Short Course in Artificial Intelligence

17th June 2024

Lecture 1: Complexity

Why a course on AI ?

- ***“The A.I. Revolution Will Change Work. Nobody Agrees How.”***
  - *New-York Times, June 2023*
- Recent improvements on Generative AI and LLM (ChatGPT, DALL-E…)
- Some new tools will change the way to do Software Development

Why a course on AI ?

- **ChatGPT?:** *Write a python code to sort a list of 5 numbers without any loop or a predefined function.*

```
def sort_numbers(numbers):
  if numbers[0] > numbers[1]:
    numbers[0], numbers[1] = numbers[1], numbers[0]
    sort_numbers(numbers)
  if numbers[1] > numbers[2]:
    numbers[1], numbers[2] = numbers[2], numbers[1]
    sort_numbers(numbers)
  if numbers[2] > numbers[3]:
    numbers[2], numbers[3] = numbers[3], numbers[2]
    sort_numbers(numbers)
  if numbers[3] > numbers[4]:
    numbers[3], numbers[4] = numbers[4], numbers[3]
    sort_numbers(numbers)
  return numbers

if __name__ == "__main__":
  numbers = [5, 2, 9, 1, 7]
  sorted_numbers = sort_numbers(numbers.copy())
  print(sorted_numbers)
```

Why a course on AI ?

- Many jobs will be impacted by AI
- Software Development will be impacted too
- We don't know how much (yet)
  - This is currently happening
- Things will be changing quickly…
  - New skills such as prompt engineering

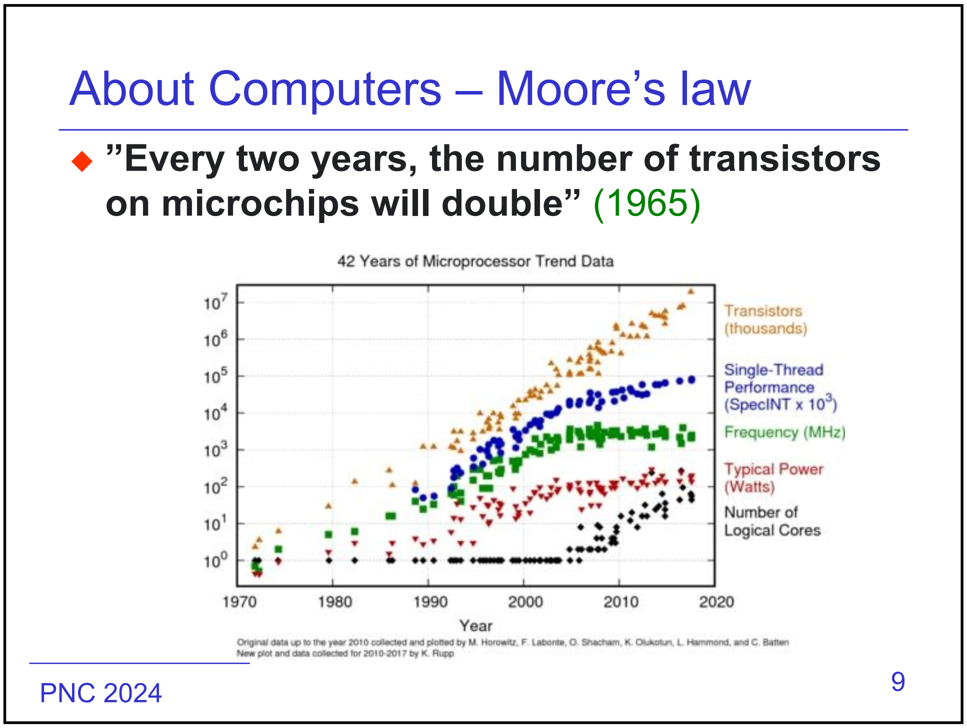
About Computers

- The computer on which I learned programming

- 32 KB RAM, 5 MB hard disk

About Computers

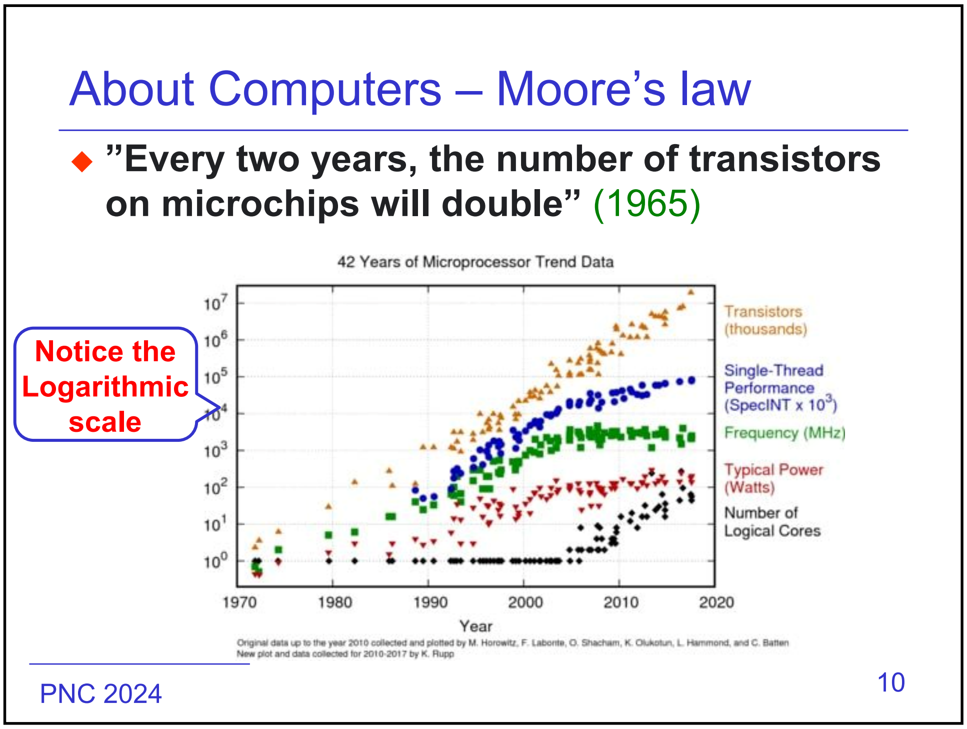
- A Smartphone today:
  - 8 GB Ram
  - 256 GB storage
  - 2340x1080 display
  - 2.5 in x 5.7 in
  - cameras: 3 rear, 1 front
  - phone, wifi, bluetooth
  - GPS….



