# Programming for Data Science

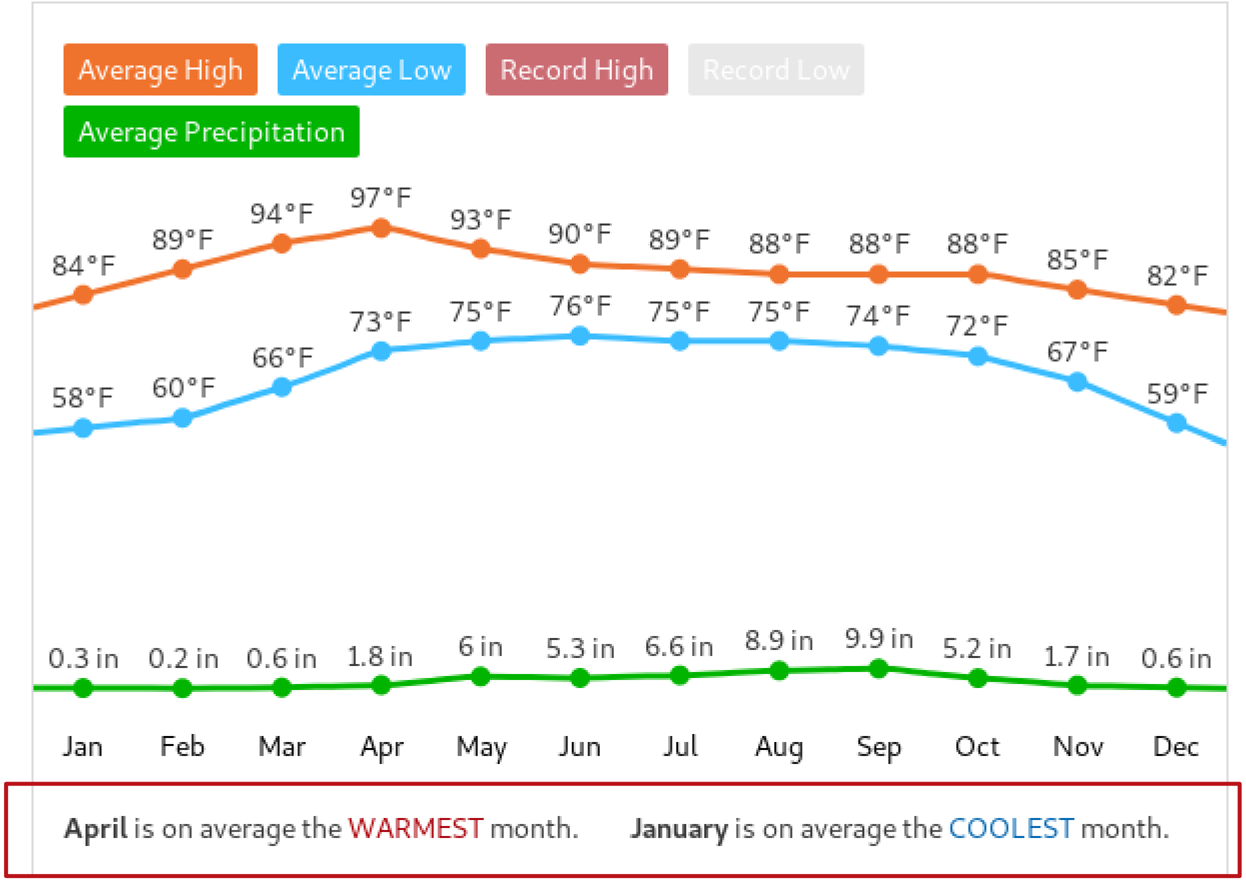
Computational Thinking and friends

# What is Data Science ?

- The scientific process of extracting information or knowledge from data
- The art of presenting the results
- Interdisciplinary field involves
  - Mathematics
  - Statistics
  - Computer science
  - And business, finance, biology, social science ...

