd.read_csv("iris.csv");
df.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

```
df['sepal_length'].head()
```

```
0    5.1
1    4.9
2    4.7
3    4.6
4    5.0
Name: sepal_length, dtype: float64
```

Summary statistic methods on Pandas columns

- There are several summary statistic methods that operate on a column or on all the columns.

```
print(df['sepal_length'].min())
print(df['sepal_length'].max())
print(df['sepal_length'].mean())
print(df['sepal_length'].median())
```

```
4.3
7.9
5.843333333333335
5.8
```

Drop a column

- We can drop some columns from the DataFrame with the `drop` method.
- We can use the `inplace` parameter of the `drop` method to modify the original DataFrame.
- Many of the modifying methods of the DataFrame have the `inplace` parameter.

```
df2 = df.drop(["sepal_length", "sepal_width"], axis = 1)  
df2.head()
```

	petal_length	petal_width	species
0	1.4	0.2	setosa
1	1.4	0.2	setosa
2	1.3	0.2	setosa
3	1.5	0.2	setosa
4	1.4	0.2	setosa

```
df.drop(["sepal_length", "sepal_width"], axis = 1, inplace = True)  
df.head()
```

	petal_length	petal_width	species
0	1.4	0.2	setosa
1	1.4	0.2	setosa
2	1.3	0.2	setosa
3	1.5	0.2	setosa
4	1.4	0.2	setosa

Add a new column in DataFrame

```
df = pd.read_csv("iris.csv");
print(df.head(5))
df['label'] = df['species'] == 'setosa'
print(df.head(5))
```

	sepal_length	sepal_width	petal_length	petal_width	species	label
0	5.1	3.5	1.4	0.2	setosa	True
1	4.9	3.0	1.4	0.2	setosa	True
2	4.7	3.2	1.3	0.2	setosa	True
3	4.6	3.1	1.5	0.2	setosa	True
4	5.0	3.6	1.4	0.2	setosa	True

Creation and indexing of series

- Series is one-dimensional version of DataFrame.
Each column in a DataFrame is a Series
- One can turn any one-dimensional list into a Series, which is a one-dimensional data structure.
- We can also attach a name to this series

```
s=pd.Series([1, 4, 5, 2, 5, 2])  
s
```

```
0    1  
1    4  
2    5  
3    2  
4    5  
5    2  
dtype: int64
```

```
s.name = "Grades"  
s
```

```
0    1  
1    4  
2    5  
3    2  
4    5  
5    2  
Name: Grades, dtype: int64
```

Row indices of Series

- In addition to the values of the series, also the row indices were printed. All the accessing methods from NumPy arrays also work for the Series: indexing, slicing, masking and fancy indexing.
- Note that the indices stick to the corresponding values, they are not renumbered!

```
s=pd.Series([1, 4, 5, 2, 5, 2])
print(s[1])
s2=s[[0,5]]
print(s2)
```

```
4
0    1
5    2
dtype: int64
```

```
print(s2[5])
print(s2[1])
```

```
2
```

```
KeyError
```

Series

- The values as a NumPy array are accessible via the `values` attribute.
- The indices are available through the `index` attribute.
- The index is not simply a NumPy array, but a data structure that allows fast access to the elements.

```
s=pd.Series([1, 4, 5, 2, 5, 2])
s2=s[[0,5]]
print(s2.values)
print(s2.index)
```

```
[1 2]
Int64Index([0, 5], dtype='int64')
```

```
s3=pd.Series([1, 4, 5, 2, 5, 2], index=list("abcdef"))
s3
```

```
a    1
b    4
c    5
d    2
e    5
f    2
dtype: int64
```

```
print(s3.index)
print(s3['c'])
```

```
Index(['a', 'b', 'c', 'd', 'e', 'f'], dtype='object')
5
```

Series

- It is still possible to access the series using NumPy style *implicit integer indices*.
- This can be confusing though.
- Pandas offers attributes `loc` and `iloc`. The attribute `loc` always uses the explicit index, while the attribute `iloc` always uses the implicit integer index

```
: print(s3)
print(s3[0])
```

```
a    1
b    4
c    5
d    2
e    5
f    2
dtype: int64
1
```

```
: s4 = pd.Series(["Jack", "Jones", "James"], index=[1,2,3])
s4
```

```
1    Jack
2    Jones
3    James
dtype: object
```

```
: s4[1]
```

```
'Jack'
```

```
: s4.iloc[1]
```

```
'Jones'
```

```
: s4.loc[1]
```

```
'Jack'
```

Creation of dataframes

- The DataFrame is essentially a two-dimensional object, and it can be created in three different ways:
 - out of a two-dimensional NumPy array
 - out of given columns
 - out of given rows

Creating DataFrames from a NumPy array

- In the following example a DataFrame with 2 rows and 3 column is created. The row and column indices are given explicitly.

```
# DataFrames
df=pd.DataFrame(np.random.randn(2,3), columns=["First", "Second", "Third"], index=["a", "b"])
df
```

	First	Second	Third
a	0.149066	-0.176563	-1.590523
b	-0.015790	1.329185	0.420429

- If either columns or index argument is left out, then an implicit integer index will be used.

```
df2=pd.DataFrame(np.random.randn(2,3), index=["a", "b"])
df2
```

	0	1	2
a	1.210334	-1.035517	0.494750
b	-0.076913	-0.655023	1.235455

Creating DataFrames from columns

- A column can be specified as a list, an NumPy array, or a Pandas' Series.
- Input is a dictionary of which keys give the column names and values are the actual column content.

```
s1 = pd.Series([1,2,3])
s2 = pd.Series([4,5,6])
df = pd.DataFrame({'a':s1, 'b':s2})
print(df)
```

	a	b
0	1	4
1	2	5
2	3	6

Creating DataFrames from rows

- We can give a list of rows as a parameter to the DataFrame constructor.

```
df=pd.DataFrame([{"Wage" : 1000, "Name" : "Jack", "Age" : 21}, {"Wage" : 1500, "Name" : "John", "Age" : 29}])  
df
```

	Age	Name	Wage
0	21	Jack	1000
1	29	John	1500

```
df = pd.DataFrame([[1000, "Jack", 21], [1500, "John", 29]], columns=["Wage", "Name", "Age"])  
df
```

	Wage	Name	Age
0	1000	Jack	21
1	1500	John	29

Accessing columns and rows of a dataframe

- Recommend to use attributes `loc` and `iloc` for accessing columns and rows in a dataframe.
- `loc` uses explicit indices and the `iloc` uses the implicit integer indices.

```
df = pd.DataFrame([[1000, "Jack", 21], [1500, "John", 29]], columns=["Wage", "Name", "Age"])
df
```

	Wage	Name	Age
0	1000	Jack	21
1	1500	John	29

```
print(df.loc[1, "Wage"])

print(df.iloc[0,0])

print(df.loc[1, ["Name", "Wage"]])
```

```
1500
1000
Name    John
Wage    1500
Name: 1, dtype: object
```

Accessing columns and rows of a dataframe

- With `iloc` everything works like with NumPy arrays: indexing, slicing, fancy indexing, masking and their combinations.
- With `loc` it is the same but now the names in the explicit indices are used for specifying rows and columns.

Summary statistics of Pandas

- The summary statistic methods work in a similar way as their counter parts in NumPy. By default, the aggregation is done over columns.
- The `describe` method of the `DataFrame` object gives different summary statistics for each (numeric) column. The result is a `DataFrame`. This method gives a good overview of the data and is typically used in the exploratory data analysis phase.

```
df = pd.read_csv("iris.csv");
df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Use Summary Statistics to Explore Data

Use Summary Statistics to Explore Data (in Python)

```
In [3]: # Libraries
import pandas as pd

# load data
iris = pd.read_csv("iris.csv")

iris.head()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

- Load and see the data

Feature Type

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length    150 non-null float64
sepal_width     150 non-null float64
petal_length    150 non-null float64
petal_width     150 non-null float64
species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

- Show the feature types

Summary statistic in python

```
In [6]: iris.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

- Mean
- Standard Deviation
- Min
- Q1
- Q2
- Q3
- Max

Summary statistic in python: mean; standard deviation; median

```
In [7]: iris['sepal_length'].std()
```

```
Out[7]: 0.8280661279778629
```

```
In [9]: iris['sepal_length'].mean()
```

```
Out[9]: 5.843333333333335
```

```
In [10]: iris['sepal_length'].median()
```

```
Out[10]: 5.8
```

- Compute the mean, standard deviation and median for the feature ‘sepal_length’

Summary statistic in python: Mode

```
In [13]: iris['species'].value_counts()
```

```
Out[13]: setosa      50  
versicolor   50  
virginica    50  
Name: species, dtype: int64
```

```
In [14]: iris['species'].mode()
```

```
Out[14]: 0      setosa  
1      versicolor  
2      virginica  
dtype: object
```

- Get counts of unique values for ‘species’
- Compute the mode for ‘species’

Summary statistics for different groups

- Explore more about the data

Setosa

```
In [17]: iris_setosa = iris[iris['species'] == 'setosa']
iris_setosa.describe()
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.000000
mean	5.00600	3.418000	1.464000	0.244000
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

Versicolor

```
In [18]: iris_versicolor = iris[iris['species'] == 'versicolor']
iris_versicolor.describe()
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

The petal_length of 'setosa' is always smaller than the petal_length of 'versicolor'

Summary statistics for different group

- Explore more about the data

Setosa

```
In [17]: iris_setosa = iris[iris['species'] == 'setosa']
iris_setosa.describe()
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.000000
mean	5.00600	3.418000	1.464000	0.24400
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

Versicolor

```
In [18]: iris_versicolor = iris[iris['species'] == 'versicolor']
iris_versicolor.describe()
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

The petal_width of 'setosa' is always smaller than the petal_width of 'versicolor'

Summary statistics for different group

- Explore more about the data

Setosa

```
In [17]: iris_setosa = iris[iris['species'] == 'setosa']
iris_setosa.describe()
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.000000
mean	5.00600	3.418000	1.464000	0.244000
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

Versicolor

```
In [18]: iris_versicolor = iris[iris['species'] == 'versicolor']
iris_versicolor.describe()
```

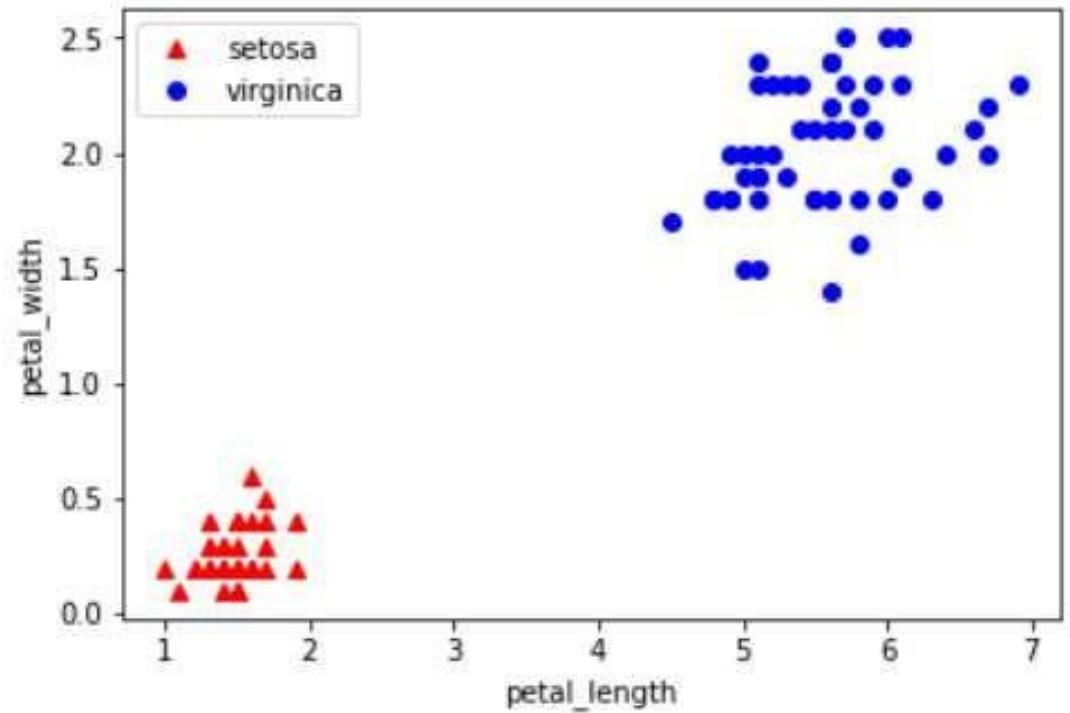
Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

- We have our own way to distinguish ‘setosa’ and ‘versicolor’ (i.e., using petal_length or petal_width)
- The knowledge is gained by performing simple summary statistics techniques on data.

The pattern is more obvious when we visualize it

- The petal_length of ‘setosa’ is always smaller than the petal_length of ‘versicolor’
- The petal_width of ‘setosa’ is always smaller than the petal_width of ‘versicolor’
- It is very easy to classify ‘setosa’ and ‘virginica’ just based on petal_length and petal_width.



IDAT7215

Computer Programming for Product Development and Applications

Lecture 2-2: Python Libraries: Matplotlib

Dr. Zulfiqar Ali

Outline

- Why Data Visualization
- What is Data Visualization
- Introduction to Matplotlib
- Types of Plots
 - Line Plot
 - Bar Graph
 - Histograms
 - Scatter Plot
 - Area Plot
 - Pie Plot

Why Data Visualization?

- Human brain can process information easily when it is in pictorial or graphical form



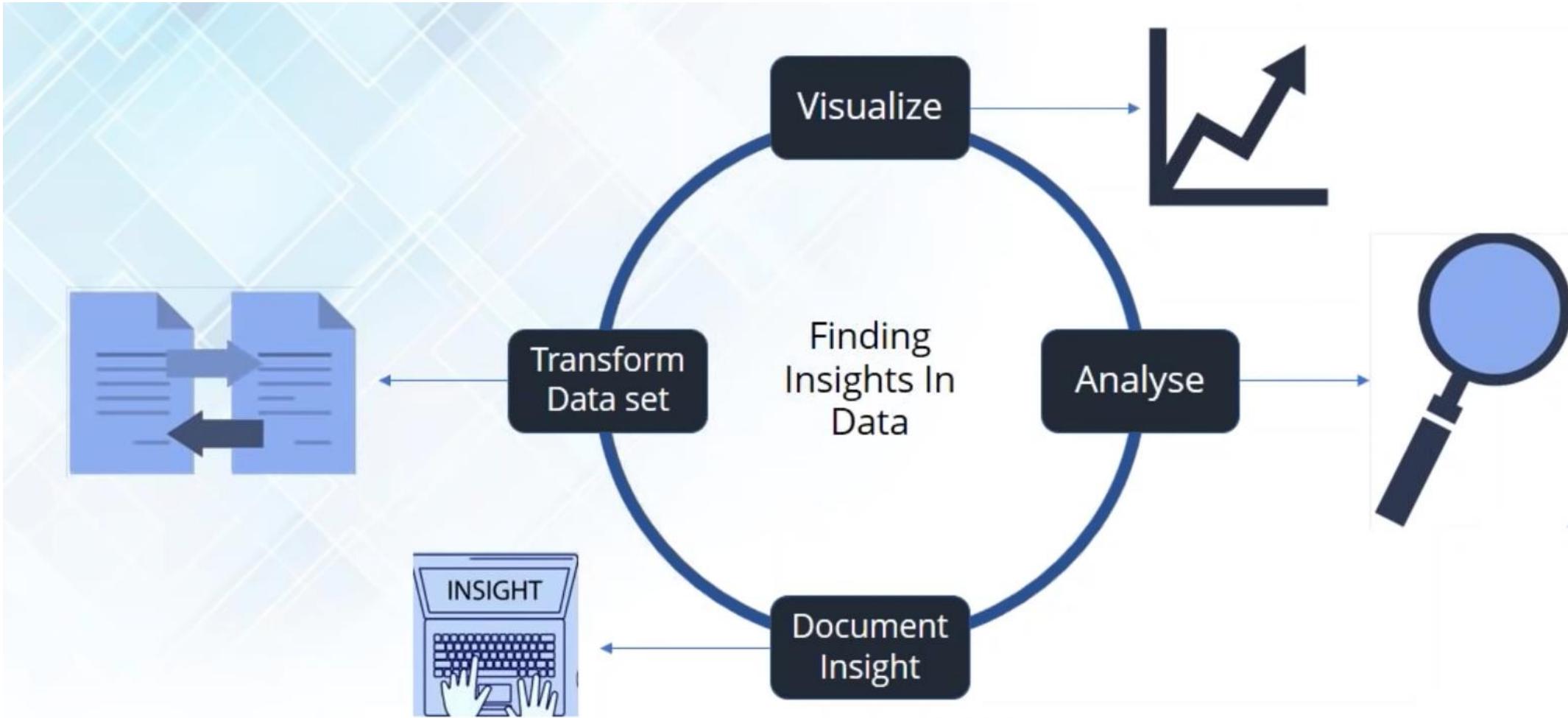
Why Data Visualization?

- Data visualization allows us to quickly interpret the data and adjust different variables to see their effect.



What is Data Visualization?

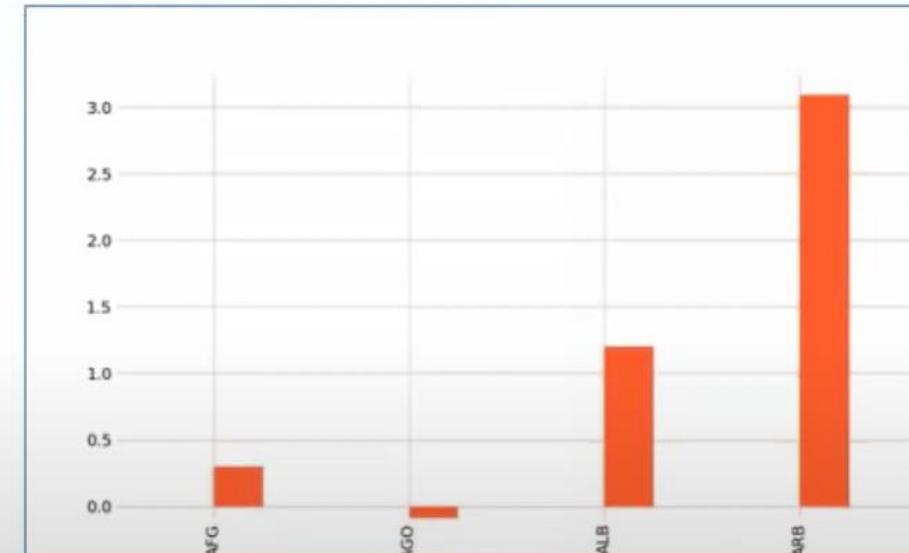
- Data visualization is the presentation of data in a pictorial or graphical format.



What is Matplotlib?

- Matplotlib is a Python package used for 2D graphics.

Country Name	Country Code	2010	2011	2012	2013	2014
Afghanistan	AFG	20.6	20.9	19.7	21.1	20.8
Angola	AGO	10.8	10.7	10.7	10.6	10.5
Albania	ALB	25.799999	27	28.3	28.7	29.2
Arab World	ARB	25.022214	28.11752	29.11321	29.33531	29.70457
United Arab Emirates	ARE	9.8000002	9.8	9.8	9.9	10
Argentina	ARG	19.5	18.8	18.4	19.7	21.3
Armenia	ARM	38.299999	38.7	35	32.5	35.1
Australia	AUS	11.4	11.4	11.7	12.2	13.1
Austria	AUT	8.8000002	8.2	8.7	9.1	9.2
Azerbaijan	AZE	14.6	14.5	14.3	13.4	13.6
Burundi	BDI	10.8	10.8	10.8	10.8	10.7
Belgium	BEL	22.5	18.6	19.7	23.1	23.6
Benin	BEN	2	2	2	1.8	1.7
Burkina Faso	BFA	5.1999998	5.3	5.2	5.2	5
Bangladesh	BGD	8.1999998	8.2	8.2	8.9	9.1
Bulgaria	BGR	22.9	25.2	28.2	29.7	25.9
Bahrain	BHR	10.2	11.4	10.5	10.6	10.9
Bahamas, The	BHS	36	27.2	30.4	30.8	30.1
Bosnia and Herzegovina	BIH	57.200001	57.1	61.7	57.4	57.5
Belarus	BLR	13.2	12.5	11.8	12	12
Belize	BLZ	20.9	24.3	26	22.4	22



What is Matplotlib?