About Computers – Moore’s law

- ”Every two years, the number of transistors on microchips will double” (1965)

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

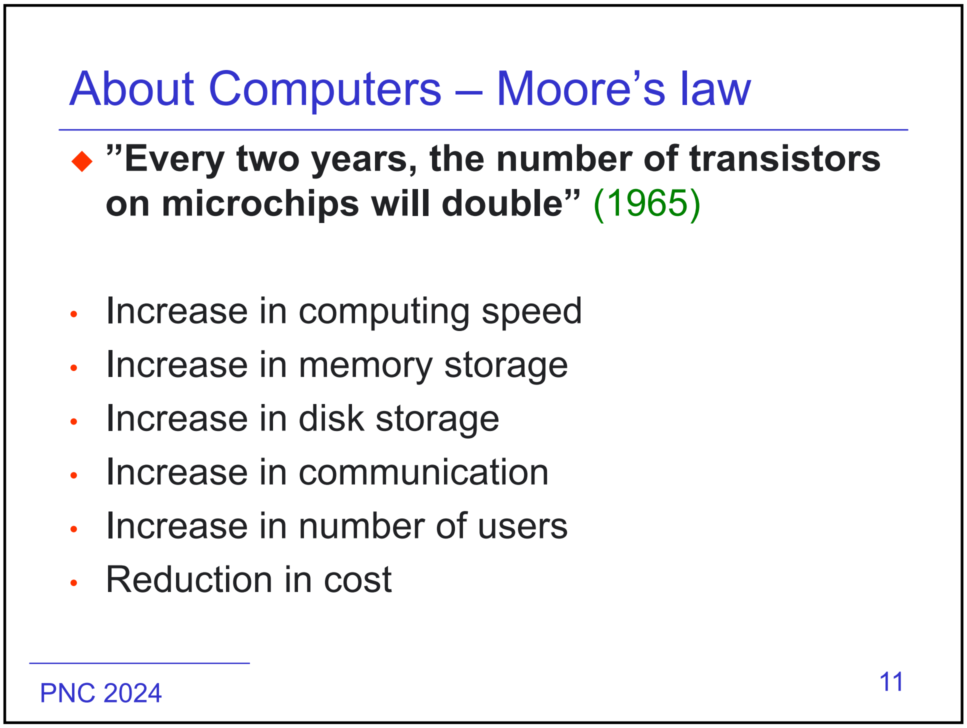


About Computers – Moore’s law

- ”Every two years, the number of transistors on microchips will double” (1965)

Notice the Logarithmic scale

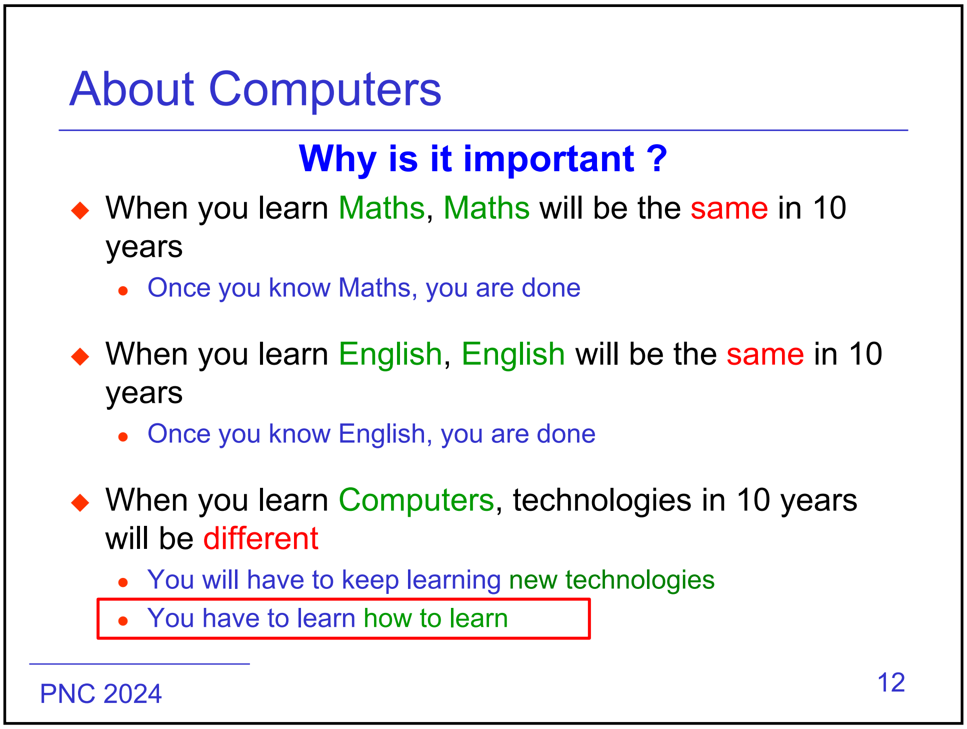
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp



About Computers – Moore's law

- **"Every two years, the number of transistors on microchips will double"** (1965)

- Increase in computing speed
- Increase in memory storage
- Increase in disk storage
- Increase in communication
- Increase in number of users
- Reduction in cost



About Computers

Why is it important ?

- When you learn Maths, Maths will be the same in 10 years
  - Once you know Maths, you are done
- When you learn English, English will be the same in 10 years
  - Once you know English, you are done
- When you learn Computers, technologies in 10 years will be different
  - You will have to keep learning new technologies
  - You have to learn how to learn

Artificial Intelligence

- Can a machine act / think as a person ?
- A very old idea (dream): 1769: Von Kempelen

Artificial Intelligence

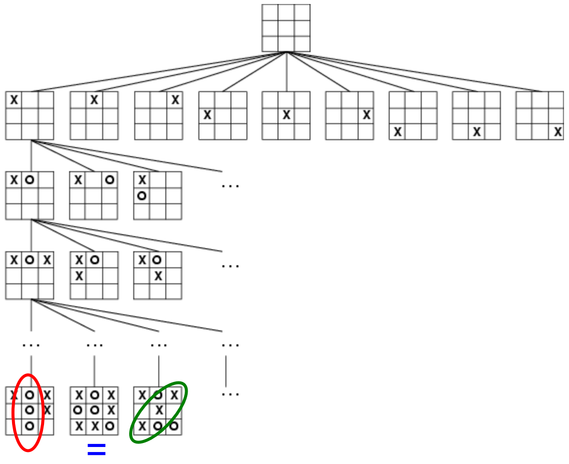
- Can a machine act / think as a person ?
- A very old idea: 1769: Von Kempelen

Artificial Intelligence: the Early Days

- The First Computer: ENIAC 1943
  - Electronic Numerical Integrator and Computer
  - Programmable, first used for ballistic trajectories
- First AI programs: 1950-
- “Solving complex problems”
  - complex = many possible solutions
  - problem = find the best solution
  - too many solutions to check them all → heuristics
- Typical Applications
  - Theorem Proving, Logic
  - Find best path in a tree/graph
  - Games

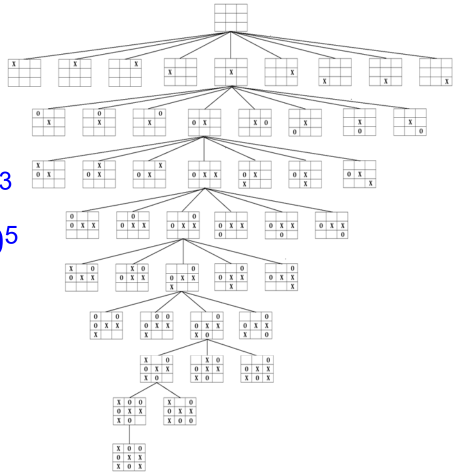
Artificial Intelligence

- Game complexity: Tic-tac-toe



Artificial Intelligence

- Game complexity: Tic-tac-toe (caro)