# Types of knowledge [1]

- Descriptive statistics

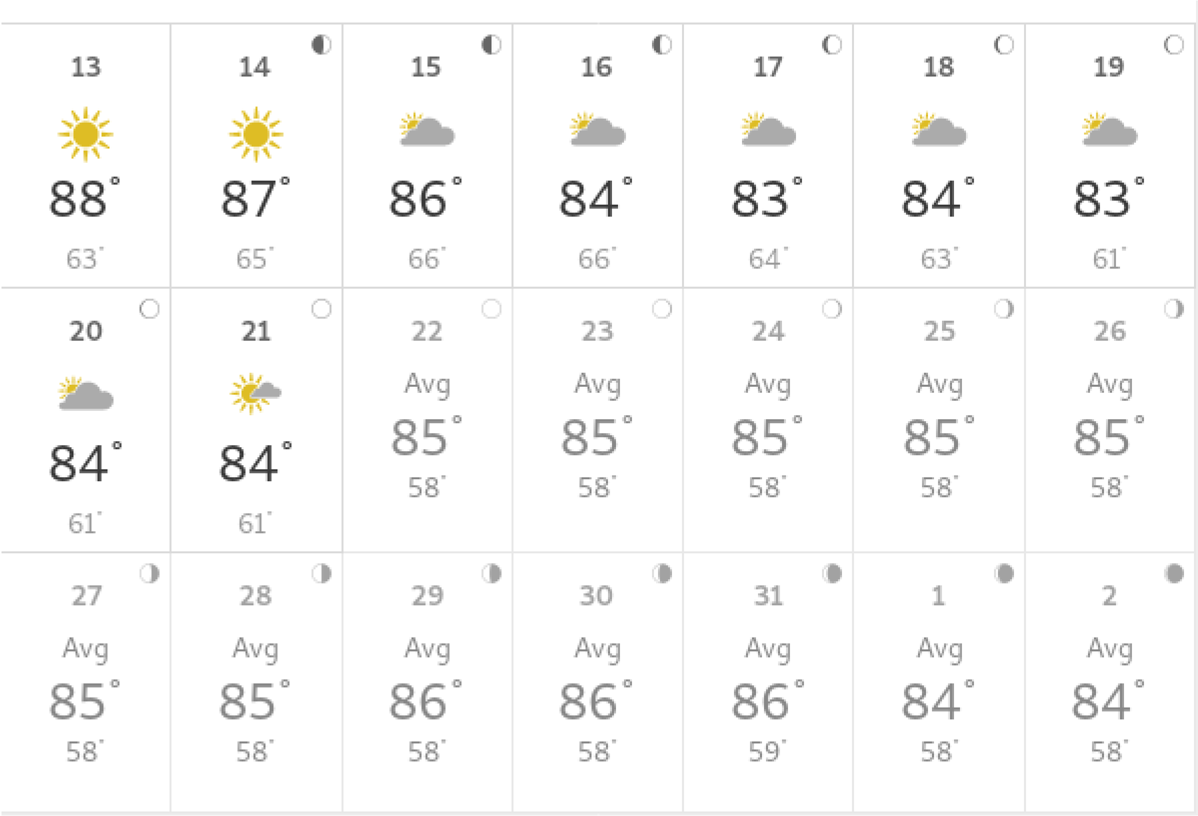


**April** is on average the WARMEST month. **January** is on average the COOLEST month.

Ref: weather.com

# Types of knowledge [2]

- Forecasting model

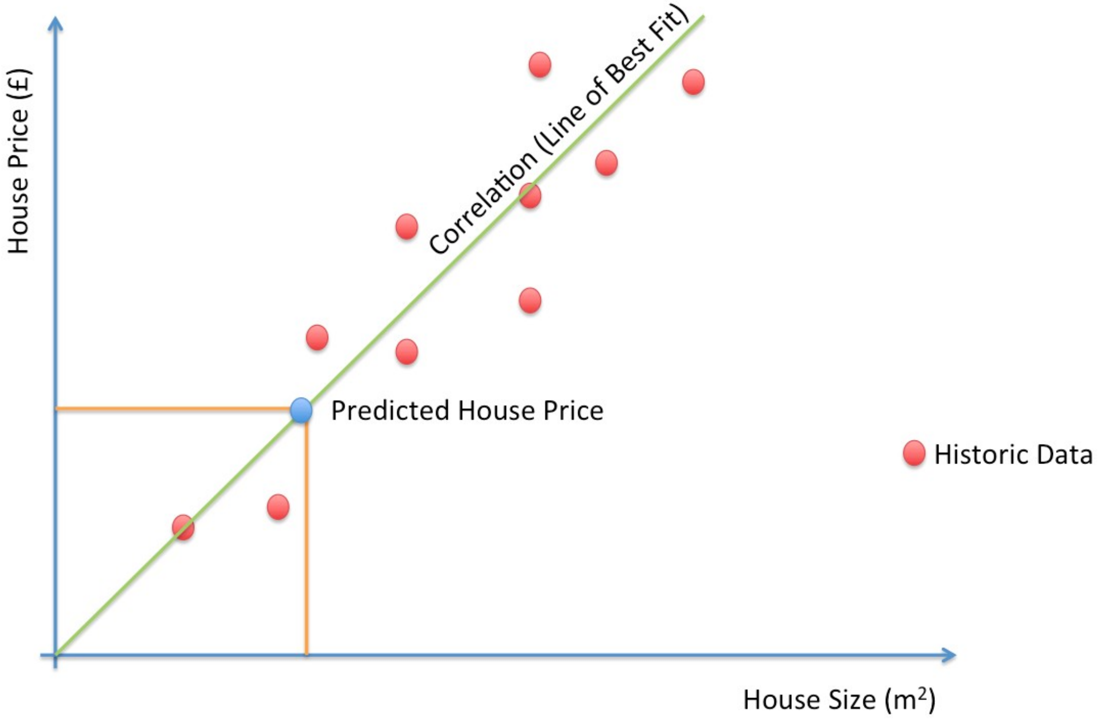


| | | | | | | |
|---|---|---|---|---|---|---|
| 13<br>☀<br>88°<br>63° | 14<br>☀<br>87°<br>65° | 15<br>⛅<br>86°<br>66° | 16<br>⛅<br>84°<br>66° | 17<br>⛅<br>83°<br>64° | 18<br>⛅<br>84°<br>63° | 19<br>⛅<br>83°<br>61° |
| 20<br>⛅<br>84°<br>61° | 21<br>⛅<br>84°<br>61° | 22<br>Avg<br>85°<br>58° | 23<br>Avg<br>85°<br>58° | 24<br>Avg<br>85°<br>58° | 25<br>Avg<br>85°<br>58° | 26<br>Avg<br>85°<br>58° |
| 27<br>Avg<br>85°<br>58° | 28<br>Avg<br>85°<br>58° | 29<br>Avg<br>86°<br>58° | 30<br>Avg<br>86°<br>58° | 31<br>Avg<br>86°<br>59° | 1<br>Avg<br>84°<br>58° | 2<br>Avg<br>84°<br>58° |