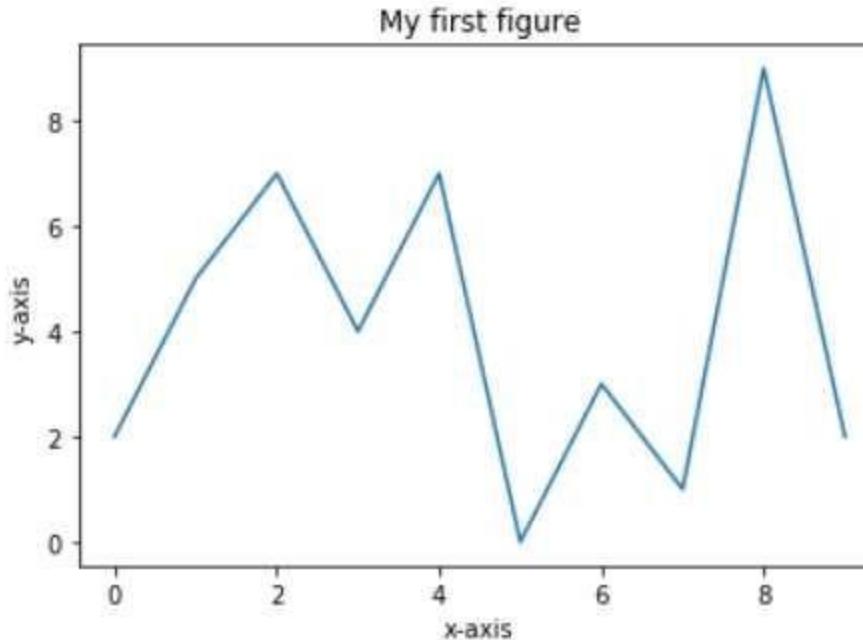
- Matplotlib is an amazing visualization library in python for 2D plots of arrays.
- Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with broader SciPy stack. It was **introduced by John Hunter in the year 2002**.
- One of the greatest benefits of visualization is that it allows us visual access to huge amount of data in easily digestible visuals.
- Matplotlib is the most common low-level visualization library for Python.
- It can create line graphs, bar plots, scatter plots, density plots, histograms, heatmaps, and so on.

Simple Figure

- Simply line plot

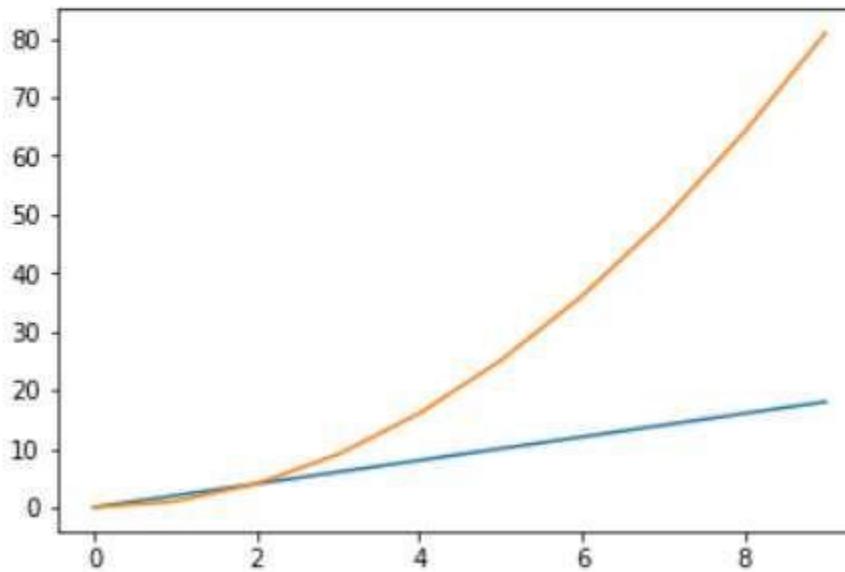
```
import numpy as np
import matplotlib.pyplot as plt
# add IPython magic command %matplotlib inline
# to print pyplot output inline in the notebook without calling plt.show()
%matplotlib inline

a=np.array([2, 5, 7, 4, 7, 0, 3, 1, 9, 2])
plt.plot(a)                      # plot the points in the array a
plt.title("My first figure")     # Add a title to the figure
plt.xlabel("x-axis")              # Give a label to the x-axis
plt.ylabel("y-axis")              # Give a label to the y-axis
plt.show()                       # Tell matplotlib to output the figure.
                                # Not strictly required in notebooks (but a bit neater).
```



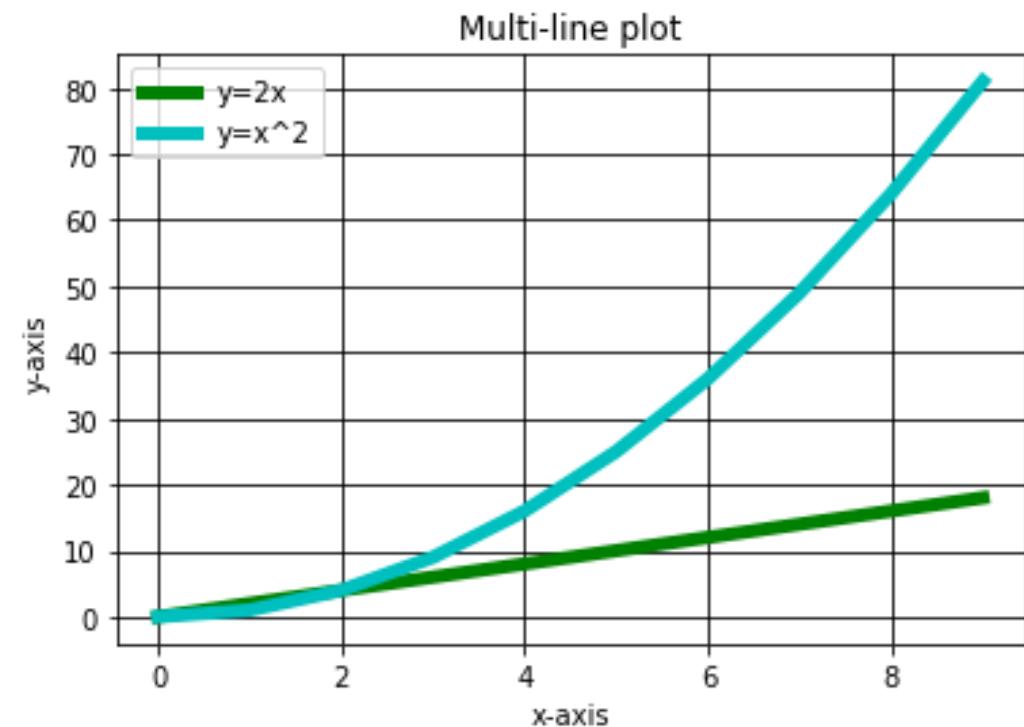
Plotting two lines in the same figure

```
# if plt.plot(*args) are run in the same cell  
# we will plot on the same axes  
x = np.arange(10)  
plt.plot(x,2*x)  
plt.plot(x,x**2)  
plt.show()
```



Plotting two lines in the same figure

```
# If plt.plot(*args) are run in the same cell
# we will plot on the same axes
x = np.arange(10)
y1=2*x
y2=x**2
plt.plot(x,y1,'g',label='y=2x',linewidth=5)
plt.plot(x,y2, 'c',label='y=x^2',linewidth=5)
plt.title('Multi-line plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.grid(True, color='k')
plt.show();
```

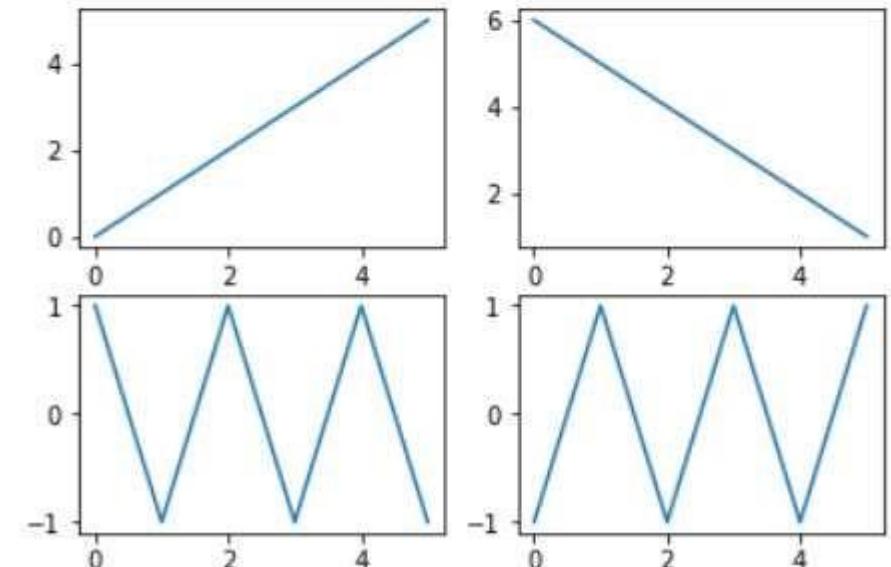


Sub-figures

- One can create a figure with several subfigures using the command `plt.subplots`.
- It creates a grid of subfigures, where the number of rows and columns in the grid are given as parameters.
- It returns a pair of a figure object and an array containing the subfigures. In matplotlib the subfigures are called axes.

```
fig, ax = plt.subplots(2,2)
print(ax.shape)
ax[0,0].plot(np.arange(6))          # top left
ax[0,1].plot(np.arange(6,0,-1))    # top right
ax[1,0].plot((-1)**np.arange(6))   # bottom left
ax[1,1].plot((-1)**np.arange(1,7)) # bottom right
plt.show()
```

(2, 2)



Bar Graph

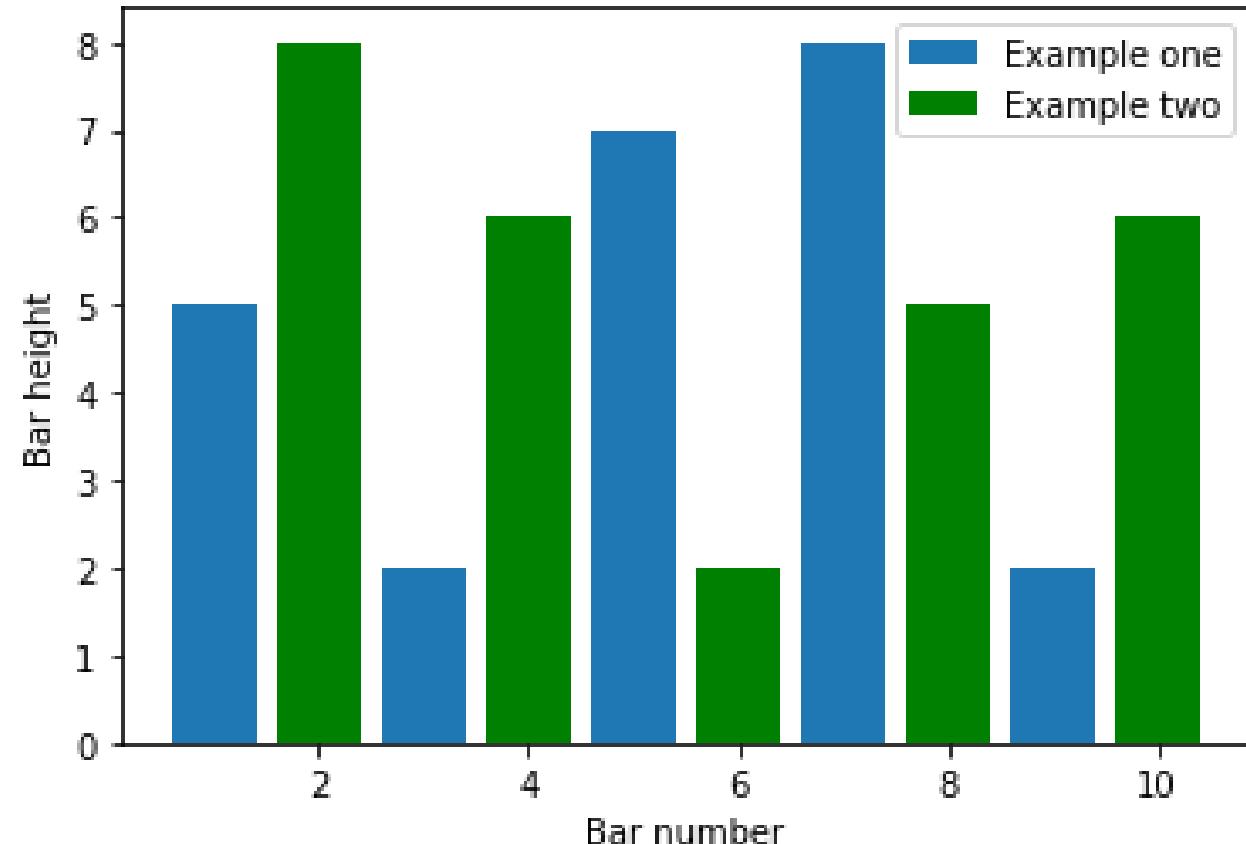
- Bar graph are useful when we want to compare things between groups.
- Useful when changes are larger.

```
import matplotlib.pyplot as plt

x1 = [1,3,5,7,9]    # x-axis values
y1 = [5,2,7,8,2]    # y-axis values

x2 = [2,4,6,8,10]   # x-axis values
y2 = [8,6,2,5,6]    # y-axis values

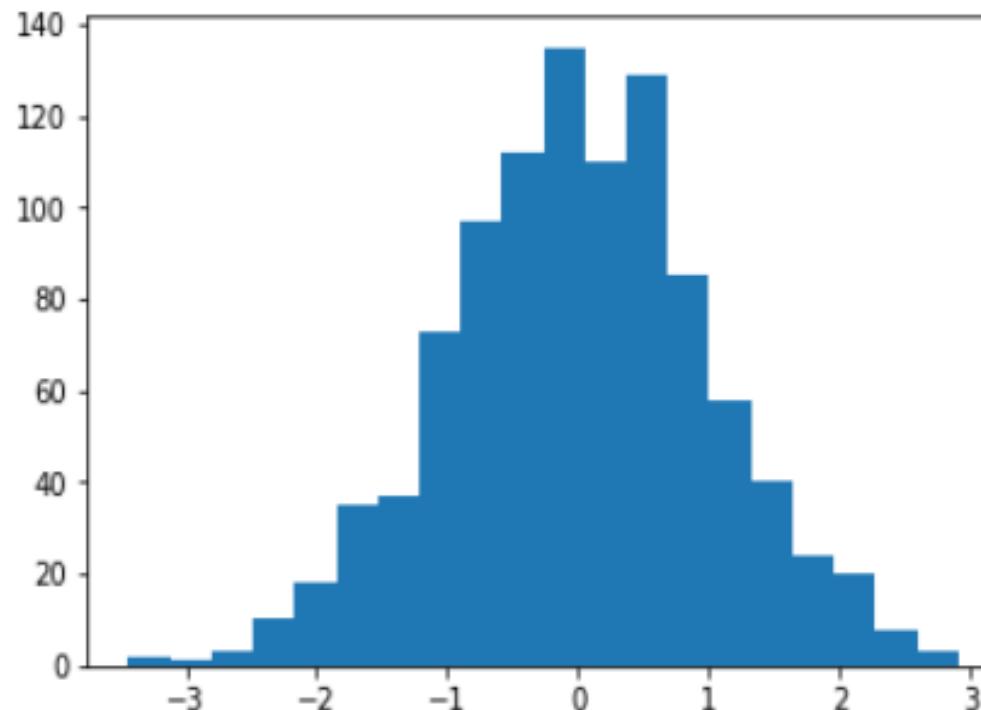
plt.bar(x1,y1, label = "Example one")
plt.bar(x2,y2, label = 'Example two', color='g')
plt.legend()
plt.xlabel('Bar number')
plt.ylabel('Bar height')
plt.show;
```



Histogram

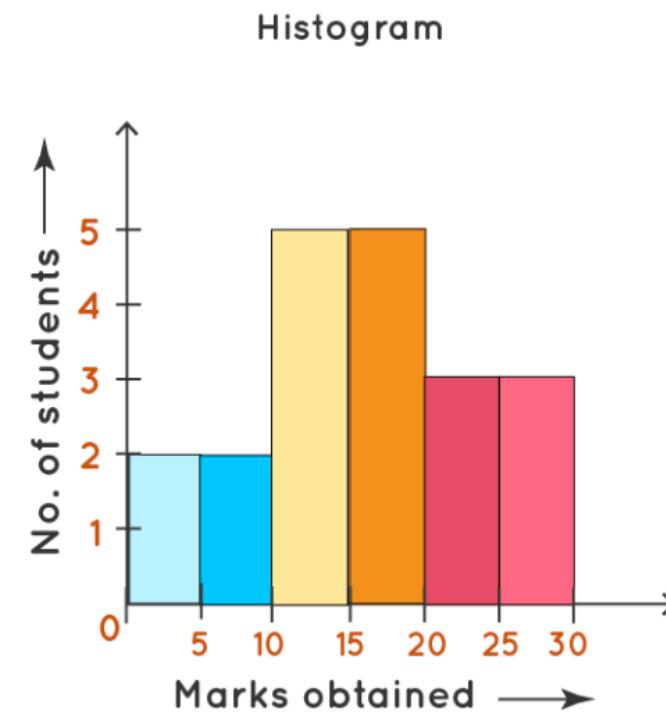
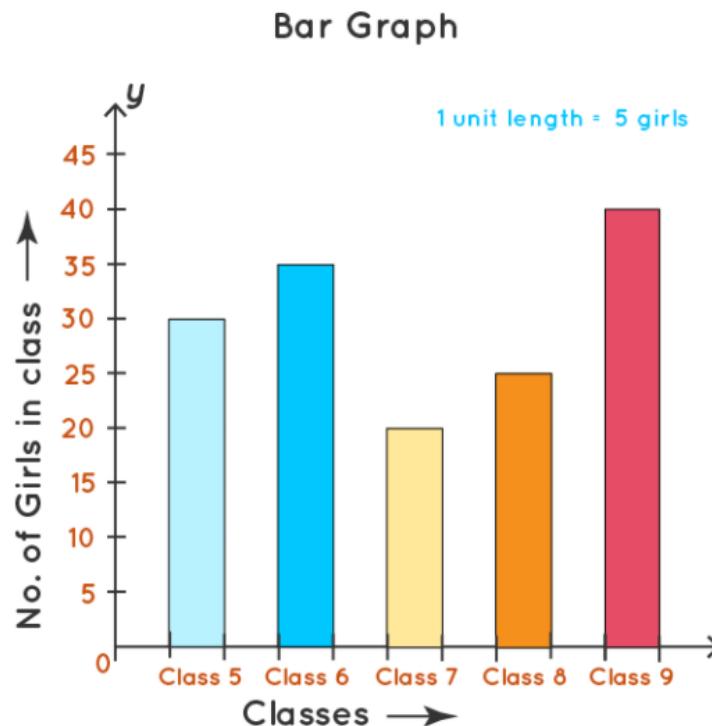
- A **histogram** is a way to graphically summarize or describe a data set by visually conveying its distribution using vertical bars.

```
x = np.random.randn(1000)  
plt.hist(x,20);
```



Histogram

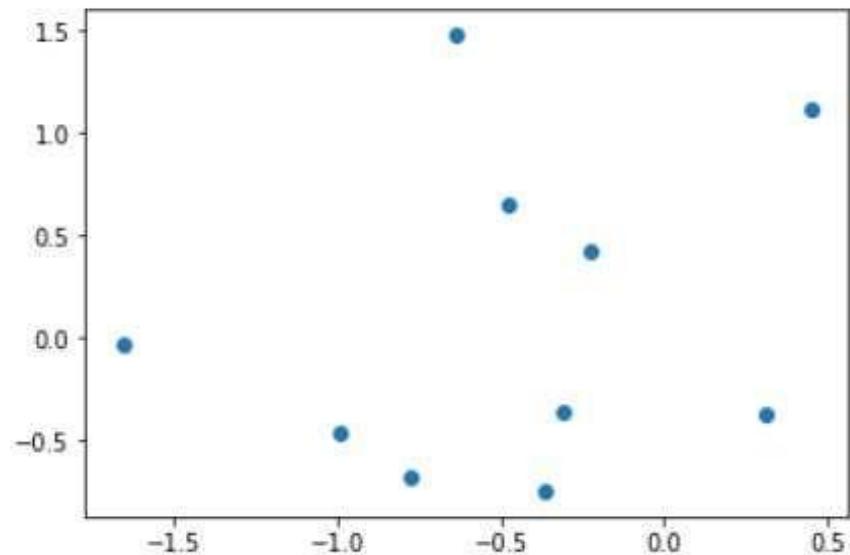
- A **histogram** is a way to graphically summarize or describe a data set by visually conveying its distribution using vertical bars.
- In histogram we have **quantitative variable** whereas in bar graph we have **categorical variable**.