Board size: 9

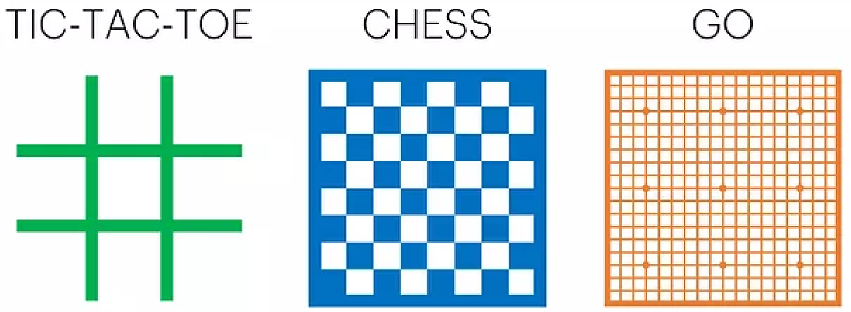
Number of moves: ~ 4

Number of positions: ~ 10^3

Number of games : ~ 10^5

Artificial Intelligence

- Games and intelligence:
  - Games complexity:



3x3 8x8 19x19

Artificial Intelligence

- Games and intelligence:
  - Games complexity:

Game	Board size	Number of moves (average)	Number of positions	Number of games	Average game length
Tic-tac-toe	9	4	10^{3}	10^{5}	9
Chess	64	35	10^{47}	10^{123}	80
Go (19x19)	361	250	10^{170}	10^{360}	150

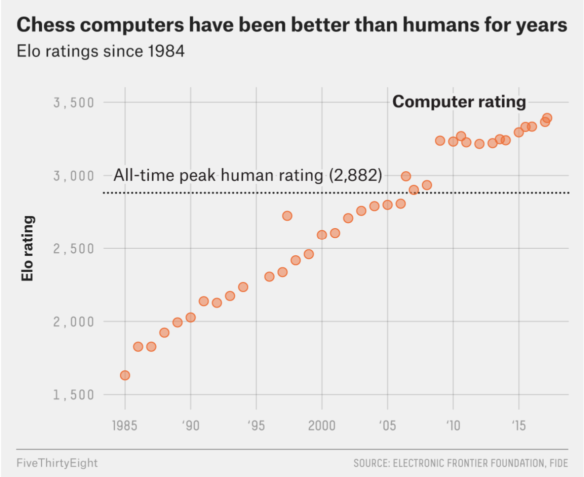
(Number of atoms in the Universe ≈ 10^{80})

Artificial Intelligence

- Chess was considered a very difficult game:
  - 1997: Kasparov vs DeepBlue

Artificial Intelligence

- Chess was considered a very difficult game:



Chess computers have been better than humans for years

Elo ratings since 1984

FiveThirtyEight

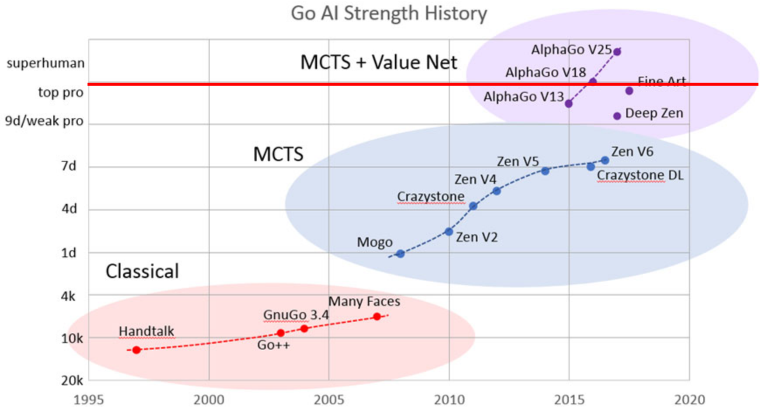
SOURCE: ELECTRONIC FRONTIER FOUNDATION, FIDE

Artificial Intelligence

- The progress of Computer Chess were due to the increase in computer power, which allowed programs to enumerate more possibilities.
- Despite this increase, Go was considered infeasible because of the gap in complexity

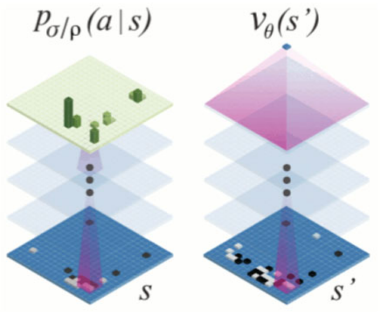
Artificial Intelligence

- ◆ Go was considered infeasible but:
  - 2017 AlphaGo



AlphaGo

- ◆ Chess was solved because computers became faster
- ◆ Go was solved because of a new learning technique: Deep Learning
  - Policy (Deep) network
  - Value (Deep) network



Artificial Intelligence - Machine Learning

- In traditional programming, the software developer describes each step of the processing
- In Machine Learning, the AI scientist provides a model and examples, then lets the machine find the best instance of the model (this is the training). When the model is trained, it can be used to process new data.

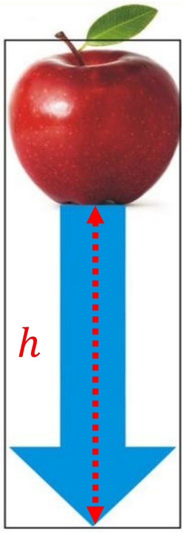
Artificial Intelligence - Machine Learning

- Problems are usually solved by a

formula: $h = \frac{1}{2} g t^2$ so $t = \sqrt{\frac{2h}{g}}$

$h = 10\,m$
$g = 9.8\,m/s^2$

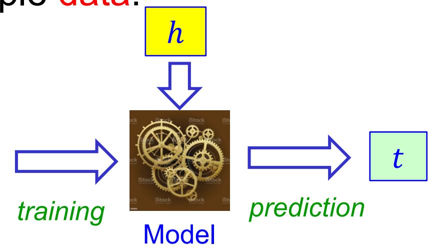
$$t = \sqrt{\frac{2 \times 10}{9.8}} = \sqrt{2.04} = 1.43s$$