Ref: weather.com

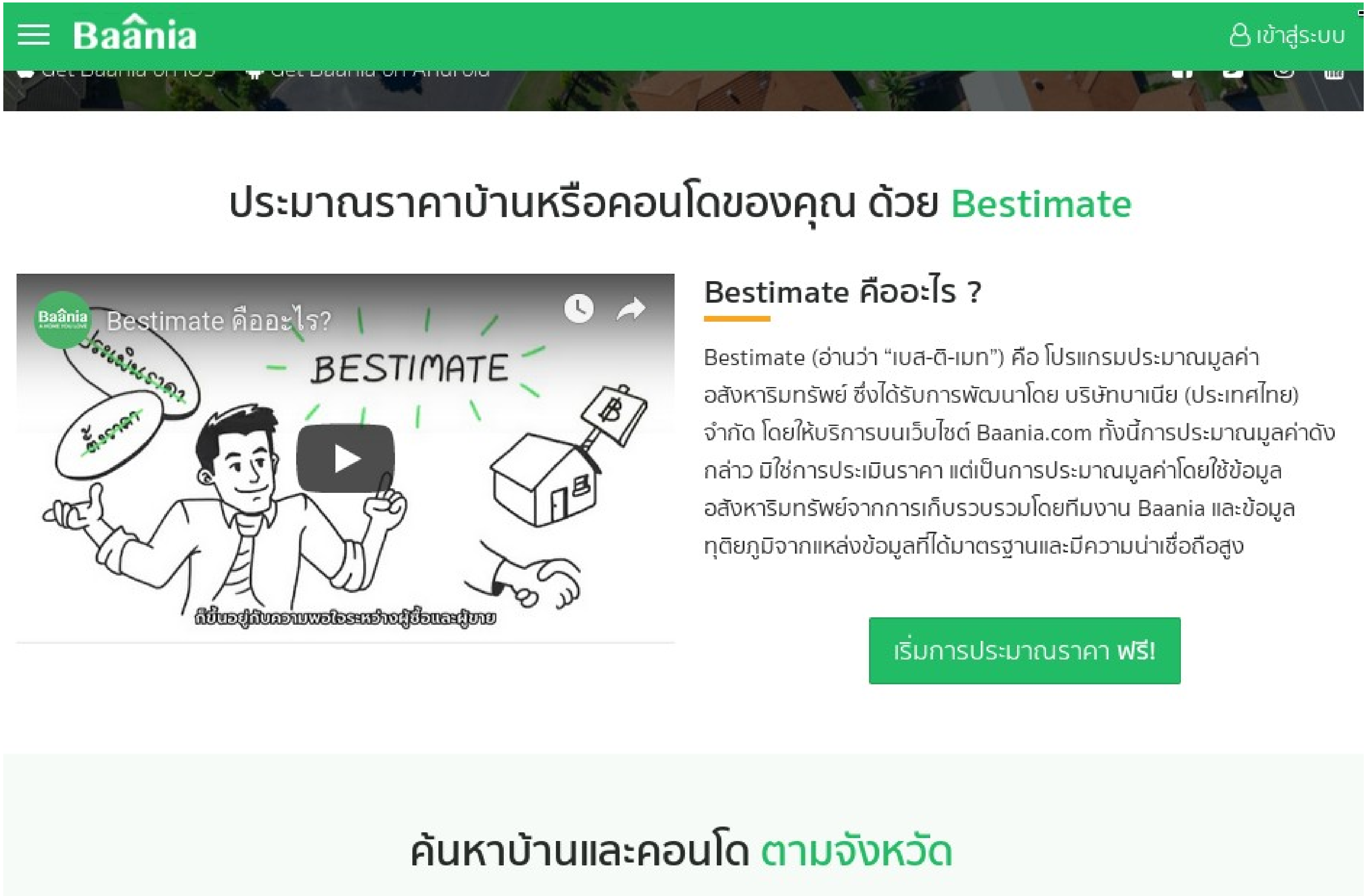
# Types of knowledge [3]

- Regressor



Ref: David Gildeh

# Real world example



## ประมาณราคาบ้านหรือคอนโดของคุณ ด้วย Bestimate

### Bestimate คืออะไร ?

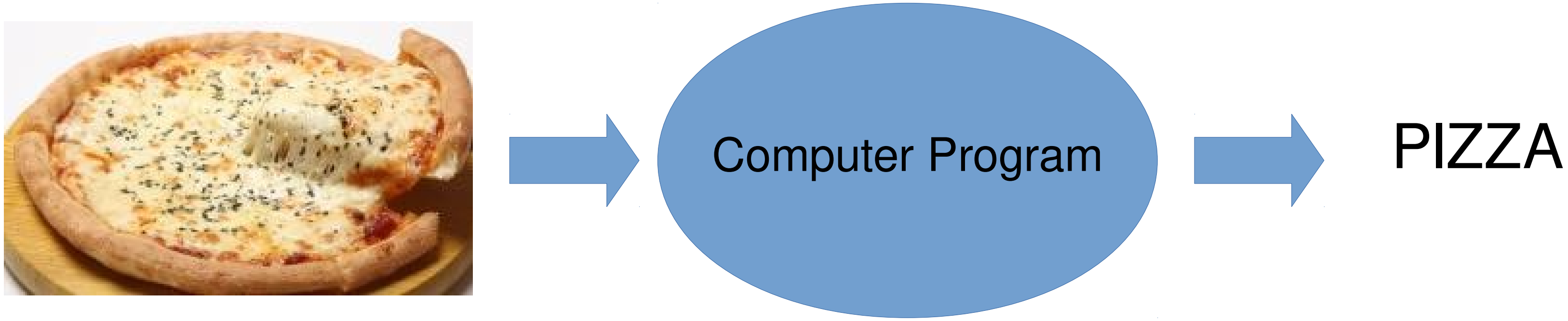
Bestimate (อ่านว่า “เบส-ติ-เมท”) คือ โปรแกรมประมาณมูลค่าอสังหาริมทรัพย์ ซึ่งได้รับการพัฒนาโดย บริษัทบาเนีย (ประเทศไทย) จำกัด โดยให้บริการบนเว็บไซต์ Baania.com ทั้งนี้การประมาณมูลค่าดังกล่าว มิใช่การประเมินราคา แต่เป็นการประมาณมูลค่าโดยใช้ข้อมูลอสังหาริมทรัพย์จากการเก็บรวบรวมโดยทีมงาน Baania และข้อมูลทุติยภูมิจากแหล่งข้อมูลที่ได้มาตรฐานและมีความน่าเชื่อถือสูง

เริ่มการประมาณราคา ฟรี!

