SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

10.020 Data Driven World

Working with Data

Peng Song, ISTD

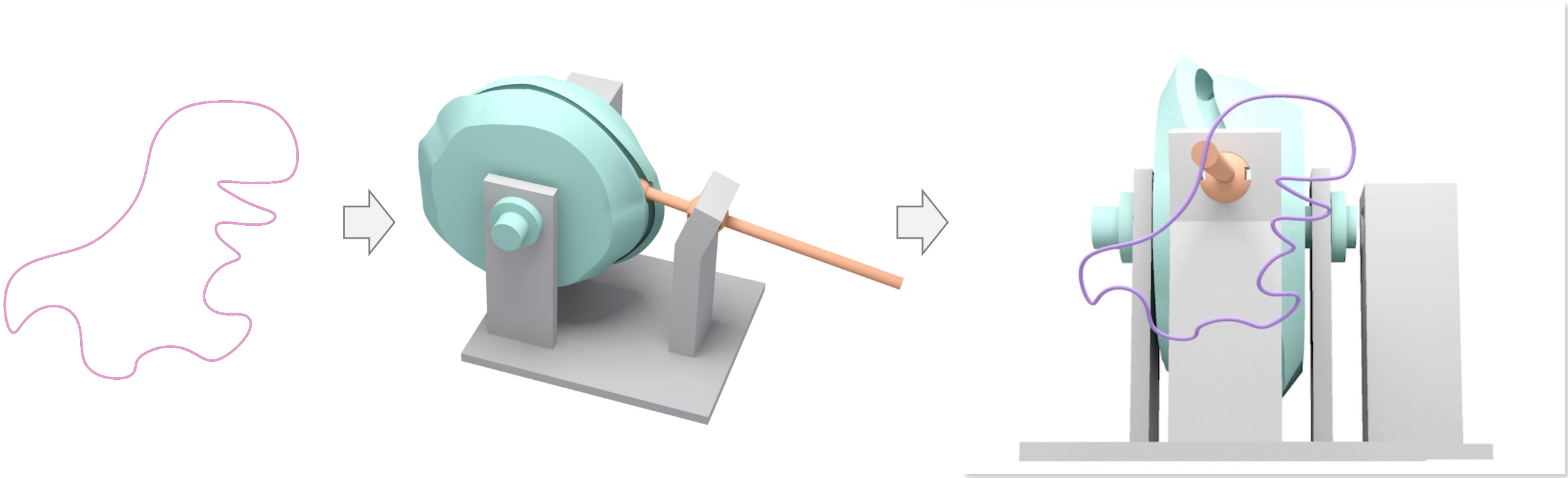
Week 8, Lesson 1, 2021

Peng SONG

- Assistant Professor
- Pillar of Information Systems Technology and Design (ISTD)
- PhD from NTU
- Email: peng_song@sutd.edu.sg
- Office: 1.202.17
- Research group: https://sutd-cgl.github.io/
- Research interest: Computer Graphics
  - Computational Design
  - Geometry Modeling

Computational Design

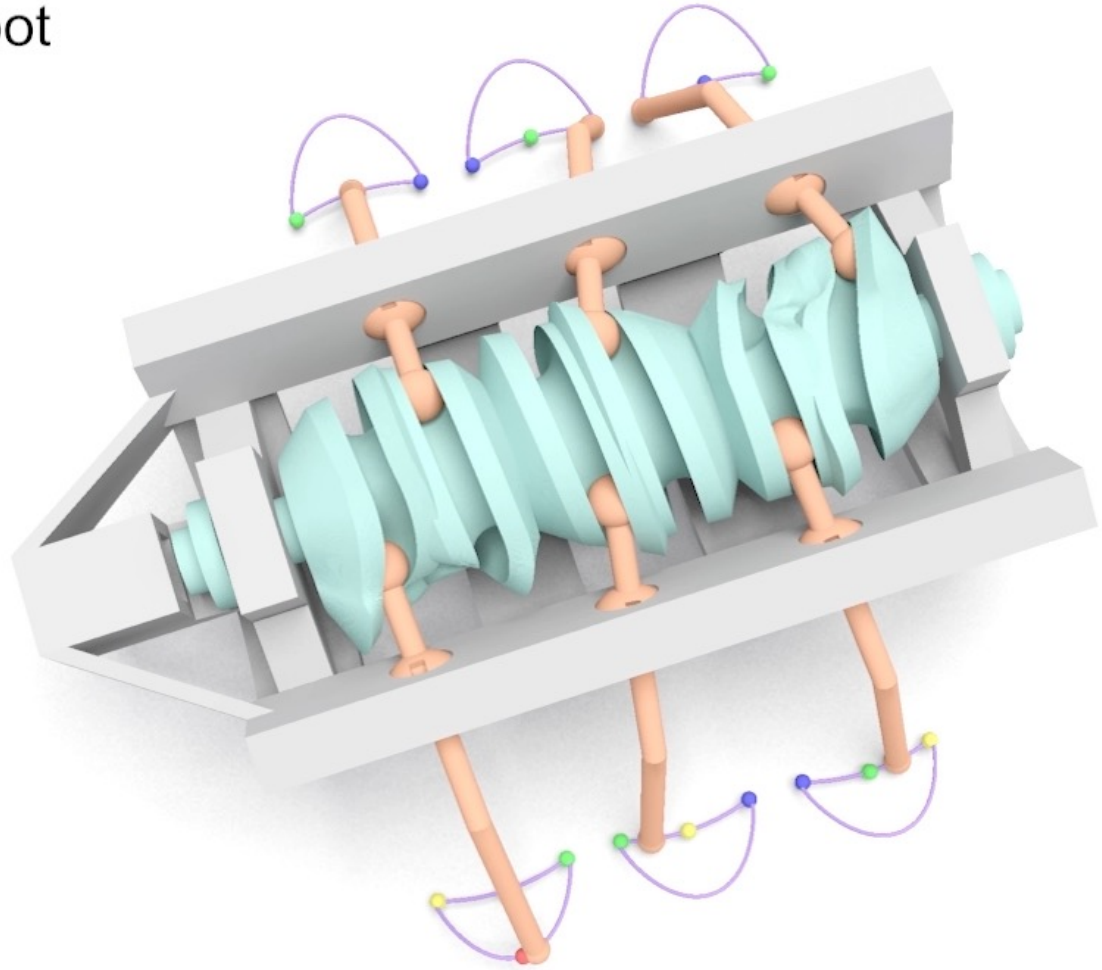
- Develop computational methods, algorithms, and tools to design fabricable and functional real-world objects



Cheng et al. Spatial-Temporal Motion Control via Composite Cam-follower Mechanisms. SIGGRAPH Asia 2021.