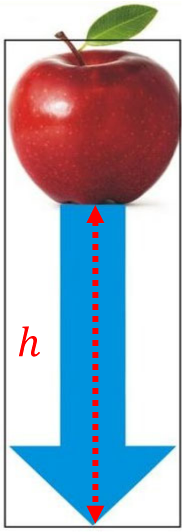


Artificial Intelligence - Machine Learning

- Problems are usually solved by a formula: $h = \frac{1}{2}gt^2$ so $t = \sqrt{\frac{2h}{g}}$
- Machine Learning builds a model from example data:

h	t
1m	0.45s
2m	0.64s
3m	0.78s
4m	0.90s
…	…





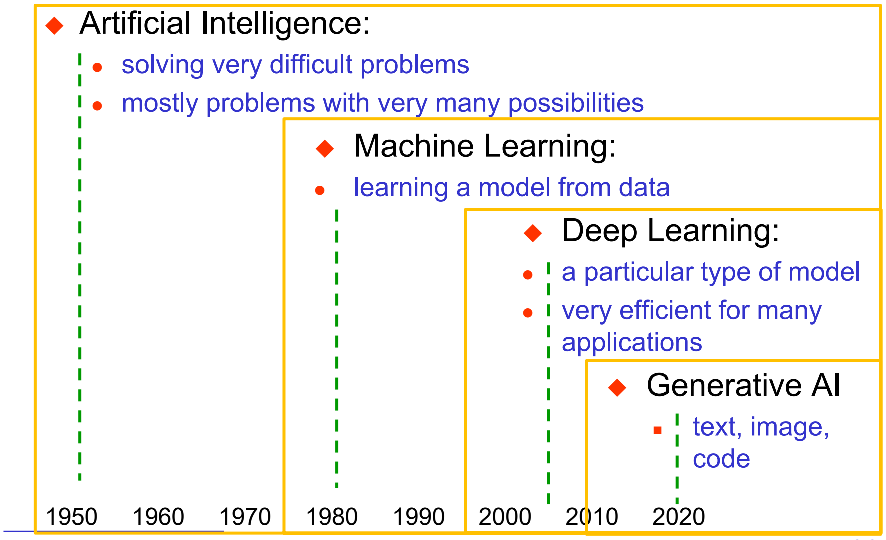
Artificial Intelligence

- Why is ML useful ?
- Because if can be applied to problems for which we have no formula, provided that we have examples.
- Applications:
  - Image recognition
  - Language translation
  - Targeted advertising
  - …

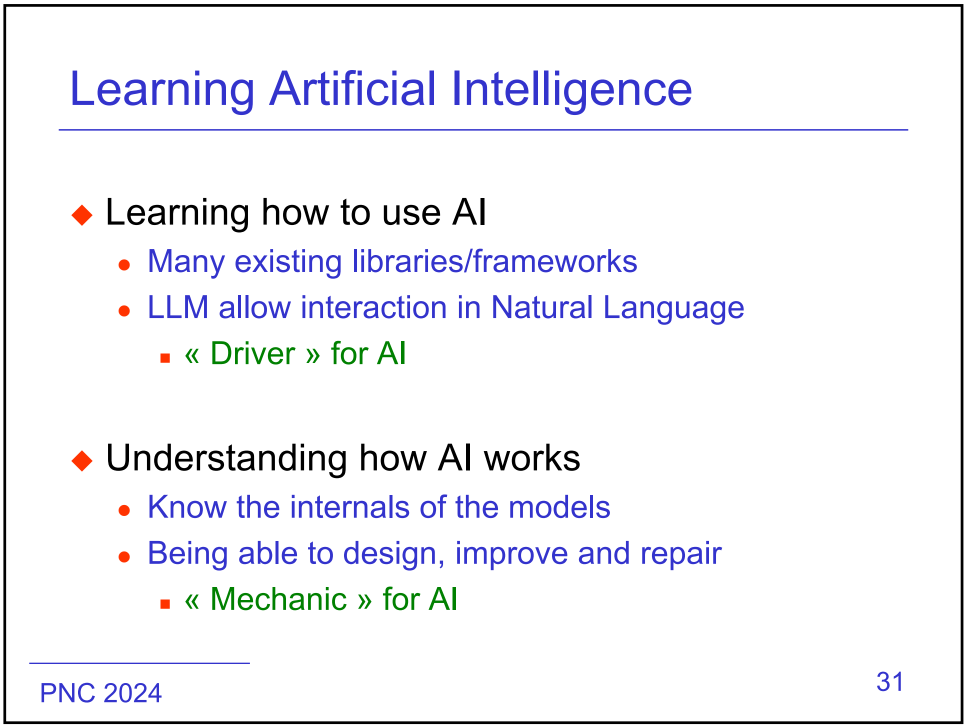
Artificial Intelligence

- Why is AI so important today ?
- Because of this new technique called « Deep Learning », which is one instance of Machine Learning
- It has been shown to be VERY effective on many difficult problems:
  - Image Recognition
  - Natural Language Processing
  - Data Analysis
- Recently, Generative AI allows to produce realistic content: text, images, code…

Artificial Intelligence



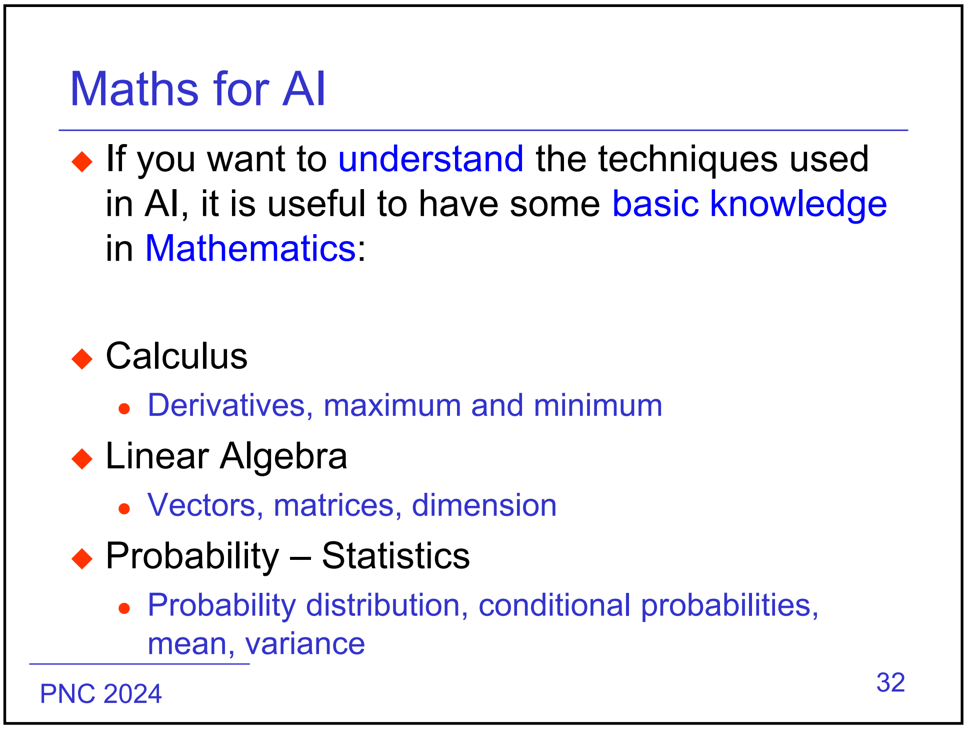
- Artificial Intelligence:
  - solving very difficult problems
  - mostly problems with very many possibilities
- Machine Learning:
  - learning a model from data
- Deep Learning:
  - a particular type of model
  - very efficient for many applications
- Generative AI
  - text, image, code