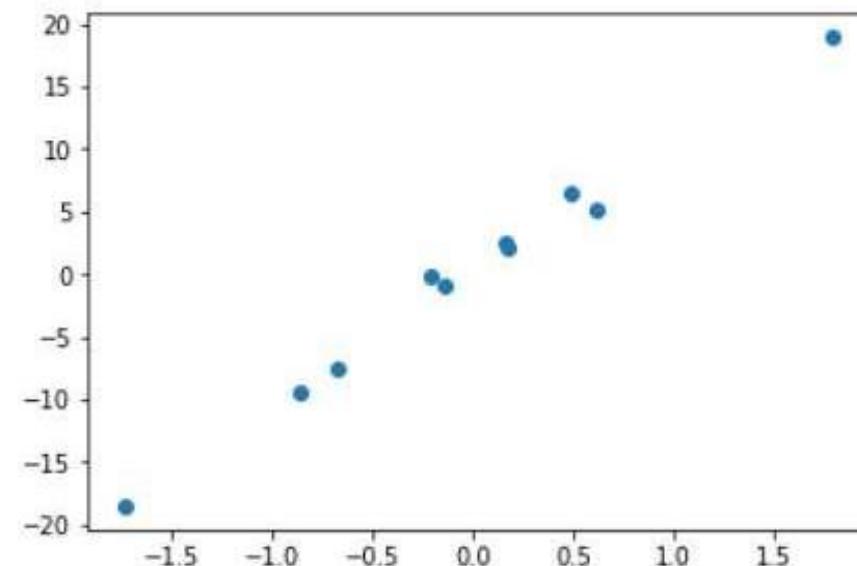
Scatter plots

- The scatterplot is a visualization technique that enjoys widespread use in data analysis and is a powerful way to convey information about the relationship between two variables.

```
x = np.random.randn(10)  
y = np.random.randn(10)|  
plt.scatter(x,y);
```



```
x = np.random.randn(10)  
y = x*10 + np.random.randn(10)|  
plt.scatter(x,y);
```



Area Plot or Stack Plot

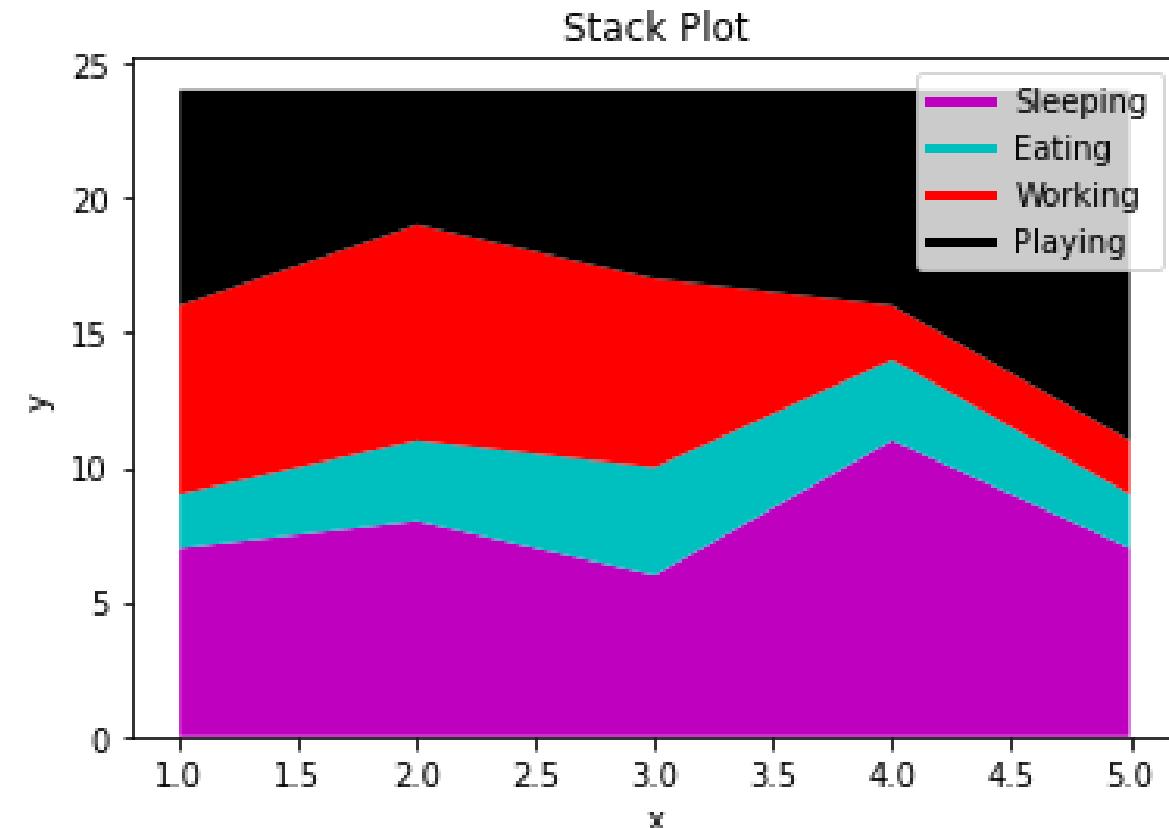
- Area or stack plot are used to draw a stacked area plot. It displays the complete data for visualization.

```
# Data
days = [1,2,3,4,5]
sleeping = [7,8,6,11,7]
eating=[2,3,4,3,2]
working=[7,8,7,2,2]
playing=[8,5,7,8,13]

# Stack plot
plt.stackplot(days,sleeping,eating,working,playing,colors=['m','c','r','k'])

# For Labels in Legend
plt.plot([],[],color='m',label='Sleeping',linewidth=3)
plt.plot([],[],color='c',label='Eating',linewidth=3)
plt.plot([],[],color='r',label='Working',linewidth=3)
plt.plot([],[],color='k',label='Playing',linewidth=3)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Stack Plot')
plt.legend()
plt.show;
```



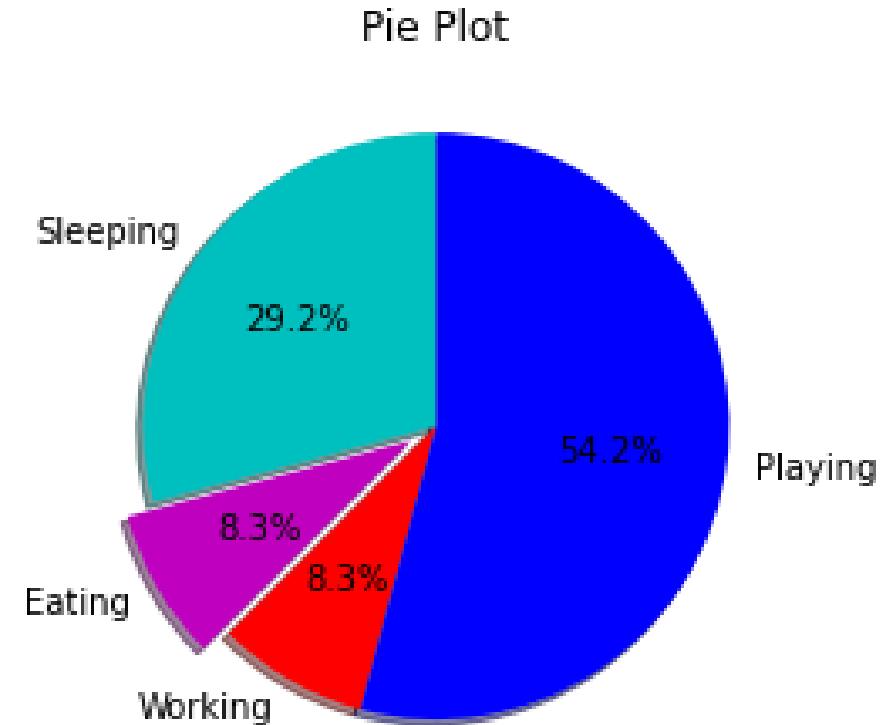
Pie Plot

- A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion.

```
slices = [7,2,2,13]
activities = ['Sleeping','Eating','Working','Playing']
cols = ['c','m','r','b']

plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow=True,
        explode=(0,0.1,0,0),
        autopct='%1.1f%%')

plt.title('Pie Plot')
plt.show()
```



https://matplotlib.org/tutorials/introductory/sample_plots.html

IDAT7215

Computer Programming for Product Development and Applications

Lecture 2-1: Python Libraries: NumPy

Dr. Zulfiqar Ali

Outline

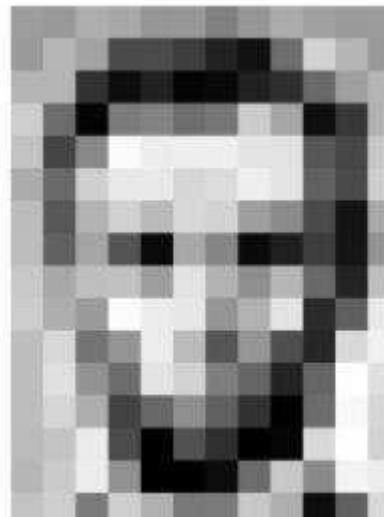
- NumPy Introduction
- Creation of Arrays
- Data Retrieve and Restructure
- Fast Computation with NumPy

NumPy

- NumPy stands for Numerical Python and it is the fundamental package for scientific computing with Python.
- NumPy is a Python library for handling multi-dimensional arrays.
- It contains both the **data structures** needed for the storing and accessing arrays, and **operations and functions** for computation using these arrays.
- Unlike lists, the arrays must have the same data types for all its elements.
- The **homogeneity** of arrays allows highly optimized functions that use arrays as their inputs and outputs.

Usages of high-dimensional arrays in data analysis

- Store matrices, solve systems of linear equations, compute eigenvalues/eigenvectors, matrix decompositions, ...
- Images and videos can be represented as NumPy arrays



157	158	174	168	155	153	129	151	172	167	165	156
158	162	160	74	75	62	33	17	118	210	162	154
159	160	50	34	34	8	10	33	48	106	159	181
256	138	6	134	131	111	126	204	148	16	66	180
194	48	137	251	237	239	239	208	237	67	71	201
172	129	267	233	233	214	220	239	238	88	74	205
188	68	170	269	185	215	211	188	135	75	28	169
185	87	165	84	10	148	154	11	31	62	27	148
199	166	191	193	168	227	178	149	182	106	36	195
205	176	165	252	236	291	149	178	238	45	95	234
190	216	186	149	238	187	85	150	79	26	218	241
190	224	147	162	227	210	127	102	36	101	255	224
190	214	123	99	103	142	81	90	2	105	249	218
187	196	230	25	1	61	47	9	5	217	255	213
183	202	237	141	8	0	12	108	200	138	263	236
185	206	121	297	177	121	123	200	175	11	93	218

Usages of high-dimensional arrays in data analysis

- Store matrices, solve systems of linear equations, compute eigenvalues/eigenvectors, matrix decompositions, ...
- Images and videos can be represented as NumPy arrays
- A 2-dimensional table might store an input data matrix in data analysis, where row represents a sample, column represents a feature (Commonly used in scikit-learn).

Creation of arrays

- Import the NumPy library
 - Suggested to use the standard abbreviation np

```
import numpy as np
# creation of arrays
a = np.array([1,2,3]) # one dimensional array
a
array([1, 2, 3])

np.array([[1,2,3], [4,5,6]]) # two dimensional array
array([[1, 2, 3],
       [4, 5, 6]])

np.array([[[1,2], [3,4]], [[5,6], [7,8]]]) # three dimensional array
array([[[1, 2],
       [3, 4]],
      [[5, 6],
       [7, 8]]])
```

Creation of arrays

- Import the NumPy library
 - Suggested to use the standard abbreviation np
- Give a (nested) list as a parameter to the array constructor
 - One dimensional array: list
 - Two dimensional array: list of lists
 - Three dimensional array: list of lists of list

```
import numpy as np
# creation of arrays
a = np.array([1,2,3]) # one dimensional array
a

array([1, 2, 3])
```

```
np.array([[1,2,3], [4,5,6]]) # two dimensional array

array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.array([[[1,2], [3,4]], [[5,6], [7,8]]]) # three dimensional array

array([[[1, 2],
       [3, 4]],
      [[5, 6],
       [7, 8]]])
```

1D Array

3	2
3	4

2D Array

1	0	1
3	4	1

3D Array

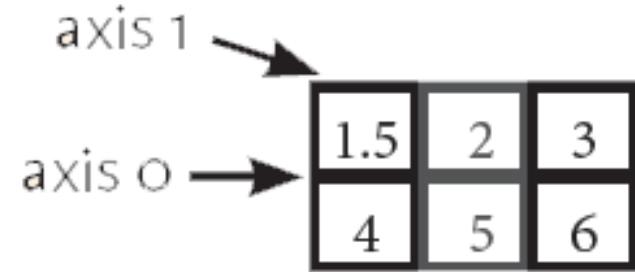
1	7	9
5	9	3
7	9	9

One dimensional array, Two dimensional array, Three dimensional array

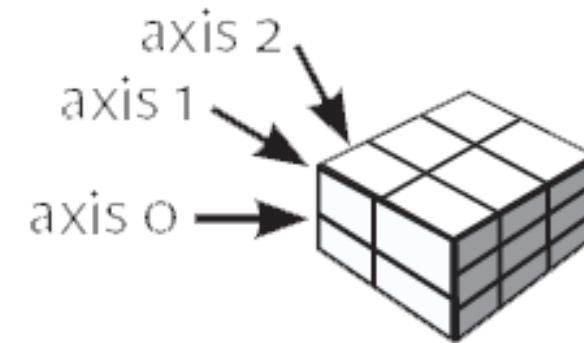
1D array



2D array



3D array



- In a two-dimensional array, you have rows and columns. The rows are indicated as “axis 0” while the columns are the “axis 1”.
- The number of the axis goes up accordingly with the number of the dimensions.

Creation of arrays

- Useful function to create common types of arrays
 - np.zeros() : all elements are 0s
 - np.ones() : all elements are 1s
 - np.full() : all elements to a specific value
 - np.empty() : all elements are uninitialized
 - np.eye() : identity matrix: a matrix with elements on the diagonal are 1s, others are 0s

```
# create common types of arrays:  
np.zeros((3,4))
```

```
array([[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]])
```

```
np.ones((3,3))
```

```
array([[1., 1., 1.],  
       [1., 1., 1.],  
       [1., 1., 1.]])
```

```
np.full((3,3), fill_value=3.0)
```

```
array([[3., 3., 3.],  
       [3., 3., 3.],  
       [3., 3., 3.]])
```

```
np.empty((2,2))
```

```
array([[1.19956159e-311, 2.13620807e-306],  
      [3.44897822e-307, 6.89794965e-307]])
```

```
np.eye(5)
```

```
array([[1., 0., 0., 0., 0.],  
       [0., 1., 0., 0., 0.],  
       [0., 0., 1., 0., 0.],  
       [0., 0., 0., 1., 0.],  
       [0., 0., 0., 0., 1.]])
```

Creation of arrays

```
np.arange(1,10,3) # the start value is 1 (inclusive), stop value is  
array([1, 4, 7])  
  
np.linspace(0, np.pi, 5) # Evenly spaced range with 5 elements  
array([0.0, 0.78539816, 1.57079633, 2.35619449, 3.14159265])
```

- Generate evenly spaced values within a given interval.
- np.arange() : It works like Python built-in range() function
- For non-integer ranges it is better to use np.linspace().
- With np.linspace() one does not have to compute the length of the step, but instead one specifies the wanted number of elements. By default, the endpoint is included in the result, unlike with arange.

Creation of arrays with random elements

- We may need some random generated data to test our program
- NumPy can easily produce arrays of wanted shape with random numbers.
- `np.random.random()` : uniformly distributed from [0.0, 1.0)
- `np.random.normal()` : normally distributed
- `np.random.randint()` : uniformly distributed integers

Creation of arrays with random elements

```
# Creation of arrays with random elements
```

```
np.random.random((3,4)) # Elements are uniformly distributed from [0.0,1.0)
```

```
array([[0.22051398, 0.7076659 , 0.42798583, 0.01112018],  
       [0.97417167, 0.26964842, 0.36510724, 0.28772977],  
       [0.04420915, 0.51836288, 0.92733042, 0.55925031]])
```

```
np.random.normal(0, 1, (3,4)) # Elements are normally distributed with mean 0 and standard deviation 1
```

```
array([[-1.25509693, 1.25917427, 0.16378993, -0.3458001 ],  
      [ 0.52594032, 0.55853451, -0.56651307, 1.20601537],  
      [-0.2643899 , 1.18457211, -0.32818969, -0.06231104]])
```

```
np.random.randint(-2, 10, (3,4)) # Elements are uniformly distributed integers from [-2,10)
```

```
array([[-1, 0, 8, -2],  
      [ 3, 5, 3, 9],  
      [ 1, 8, 9, 8]])
```

Creation of arrays with random elements

- To debug our code, sometimes it is useful to re-create exactly the same random data in every run of our program.
- We can create random numbers deterministically using seed.

```
# create random numbers deterministically using seed.  
np.random.seed(0)  
print(np.random.randint(0, 100, 10))  
  
[44 47 64 67 67  9 83 21 36 87]
```

If you run the code multiple times,
it will always give the same
numbers,

Array types and attributes

- An array has several attributes:
 - `ndim`: the number of dimensions
 - `shape`: size in each dimension
 - `size`: the number of elements
 - `dtype`: the type of element

```
b=np.array([[1,2,3], [4,5,6]])
```

```
print(b.ndim)
print(b.shape)
print(b.size)
print(b.dtype)
```

```
2
(2, 3)
6
int32
```

```
b=np.array([[[1,2,3], [4,5,6]], [[1,2,3], [4,5,6]]])
print(b.ndim)
print(b.shape)
print(b.size)
print(b.dtype)
```

```
3
(2, 2, 3)
12
int32
```

Indexing

- One dimensional array works like the list.
- For multi-dimensional array, the index is a comma separated tuple instead of single integer
- Note that if you give only a single index to a multi-dimensional array, it indexes the first dimension of the array.

```
# Indexing
a=np.array([1,4,2,7,9,5])
print(a[1])

b=np.array([[1,2,3], [4,5,6]])
print(b)
print(b[1,2])      # row index 1, column index 2

# As with lists, modification through indexing is possible
b[0,0] = 10
print(b)

# if give only a single index to a multi-dimensional array
print(b[0])        # First row
print(b[1])        # Second row
```

```
4
[[1 2 3]
 [4 5 6]]
6
[[10  2   3]
 [ 4   5   6]]
[10  2   3]
[4 5 6]
```

Slicing

- Slicing works similarly to lists, but now we can have slices in different dimensions.
- We can even assign to a slice
- Extract rows or columns from an array

```
: # Slicing
a=np.array([1,4,2,7,9,5])
print(a[1:3])

b=np.array([[1,2,3], [4,5,6]])
print(b[:,0])
print(b[0,:])
print(b[:,1:])
```

```
[4 2]
[1 4]
[1 2 3]
[[2 3]
 [5 6]]
```

```
: # assign to a slice
b[:,1:] = 7
print(b)
```

```
[[1 7 7]
 [4 7 7]]
```

```
: print(b[:,0])      # First column
print(b[1,:])       # Second row
```

```
[1 4]
[4 7 7]
```

Reshaping

- When an array is reshaped, its number of elements stays at the same, but they are reinterpreted into a different shape.
- E.g., one dimensional array into two dimension array

```
# reshape
a=np.arange(9)
anew=a.reshape(3,3)
print(a)
print(anew)
```

```
[0 1 2 3 4 5 6 7 8]
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Combining Arrays

- Combining several arrays into one bigger array:
concatenate and stack
- Concatenate: It takes n-dimensional arrays
and return an n-dimensional array.
- Stack: it takes n-dimensional arrays and return
(n+1)-dimensional array

```
: # Combining Arrays
# concatenation
a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])
print(a)
print(b)
c = np.concatenate((a,b))
print(a.ndim)
print(c.ndim)
```

```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
2
2
```

```
: # stack
print(np.stack((a,b)))
print(np.stack((a,b)).ndim)
```

```
[[[1 2]
 [3 4]]