Computational Design

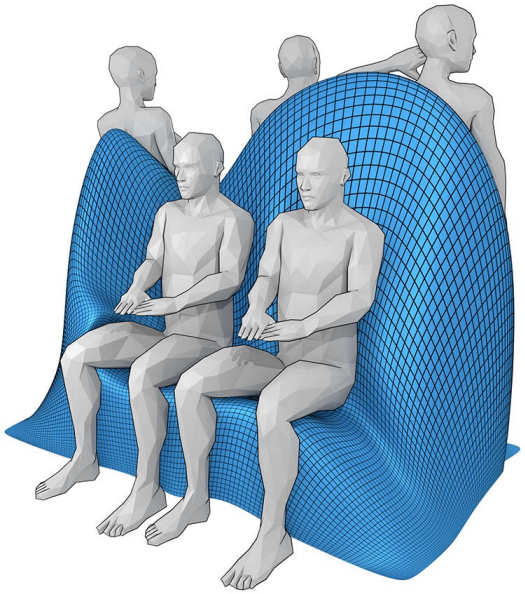
Walking Robot



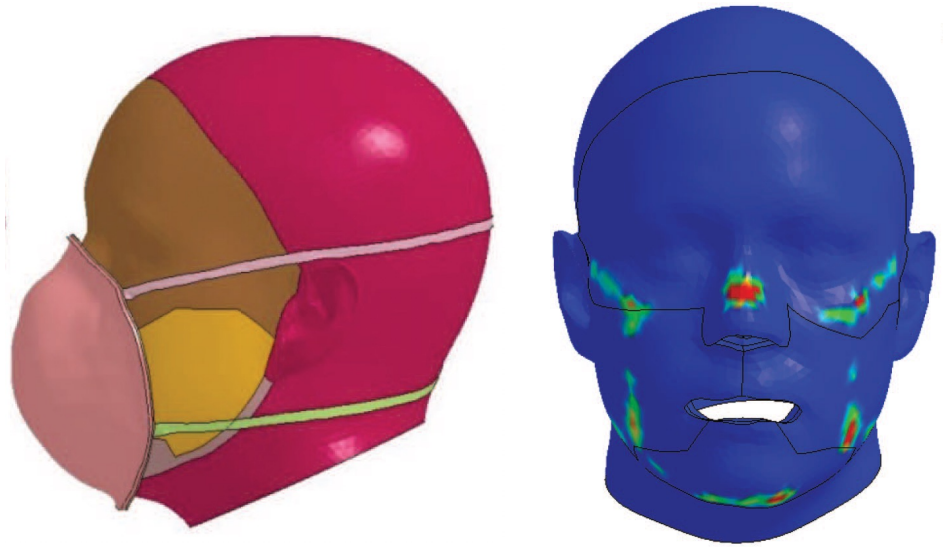
Cheng et al. Spatial-Temporal Motion Control via Composite Cam-follower Mechanisms. SIGGRAPH Asia 2021.

Geometry Modeling

- Model geometry of man-made objects that are comfortable for people to use or to interact with.



[Leimer et al. 2018]



[Lei et al. 2012]

Revision: Weeks 1-6

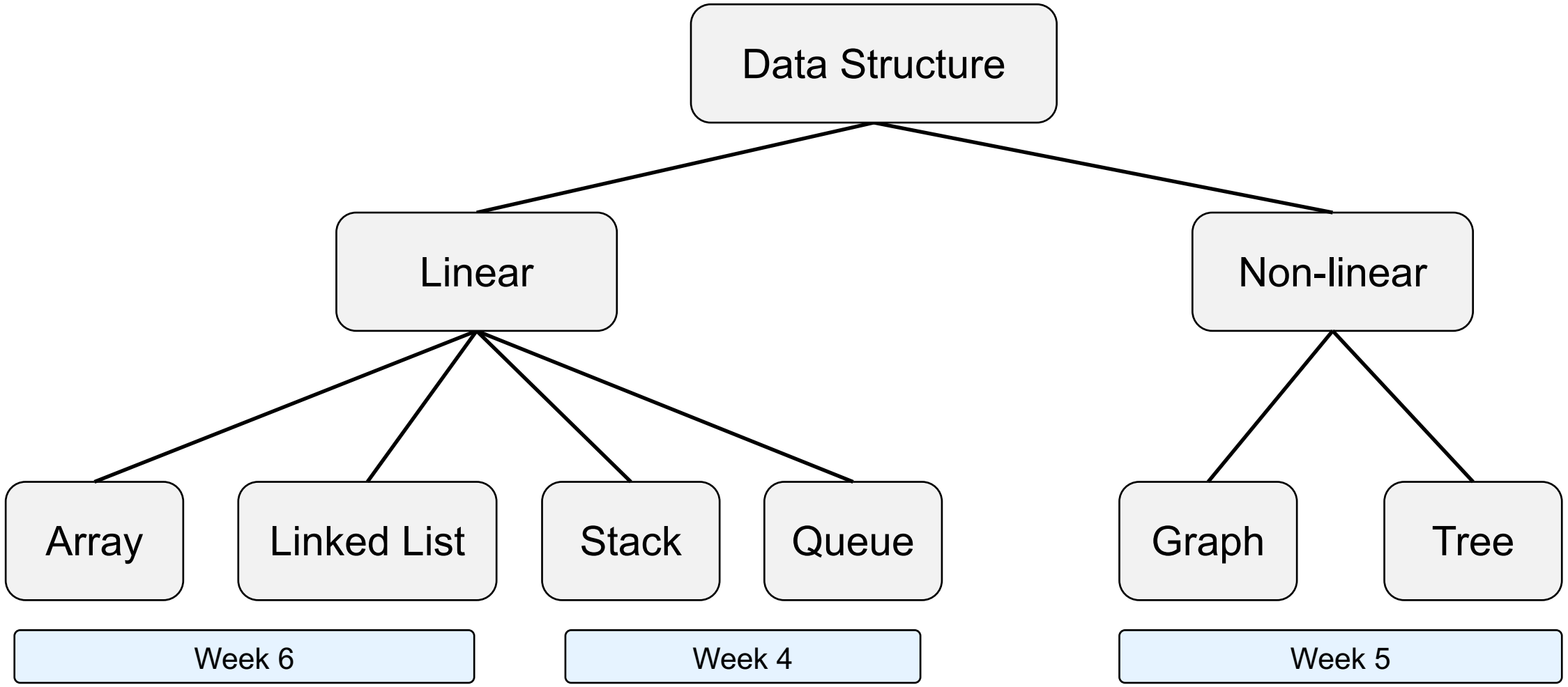
- Sorting algorithms
- User-defined data structures
- Object oriented programming