Learning Artificial Intelligence

- Learning how to use AI
  - Many existing libraries/frameworks
  - LLM allow interaction in Natural Language
    - « Driver » for AI

- Understanding how AI works
  - Know the internals of the models
  - Being able to design, improve and repair
    - « Mechanic » for AI



Maths for AI

- If you want to understand the techniques used in AI, it is useful to have some basic knowledge in Mathematics:

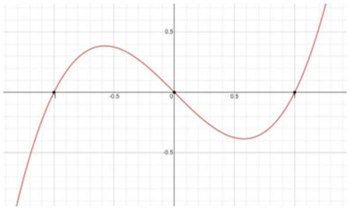
- Calculus
  - Derivatives, maximum and minimum
- Linear Algebra
  - Vectors, matrices, dimension
- Probability – Statistics
  - Probability distribution, conditional probabilities, mean, variance

Maths for AI – Calculus $f(x)$

- Remember derivatives $f'(x)$ (also noted $\frac{\partial f}{\partial x}(x)$)
  - $(x^2)' = 2x,\ \left(\frac{1}{x}\right)' = -\frac{1}{x^2},\ (\sin x)' = \cos x,\ \dots$
- Used to find extrema:
  - $f(x) = x^3 - x,\ f'^{(x)} = 3x^2 - 1$

x		$-\sqrt{3}/3$		0		$\sqrt{3}/3$	
$f'(x)$	+	0	-		-	0	+
$f(x)$	↗		↘	0	↘		↗

local maximum *local minimum*



Maths for AI – Probabilities, Statistics

- Probability distribution:
  - Example : people height

Height	1m50	1m60	1m65	
Number	🧍🧍🧍🧍	🧍🧍🧍🧍🧍	🧍🧍🧍	12
Probability	$\frac{4}{12} = 0.333$	$\frac{5}{12} = 0.416$	$\frac{3}{12} = 0.25$	$\sum P(x) = 1$

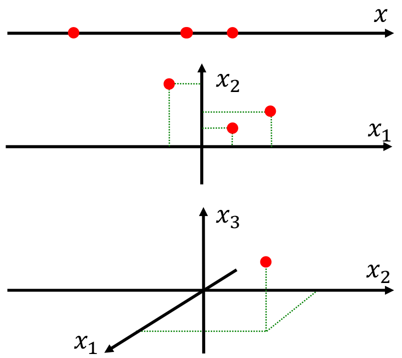
- Mean (average) height:

$P(h = 1.50) \times 1.50 + P(h = 1.60) \times 1.60 + P(h = 1.65) \times 1.65$

- $\frac{4}{12} \times 1.50 + \frac{5}{12} \times 1.60 + \frac{3}{12} \times 1.65 = 1.58$

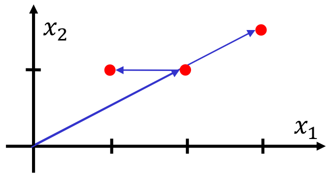
Maths for AI - Linear Algebra

- Vectors: $(x_1, x_2, x_3, x_4) = (1,\ 2.5, 0, -5.2)$
  - list of numbers of fixed size (dimension)
- Vector Spaces:
  - Dimension 1: line
  - Dimension 2: plane
    - (x_1, x_2)
  - Dimension 3: space
    - $(x_1, x_2, x_3\ )$
  - Dimension 4, 5, … N…

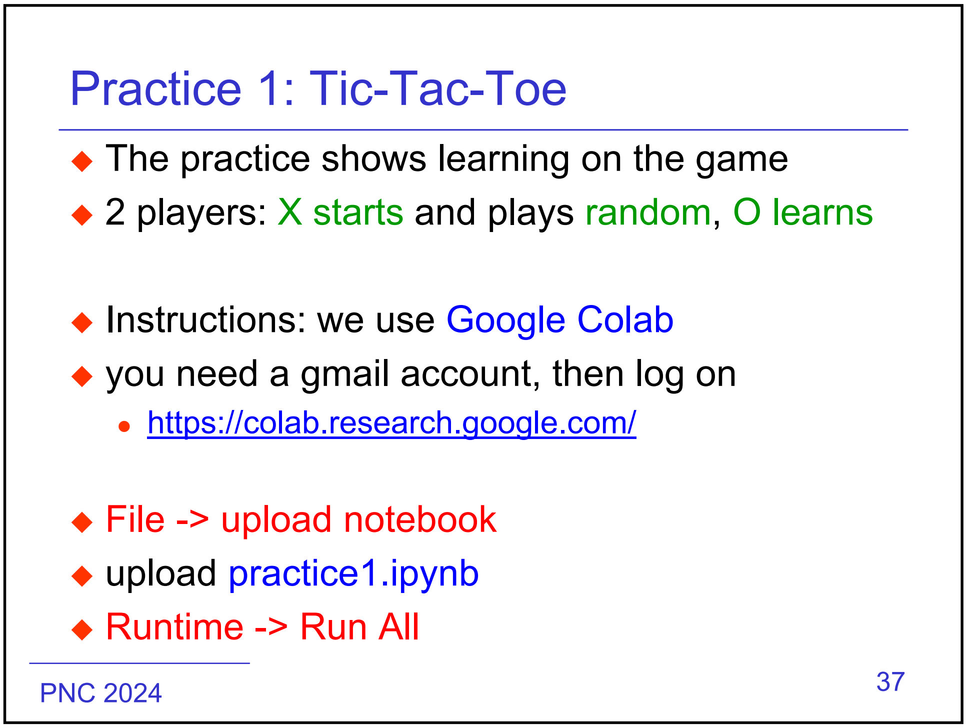


Maths for AI - Linear Algebra

- Operations on vectors
  - $(2,1) + (-1, 0) = (1, 1)$
  - $1.5 \times (2,1) = (3,1.5)$



- Matrices:
  - a linear relation $\begin{cases} y_1 = a \times x_1 + b \times x_2 \\ y_2 = c \times x_1 + d \times x_2 \end{cases}$
  - can be written as $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
  - can also be written as $Y = AX$