 [[5 6]
 [7 8]]]
3
```

Concatenate

- By default, concatenate joins the arrays along axis 0.
- To joint array horizontally, add parameter
axis = 1

```
# concatenate
a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])
print(a)
print(b)
print(np.concatenate((a,b)))
```

```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

```
print(np.concatenate((a,b), axis = 1))
```

```
[[1 2 5 6]
 [3 4 7 8]]
```

Concatenate different dimensions

- If you want to concatenate arrays with different dimensions, you must first reshape the arrays to have the same number of dimensions.
 - E.g, add a new row (column) to a 2d array

```
a = np.array([[1,2],[3,4]])
b = np.array([5,6])
print(np.concatenate((a,b)))
```

```
ValueError                                 Traceback (most recent call last)
<ipython-input-66-01163ce3ad68> in <module>
      1 a = np.array([[1,2],[3,4]])
      2 b = np.array([5,6])
----> 3 print(np.concatenate((a,b)))

ValueError: all the input arrays must have same number of dimensions
```

```
a = np.array([[1,2],[3,4]])
b = np.array([5,6])
print(np.concatenate((a,b.reshape(1,2))))
```

```
[[1 2]
 [3 4]
 [5 6]]
```

```
print(np.concatenate((a,b.reshape(2,1)), axis = 1))
```

```
[[1 2 5]
 [3 4 6]]
```

Stack

- Use `stack` to create higher dimensional arrays from lower dimensional arrays:

```
b = np.array([4,5,6])
print(np.stack((b,b)))
```

```
[[4 5 6]
 [4 5 6]]
```

Split

- `split` is the inverse operation of `concatenate`.
- The input argument to `split` can be the number of equal parts the arrays is divided into.

```
# Split
d=np.arange(12).reshape(6,2)
print("d:")
print(d)
d1,d2 = np.split(d, 2)
print("d1:")
print(d1)
print("d2:")
print(d2)
```

```
d:
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]
d1:
[[0 1]
 [2 3]
 [4 5]]
d2:
[[ 6  7]
 [ 8  9]
 [10 11]]
```

Split

- The input argument to `split` can also be indices that specified explicitly the break points.
- The entries indicate where along axis (default axis = 0) the array is split.
- E.g., `np.split(d, (2, 3, 5))` split array `d` into
 - `d[:2]`
 - `d[2:3]`
 - `d[3:5]`
 - `d[5:]`

```
d=np.arange(12).reshape(6,2)
print("d:")
print(d)
sub_d = np.split(d, (2,3,5))
for d_i in sub_d:
    print(d_i)
```

```
d:
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]
[[0 1]
 [2 3]]
[[4 5]]
[[6 7]
 [8 9]]
[[10 11]]
```

Less Memory in NumPy

- Space occupied by NumPy is less compare to list.

```
: import numpy as np  
import time  
import sys  
S =range(1000)  
print(sys.getsizeof(5)*len(S))
```

```
D = np.arange(1000)  
print(D.size*D.itemsize)
```

28000

4000

Fast computation on arrays

- In addition to providing a way to store and access multi-dimension arrays, NumPy also provides several routines to perform computations on them.
- One of the reasons for the popularity of NumPy is that these computations can be very efficient, much more efficient than what Python can normally do.
- The biggest bottle-necks in efficiency are the loops, which can be iterated millions, billions, or even more times.
- What slows down loops in Python is the fact that Python is dynamically typed language: at each expression Python has to find out the types of the arguments of the operations.

Fast computation examples

- Let consider multiply 2 to a collection of numbers.
- At each iteration of the loop, Python has find out the type of the variable `x`, which can in this example be an `int`, a `float` or a `string`, and depending on this type call a different function to perform the “multiplication” by two.
- What makes NumPy efficient, is the requirement that each element in an array must be of the same type. This homogeneity of arrays makes it possible to create *vectorized* operation, which don’t operate on single elements, but on arrays (or subarrays).

```
# fast computation
def double_in_list(L):
    L2 = []
    for x in L:
        L2.append(x*2)
    return L2

l = range(1000)
#print(double_in_list(l))
%timeit double_in_list(l)

a=np.arange(1000)
#print(a*2)
%timeit a*2
```

239 µs ± 32.5 µs per loop (mean ±
3.89 µs ± 35.6 ns per loop (mean

Fast computation examples

```
a=np.arange(10)
a2=a*2
print(a)
print(a2)
```

```
[0 1 2 3 4 5 6 7 8 9]
[ 0  2  4  6  8 10 12 14 16 18]
```

- Because each iteration in NumPy is using identical operations only the data differs, this can be compiled into machine language, and then performed in one go, hence avoiding Python's dynamic typing.
- The name vector operation comes from linear algebra
 - addition of two vectors $\mathbf{a} = [a_1, a_2]$, $\mathbf{b} = [b_1, b_2]$ is element-wise addition

$$\mathbf{a} + \mathbf{b} = [a_1 + b_1, a_2 + b_2]$$

Arithmetic Operations in NumPy

- The basic arithmetic operations in NumPy are defined in the vector form.
 - + : addition
 - : subtraction
 - * : multiplication
 - / : division
 - // : floor division
 - ** : power
 - % : remainder

```
# Arithmetic Operations in NumPy
```

```
a = np.arange(1, 10)
b = np.arange(2, 20, 2)
print(a)
print(b)
print("a + b:")
print(a+b)
print("a - b:")
print(a-b)
print("a * b:")
print(a*b)
print("a / b:")
print(a/b)
print("a // b:")
print(a//b)
print("a**b:")
print(a**b)
print("a%b:")
print(a%b)
```

```
[1 2 3 4 5 6 7 8 9]
[ 2  4   6   8 10 12 14 16 18]
a + b:
[ 3  6   9 12 15 18 21 24 27]
a - b:
[-1 -2 -3 -4 -5 -6 -7 -8 -9]
a * b:
[ 2   8   18  32  50  72  98 128 162]
a / b:
[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]
a // b:
[0 0 0 0 0 0 0 0 0]
a**b:
[           1           16          729
           -381759919            0 -1953380655]
a%b:
[1 2 3 4 5 6 7 8 9]
```

Aggregations: max, min, sum, mean, standard deviations...

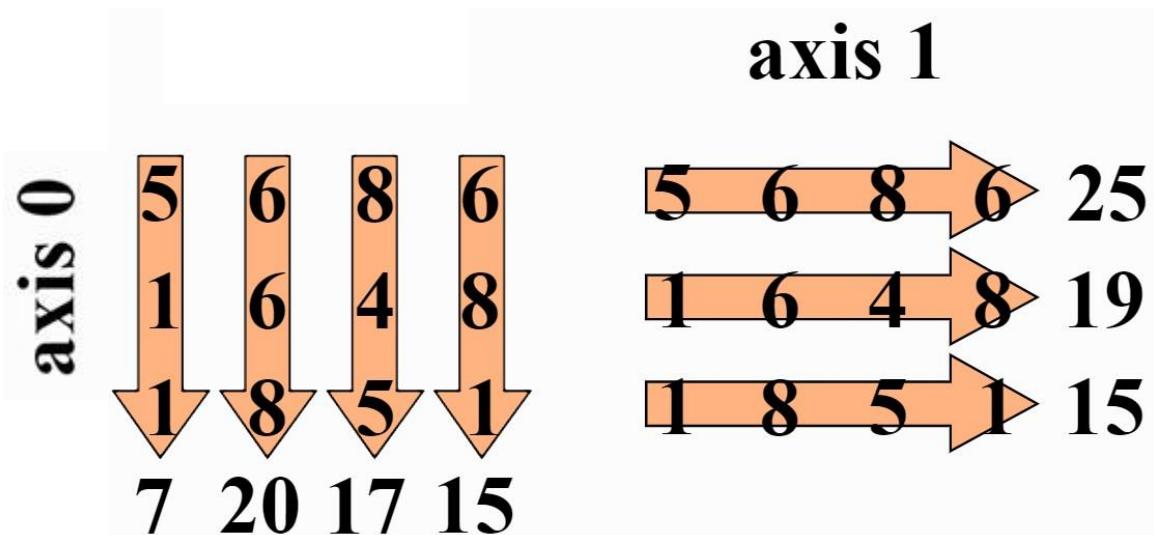
- Aggregations allow us to describe the information in an array by using few numbers

```
# aggregation
np.random.seed(0)
a=np.random.randint(-100, 100, (4,5))
print(a)
print(f"Minimum: {a.min()}, maximum: {a.max()}")
print(f"Sum: {a.sum()}")
print(f"Mean: {a.mean()}, standard deviation: {a.std()}")
```

[[72 -53 17 92 -33]
 [95 3 -91 -79 -64]
 [-13 -30 -12 40 -42]
 [93 -61 -13 74 -12]]
Minimum: -91, maximum: 95
Sum: -17
Mean: -0.85, standard deviation: 58.39886557117355

Aggregation over certain axes

- Instead of aggregating over the whole array, we can aggregate over certain axes only as well.



```
b=np.array([[5, 6, 8, 6],  
           [1, 6, 4, 8],  
           [1, 8, 5, 1]])  
print(b)  
print("Column sums:", b.sum(axis=0))  
print("Row sums:", b.sum(axis=1))
```

```
[[5 6 8 6]  
 [1 6 4 8]  
 [1 8 5 1]]  
Column sums: [ 7 20 17 15]  
Row sums: [25 19 15]
```

Python function, NumPy function, NumPy Method

Python function	NumPy function	NumPy method
sum	np.sum	a.sum
*	np.prod	a.prod
*	np.mean	a.mean
*	np.std	a.std
*	np.var	a.var
min	np.min	a.min
max	np.max	a.max

- Most of the aggregation functions in NumPy have corresponding methods.
- Python language has builtin functions sum, min, max, etc.
- Do not accidentally use Python built-in functions for arrays, since they will be significantly slower than NumPy's functions and methods.

Efficiency of NumPy functions

```
a=np.arange(1000)
%timeit np.sum(a)

%timeit sum(a)
```

```
7.12 µs ± 187 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
191 µs ± 1.56 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

- The speed of NumPy partly comes from the fact that its arrays must have same type for all the elements. This requirement allows some efficient optimizations.

Broadcasting

- We have seen that NumPy allows array operations that are performed element-wise.
- NumPy also allows binary operation that do not require the two arrays to have the same shape.
- E.g., add 4 to all elements of an array

```
np.arange(3) + 4
```

```
array([4, 5, 6])
```

Broadcasting

- How binary operation is performed?
- NumPy tries to stretch the arrays to have the same shape, then perform the element-wise operation.
- In NumPy, this stretching is called broadcasting.

```
np.arange(3) + 4  
array([4, 5, 6])
```

NumPy first stretched the scalar 4 to the array `np.array([4, 4, 4])` and then performed the element-wise addition.

Broadcasting

```
a=np.full((3,3), 5)
b=np.arange(3)
print("a:", a, sep="\n")
print("b:", b)
print("a+b:", a+b, sep="\n")
```

```
a:
[[5 5 5]
 [5 5 5]
 [5 5 5]]
b: [0 1 2]
a+b:
[[5 6 7]
 [5 6 7]
 [5 6 7]]
```

- In this example the second argument `b` was first broadcasted to the array

```
np.array([[0, 1, 2],
          [0, 1, 2],
          [0, 1, 2]])
```

- Then the addition was performed.

Comparisons and Masking

- Just like NumPy allows element-wise arithmetic operations between arrays, it is also possible to compare two arrays element-wise.
- We can also count the number of comparisons that were True. This solution relies on the interpretation that True corresponds to 1 and False corresponds to 0.
- Broadcasting rules also apply to comparison

```
a=np.array([1,3,4])  
b=np.array([2,2,7])  
c = a < b  
print(c)
```

```
[ True False  True]
```

```
print(np.sum(c))
```

```
2
```

```
print(a > 0)
```

```
[ True  True  True]
```

Masking

- Another use of Boolean arrays is that they can be used to select a subset of elements. It is called *masking*.
- It can also be used to assign a new value. For example, the following zeroes out the negative numbers.

```
: a = np.random.randint(-2, 10, (3,4))
print("a: ")
print(a)
c = a > 0
print("Positive numbers in a:")
print(a[c])      # Select only the positive numbers

a[~c] = 0
print("zeroes out negative numbers in a:")
print(a)
```

```
a:
[[ 3  4  6  2]
 [-1  2  7  8]
 [ 8  6 -1 -1]]
Positive numbers in a:
[3 4 6 2 2 7 8 8 6]
zeroes out negative numbers in a:
[[3 4 6 2]
 [0 2 7 8]
 [8 6 0 0]]
```

Fancy Indexing

- Using indexing we can get a single elements from an array. If we wanted multiple (not necessarily contiguous) elements, we would have to index several times.
- That's quite verbose. *Fancy indexing* provides a concise syntax for accessing multiple elements.
- We can also assign to multiple elements through fancy indexing.

```
# Fancy indexing
a=np.arange(10)
a2=np.array([a[2], a[5], a[7]])
print(a)
print(a2)

[0 1 2 3 4 5 6 7 8 9]
[2 5 7]
```

```
a=np.arange(10)          # List
idx=[2,5,7]               # In
print(a[idx])              # Or
print(a[[2,5,7]])

# assign
a[idx] = 0
print(a)

[2 5 7]
[2 5 7]
[0 1 0 3 4 0 6 0 8 9]
```

Fancy Indexing

- Fancy indexing works also for higher dimensional arrays.
- We can also combine normal indexing, slicing and fancy indexing.

```
b=np.arange(16).reshape(4,4)
print(b)
row=np.array([0,2])
col=np.array([1,3])
print(b[row, col])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[ 1 11]
```

```
b=np.arange(16).reshape(4,4)
print("b:")
print(b)
print("b[:,[0,3]]:")
print(b[:, [0, 3]])
```

```
b:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
b[:,[0,3]]:
[[ 0  3]
 [ 4  7]
 [ 8 11]
 [12 15]]
```

Sorting arrays

- Sorting one dimensional array is similar to sort a list.
- We can also sort a high dimensional array along different axes.

```
# sorting arrays
a=np.array([2,1,4,3,5])
print(np.sort(a))           # Does not modify the argument
print(a)
```

```
a.sort()                  # Modifies the argument
print(a)
```

```
[1 2 3 4 5]
[2 1 4 3 5]
[1 2 3 4 5]
```

```
b=np.random.randint(0,10, (4,4))
print(b)

print(np.sort(b, axis=0))   # sort each column
print(np.sort(b, axis=1))   # Sort each row
```

```
[[7 9 9 3]
 [6 7 2 0]
 [3 5 9 4]
 [4 6 4 4]]
 [[3 5 2 0]
 [4 6 4 3]
 [6 7 9 4]
 [7 9 9 4]]
 [[3 7 9 9]
 [0 2 6 7]
 [3 4 5 9]
 [4 4 4 6]]
```

Sorting arrays

- A related operation is the `argsort` function. Which doesn't sort the elements but returns the indices of the sorted elements.

```
a=np.array([2,5,6,1,3])
print("Array a:", a)
idx = np.argsort(a)
print("Indices:", idx)
print(a[idx])
```