Revision: Sorting algorithms

	Method	Complexity	Week
Bubble sort	Exchanging: Repeatedly swap the adjacent elements if they are in wrong order	$O(n^2)$	1
Insertion sort	Insertion: Pick elements from the unsorted part and insert them at the correct position in the sorted part.	$O(n^2)$	1
Heapsort	Binary heap: Iteratively extract the largest element from the unsorted part using a binary heap, and place it into the sorted part	$O(n \log n)$	2
Merge sort	Divide and conquer: Divide the input array into two halves, calls itself for the two halves, and then merges the two sorted halves.	$O(n \log n)$	3

Wikipedia

Revision: User-defined Data Structures



Revision: Object Oriented Programming (OOP)

Encapsulation
Hide the unnecessary

Composition
has-a relation
between objects

Class
Data: attributes
Operation: methods

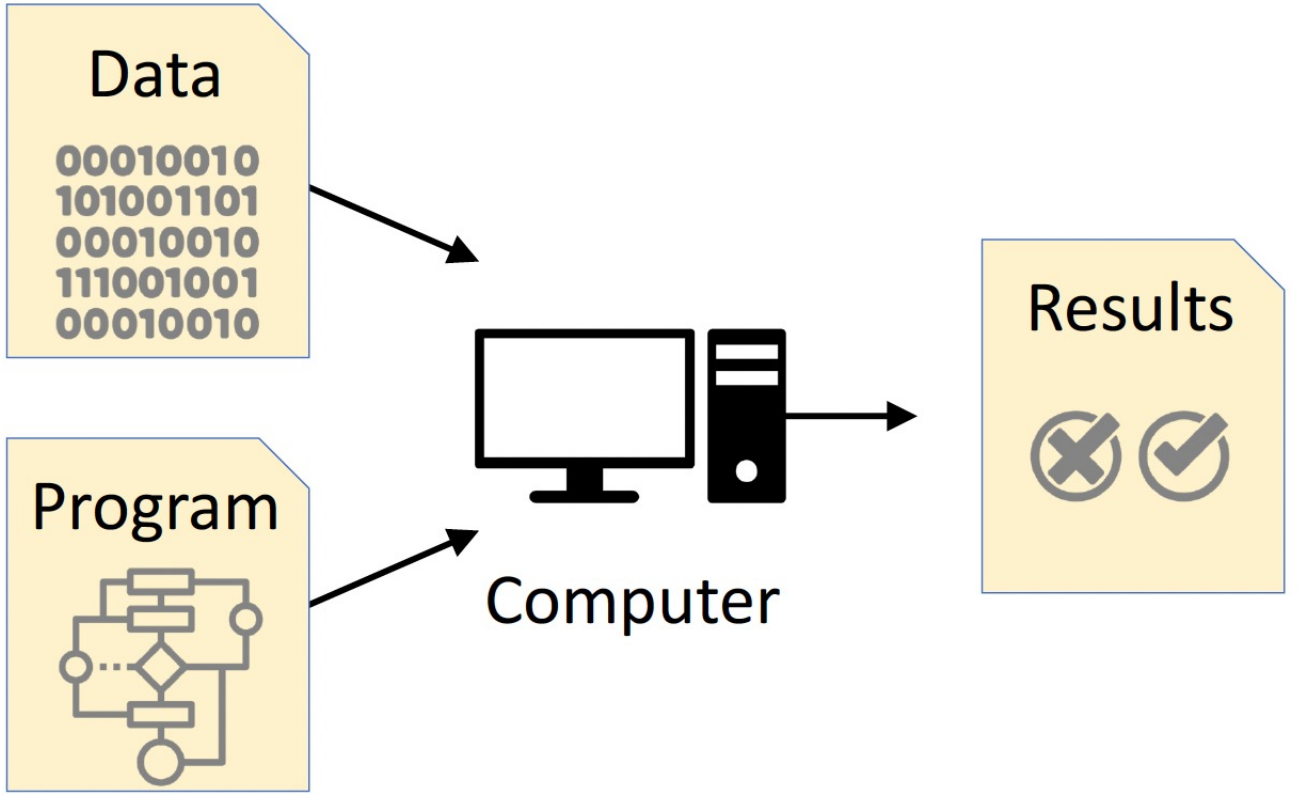
Object
Instance of a class

Abstraction
blueprint for
other classes

Inheritance
is-a relation
between classes

Traditional Programming

[5, 4, 1, 8, 3, 6]

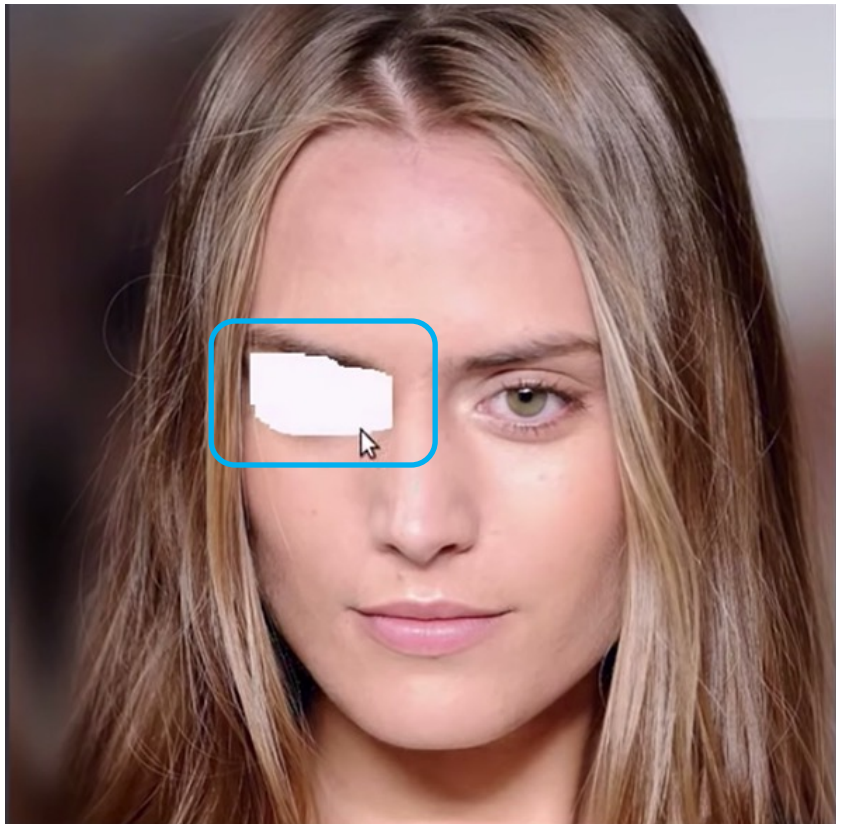


[1, 3, 4, 5, 6, 8]

Bubble sort algorithm

Image Inpainting Problem

Can we solve this problem with traditional programming?