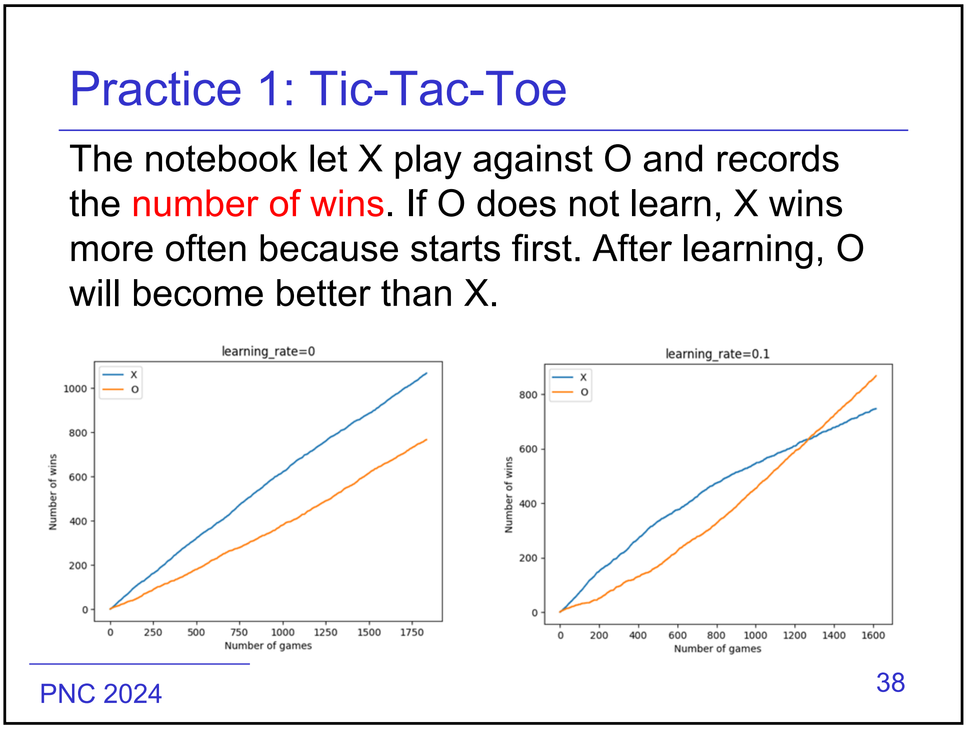


Practice 1: Tic-Tac-Toe

- The practice shows learning on the game
- 2 players: X starts and plays random, O learns

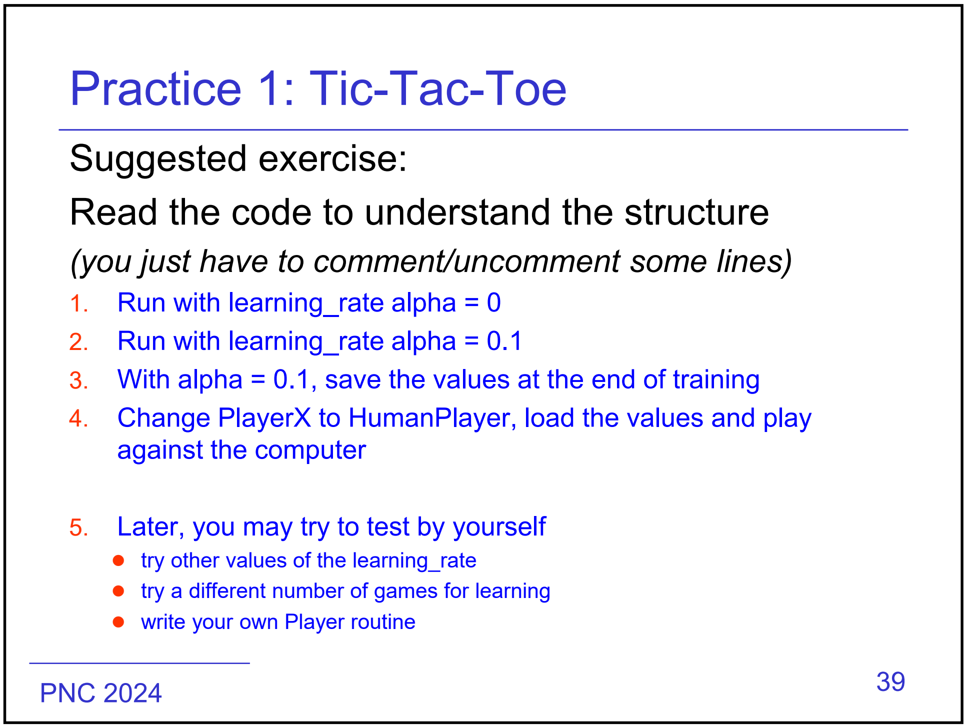
- Instructions: we use Google Colab
- you need a gmail account, then log on
  - https://colab.research.google.com/

- File -> upload notebook
- upload practice1.ipynb
- Runtime -> Run All



Practice 1: Tic-Tac-Toe

The notebook let X play against O and records the number of wins. If O does not learn, X wins more often because starts first. After learning, O will become better than X.



Practice 1: Tic-Tac-Toe

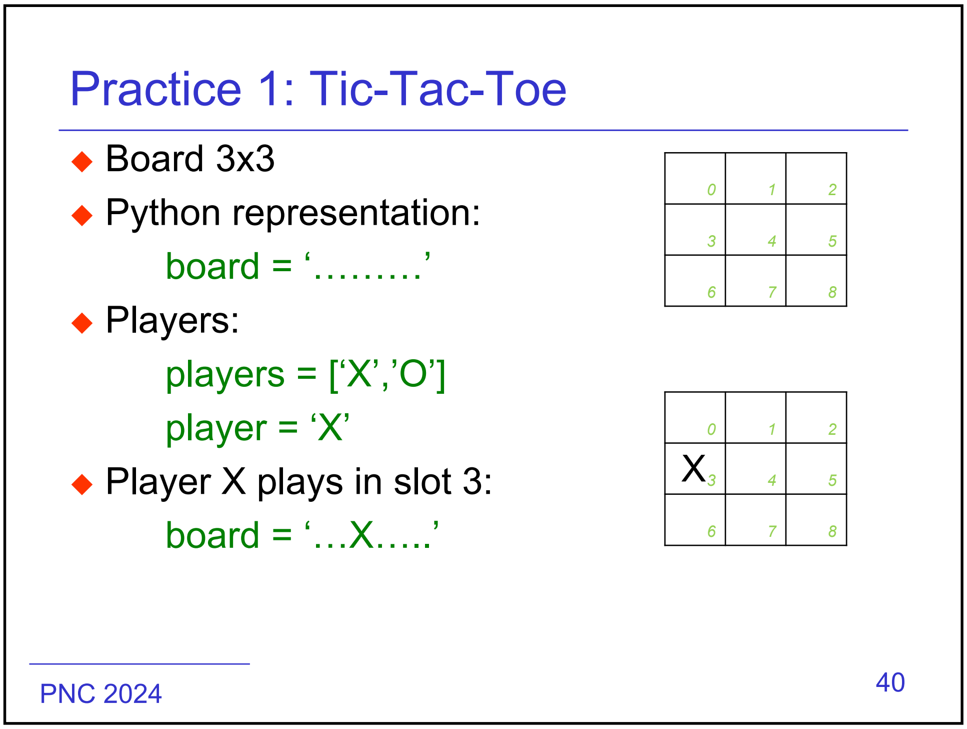
Suggested exercise:

Read the code to understand the structure

(you just have to comment/uncomment some lines)