```
Array a: [2 5 6 1 3]
Indices: [3 0 4 1 2]
[1 2 3 5 6]
```

These indices [3, 0, 4, 1, 2] say that the smallest element of the array is in position 3 of a, second smallest elements is in position 0 of a, third smallest is in position 4, and so on.

Matrix operations

- NumPy support a wide variety of matrix operations, such as matrix multiplication, solve systems of linear equations, compute eigenvalues/eigenvectors, matrix decompositions and other linear algebra related operations.

Applications of NumPy

- Mathematics (MATLAB Replacement)
- Plotting (Matplotlib)
- Backend (Pandas, Digital Photography)
- Machine Learning
- Signal Processing

IDAT7215

Computer Programming for

Product Development and

Applications

Lecture 1-1: Introduction to Programming

Dr. Zulfiqar Ali

What is a Programming Language?



A programming language is a **set of written symbols/rules** that **instructs the computer hardware to perform specific tasks**.

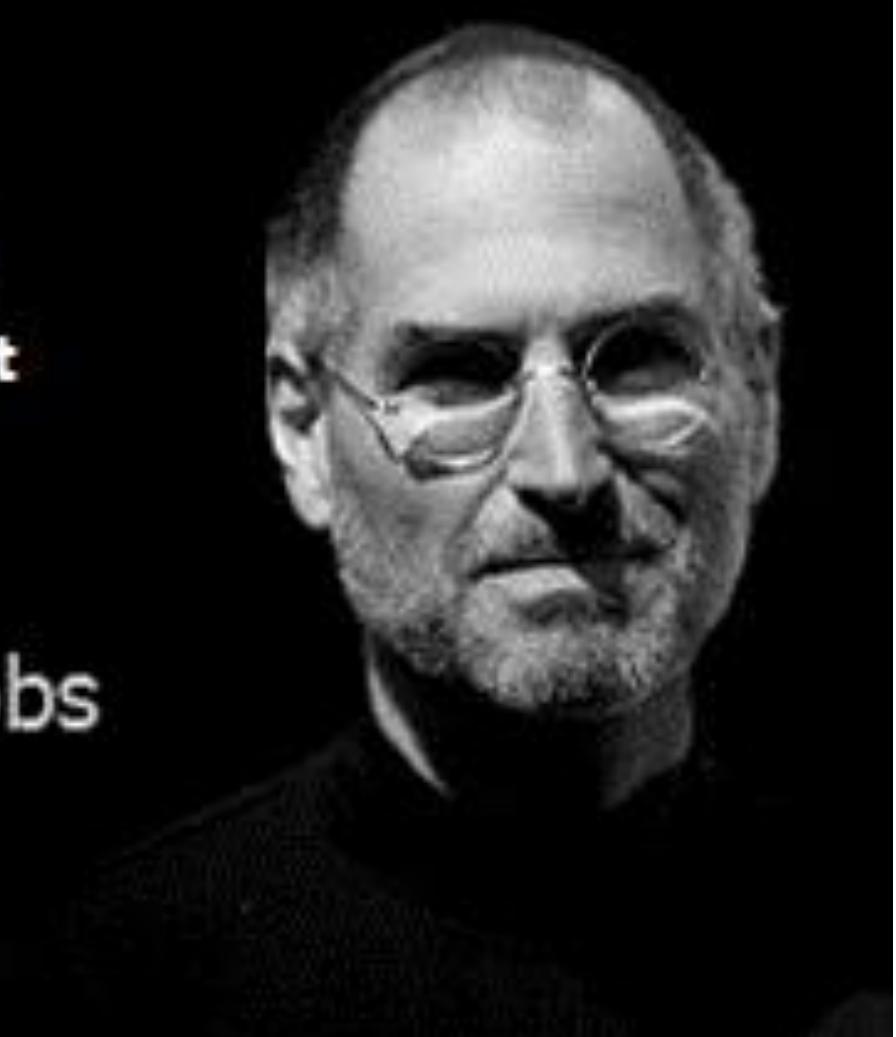
Why Programming Is Important?

- Programming helps you understand computers.
- A computer is only a tool. If you learn how to write simple programs, you will gain more knowledge about how a computer works.
- Writing a few simple programs increases your confidence level.
- Many people find great personal satisfaction in creating a set of instructions that solve a problem.
- Learning programming lets you find out quickly whether you like programming and whether you have the analytical turn of mind programmers need.
- Even if you decide that programming is not for you, understanding the process certainly will increase your appreciation of what programmers and computers can do.

Why Programming Is Important?

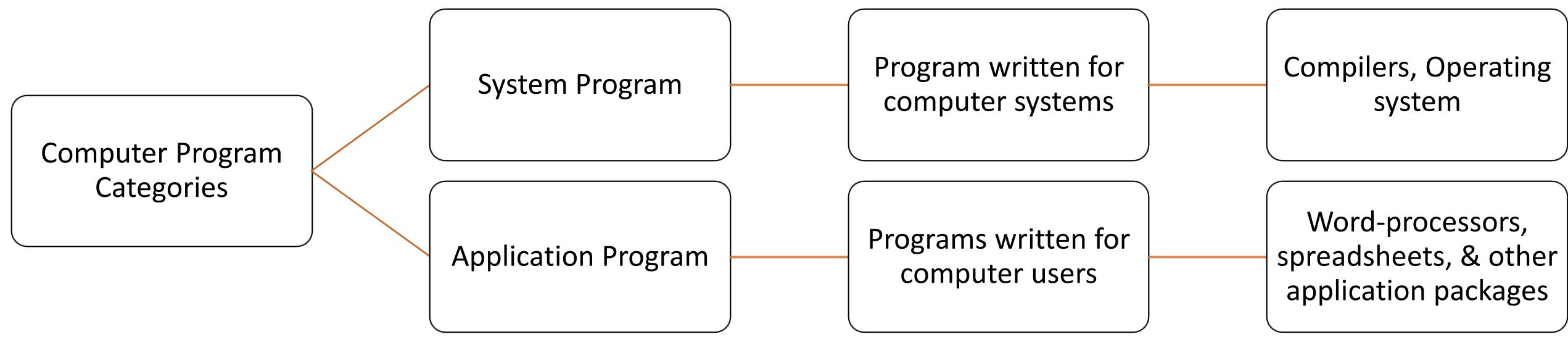
**Everyone should know how to
program a computer, because it
teaches you how to think!**

- Steve Jobs

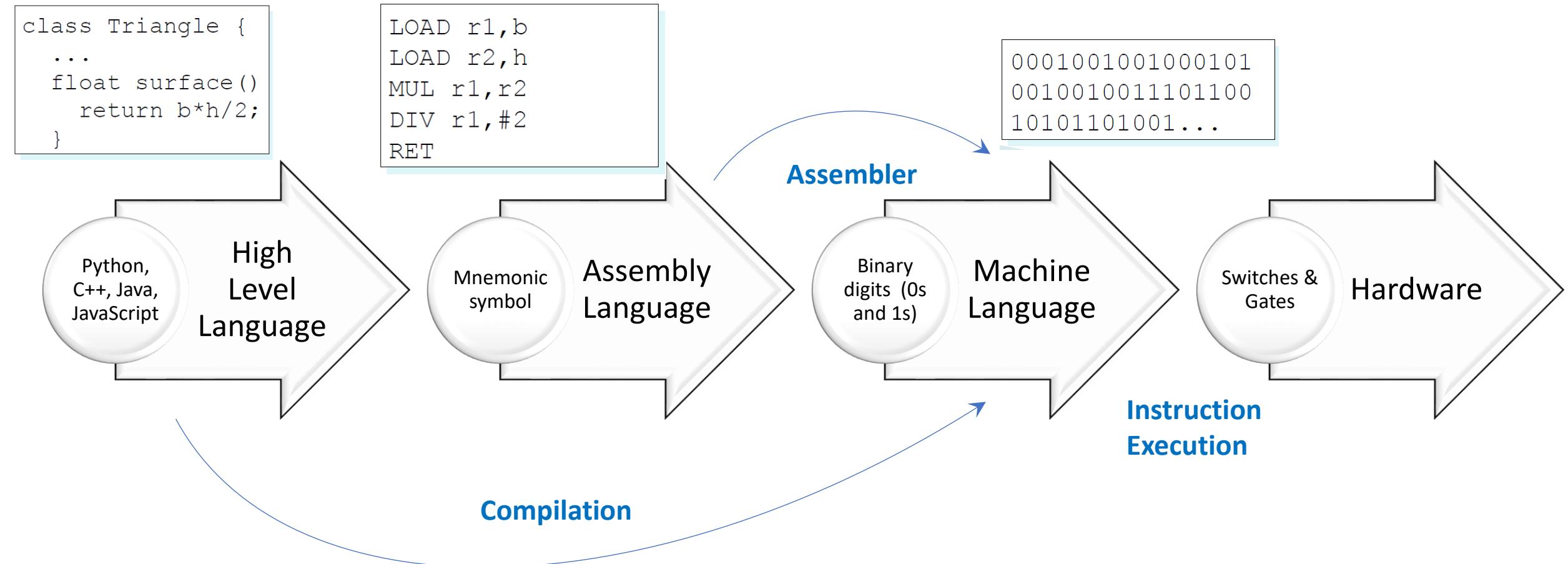


Why are there so many programming languages?

- Why does some people speak French?
- Programming languages have evolved over time as better ways have been developed to design them.
 - First programming languages were developed in the 1950s.
 - Since then, thousands of languages have been developed.
- Different programming languages are designed for different types of programs.



Computer Languages



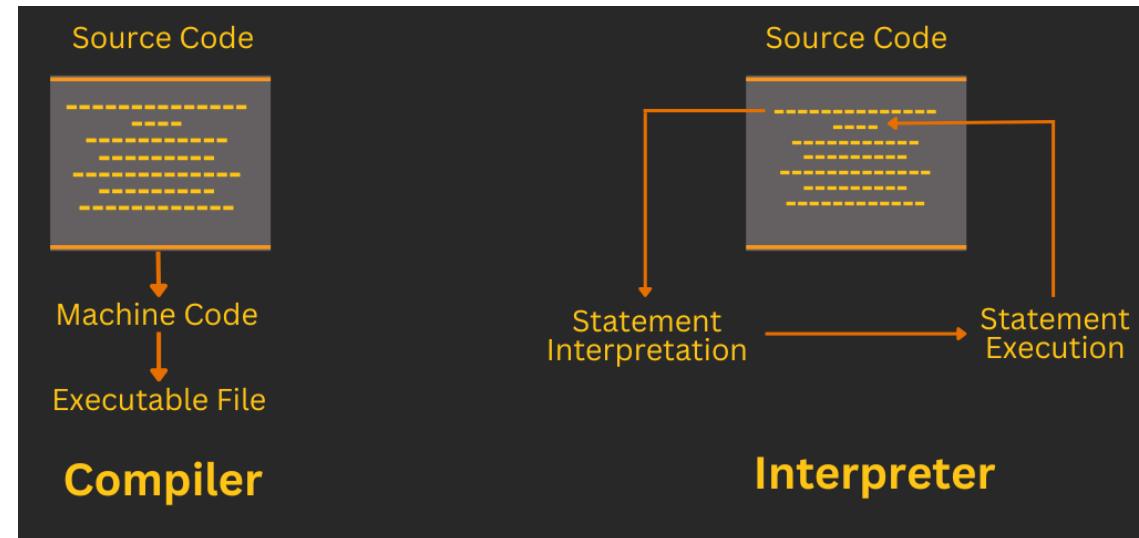
Compilers & Programs

▪ Compiler

- Compiler translates the entire source code into machine code before execution.
- Faster execution since no translation is needed during runtime.
- E.g., C, C#, C++

▪ Interpreter

- An interpreter translates code line by line during execution, making it easier to detect errors.
- Potentially slowing down the program.
- E.g., Python, Ruby, JavaScript.



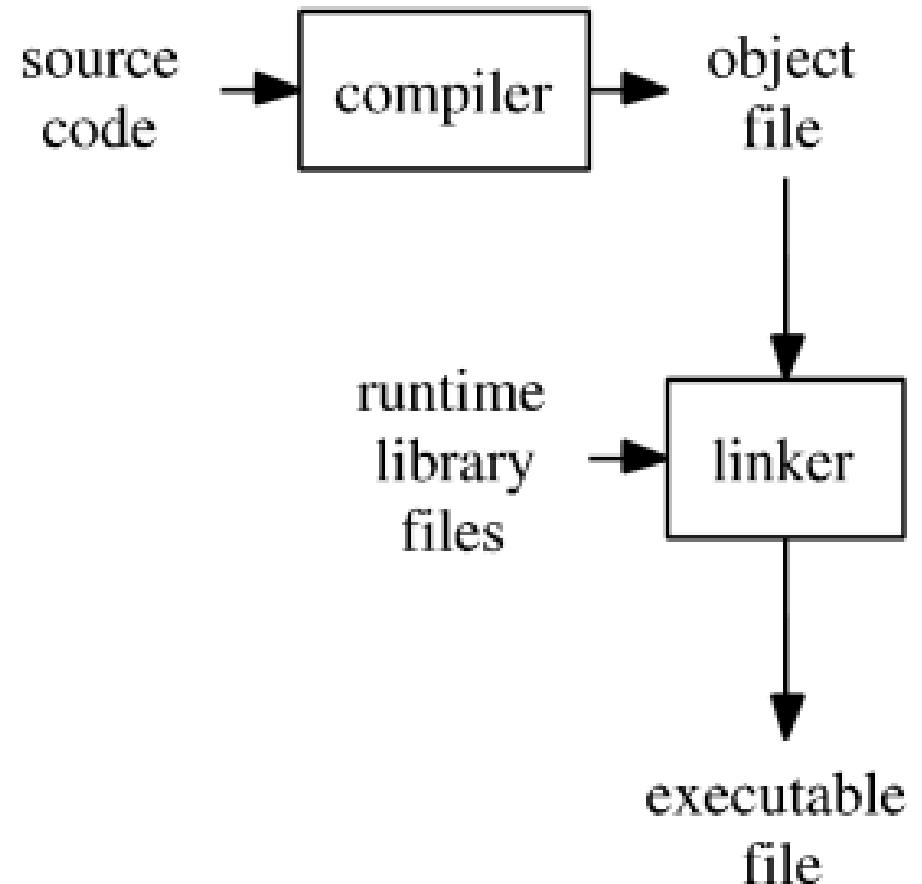
Compilers & Programs

▪ Source program

- The form in which a computer program, written in some formal programming language, is written by the programmer.
- Can be compiled automatically into object code or machine code or executed by an interpreter.

▪ Object program

- Output from the compiler
- Equivalent machine language translation of the source program
- Files usually have extension ‘.obj’



Compilers & Programs

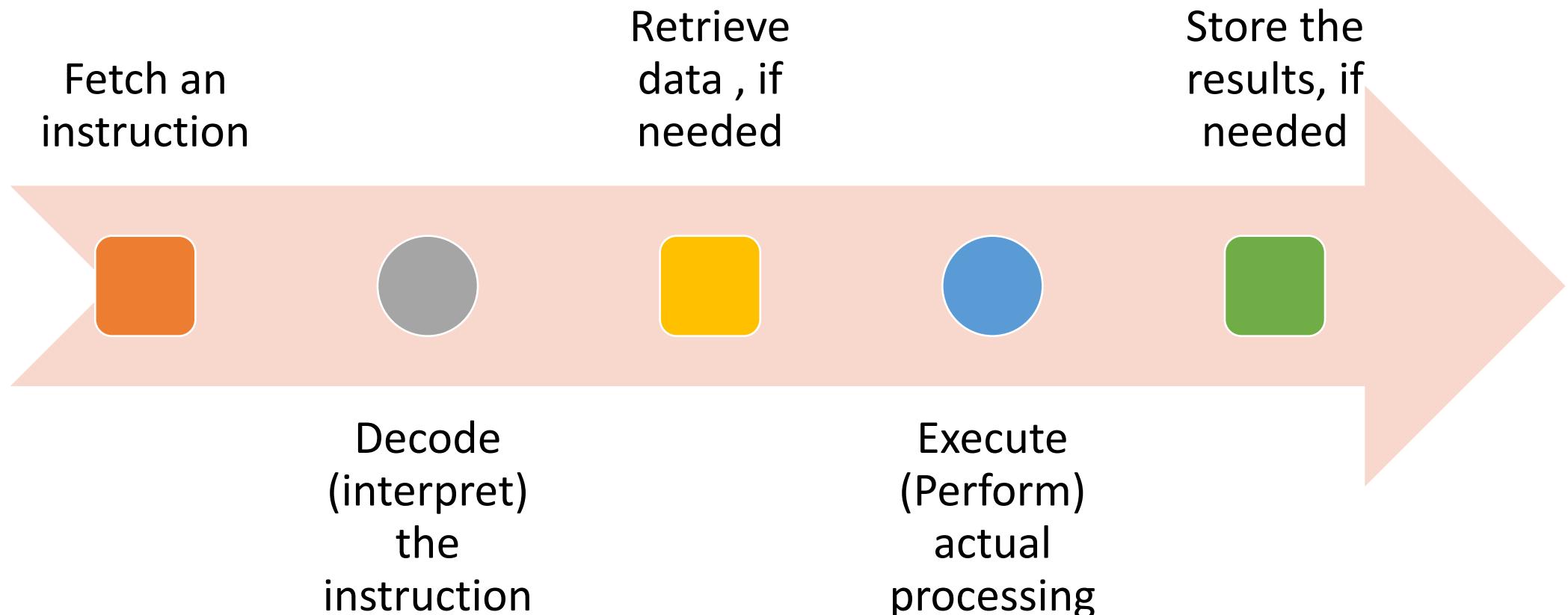
- **Executable program**

- Output from linker/loader.
- Machine language program linked with necessary libraries & other files.
- Files usually have extension '.exe'

- **What is Linker?**

- A **program that pulls other programs** together so that they can **run**.
- Most programs are very large and consist of several modules.
- Even small programs use existing code provided by the programming environment called libraries.
- The linker pulls everything together, makes sure that references to other parts of the program (code) are resolved.

Program Execution



Steps taken by the CPU to run a program
(instructions are in machine language)

```
x = 2  
y = 14  
average = x + y / 2  
print(average)
```

- Deliver incorrect results
- Incorrect usage of parentheses

Logical Error

Program Errors

Syntax Error

- Error in grammar of the language

A screenshot of a Python shell window. The code is:

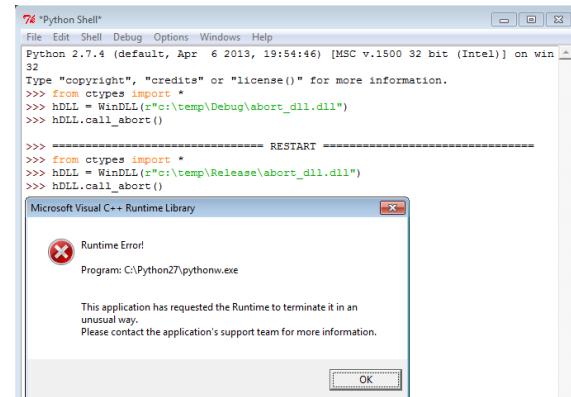
```
1 number = 55
2 if number == 55
3     print('True')
4 else
5     print('False')
```

Red arrows point from the text "Missing Colon" to the closing brace of the `if` statement and the opening brace of the `else` statement. The error message in the shell is:

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<string>", line 2
    if number == 55
               ^
SyntaxError: expected ':'
```

Runtime Error

- Divide by zero
- Invalid input data



Coding

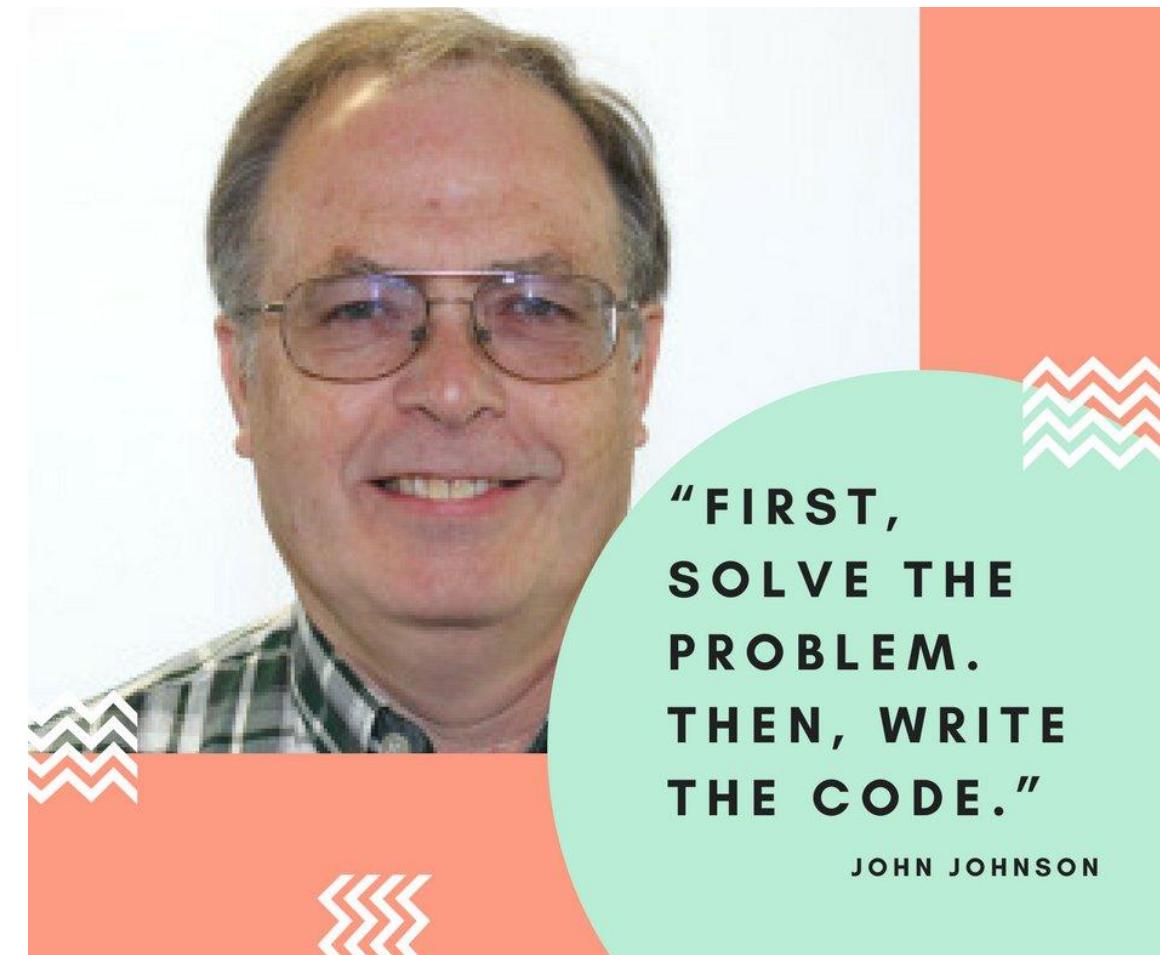
vs

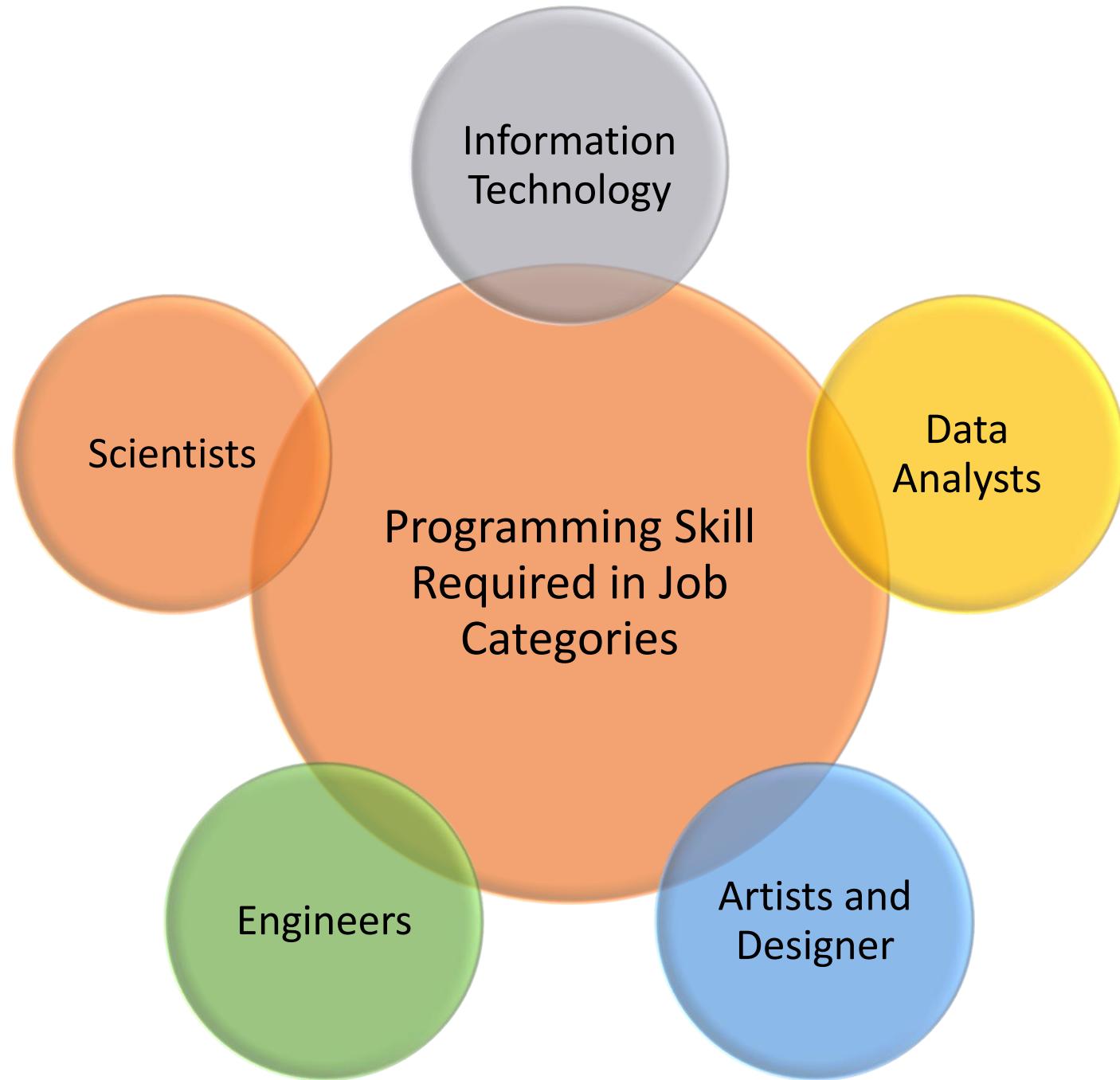
Programming

Programming is a process

- Problem Identification and Solving Phase
- Maintenance Phase
- Implementation Phase

All three of them are important for computer programmers or software developers to be able to solve problems. A lot of developers misunderstood that the programming process is only algorithm implementation, which is incorrect.