Input: image with missing pixels

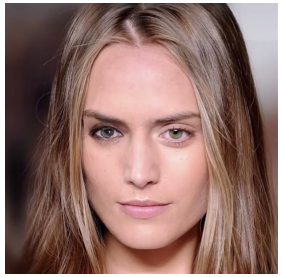
inpainting

Output: a **complete face** image

Machine Learning

Data

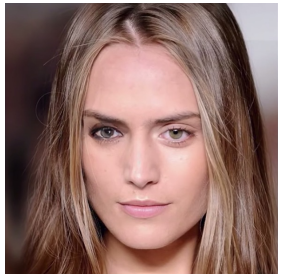
00010010
101001101
00010010
111001001
00010010



Results

Machine Learning

Key Idea: learn from existing data and generalize it



Data

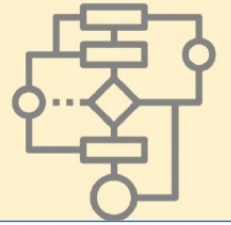
00010010
101001101
00010010
111001001
00010010

Results

Computer

Inpainting algorithm

Program

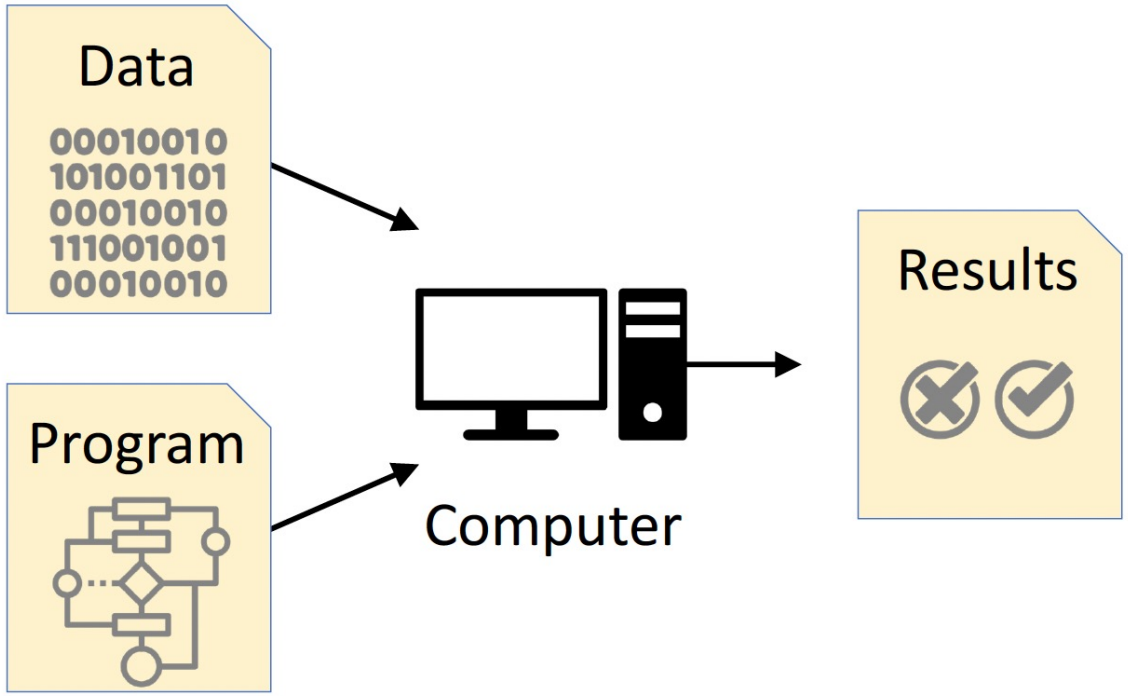


New data

Machine Learning

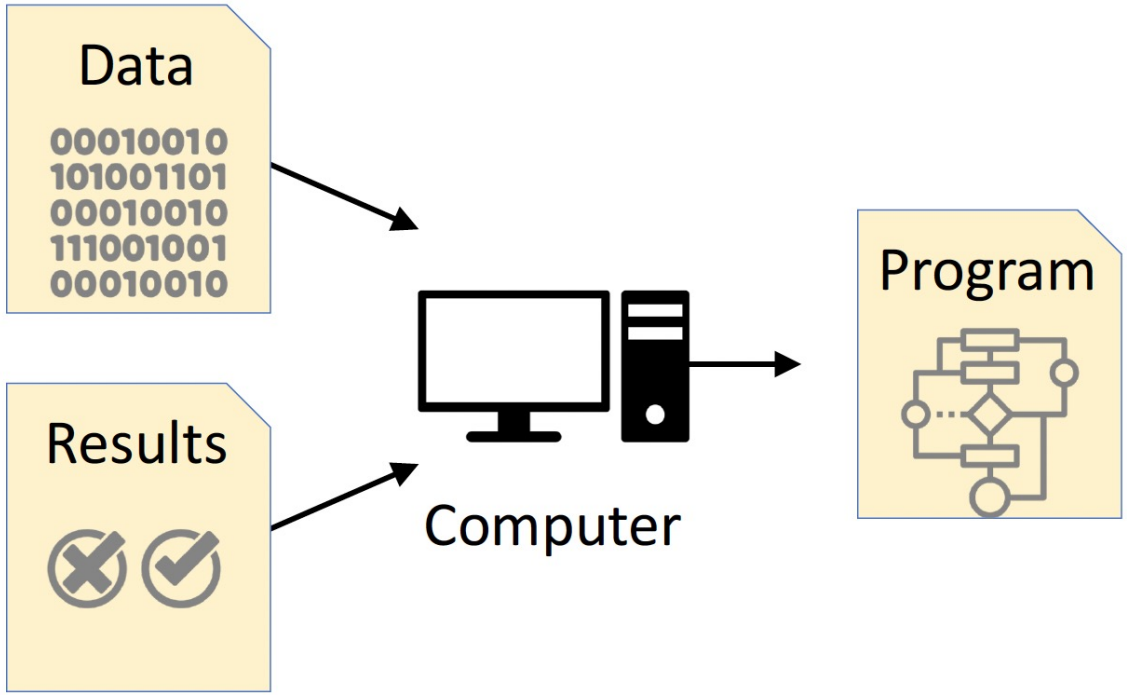
Traditional Programming

Works well when we know how to specify the program



Machine Learning

Needed when we don't know how to specify the program



Source: Eric Eaton

When We Need Machine Learning?

- Tasks for which it is challenging to specify our knowledge
  - Facial recognition
  - Understanding speech
  - Medical diagnosis
- Tasks for which we don’t have human expertise
  - Space exploration
  - Undersea manipulation
  - Cellular robotics
- Tasks requiring customization
  - Image inpainting
  - Email filters
  - Personalized medicine

Defining Machine Learning

"**Machine learning** is the subfield of computer science that gives computers the ability to learn without being explicitly programmed."

- term coined by Arthur Samuel 1959 while at IBM

"**Machine learning** is any process by which a system improves performance from experience."