1. Run with learning_rate alpha = 0
2. Run with learning_rate alpha = 0.1
3. With alpha = 0.1, save the values at the end of training
4. Change PlayerX to HumanPlayer, load the values and play against the computer

5. Later, you may try to test by yourself
   - try other values of the learning_rate
   - try a different number of games for learning
   - write your own Player routine



Practice 1: Tic-Tac-Toe

- Board 3x3
- Python representation:

  board = '………'
- Players:

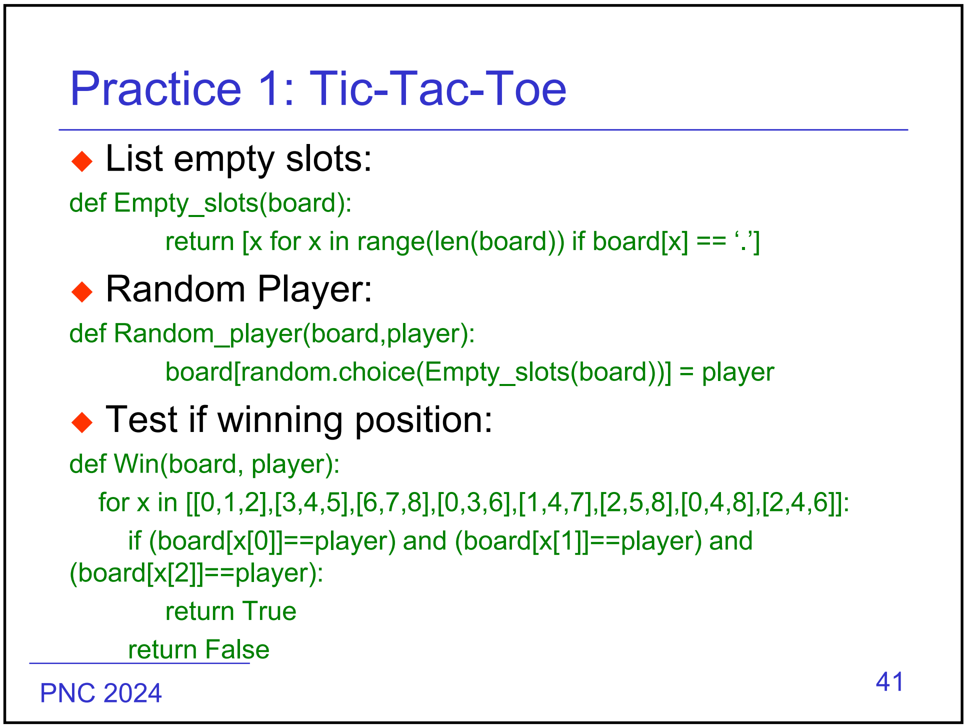
  players = ['X','O']

  player = 'X'
- Player X plays in slot 3:

  board = '…X…..'

0	1	2
3	4	5
6	7	8

0	1	2
X 3	4	5
6	7	8



Practice 1: Tic-Tac-Toe

- List empty slots:

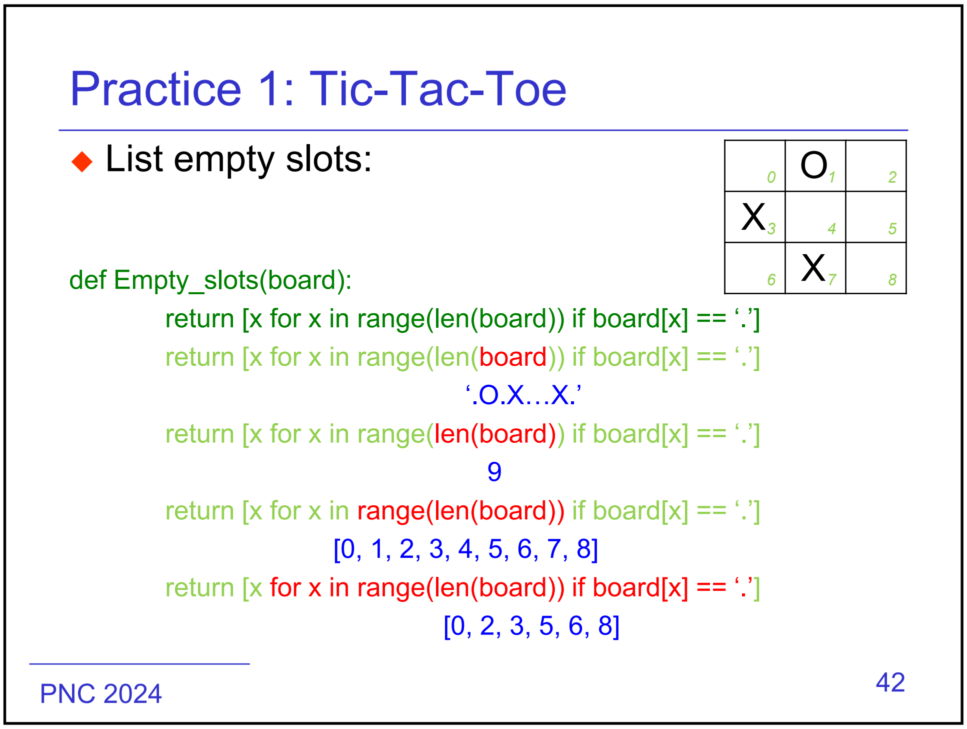
```
def Empty_slots(board):
          return [x for x in range(len(board)) if board[x] == ‘.’]
```

- Random Player:

```
def Random_player(board,player):
          board[random.choice(Empty_slots(board))] = player
```

- Test if winning position:

```
def Win(board, player):
   for x in [[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]:
      if (board[x[0]]==player) and (board[x[1]]==player) and
(board[x[2]]==player):
          return True
      return False
```