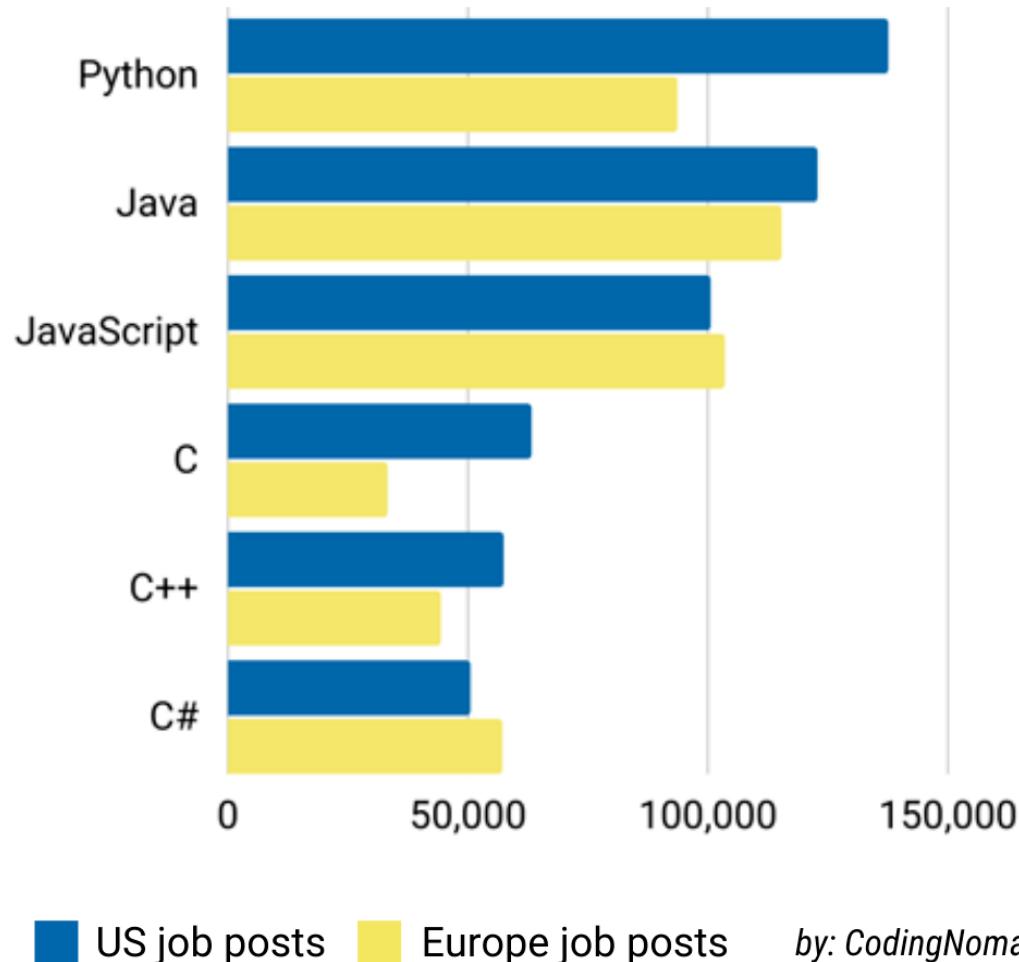


What's most in demand?

The report found that the highest demand was for programming languages with broad applicability. Other skills in demand include:

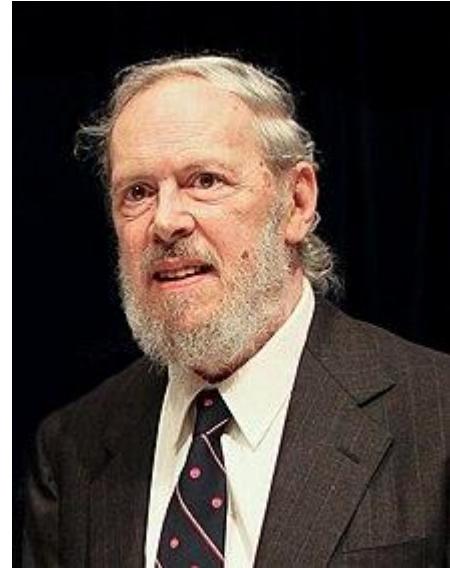


Most in-demand programming languages 2021-2022



C Language

- C is a general-purpose, **high-level language** that Dennis M. Ritchie originally developed to develop the UNIX operating system at **Bell Labs**.
- C was originally first implemented on the **DEC PDP-11 computer** in **1972**.
- In **1978**, **Brian Kernighan and Dennis Ritchie** produced the first publicly available description of C, now known as the **K&R standard**.
- The UNIX operating system, the C compiler, and all UNIX application programs have been written in C.



Structure of C Program

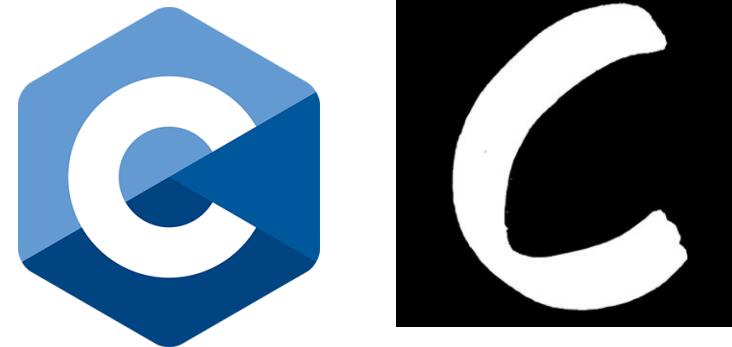
1	#include <stdio.h>	Header
2	int main(void)	Main
3	{	
4	printf("Hello World");	Statement
5	return 0;	Return
6	}	

```
5 </head>
6 <!-- Type an abbreviation & TAB -->
7 <body>|<-- Click here to insert the abbreviation
8 </body>
9 </html>
10
```

A screenshot of a code editor showing a partially typed HTML document. The cursor is at the end of the opening body tag. A tooltip or status bar at the bottom says "Type an abbreviation & TAB -->". The code editor interface includes line numbers on the left and a toolbar at the top.

C Language

- C has now become a widely used professional language for various reasons.
 - Easy to learn
 - Structure language
 - It produces efficient programs
 - It can handle low-level activities.
 - It can be compiled on a variety of computer platforms.



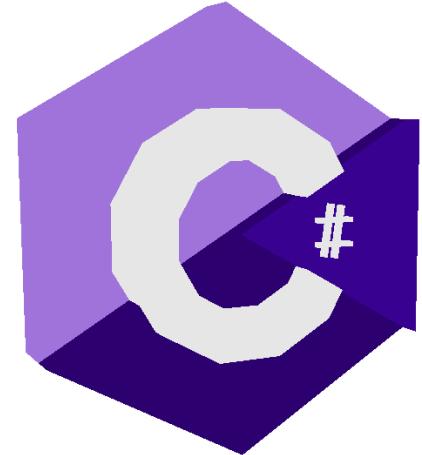
Structure of C Program

1	#include <stdio.h>	Header
2	int main(void)	Main
3	{	
4	printf("Hello World");	Statement
5	return 0;	Return
6	}	

```
5 </head>
6 <!-- Type an abbreviation & TAB -->
7 <body>|<-- Click here to insert the abbreviation
8 </body>
9 </html>
10
```

C# Language

- Nowadays, C# is the most popular language in the world.
- Microsoft developed C# within its .NET framework initiative and later approved as a standard by ECMA.
- C# programming language is a general-purpose OOPS-based programming language.
- The C# development team was led by “**Anders Hejlsberg**” in 2002.
- C# programming language is one of the languages designed for Common Language Infrastructure (CLI).



Java Language

- Java is a popular general-purpose programming language and computing platform. It is fast, reliable, and secure. According to **Oracle**, the company that owns Java, **Java runs on 3 billion devices worldwide.**
- Considering the number of Java developers, devices running Java, and companies adapting it, it is safe to say that Java will be around for many years to come.



JavaScript Language

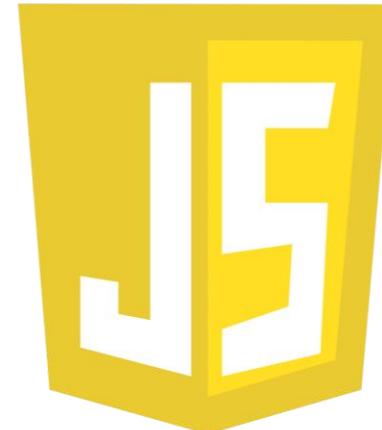
- JavaScript was initially created to “make web pages alive”.
- The programs in this language are called **scripts**. They can be written right in a web page’s HTML and executed automatically as the page loads.
- Scripts are provided and executed as plain text. They do not need special preparation or compilation to run.
- In this aspect, JavaScript is very different from another language called **Java**.

```
    })>.done(function(response) {
      for (var i = 0; i < response.length; i++) {
        var layer = L.marker(
          [response[i].latitude, response[i].longitude]
        ).addTo(map);
      }
      layers.addLayer(layers);
    })
  )
}

function changeFeature(id) {
  queryUrl = "http://www.firebaseio.com/" + id + ".json";
  $.getJSON(queryUrl, function(response) {
    if (response) {
      var layer = L.marker(
        [response.latitude, response.longitude]
      ).addTo(map);
    }
  });
}

```

JavaScript



Python Language

- Python is a widely used general-purpose, **high-level programming language**.
- It was initially designed by **Guido van Rossum in 1991** and developed by **Python Software Foundation**. It was mainly developed for emphasis on code reliability, and its syntax allows programmers to **express concepts in fewer lines of code**.
- Python is a programming language that lets you work quickly and integrate systems more efficiently.



Real World Examples



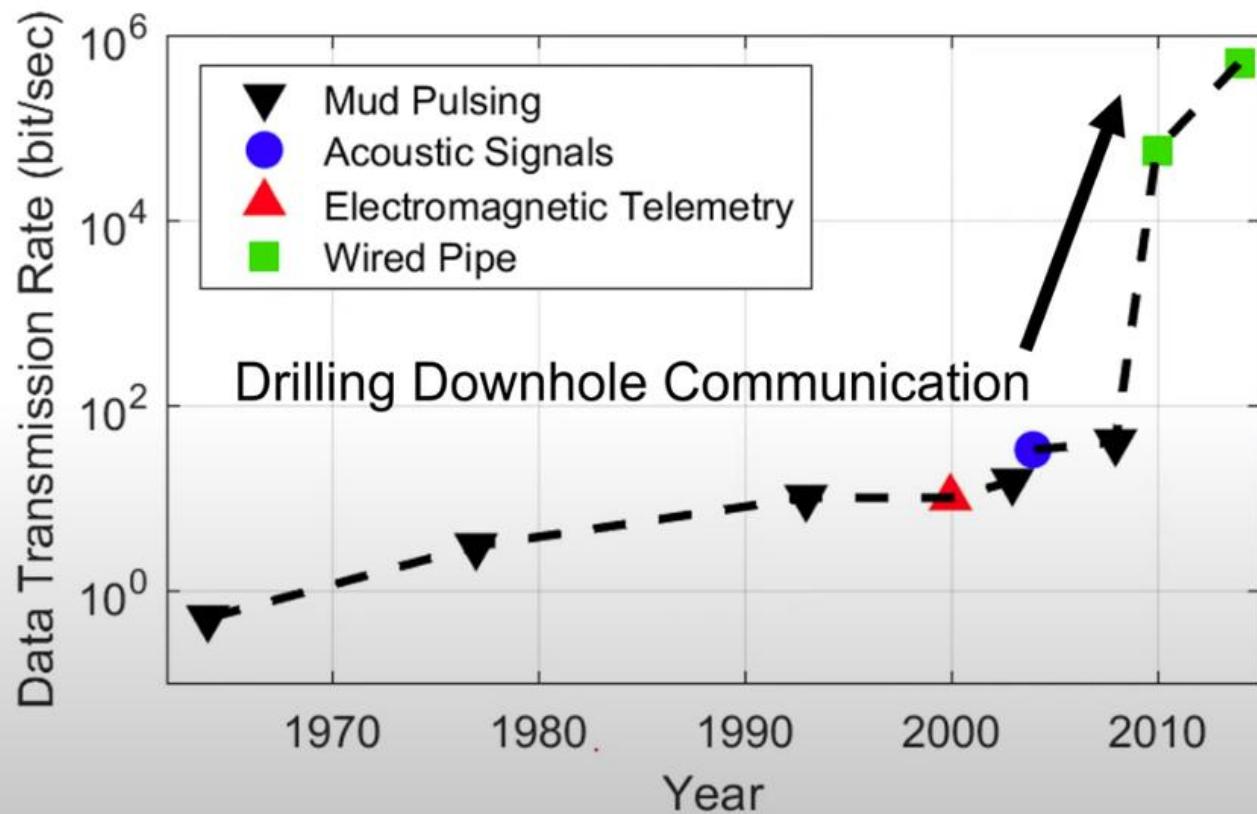
25 GB/hour



150,000
points/sec



51,200 GB/hr

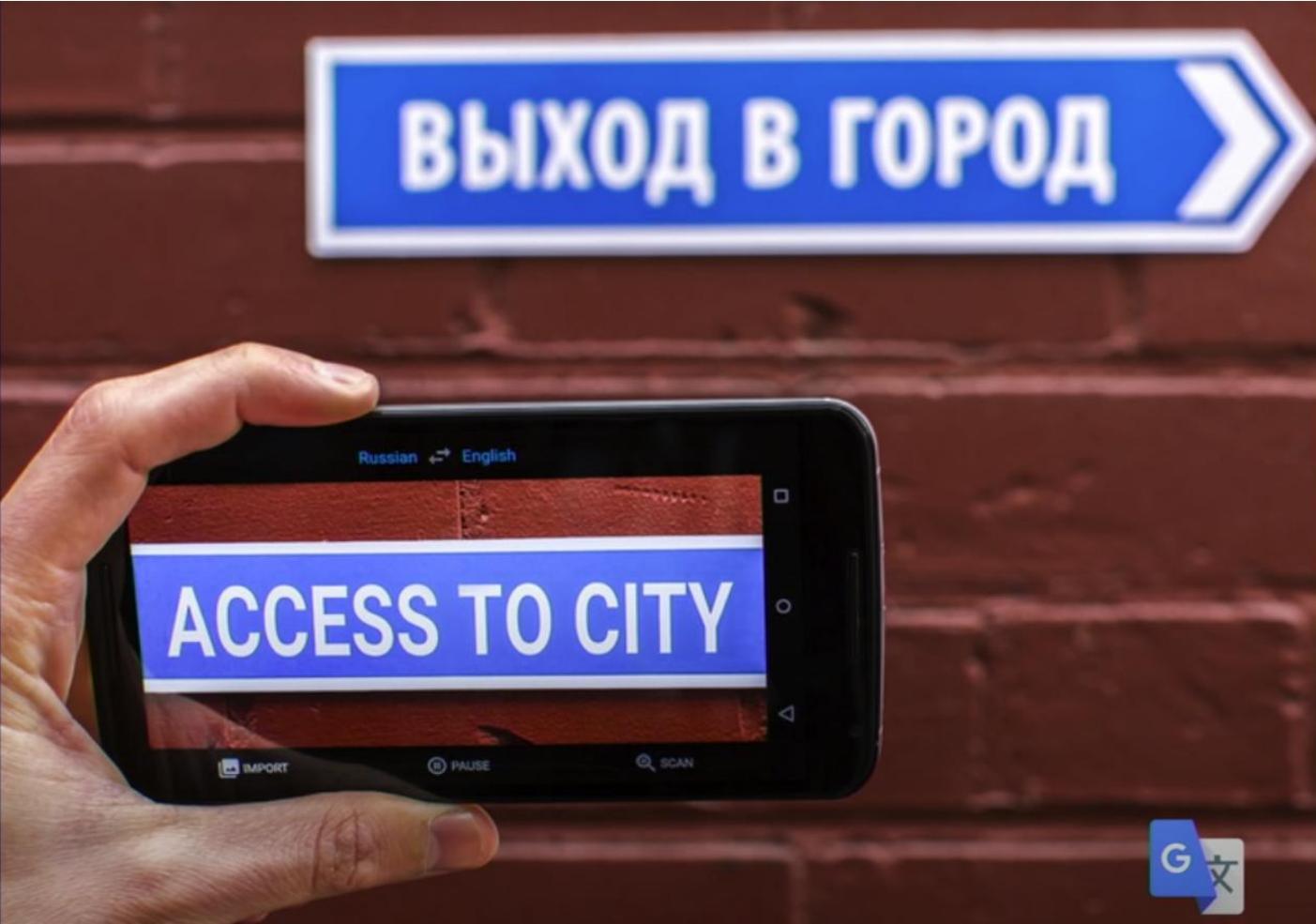


Hedengren, J. D., Eaton, A. N., Overview of Estimation Methods for Industrial Dynamic Systems, Springer, 2017, DOI: 10.1007/s11081-015-9295-9.