## ค้นหาบ้านและคอนโด ตามจังหวัด

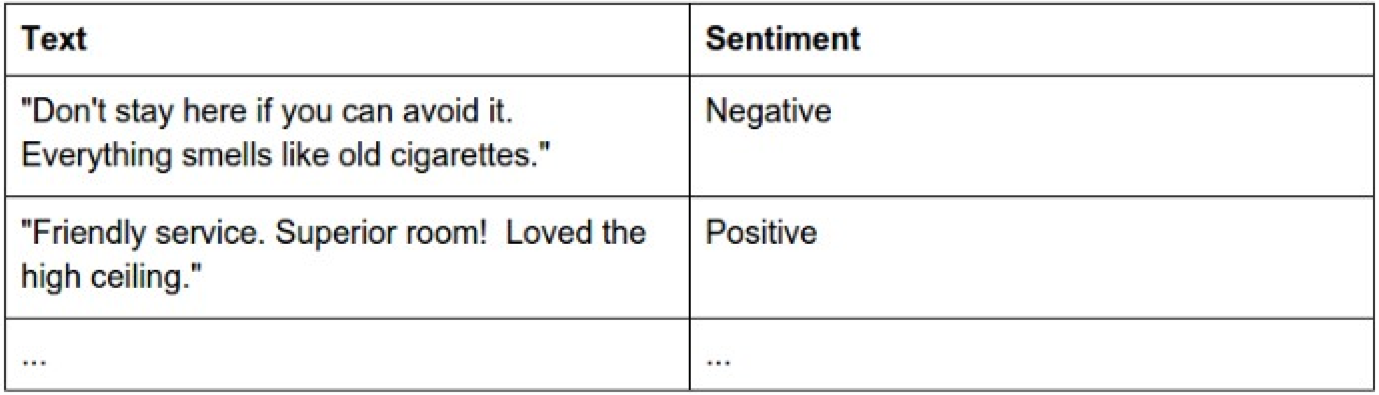
# Types of knowledge [4]

- Classification rule



Computer Program

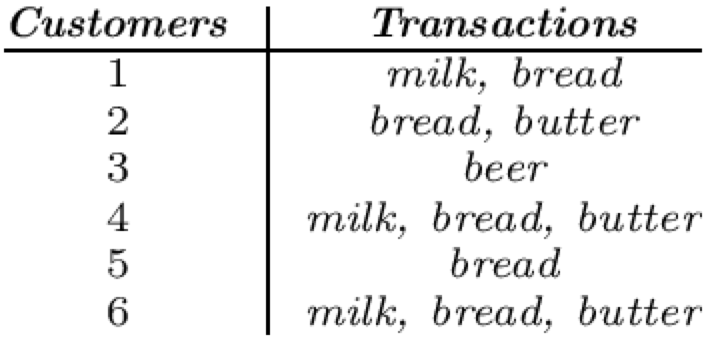
PIZZA



| Text | Sentiment |
|---|---|
| "Don't stay here if you can avoid it. Everything smells like old cigarettes." | Negative |
| "Friendly service. Superior room!  Loved the high ceiling." | Positive |
| ... | ... |

# Types of knowledge [5]

- Association rule
  - Given huge transactional data
  - Find items that are frequently occur together



| *Customers* | *Transactions* |
|---|---|
| 1 | *milk, bread* |
| 2 | *bread, butter* |
| 3 | *beer* |
| 4 | *milk, bread, butter* |
| 5 | *bread* |
| 6 | *milk, bread, butter* |