- Herbert Simon

Types of Machine Learning

- Supervised learning
  - Given: training data + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without labels)
- Reinforcement learning
  - Rewards from sequence of actions

Two relevant courses in SUTD:

51.504 Machine Learning

50.039 Theory and Practice of Deep Learning

Types of Machine Learning

- **Supervised learning**
  - Given: training data + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without labels)
- Reinforcement learning
  - Rewards from sequence of actions

Our focus in 10.020
(very basic knowledge)

Types of Machine Learning

- **Supervised learning**
  - Given: training **data** + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without labels)
- Reinforcement learning
  - Rewards from sequence of actions

What is data?

Data

- **Definition**: data is factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation
- Data can have various forms

Text | **Audio** | **Video** | **Image** | **3D Shape**

Natural Language Processing (NLP)

Computer Vision (CV)

Geometric Deep Learning (GDL)

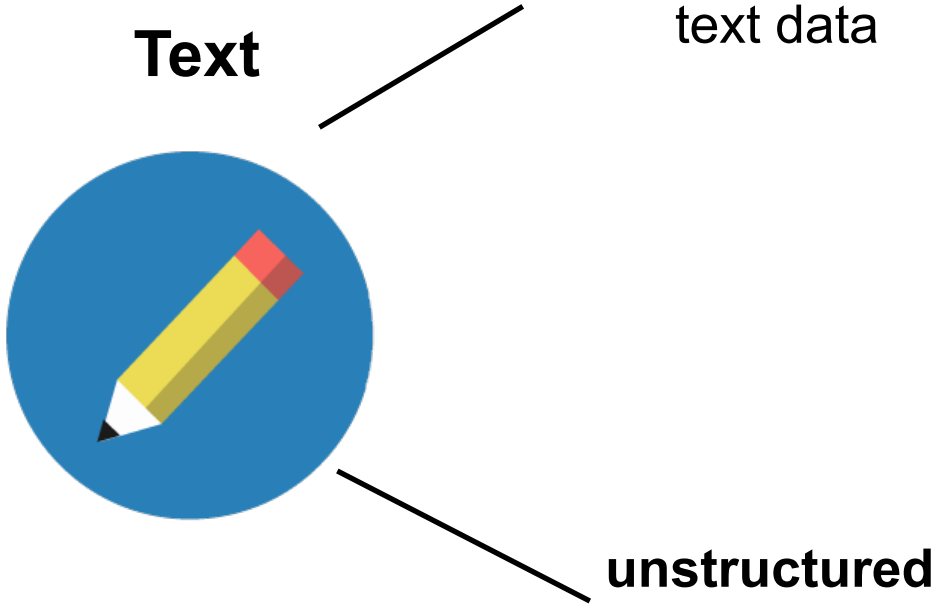
Data

- We consider text data in this course

Text Audio Video Image 3D Shape

Text Data

- We consider <u>structured text data</u> in this course



	A	B	C	D	E	F	G
1	Purchase ID	Last name	First name	Birthday	Country	Date of purchase	Amount of purchase
2	1	Davidson	Michael	04/03/1986	United States	10/12/2016	37
3	2	Vito	Jim	09/01/1994	United Kingdom	02/02/2016	85
4	3	Johnson	Tom	23/08/1972	France	02/11/2016	83
5	4	Lewis	Peter	18/10/1979	Germany	22/11/2016	27
6	5	Koenig	Edward	13/05/1983	Argentina	26/03/2015	43
7	6	Preston	Jack	16/06/1991	United States	06/11/2016	77
8	7	Smith	David	11/03/1965	Canada	15/11/2016	23
9	8	Brown	Luis	03/09/1997	Australia	03/07/2015	74
10	9	Miller	Thomas	07/01/1980	Germany	07/11/2016	13
11	10	Williams	Bill	26/07/1960	United States	20/11/2015	80
12	11	Gemini	Alexia	12/09/1995	Canada	11/03/2017	35
13	12	Bond	James	25/02/1975	United Kingdom	12/08/2017	40
14	13	Burgle	Patricia	01/12/1990	United States	18/01/2015	55
15	14	Reding	Michelle	07/04/1985	Canada	23/02/2017	28
16	15	Harvey	Billy	14/07/1971	United Kingdom	12/01/2016	41
17							

This interesting domestic drama, by Mr. W. S. Gilbert, has continued to engage the sympathies of a nightly sufficient audience at the Haymarket Theatre, where it has now been represented more than sixty times. Its subject and character were described by us, in the ordinary report of theatrical novelties, about two months ago. Our readers will probably not need to be reminded that the hero of the story, Dan'l Druce, the blacksmith, is a solitary recluse dwelling on the coast of Norfolk, where his lone cottage is visited by fugitives from party vengeance during the civil wars of the Commonwealth. His hoard of money is stolen; but a different sort of

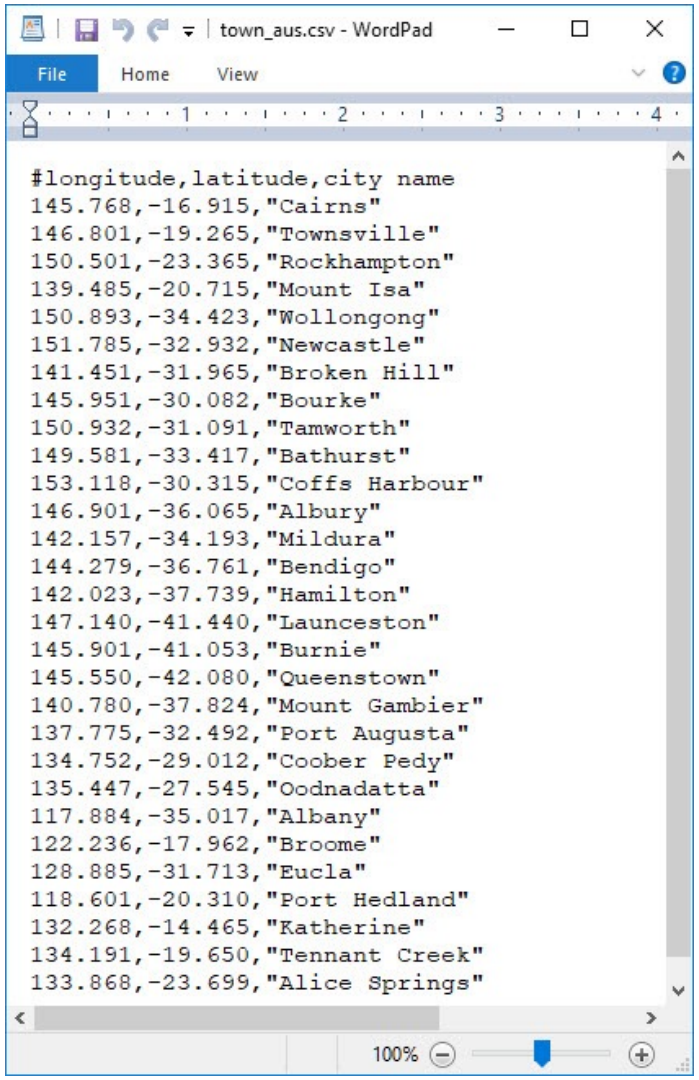
Tabular Text Data

- An arrangement of text data, typically in columns and rows.