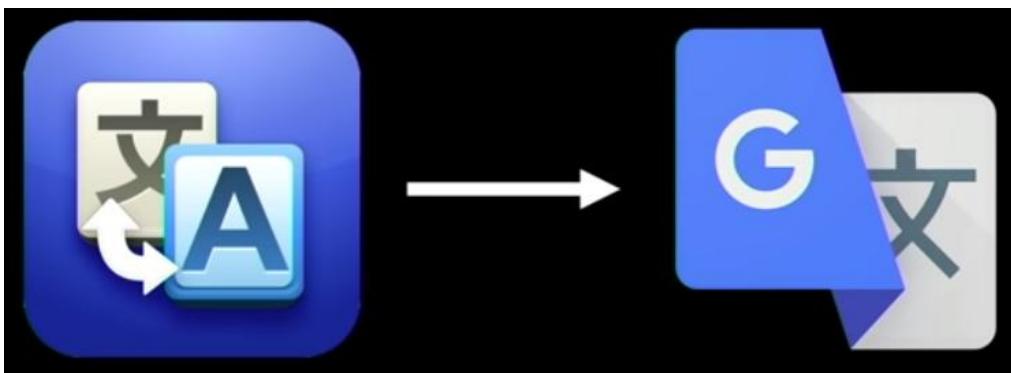
Sensors, Actuators, and Algorithms





500 GB

30 million personal computer



Google Data

10-15 Exabyte

1 EB = 1,000,000 TB
1 TB = 1000 GB

Jobs in Data Analytics

Top 10 Best Jobs in America in 2020

Rank	Job Title	Median Base Salary	Job Satisfaction	Job Openings
1	Front End Engineer	\$105,240	3.9	13,122
2	Java Developer	\$83,589	3.9	16,136
3	Data Scientist	\$107,801	4.0	6,542
4	Product Manager	\$117,713	3.8	12,173
5	Devops Engineer	\$107,310	3.9	6,603
6	Data Engineer	\$102,472	3.9	6,941
7	Software Engineer	\$105,563	3.6	50,438
8	Speech Language Pathologist	\$71,867	3.8	29,167
9	Strategy Manager	\$133,067	4.3	3,515
10	Business Development Manager	\$78,480	4.0	6,560

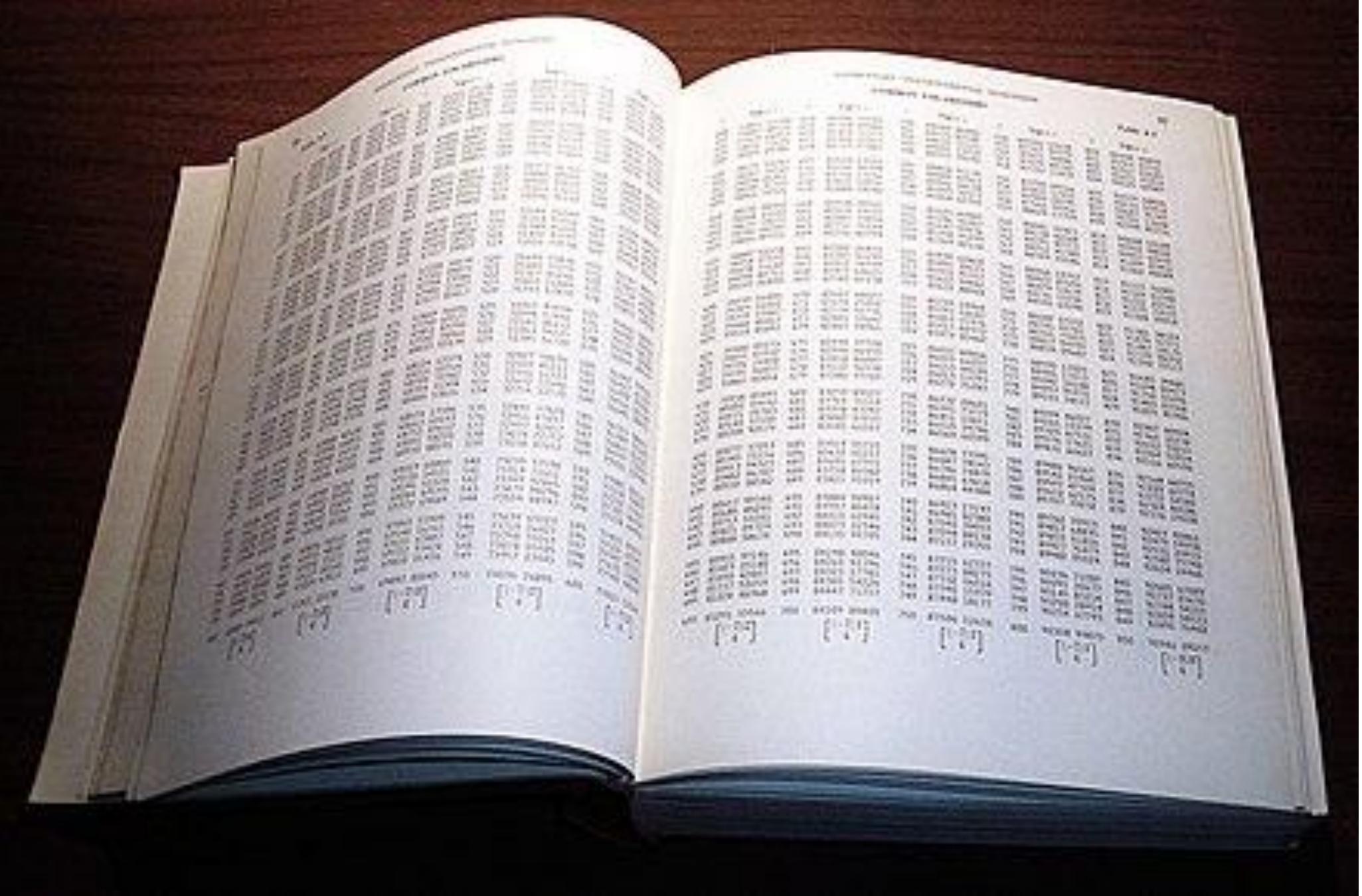
Indeed's best jobs of 2019

Rank	Job title	Average Y/Y % change in the share of job postings, 2015–2018	Average salary	Number of postings per 1M total jobs, 2018
1	Machine learning engineer	629%	\$102,500	173
2	Veterinarian	245%	\$87,000	154
3	Full-stack developer	152%	\$87,500	1,764
4	Data scientist	128%	\$98,190	569
5	Dentist	119%	\$124,677	139
6	Senior back-end developer	109%	\$102,500	77
7	Development operations engineer	97%	\$105,000	826
8	Product owner	76%	\$90,000	589
9	Financial controller	66%	\$90,000	314
10	Salesforce administrator	56%	\$85,000	56

Source: Indeed



- Data related job roles are controlling comparable jobs reports released over the past a few years.



Lookup table



Computer Understanding for Lookup Table

Humans versus machines



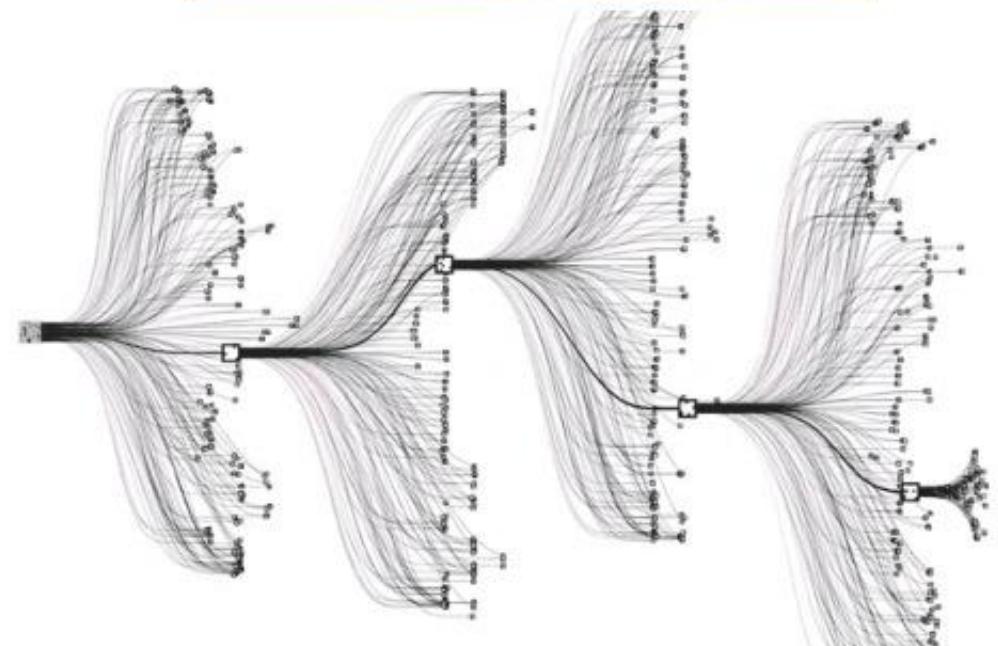
1997: Deep Blue (chess)



2011: IBM Watson (Jeopardy!)



- In 1997, Deep Blue defeated the world chess champion.
- In 2011, IBM Watson defeated two of the biggest winners at the quiz show Jeopardy
- In 2016, Google DeepMind created a program called AlphaGo, which used deep neural networks and reinforcement learning, to defeat a world champion, surprising not only the master Go player but many AI researchers as well.



Humanoid Robot and AI

- Sophia is a social **humanoid robot** developed by Hong Kong based company Hanson Robotics.
- Sophia was activated on April 19, 2015.
- She made her first public appearance at South by Southwest Festival in mid-March 2016 in United States.
- In October 2017, Sophia became a Saudi Arabian citizen, the first robot to receive citizenship in any country.



IDAT7215

Computer Programming for Product Development and Applications

Lecture 1-2: Python Installation and Python Fundamentals

Dr. Zulfiqar Ali

Python Installation

Python Installation

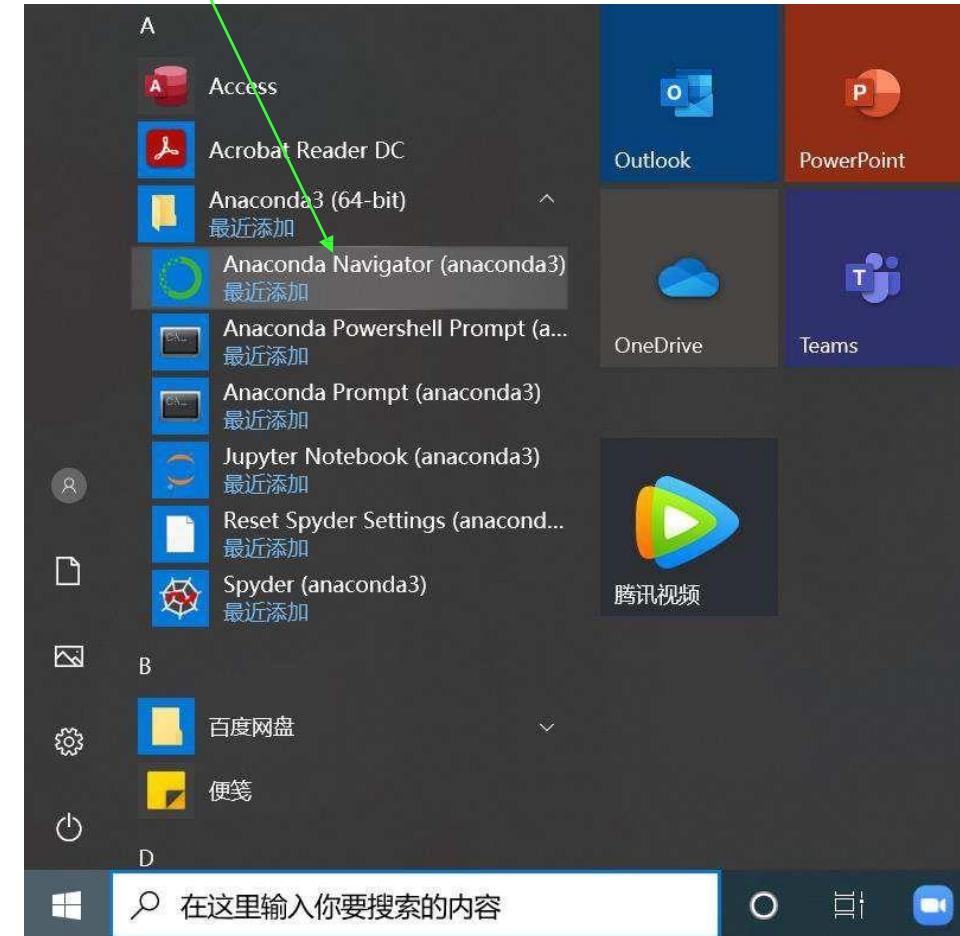
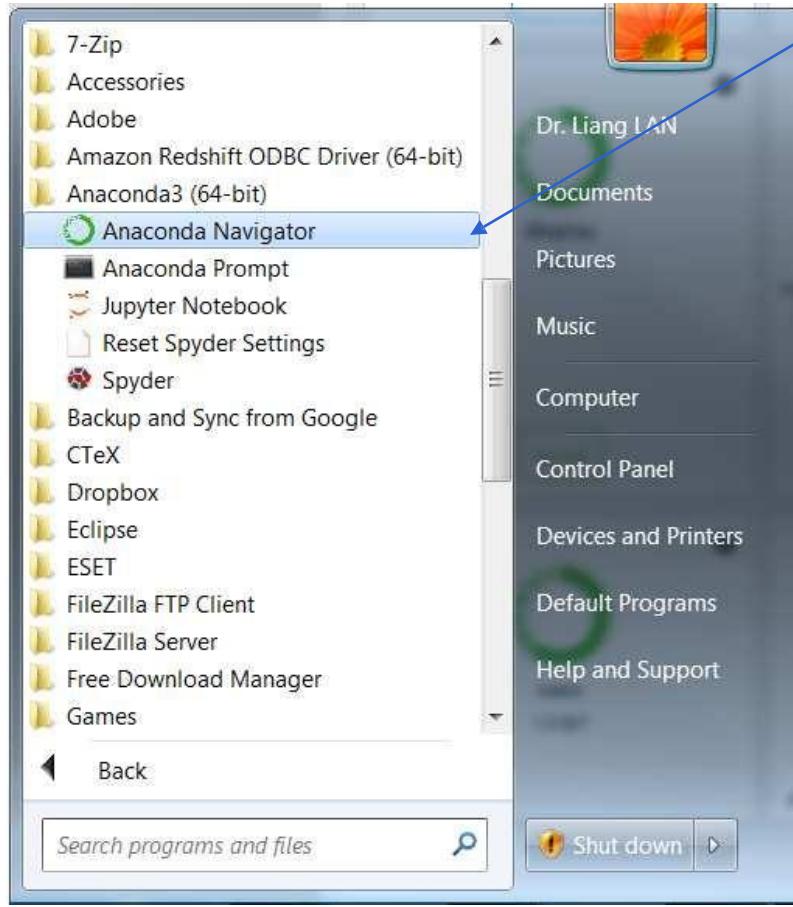
- Download and Install Anaconda
 - <https://www.anaconda.com/products/individual>
 - Choose your OS system, Recommend the 64-bit version
 - Install with the default configuration

Anaconda Installers

Windows	MacOS	Linux
 Python 3.8	 Python 3.8	 Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

Launch Jupyter Notebook

- Open Anaconda Navigator.



Launch Jupyter Notebook

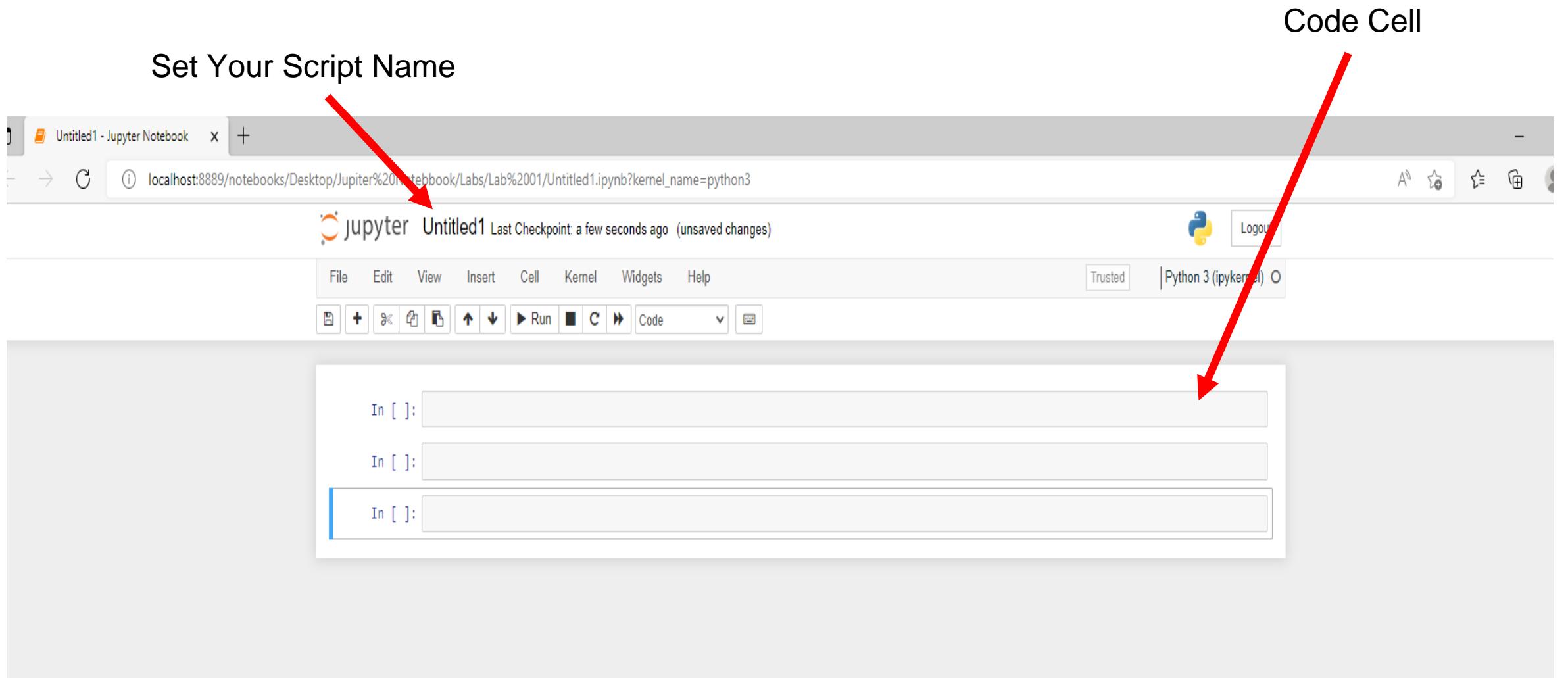
The image shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Projects (beta), Learning, and Community. The main area displays a grid of applications. The 'jupyter notebook' application is highlighted with a red arrow pointing to its 'Launch' button. The application details are as follows:

Application	Description	Version	Action Buttons
jupyterlab	An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	0.31.4	Launch, Install
jupyter notebook	Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.	5.4.0	Launch, Install
qtconsole	PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	4.3.1	Launch, Install
spyder	Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.	3.2.6	Launch, Install
vscode	Streamlined code editor with support for development operations like debugging, task running and version control.	1.25.1	Launch, Install
anypytools		0.14.1	Install
dioptas		0.4.1	Install
glueviz	Multidimensional data visualization across files. Explore relationships within and among related datasets.	0.13.3	Install
orange3	Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.	3.16.0	Install
psyplot-gui		1.1.0	Install
rstudio	A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.	1.1.456	Install
tierpsy		1.5.1b	Install

Create your first notebook

The screenshot shows the Jupyter Notebook interface. At the top, there is a navigation bar with the Jupyter logo, 'Quit', and 'Logout' buttons. Below the navigation bar, there are three tabs: 'Files' (selected), 'Running', and 'Clusters'. A message 'Select items to perform actions on them.' is displayed above a file tree. The file tree shows a directory structure: '0' (selected), 'Desktop / HKU / Teaching / IDAT 7215 Computer Programming / Jupiter Notebook / Labs'. Inside 'Labs', there are four sub-folders: 'Lab 01', 'Lab 02', 'Lab 03', and 'Lab 04'. On the right side of the interface, there is a 'New' button with a dropdown menu. The dropdown menu includes options: 'Upload', 'New', 'Notebook:', 'Text File', 'Folder', and 'Terminal'. The 'Notebook:' option is expanded, showing 'Python 3 (ipykernel)' (which is highlighted with a red arrow) and 'Create a new notebook with Python 3 (ipykernel)'. The status bar at the bottom indicates 'localhost:8888/tree/Desktop/HKU/Teaching/IDAT 7215 Computer Programming/Jupiter Notebook/Labs#'. The timestamp '18 minutes ago' is also visible.

Create your first notebook



Python fundamentals

Outline

- Variables and Types in Python
- Data Structures in Python
- Python Functions
- Control Flows

Variables and Types

- Variable
 - Specific, case-sensitive name
 - Call up value through variable name
- In Python, we do not have to explicitly declare the type of the variable (e.g., ‘dog_name’) in any way.
- Python automatically detected the type of a variable.

```
In [2]: dog_name = 'Freddie'  
       dog_age = 3  
       dog_birth_year = 2006  
       dog_weight = 30.2  
       is_vaccinated = True
```

```
In [3]: dog_name  
out[3]: 'Freddie'
```

Python Types

- The type of a variable can be queried by using the built-in function `type`.
- The type of a variable is not fixed.

```
In [3]: a = 3  
        type(a)
```

Out[3]: int

```
In [4]: a = 'hello'  
        type(a)
```

Out[4]: str

Basic data types in Python

- Basic data types
 - int (integers)
 - float (floating-point number)
 - str (string)
 - bool (boolean: True or False)
 - ...
- The name of types act as conversion operators between types

```
In [2]: dog_name = 'Freddie'  
        dog_age = 3  
        dog_birth_year = 2006  
        dog_weight = 30.2  
        is_vaccinated = True
```

```
In [3]: dog_name  
out[3]: 'Freddie'
```

```
In [6]: print(int(2.8))  
        print(float(2))  
        print(int("123"))  
        print(bool(-2), bool(2))  
        print(str(234))
```

```
2  
2.0  
123  
True True  
234
```

Arithmetic Operations in Python

- Python supports the following basic arithmetic operations:

Operator	Operation	Example	
=	Assign a value to a variable	>>> a = 2	# a = 2
+	Addition	>>> a = 199 + 201	# a = 400
-	Subtraction	>>> a = 200 - 300	# a = -100
*	Multiplication	>>> a = 15.2 * 3	# a = 45.6
/	Division (Remark: it always returns a float)	>>> a = 15 / 2	# a = 7.5
//	Floor division (or integer division)	>>> a = 15 // 2	# a = 7
**	Power	>>> a = 4 ** 3	# a = 64
%	Remainder	>>> a = 15 % 2	# a = 1

String Operations in Python

■ Create a string

- A string is a sequence of characters.
- The characters are specified either between single (') or double (") quotes
- E.g., a = "abc" or a = 'abc'

```
# Create a string
message = "Hello World!"
print(message)
message = 'Hello World!'
print(message)
```

```
Hello World!
Hello World!
```

■ Concatenate strings

- '+' operator
- join

```
# concatenate string
message1 = 'Hello'
message2 = 'World!'
message = message1 + ' ' + message2
print(message)

message = (' ').join([message1, message2])
print(message)

message = (' ').join([message1, message2, message1, message2])
print(message)
```

```
Hello World!
Hello World!
Hello World! Hello World!
```

String Operations in Python

- Repeat a string

```
In [14]: message = "Hello World!"  
        # Repeat string  
        print(message*3) # repeat the message three times
```

Hello World!Hello World!Hello World!