IF

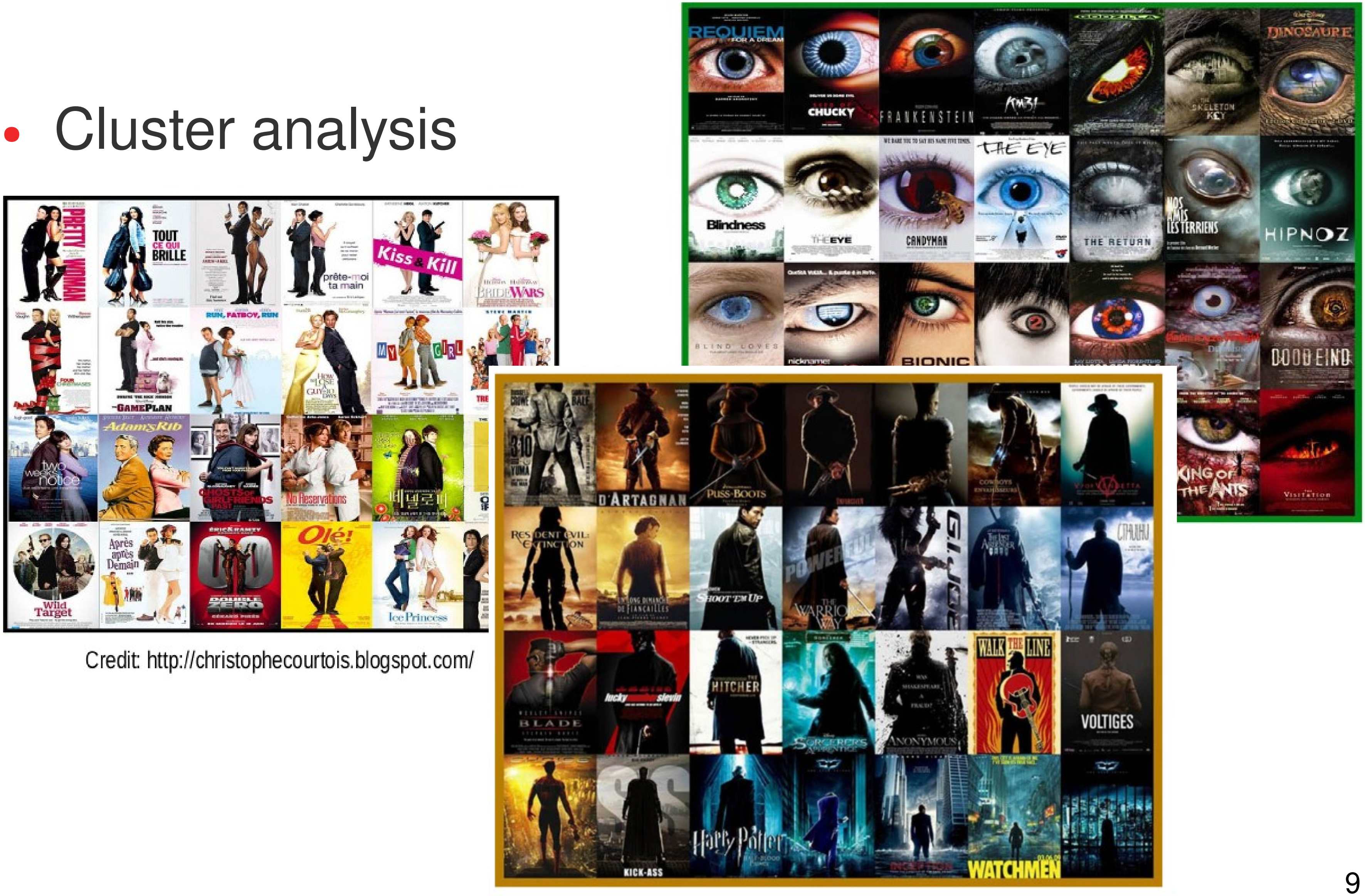
people buy MILK

THEN

they are likely to buy BREAD

# Types of knowledge [6]

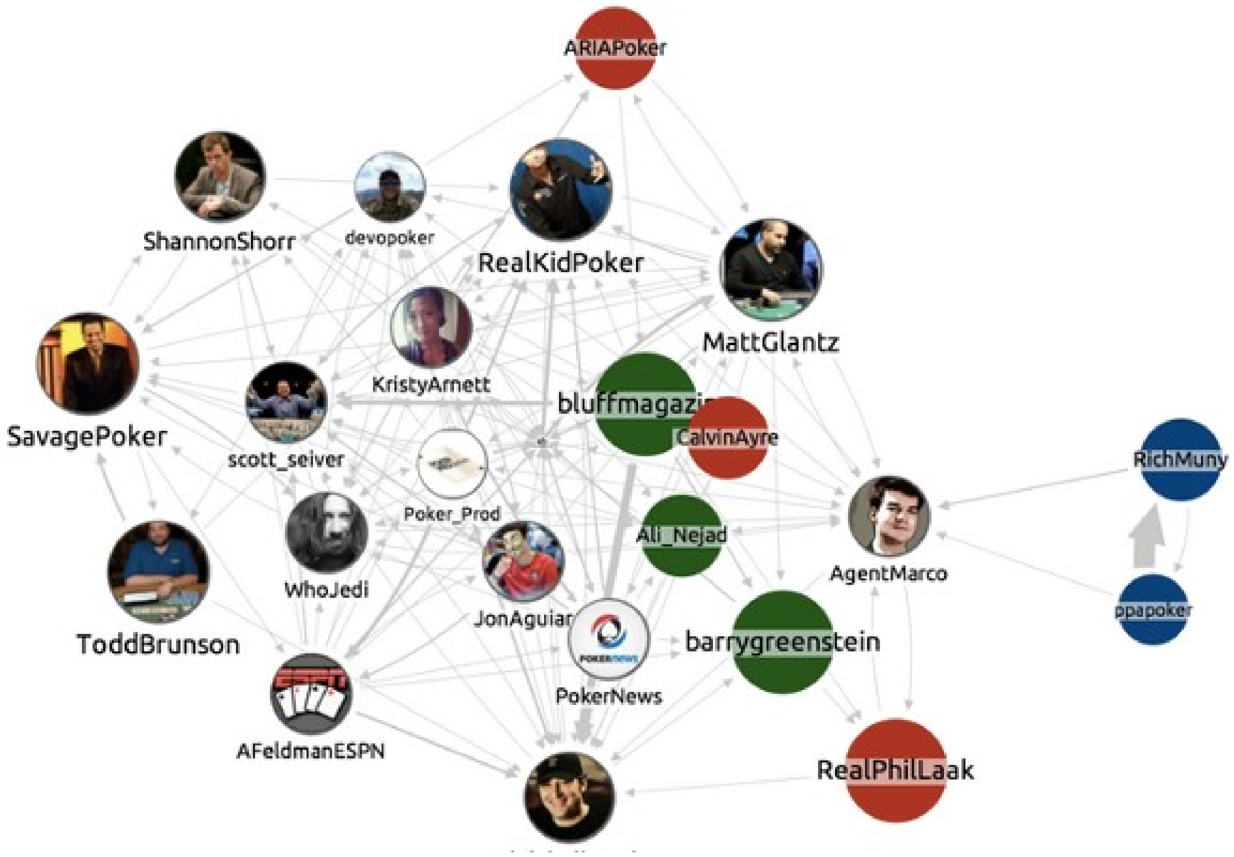
- Cluster analysis



Credit: http://christophecourtois.blogspot.com/

# Types of knowledge [6]

- Structure analysis



# To do data science you need

1) background knowledge of the problem

2) mathematical or statistical skill/ideas to solve the problem at the abstraction level