	A	B	C	D	E	F	G
1	Purchase ID	Last name	First name	Birthday	Country	Date of purchase	Amount of purchase
2	1	Davidson	Michael	04/03/1986	United States	10/12/2016	37
3	2	Vito	Jim	09/01/1994	United Kingdom	02/02/2016	85
4	3	Johnson	Tom	23/08/1972	France	02/11/2016	83
5	4	Lewis	Peter	18/10/1979	Germany	22/11/2016	27
6	5	Koenig	Edward	13/05/1983	Argentina	26/03/2015	43
7	6	Preston	Jack	16/06/1991	United States	06/11/2016	77
8	7	Smith	David	11/03/1965	Canada	15/11/2016	23
9	8	Brown	Luis	03/09/1997	Australia	03/07/2015	74
10	9	Miller	Thomas	07/01/1980	Germany	07/11/2016	13
11	10	Williams	Bill	26/07/1960	United States	20/11/2015	80
12	11	Gemini	Alexia	12/09/1995	Canada	11/03/2017	35
13	12	Bond	James	25/02/1975	United Kingdom	12/08/2017	40
14	13	Burgle	Patricia	01/12/1990	United States	18/01/2015	55
15	14	Reding	Michelle	07/04/1985	Canada	23/02/2017	28
16	15	Harvey	Billy	14/07/1971	United Kingdom	12/01/2016	41

Tabular Data in CSV format

- A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values.
  - Each line of the file is a data record.
  - Each record consists of one or more fields, separated by commas.



Panadas Library

- **pandas** is a software library written for the Python programming language for data manipulation and analysis.
- **pandas** offers data structures and operations for manipulating numerical tables.

```
import pandas as pd
```

Reading CSV Data

```
file_url = 'https://www.dropbox.com/s/jz8ck0obu9u1rng/resale-flat-prices-based-on-registration-date-from-jan-2017-onwards.csv?raw=1'
df = pd.read_csv(file_url)
df
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
95853	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	650000.0
95854	2021-04	YISHUN	EXECUTIVE	360	YISHUN RING RD	04 TO 06	146.0	Maisonette	1988	66 years 04 months	645000.0
95855	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	585000.0
95856	2021-04	YISHUN	EXECUTIVE	355	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 08 months	675000.0
95857	2021-04	YISHUN	EXECUTIVE	277	YISHUN ST 22	04 TO 06	146.0	Maisonette	1985	63 years 05 months	625000.0

95858 rows × 11 columns

Data Type in Panadas Library

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

```
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
s
```

data

a list of axis labels

```
a 0.469112
b -0.282863
c -1.509059
d -1.135632
e 1.212112
dtype: float64
```

Data Type in Panadas Library

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

```
d = {"b": 1, "a": 0, "c": 2}
pd.Series(d)
```

Python dict

```
a 0.0
b 1.0
c 2.0
dtype: float64
```

Data Type in Panadas Library

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

```
d = list(range(2, 8))
pd.Series(d)
```

Python list

```
0 2
1 3
2 4
3 5
4 6
5 7
dtype: int64
```

Data Type in Panadas Library

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types.

```
d = {"one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
     "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
df = pd.DataFrame(d)
df
```

dict of Series

```
  one two
a 1.0 1.0
b 2.0 2.0
c 3.0 3.0
d NaN 4.0
```

Data Type in Panadas Library

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types.

```
d = {"one": [1.0, 2.0, 3.0, 4.0], "two": [4.0, 3.0, 2.0, 1.0]}
df = pd.DataFrame(d)
df
```

dict of lists

```
   one two
0 1.0 4.0
1 2.0 3.0
2 3.0 2.0
3 4.0 1.0
```

DataFrame Operations

#1 Get DataFrame information

#2 Get a Column or Row as a Series

#3 Get Rows and Columns as DataFrame

#4 Select Data Using Conditions

#5 Transpose Data Frame

#6 Statistical Functions

#7 Vector Operations

#1 Get DataFrame information

```
df.columns
```

Return the column labels of the DataFrame.

```
Index(['month', 'town', 'flat_type', 'block', 'street_name',
'storey_range', 'floor_area_sqm', 'flat_model', 'lease_commence_date',
'remaining_lease', 'resale_price'], dtype='object')
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
95853	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	650000.0
95854	2021-04	YISHUN	EXECUTIVE	360	YISHUN RING RD	04 TO 06	146.0	Maisonette	1988	66 years 04 months	645000.0
95855	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	585000.0
95856	2021-04	YISHUN	EXECUTIVE	355	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 08 months	675000.0
95857	2021-04	YISHUN	EXECUTIVE	277	YISHUN ST 22	04 TO 06	146.0	Maisonette	1985	63 years 05 months	625000.0

95858 rows × 11 columns

#1 Get DataFrame information

```
df.index
```

Return the index (row labels) of the DataFrame.

```
RangeIndex(start=0, stop=95858, step=1)
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
95853	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	650000.0
95854	2021-04	YISHUN	EXECUTIVE	360	YISHUN RING RD	04 TO 06	146.0	Maisonette	1988	66 years 04 months	645000.0
95855	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	585000.0
95856	2021-04	YISHUN	EXECUTIVE	355	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 08 months	675000.0
95857	2021-04	YISHUN	EXECUTIVE	277	YISHUN ST 22	04 TO 06	146.0	Maisonette	1985	63 years 05 months	625000.0

95858 rows × 11 columns

#1 Get DataFrame information

```
df.shape
```

```
(95858, 11)
```

Return a tuple representing the dimensionality of the DataFrame.

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
95853	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	650000.0
95854	2021-04	YISHUN	EXECUTIVE	360	YISHUN RING RD	04 TO 06	146.0	Maisonette	1988	66 years 04 months	645000.0
95855	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	585000.0
95856	2021-04	YISHUN	EXECUTIVE	355	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 08 months	675000.0
95857	2021-04	YISHUN	EXECUTIVE	277	YISHUN ST 22	04 TO 06	146.0	Maisonette	1985	63 years 05 months	625000.0

95858 rows × 11 columns

#2 Get a Column as a Series

- Get a column using column name