- Tokenize a string

```
In [11]: # tokenize by space  
words = message.split(' ')  
print (words)
```

['Hello', 'World!']

- Replace

```
In [12]: # replace  
new_message = message.replace('Hello', 'Morning')  
print(new_message)
```

Morning World!

Data Structures In Python

- Lists
- Tuples
- Dictionaries
- Sets

Lists

- Probably the most fundamental data structure in Python.
- A list contains arbitrary number of elements that stored in sequential order.
- The elements are separated by commas and written between square brackets.
- Indexing begins from 0
- The elements don't need to be of the same type.
- The elements of a list can be any type, including “List” itself.
- List is mutable.

List: Collection of Python Objects. Mutable

In [18]:

```
# supports objects of different data types
z = [1,4,'c',4, 2, 6]
print(z)
```

[1, 4, 'c', 4, 2, 6]

In [19]:

```
# Append to List
z.append('a')
print(z)
```

[1, 4, 'c', 4, 2, 6, 'a']

In [20]:

```
z[0] = 'a'
print(z)
```

['a', 4, 'c', 4, 2, 6, 'a']

Common Operations on List

- Get or set the i-th element in a list by indexing with square brackets
- Get or set elements in a list by slicing with square brackets
 - General format for slicing:
 $a[\text{first}:\text{last}:\text{step}]$

inclusive not inclusive

In [24]:

```
z = [1,4,'c',4, 2, 6]
# get the i-th element by indexing
print(z[2])
# set the i-th element by indexing
z[2] = 'a'
print(z)

# get elements by slicing
print(z[0:4])
print(z[0:4:2])
print(z[4:0:-2])

z[0:2] = ['a','a']
print(z)
```

c

```
[1, 4, 'a', 4, 2, 6]
[1, 4, 'a', 4]
[1, 'a']
[2, 'a']
['a', 'a', 'a', 4, 2, 6]
```

Common Operations on List

- Sorting sequences
 - `sort`: it modifies the original list
 - `sorted`: it returns a new sorted list and leaves the original list unchanged.
 - The parameter `reverse=True` (both to `sort` and `sorted`) can be given to get descending order of elements

In [28]:

```
# sorting a list
a = [3, 9, 4, 2]
a.sort()
print(a)

b = [3, 9, 4, 2]
print(sorted(b))
print(b)

print(sorted(b, reverse = True))
```

```
[2, 3, 4, 9]
[2, 3, 4, 9]
[3, 9, 4, 2]
[9, 4, 3, 2]
```

Common Operations on List

■ Concatenate lists

- `+`: return a new list. The original list is unchanged
- `extend`: modify a list in place
- `append`: add a single element to the end of a list
 - If append another list onto a list, it will treat another list as a single object

In [32]:

```
# concatenate list
a = [1, 3, 2]
b = ['a', 'b', 'c']
a.extend(b)
print(a)
```

```
a = [1, 3, 2]
b = ['a', 'b', 'c']
print(a+b)
print(a)
```

```
a.append(0)
print(a)
```

```
a = [1, 3, 2]
a.append(b)
print(a)|
```

```
[1, 3, 2, 'a', 'b', 'c']
[1, 3, 2, 'a', 'b', 'c']
[1, 3, 2]
[1, 3, 2, 0]
[1, 3, 2, ['a', 'b', 'c']]
```

Tuples

- Tuples are lists' **immutable** cousins.
- Pretty much anything you can do to a list that doesn't involve modifying it, you can do to a tuple.
- Tuples are defined by using parentheses (or nothing) instead of square brackets.

Tuple: Collection of Python Objects. Immutable

```
t =('a','b',3)
print(t)
print(type(t))
print(t[1])
```



```
t = 'a', 'b', 3
print(type(t))
```

```
('a', 'b', 3)
<class 'tuple'>
b
<class 'tuple'>
```

```
t[1]=2
```

```
-----
```

```
TypeError
```

```
Input In [2], in <cell line: 1>()
----> 1 t[1]=2
```

```
Traceback (most recent call last)
```

```
TypeError: 'tuple' object does not support item assignment
```

Tuples

- Tuples are a convenient way to return multiple values from functions

```
def sum_and_product(x, y):  
    sum_ = x + y  
    product = x*y  
    return sum_, product  
  
sp = sum_and_product(10,5)  
print(sp)  
  
s, p = sum_and_product(10,5)  
print(s)  
print(p)
```

```
(15, 50)  
15  
50
```

Dictionaries

- Dictionary associates values with keys.
- Dictionary can be created by listing the comma separated key-value pairs in curly brackets. Keys and values are separated by a colon.
- Dictionary allows to quickly retrieve the value corresponding to a given key using square brackets.

In [47]:

```
# dictionaries
student_grades = {"Joel":80, "Tim": 95, "Anna": 100}
print(student_grades["Joel"])
```

80

Sets

- Set represents a collection of unique elements.
- Set can be created by listing its elements between curly brackets.
- Set can also be created by using `set()` itself

Sets collection of unique elements

In [48]:

```
# a set is not ordered
a = {1, 2, 3, 3, 3, 4, 5, 'a'}
print(a)
a = set([1, 2, 3, 3, 3, 4, 5, 'a'])
print (a)
a = set() # an empty set
a.add(1)
a.add(1)
print(a)
```

```
{1, 2, 3, 4, 5, 'a'}
{1, 2, 3, 4, 5, 'a'}
{1}
```

In [49]:

```
b = set('abaaaacdef')
print (b) # not ordered
```

```
{'d', 'b', 'c', 'f', 'a', 'e'}
```

Sets

- Sets are used for two main reasons:
 - Very fast membership checking (`in` is a very fast operation on sets)
 - Find the unique items in a collections
- Sets are used less frequently than lists and dictionaries

In [78]:

```
# Membership checking
import time

word = ['a', 'an', 'at', 'b', 'bc', 'z']
%timeit 'b' in word

word = set(['a', 'an', 'at', 'b', 'bc', 'z'])
%timeit 'b' in word

44.9 ns ± 0.0664 ns per loop (mean ± std. dev.
25.3 ns ± 1.15 ns per loop (mean ± std. dev. of
```

In [79]:

```
# Unique items in a list
a = [1, 2, 1, 2, 1, 1, 3]
print(len(a))
unique_a = set(a)
print(unique_a)
print(len(unique_a))
```

7

{1, 2, 3}

3

Data Structures In Python

Type	Example	Description	Can we change the content within this collection?	Are the items in the collection ordered stored?
list	[1, 2, 3]	List: an <i>mutable, ordered</i> collection	Yes – mutable	Yes – ordered
tuple	(1, 2, 3)	Tuple: an <i>immutable, ordered</i> collection	No – immutable	Yes – ordered
set	{1, 2, 3}	Set: <i>Unordered</i> collection of unique values	Yes – mutable	No – unordered
dict	{'a':1, 'b':2, 'c':3}	Dictionary: <i>Unordered</i> (key, value) mapping	Yes – mutable	No – unordered

Python Functions

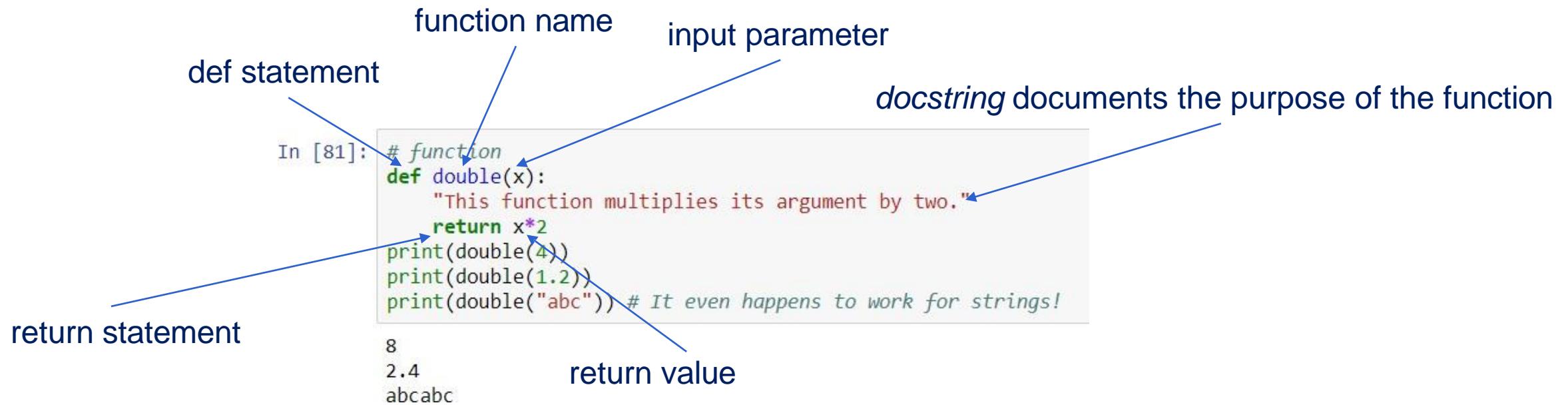
- A function is a rule for taking zero or more inputs and returning a corresponding output by performing a sequence of operations.
- When you define a function, you specify the name and the sequence of operations.
- Later, you can “call” the function by name.
- In Python, a function is defined with `def` statement.

In [81]:

```
# function
def double(x):
    "This function multiplies its argument by two."
    return x*2
print(double(4))
print(double(1.2))
print(double("abc")) # It even happens to work for strings!
```

```
8
2.4
abcabc
```

Python Functions



Python Functions

- Function parameter can also be given default values.

```
In [83]: def my_print(person = "Joe"):  
    message = "Hello, " + person  
    print(message)
```

```
my_print("Peter")  
my_print()
```

Hello, Peter

Hello, Joe

Python Functions

- In addition to positional arguments we have seen so far, a function call can also have *named arguments*.

```
In [84]: def named(a, b, c):
    print("First:", a, "Second:", b, "Third:", c)
```

```
named(5, 8, 7)
named(5, c=7, b=8)
```

```
First: 5 Second: 8 Third: 7
```

```
First: 5 Second: 8 Third: 7
```

- Note that the named arguments didn't need to be in the same order as in the function definition. The named arguments must come after the positional arguments.

```
In [85]: named(a=5, 7, 8)
```

```
File "<ipython-input-85-33e6d8dbc2a3>", line 1
```

```
    named(a=5, 7, 8)
        ^
```

```
SyntaxError: positional argument follows keyword argument
```

Control Flows in Python

- With control flow, you can execute certain code blocks conditionally and/or repeatedly: these basic building blocks can be combined to create sophisticated programs.
- Three different categories for control flows
 - Loops for repetitive tasks (`while`, `for`)
 - Decision making with the `if-else` statement
 - Exception-Handling (`break`, `continue`, `try-except`)

Loops for repetitive tasks

- In Python we have two kinds of loops: `while` and `for`
- While loop is used to execute a block of statements repeatedly until a given condition is satisfied.

```
In [87]: # while loop
i=1
while i*i < 100:
    print("Square of", i, "is", i*i)
    i = i + 1
print("Finished printing all the squares below 100.")
```

```
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
Square of 6 is 36
Square of 7 is 49
Square of 8 is 64
Square of 9 is 81
Finished printing all the squares below 100.
```

Loops for repetitive tasks

- For loop are used for sequential traversal.

```
In [90]: # for loop
for i in [1,1,2,3,4,5,6,7,8,9]:
    print("Square of", i, "is", i*i)
```

```
Square of 1 is 1
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
Square of 6 is 36
Square of 7 is 49
Square of 8 is 64
Square of 9 is 81
```

- At each iteration, the variable `i` refers to another value from the list in order.

Decision making with the if statement

- The `if-else` statement is used to decide whether a certain statement or block of statements will be executed or not.
 - if a certain condition is true then a block of statement is executed otherwise not.

In [91]:

```
# if-else
def absolute_value(x):
    if x >= 0:
        a=x
    else:
        a=-x
    return a

print(absolute_value(2))
print(absolute_value(-2))
```

```
2
2
```

General form of an if-else statements

```
if condition1:  
    statement1_1  
    statement1_2  
    ...  
elif condition2:  
    statement2_1  
    statement2_2  
    ...  
...  
else:  
    statementn_1  
    statementn_2  
    ...
```

```
In [1]: c = -2  
if c > 0:  
    print("c is positive")  
elif c<0:  
    print("c is negative")  
else:  
    print("c is zero")
```

c is negative

Exception handling

- Sometimes, you will find that you will need to handle for exceptions.
- The exceptions could be invalid operations (e.g., divide by 0 or compute the square root of a negative number) or a class of values that you simply are not interested in.
- To handle these exceptions, you can use
 - Break
 - Continue
 - Try-except

Break

- `break` statement is only allowed inside a loop body.
- When `break` executes, the loop terminates.
- In practical use, a `break` statement is usually used with `if` statement to break a loop conditionally.

Break

- Breaking the loop, when the wanted element is found

```
In [2]: # break
# find the first negative number in a list
l=[1,3,65,3,-1,56,-10]
for x in l:
    if x < 0:
        break
print("The first negative list element was", x)
```

```
The first negative list element was -1
```

Continue

- Continue statement is only allowed inside a loop body.
- When continue executes, the current iteration of the loop terminates, and the execution continues with next iteration of the loop.
- In practical use, a continue statement is usually used with if statement to executes conditionally.

Continue

- Stopping current iteration and continuing to the next one (do not break the loop).

In [5]:

```
# continue
# compute the square root of each element in a list
from math import sqrt
l=[1,3,65,3,-1,56,-10]
for x in l:
    if x < 0:
        continue
    print("Square root of " +str(x) + " is " + str(sqrt(x)))
```

```
Square root of 1 is 1.0
Square root of 3 is 1.7320508075688772
Square root of 65 is 8.06225774829855
Square root of 3 is 1.7320508075688772
Square root of 56 is 7.483314773547883
```

Try-except

- Unhandled exceptions will cause your program to crash. You can handle them using `try` and `except` statement
- If the code in the `try` block works, the `except` block is skipped.
- If the code in the `try` block fails, it jumps to the `except` section.

In [7]: # try-exception example

```
try:  
    print(0 / 0)  
except ZeroDivisionError:  
    print("cannot divide by zero")
```

cannot divide by zero

Without the `try` block, the program will crash and raise an error.

```
print(0/0)
```

```
-----  
ZeroDivisionError  
<ipython-input-8-1babbb3b33639> in <module>()  
----> 1 print(0/0)
```

```
ZeroDivisionError: division by zero
```

Try-except

```
while True:  
    x = int(input("Please enter a number: "))  
    print("%s squared is %s" % (x, x**2))
```

Please enter a number: 2

2 squared is 4

Please enter a number: a

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-9-b935f19ed274> in <module>()  
      1 while True:  
----> 2     x = int(input("Please enter a number: "))  
      3     print("%s squared is %s" % (x, x**2))  
  
ValueError: invalid literal for int() with base 10: 'a'
```



```
while True:  
    try:  
        x = int(input("Please enter a number: "))  
        print("%s squared is %s" % (x, x**2))  
    except ValueError:  
        print("Please enter a valid number!")
```

Please enter a number: 2

2 squared is 4

Please enter a number: a

Please enter a valid number!

Please enter a number: 3

3 squared is 9

Please enter a number:

<https://www.w3schools.com/python/default.asp>

IDAT7215

Computer Programming for

Product Development and

Applications

Course Instructor: Dr. Zulfiqar Ali

About Me

■ Dr. Zulfiqar Ali

- Lecturer in the Department of Mechanical Engineering, HKU
- Postdoctoral Fellow from HKUST
- PhD., City University of Hong Kong
- CAD/CAM, Isogeometric Analysis, FEA, 3D Printing

■ Contact information

- Email: zulfiali@hku.hk
- Office: HW 5-28A
- Office Phone: +852-22194813
- Mobile Phone: +852-55872706
- Office hours: 3:00pm – 6:00 pm(Wednesday) or by appointments



Outline

- Timetable
- Course Contents
- Learning Outcomes
- Assessment Methods

Timetable

- Time for our classes

- 15 weeks from Jan 20, 2025 to May 03, 2025

- Time: (19:00 ~ 21:30) (Thursday)

- Classroom of our class

- Lectures & Labs: KK202

Course Contents

- Introduction to Python
 - The fundamentals of Python, Python Libraries: NumPy, SciPy, Pandas, and the Application of Python in system control applications.
- Programming and Artificial Intelligence
 - Introduction to Artificial Intelligence, Machine Learning and Expert Systems
- Advanced Technologies in Product Development
 - Technology and New Product Development, Complaint Mechanism Engineering, 3D Printing, Virtual Reality, Augmented Reality, Internet of Things

Outline

Week	Lecture
Week 1 – Week 2	Python Fundamentals
Week 3 – Week 5	Application of Python in Control System
Week 6 – Week 7	Programming and Artificial Intelligence
Week 8	Reading Week
Week 9	Midterm Exam
Week 10– Week 12	Programming for Expert System
Week 13– Week 15	Advanced Technologies in Product Development

Intended Learning Outcomes

- Course Aims:
 - To provide the fundamental computer programming for advanced product design and development
 - Apply programming techniques to achieve modern design elements, e.g., Artificial Intelligence.
 - Describe the essential concepts of product development.
- Professional Skills (based on Python Programming):
 - Implement the methods via a programming language.

Assessment Methods

- Continuous Assessment (50%)
 - Assignment (10 %)
 - Midterm (15 %)
 - Group Project (25%) (Project Report(10%) + Source Code/Demo (10%) + Presentation(5%))
- Examination (50%)
 - Final examination
- Important Notices
 - Plagiarism: Students who plagiarized and who were plagiarized will be given **zero mark**.

Assessment Methods

- Submission of Assignment

- Normally a penalty will be applied for late submission, and no mark will be awarded after an absolute deadline.

- Missed Midterm

- There will be no ‘make-up’ for a missed midterm exam under normal circumstances. For students who are absent from continuous assessments such as mid-term tests for genuine reasons, the percentage marks obtained in the final examination of that subject will be used as the continuous assessment marks in percent. Genuine reasons include medical conditions with a doctor's certificate and urgent family matters with proof are required.

Recommended books

1. Martelli, A. (2006). *Python in a Nutshell*. " O'Reilly Media, Inc.".
2. Payne, J. (2010). Beginning Python: Using Python 2.6 and Python 3.1. Wrox Press Ltd.
3. Alpaydin, E. (2020). Introduction to machine learning. MIT press.
4. Flinn, P. (2019). Managing technology and product development programmes: A framework for success. John Wiley & Sons.
5. Ulrich, K., Eppinger, S., & Yang M. (2019). : Product Design and Development. McGraw Hill.
6. Jung, T. (2019). Augmented Reality and Virtual Reality. The Power of AR and VR for Business. Springer Nature Switzerland AG.
7. Hanes, D., Salgueiro, G., Grosssetete, P., Barton, R., & Henry, J. (2017). IoT fundamentals: Networking technologies, protocols, and use cases for the internet of things. Cisco Press.

Moodle

The image shows a Moodle course page. At the top, there is a banner with the course title "IDAT7215 Computer programming for product development and applications [Section 2A, 2024]" over a background image of a building and palm trees. Below the banner is a navigation bar with links for Course, Settings, Participants, Grades, Reports, and More. A "Top Block Position" section contains a button to "Add a block". The main content area displays course modules: General (with edit icon), Course Books (hidden from students), Group Projects (hidden from students), Assignment (hidden from students), and Week-01: Introduction and Python Fundamentals (hidden from students). Each module has an edit icon and a more options menu. At the bottom, there are buttons for Bulk actions and Add an activity or resource.

IDAT7215 Computer programming for product development and applications
[Section 2A, 2024]

Course Settings Participants Grades Reports More

Top Block Position

Add a block

General

Course Books Hidden from students

Group Projects Hidden from students

Assignment Hidden from students

Week-01: Introduction and Python Fundamentals Hidden from students

Bulk actions

Add an activity or resource

Any Question?