3) programming skill to implement the ideas (efficiently)

   – what we'll practice in this class –

# Programming for Data Science

- To program is
  - to tell a computer what to do using a programming language
- Programing for data science is no different from programming for XYZ
  - The basics are all the same
- Also true for programming language
  - Learning the second language takes less effort

# To become a capable programmer

- You need
  - A computational mindset
    - Or computational thinking
  - A lot of practices
    - Writing code
    - Reading code

# Computational Thinking

- To formulate real-life problem as computational problem
- To think about possible solutions
- To express solutions in ways that a computer could execute
  - Express clearly and accurately
  - A set of steps to solve something is known as an algorithm

# Computational problems

- Computational problems can be catagorised into
  - Calculation problems
  - Decision problems
  - Search/Sort problems
  - Counting problems
  - Optimisation problems

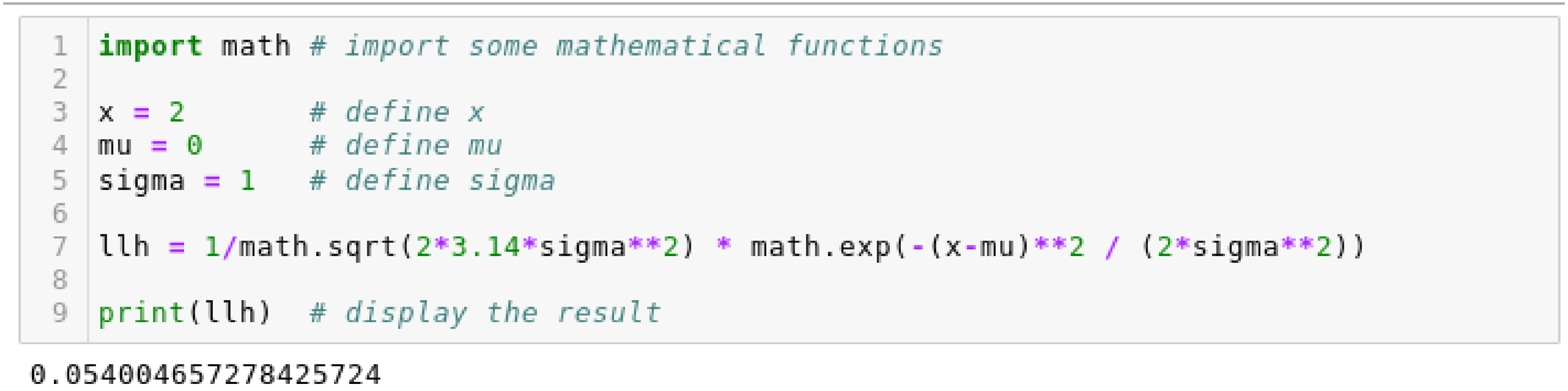
# Calculation problems

- The easiest to program
- Usually involves evaluating a mathematical expression

$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

with $\mu = 0, \sigma = 1$,

# Example solution



```
import math # import some mathematical functions

x = 2        # define x
mu = 0       # define mu
sigma = 1    # define sigma

llh = 1/math.sqrt(2*3.14*sigma**2) * math.exp(-(x-mu)**2 / (2*sigma**2))

print(llh)  # display the result
```

```
0.05400465727842572‌4
```

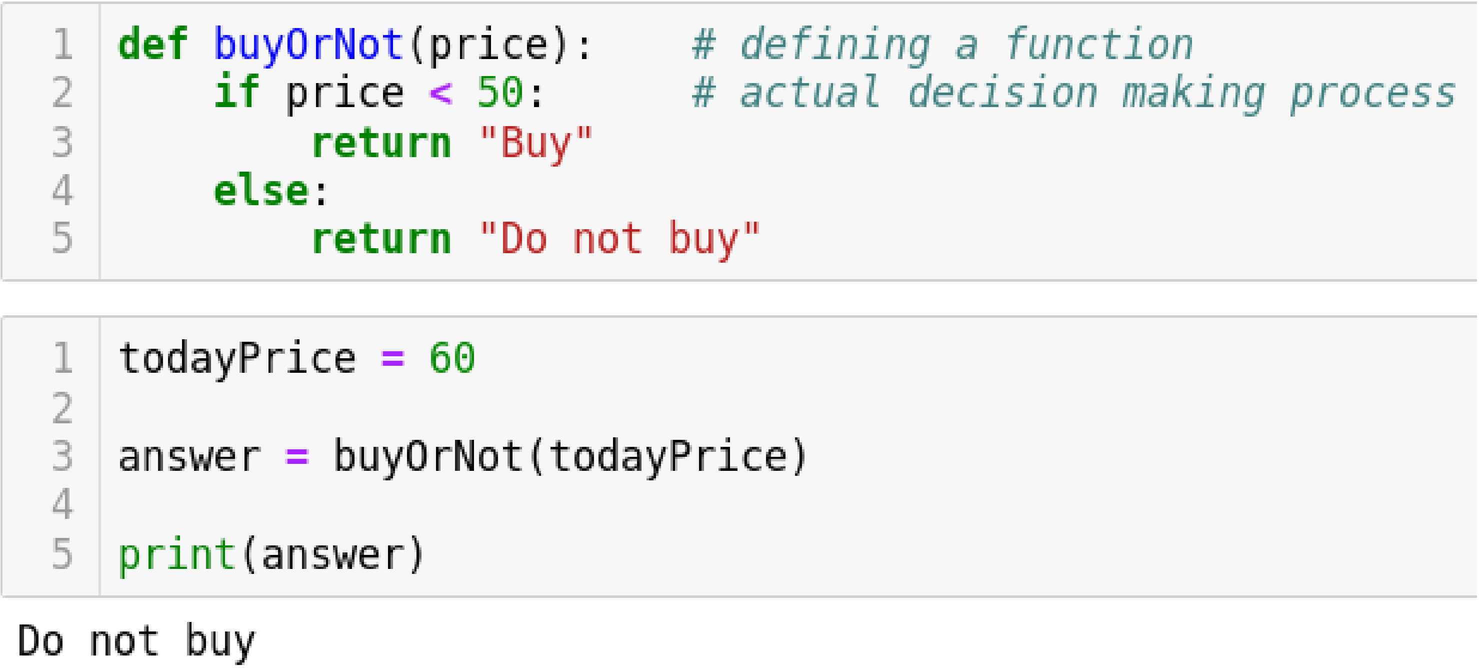
- one needs to be careful when working with mathematical formulas
- Computer cannot represent all values on the real line.
- Techniques for computing maths expression with minimal errors are studied in the field of numerical methods