```
df[column_name]
```

```
print(df['resale_price’])
print(type(df['resale_price']))
```

```
0 232000.0
1 250000.0
2 262000.0
...
95855 585000.0
95856 675000.0
95857 625000.0
Name: resale_price, Length: 95858, dtype: float64
<class 'pandas.core.series.Series'>
```

#2 Get a Column as a Series

- Get a column using column name

```
df.loc[:, column_name]
```

```
print(df.loc[:, 'resale_price’])
print(type(df.loc[:, 'resale_price']))
```

```
0 232000.0
1 250000.0
2 262000.0
...
95855 585000.0
95856 675000.0
95857 625000.0
Name: resale_price, Length: 95858, dtype: float64
<class 'pandas.core.series.Series'>
```

#2 Get a Row as a Series

- Get a row using row index

```
df.loc[row_index, :]
```

```
print(df.loc[0, :])
print(type(df.loc[0, :]))
```

```
month 2017-01
town ANG MO KIO
flat_type 2 ROOM
block 406
street_name ANG MO KIO AVE 10
storey_range 10 TO 12
floor_area_sqm 44
flat_model Improved
lease_commence_date 1979
remaining_lease 61 years 04 months
resale_price 232000
Name: 0, dtype: object
<class 'pandas.core.series.Series'>
```

The output is a Series

#2 Get a Row as a Series

- Get a row using row index

```
df.loc[row_index, :]
```

```
df_row0 = pd.DataFrame(df.loc[0, :])
df_row0
```

Convert a Series into a DataFrame

	0
month	2017-01
town	ANG MO KIO
flat_type	2 ROOM
block	406
street_name	ANG MO KIO AVE 10
storey_range	10 TO 12
floor_area_sqm	44
flat_model	Improved
lease_commence_date	1979
remaining_lease	61 years 04 months
resale_price	232000

#3 Get Rows and Columns as DataFrame

- Get a subset of rows

```
df.loc[0:10,:]
```

Get the first 11 rows

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...

#3 Get Rows and Columns as DataFrame

- Get a subset of rows and a subset of columns

```
df.loc[0:10,'month':'remaining_lease’]
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months
5	2017-01	ANG MO KIO	3 ROOM	150	ANG MO KIO AVE 5	01 TO 03	68.0	New Generation	1981	63 years
6	2017-01	ANG MO KIO	3 ROOM	447	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1979	61 years 06 months
7	2017-01	ANG MO KIO	3 ROOM	218	ANG MO KIO AVE 1	04 TO 06	67.0	New Generation	1976	58 years 04 months
8	2017-01	ANG MO KIO	3 ROOM	447	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1979	61 years 06 months
9	2017-01	ANG MO KIO	3 ROOM	571	ANG MO KIO AVE 3	01 TO 03	67.0	New Generation	1979	61 years 04 months
10	2017-01	ANG MO KIO	3 ROOM	534	ANG MO KIO AVE 10	01 TO 03	68.0	New Generation	1980	62 years 01 month

#3 Get Rows and Columns as DataFrame

- Get a subset of columns

```
columns = ['town', 'block', 'resale_price']
df.loc[:, columns]
```

	town	block	resale_price
0	ANG MO KIO	406	232000.0
1	ANG MO KIO	108	250000.0
2	ANG MO KIO	602	262000.0
3	ANG MO KIO	465	265000.0
4	ANG MO KIO	601	265000.0
...	...	...	...
95853	YISHUN	326	650000.0
95854	YISHUN	360	645000.0
95855	YISHUN	326	585000.0
95856	YISHUN	355	675000.0
95857	YISHUN	277	625000.0

95858 rows × 3 columns

#3 Get Rows and Columns as DataFrame

- Get a subset of columns

```
columns = ['town', 'block', 'resale_price']
df[columns]
```

	town	block	resale_price
0	ANG MO KIO	406	232000.0
1	ANG MO KIO	108	250000.0
2	ANG MO KIO	602	262000.0
3	ANG MO KIO	465	265000.0
4	ANG MO KIO	601	265000.0
...	...	...	...
95853	YISHUN	326	650000.0
95854	YISHUN	360	645000.0
95855	YISHUN	326	585000.0
95856	YISHUN	355	675000.0
95857	YISHUN	277	625000.0

95858 rows × 3 columns

#3 Get Rows and Columns as DataFrame

- Get a subset of rows and a subset of columns

```
df.loc[0:10, columns]
```

	town	block	resale_price
0	ANG MO KIO	406	232000.0
1	ANG MO KIO	108	250000.0
2	ANG MO KIO	602	262000.0
3	ANG MO KIO	465	265000.0
4	ANG MO KIO	601	265000.0
5	ANG MO KIO	150	275000.0
6	ANG MO KIO	447	280000.0
7	ANG MO KIO	218	285000.0
8	ANG MO KIO	447	285000.0
9	ANG MO KIO	571	285000.0
10	ANG MO KIO	534	288500.0

#3 Get Rows and Columns as DataFrame

- Get rows and columns using indices

```
columns = [1, 3, -1]
df.iloc[0:10, columns]
```

Index starts from 0

	town	block	resale_price
0	ANG MO KIO	406	232000.0
1	ANG MO KIO	108	250000.0
2	ANG MO KIO	602	262000.0
3	ANG MO KIO	465	265000.0
4	ANG MO KIO	601	265000.0
5	ANG MO KIO	150	275000.0
6	ANG MO KIO	447	280000.0
7	ANG MO KIO	218	285000.0
8	ANG MO KIO	447	285000.0
9	ANG MO KIO	571	285000.0
10	ANG MO KIO	534	288500.0

#4 Select Data Using Conditions

- Create a new DataFrame

```
columns = ['town', 'block', 'resale_price’]
df.loc[:, columns]
```

	town	block	resale_price
0	ANG MO KIO	406	232000.0
1	ANG MO KIO	108	250000.0
2	ANG MO KIO	602	262000.0
3	ANG MO KIO	465	265000.0
4	ANG MO KIO	601	265000.0
...	...	...	...
95853	YISHUN	326	650000.0
95854	YISHUN	360	645000.0
95855	YISHUN	326	585000.0
95856	YISHUN	355	675000.0
95857	YISHUN	277	625000.0

95858 rows × 3 columns

#4 Select Data Using Conditions

- Specify conditions on rows

```
df.loc[df['resale_price'] > 500_000, columns]
```

	town	block	resale_price
43	ANG MO KIO	304	518000.0
44	ANG MO KIO	646	518000.0
45	ANG MO KIO	328	560000.0
46	ANG MO KIO	588C	688000.0
47	ANG MO KIO	588D	730000.0
...	...	...	...
95853	YISHUN	326	650000.0
95854	YISHUN	360	645000.0
95855	YISHUN	326	585000.0
95856	YISHUN	355	675000.0
95857	YISHUN	277	625000.0