# Decision problem

- Problem that requires decision making
- Usually the output will either be Yes or No
- May exist on its own or is part of more complicated problem

# Example

- A python function that decides whether to buy a stock market or not



```python
def buyOrNot(price):    # defining a function
    if price < 50:      # actual decision making process
        return "Buy"
    else:
        return "Do not buy"
```

```python
todayPrice = 60

answer = buyOrNot(todayPrice)

print(answer)
```

```
Do not buy
```

# Search/Sort problem

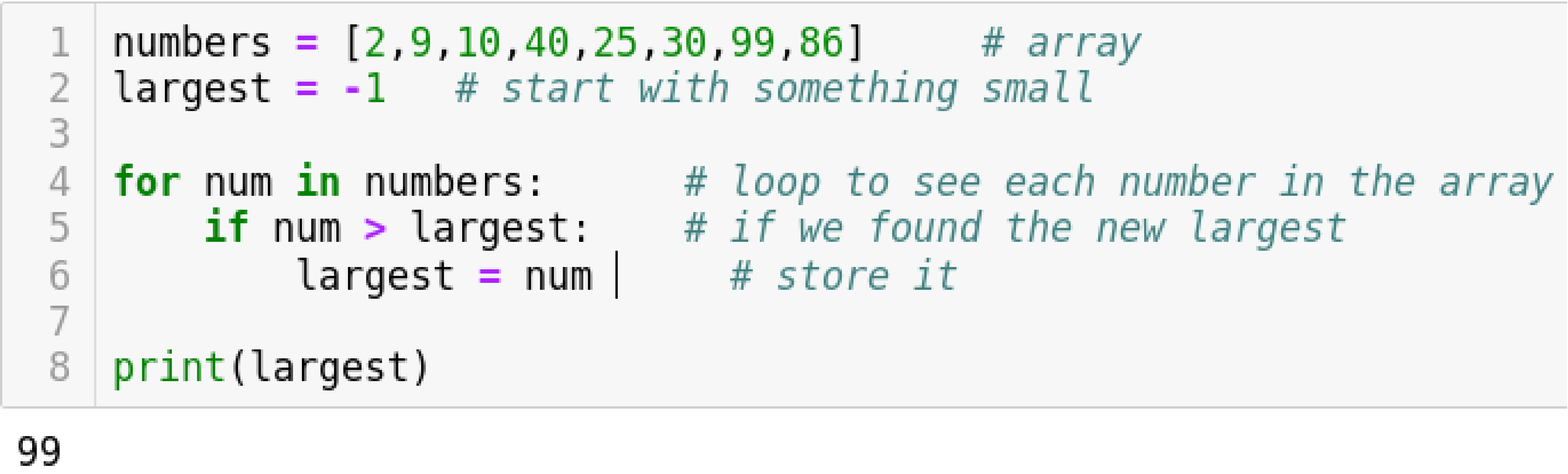
- Searching the problem of finding the solution (often there is only one solution) in a solution space efficiently

- Sorting is the problem of re-arranging the items in our collection of items into some order.

# Example of searching

- What is the largest number in set {2, 9, 10, 40, 25, 30, 99, 86} ?
- How would you solve this ?

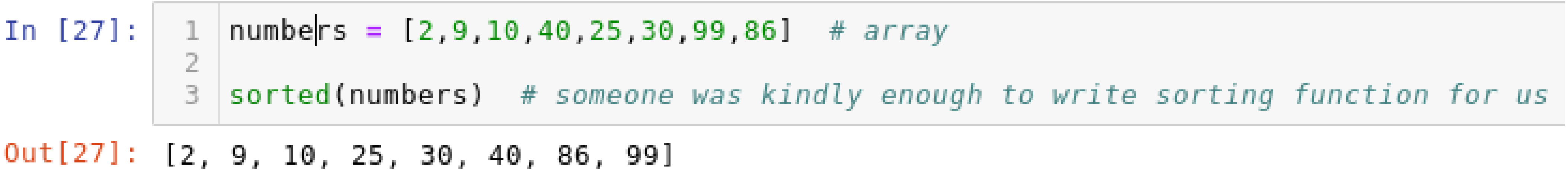
# Possible solution



```
numbers = [2,9,10,40,25,30,99,86]     # array
largest = -1   # start with something small

for num in numbers:      # loop to see each number in the array
    if num > largest:    # if we found the new largest
        largest = num      # store it

print(largest)
```

```
99
```

# Example of sorting



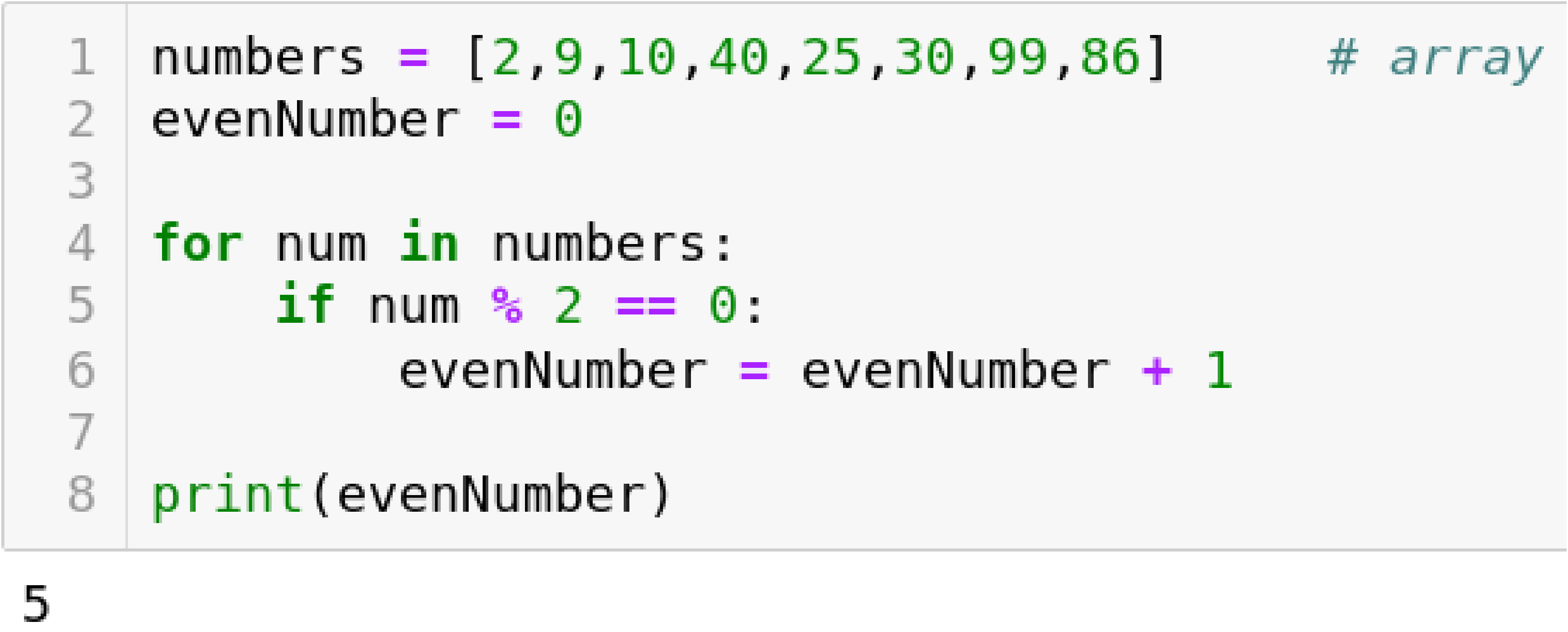
```
In [27]:  numbers = [2,9,10,40,25,30,99,86]   # array

          sorted(numbers)  # someone was kindly enough to write sorting function for us
Out[27]: [2, 9, 10, 25, 30, 40, 86, 99]
```

# Counting problem

- To count the number of solutions in the solution space, efficiently
- For example one might ask you to count how many even number in a set

  {2, 9, 10, 40, 25, 30, 99, 86} ?
- How would you solve this ?
  - "ก็ไล่ไปทีละตัวแล้วดูว่ามันเป็นเลขคู่หรือเปล่า ถ้าใช่ก็จำไว้”

# Possible solution



```python
numbers = [2,9,10,40,25,30,99,86]     # array
evenNumber = 0

for num in numbers:
    if num % 2 == 0:
        evenNumber = evenNumber + 1

print(evenNumber)
```

```
5
```

# Optimisation problem

- Concerns with finding the best possible solution from many possible solutions
- Example
  - Linear programming
  - (stock) portfolio management problem

# Problem solving techniques [1]

- Abstraction
  - solving the problem in a model of the system before applying it to the real system
- Analogy
  - using a solution that solves an analogous problem

# Problem solving techniques [2]

- Divide and Conquer
  - breaking down a large, complex problem into smaller, solvable problems
  - Solve the small problems
  - Combine the solutions to get the solution to the original problem
- Proof
  - try to prove that the problem cannot be solved
  - Or show that the problem is equivalent to solving something easier

# Problem solving techniques [3]

- Research
  - employing existing ideas or adapting existing solutions
- Reduction
  - Transform problem into something we know how to solve
- Trial-and-error (not recommended)
  - Time consuming but worth trying if everything above fails

# Problem solving with programming

- First, you have to understand the problem.
- After understanding, then make a plan.
- Carry out the plan.
- Look back on your work. How could it be better?

# Step 1: Understand the Problem

- Make sure you
  - Fully understand the problem
    - Can describe the problem in your own words
  - Know what are the inputs
  - Know what are the final output

# Step 2: Devise a Plan

- Using aforementioned problem solving techniques
  - Divide-and-conquer, analogy, reduction, etc...
- Immediate concern: to get correct solution
- Future concerns (after more CS courses)
  - Security, efficiency, speed, reliability

# Step 3: Carry out the Plan

- Translate your solution into code
- Simulate test cases
  - A test case is a correct input-output pair which is used to test the correctness of the solution
- Test your solution (program)

# Step 4: Examine and Review

- Review/discuss your solution with others
- Keep the solution in good place for later usage (analogy-typed problem solving)