27233 rows × 3 columns

#4 Select Data Using Conditions

- Specify conditions on rows

```
df.loc[(df['resale_price'] >= 500_000) & (df['resale_price'] <= 600_000),
columns]
```

	town	block	resale_price
43	ANG MO KIO	304	518000.0
44	ANG MO KIO	646	518000.0
45	ANG MO KIO	328	560000.0
49	ANG MO KIO	101	500000.0
110	BEDOK	185	580000.0
...	...	...	...
95849	YISHUN	504C	550000.0
95850	YISHUN	511B	600000.0
95851	YISHUN	504C	590000.0
95852	YISHUN	838	571888.0
95855	YISHUN	326	585000.0

13478 rows × 3 columns

#4 Select Data Using Conditions

- Specify conditions on rows

```
df.loc[(df['resale_price'] >= 500_000) & (df['resale_price'] <= 600_000) &
(df['town'] == 'ANG MO KIO'), columns]
```

	town	block	resale_price
43	ANG MO KIO	304	518000.0
44	ANG MO KIO	646	518000.0
45	ANG MO KIO	328	560000.0
49	ANG MO KIO	101	500000.0
1219	ANG MO KIO	351	530000.0
...	...	...	...
94741	ANG MO KIO	545	590000.0
94742	ANG MO KIO	545	600000.0
94743	ANG MO KIO	551	520000.0
94746	ANG MO KIO	642	545000.0
94749	ANG MO KIO	353	588000.0

329 rows × 3 columns

#5 Transpose Data Frame

- Before transpose

```
df_row0
```

	0
month	2017-01
town	ANG MO KIO
flat_type	2 ROOM
block	406
street_name	ANG MO KIO AVE 10
storey_range	10 TO 12
floor_area_sqm	44
flat_model	Improved
lease_commence_date	1979
remaining_lease	61 years 04 months
resale_price	232000

#5 Transpose Data Frame

- After transpose

```
df_row0_transposed = df_row0.T
```

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44	Improved	1979	61 years 04 months	232000

#6 Statistical Functions

- Get the five point summary using .describe() method.

```
df.describe()
```

Pandas will only try to get the statistics of the columns that contain numeric numbers.

	month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
0	2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
1	2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2	2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
3	2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
4	2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0
...	...	...	...	...	...	...	...	...	...	...	...
95853	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	650000.0
95854	2021-04	YISHUN	EXECUTIVE	360	YISHUN RING RD	04 TO 06	146.0	Maisonette	1988	66 years 04 months	645000.0
95855	2021-04	YISHUN	EXECUTIVE	326	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 04 months	585000.0
95856	2021-04	YISHUN	EXECUTIVE	355	YISHUN RING RD	10 TO 12	146.0	Maisonette	1988	66 years 08 months	675000.0
95857	2021-04	YISHUN	EXECUTIVE	277	YISHUN ST 22	04 TO 06	146.0	Maisonette	1985	63 years 05 months	625000.0

95858 rows × 11 columns

#6 Statistical Functions

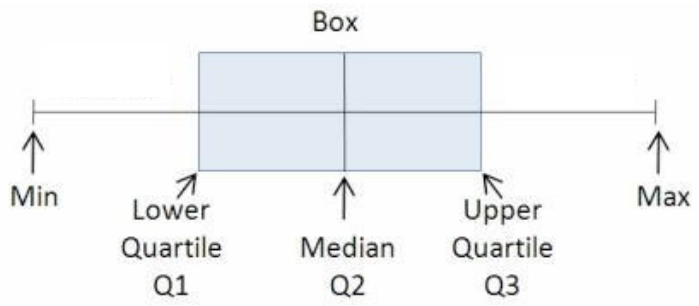
- Get the five point summary using .describe() method.

```
df.describe()
```

Number of non-NA values.

Mean of the values.

Standard deviation of the values.



	floor_area_sqm	lease_commence_date	resale_price
count	95858.000000	95858.000000	9.585800e+04
mean	97.772234	1994.553934	4.467242e+05
std	24.238799	13.128913	1.552974e+05
min	31.000000	1966.000000	1.400000e+05
25%	82.000000	1984.000000	3.350000e+05
50%	95.000000	1995.000000	4.160000e+05
75%	113.000000	2004.000000	5.250000e+05
max	249.000000	2019.000000	1.258000e+06

#6 Statistical Functions

- Get the individual statistical functions

```
print(df['resale_price'].mean())
print(df['resale_price'].std())
print(df['resale_price'].min())
print(df['resale_price'].max())
print(df['resale_price'].quantile(q=0.75))
```

```
446724.22886801313
155297.43748684428
140000.0
1258000.0
525000.0
```

#7 Vector Operations

- Apply some function to all the data in the column or row

```
def divide_by_1000(data):
    return data / 1000
df['resale_price_in1000'] = df['resale_price'].apply(divide_by_1000)
df['resale_price_in1000']
```

```
0         232.0
1         250.0
2         262.0
3         265.0
4         265.0
          ...
95853     650.0
95854     645.0
95855     585.0
95856     675.0
95857     625.0
Name: resale_price_in1000, Length: 95858, dtype: float64
```

#7 Vector Operations

- We can make use of Python's lambda function

```
df['resale_price_in1000'] = df['resale_price'].apply(lambda data:
data/1000)
df['resale_price_in1000']
```

```
0         232.0
1         250.0
2         262.0
3         265.0
4         265.0
          ...
95853     650.0
95854     645.0
95855     585.0
95856     675.0
95857     625.0
Name: resale_price_in1000, Length: 95858, dtype: float64
```

#7 Vector Operations

- Create new data

```
df['pricey'] = df['resale_price_in1000'].apply(lambda price: 1 if price > 500 else 0 )
df[['resale_price_in1000', 'pricey']]
```

	resale_price_in1000	pricey
0	232.0	0
1	250.0	0
2	262.0	0
3	265.0	0
4	265.0	0
...	...	...
95853	650.0	1
95854	645.0	1
95855	585.0	1
95856	675.0	1
95857	625.0	1

95858 rows × 2 columns

Cohort Problem CS1

CS1. *Reading Data:* Read CSV file for Boston Housing.

Task 1: Read the data set.

```
pd.read_csv(csv_file_name)
```

Task 2: Display the number of rows and columns.

```
df.shape
```

Task 3: Display the name of all the columns.

```
df.columns
```

Task 4: Create a subset data set

```
df[column_names]
```

Cohort Problem CS2

CS2. Data Frame Operation.

Task 1: All records with weighted distances to five Boston employment centers between 0 to 3.

Task 2: All records with average number of room between 5 to 8.

Task 3: The first 15 records in the table.

Task 4: The last 15 records in the table.

Task 5: All records with even index numbers, i.e. index 0, 2, 4, ...

```
columns = ["RM", "DIS", "INDUS", "MEDV"]
df_1 = df.loc[TODO, columns]
```

Thank You!