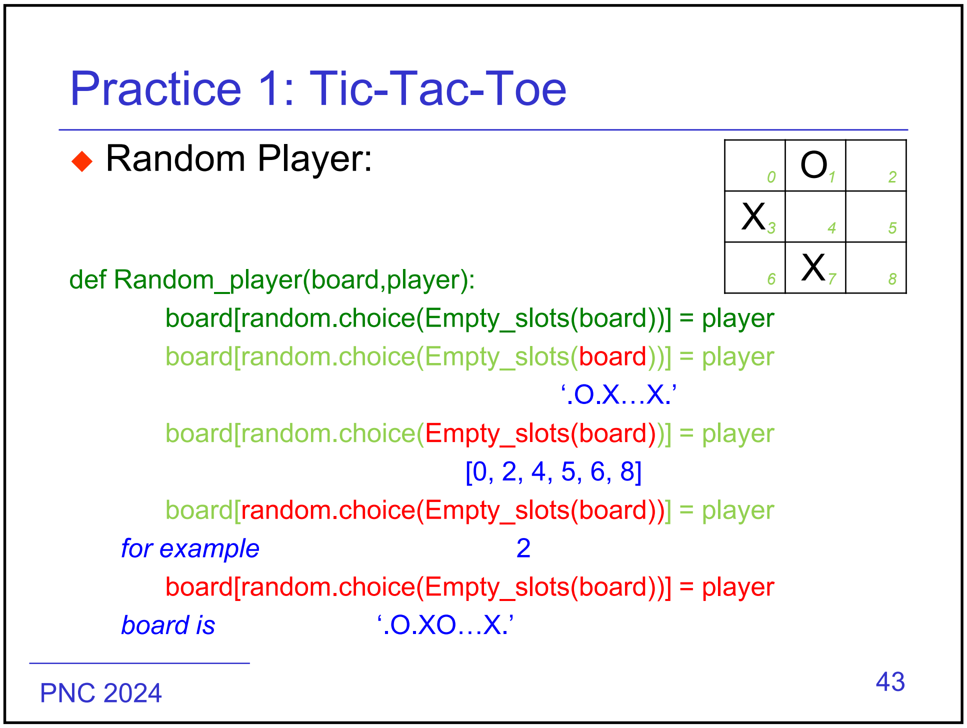


Practice 1: Tic-Tac-Toe

- List empty slots:

0	O$_1$	2
X$_3$	4	5
6	X$_7$	8

```
def Empty_slots(board):
          return [x for x in range(len(board)) if board[x] == ‘.’]
          return [x for x in range(len(board)) if board[x] == ‘.’]
                                 ‘.O.X…X.’
          return [x for x in range(len(board)) if board[x] == ‘.’]
                                    9
          return [x for x in range(len(board)) if board[x] == ‘.’]
                         [0, 1, 2, 3, 4, 5, 6, 7, 8]
          return [x for x in range(len(board)) if board[x] == ‘.’]
                                  [0, 2, 3, 5, 6, 8]
```

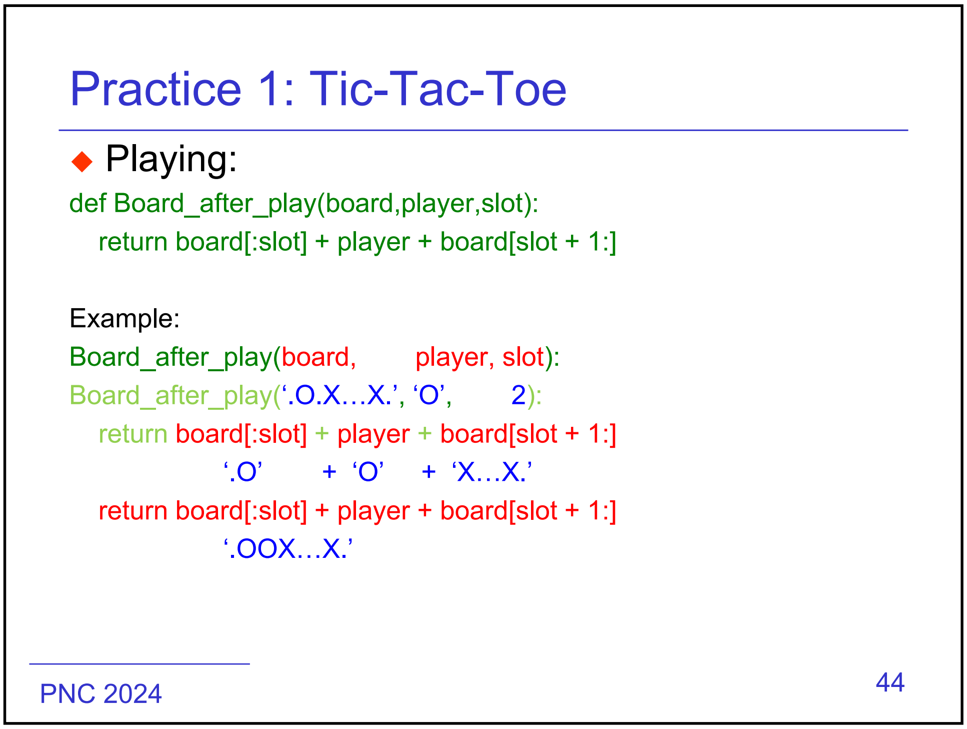


Practice 1: Tic-Tac-Toe

◆ Random Player:

0	O 1	2
X 3	4	5
6	X 7	8

```
def Random_player(board,player):
        board[random.choice(Empty_slots(board))] = player
        board[random.choice(Empty_slots(board))] = player
                                     '.O.X...X.'
        board[random.choice(Empty_slots(board))] = player
                              [0, 2, 4, 5, 6, 8]
        board[random.choice(Empty_slots(board))] = player
    for example                         2
        board[random.choice(Empty_slots(board))] = player
    board is              '.O.XO...X.'
```

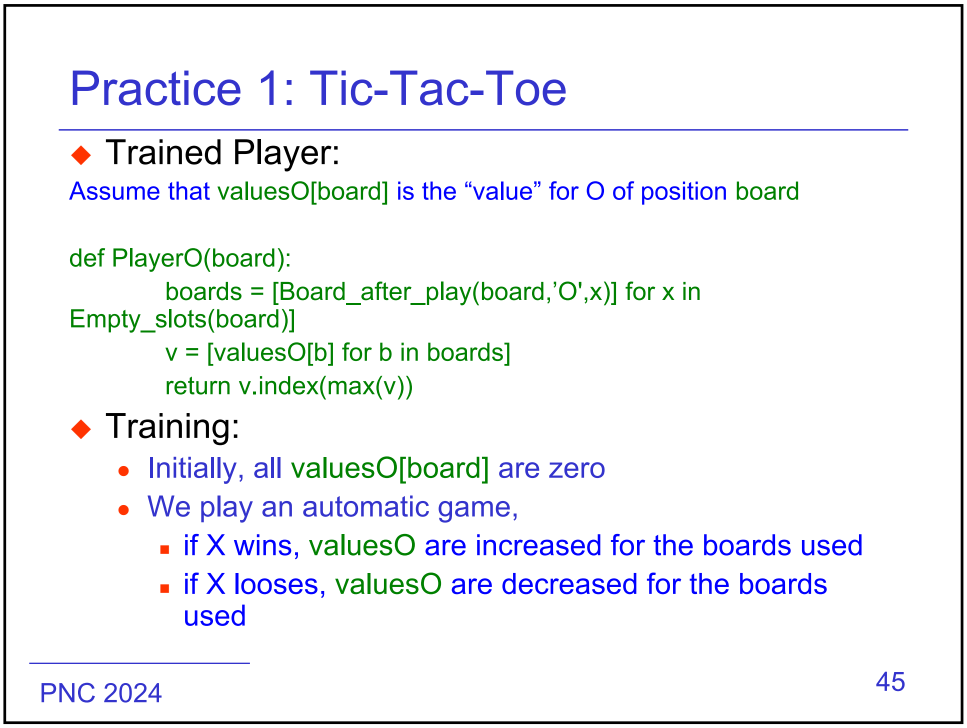


Practice 1: Tic-Tac-Toe

◆ Playing:

```
def Board_after_play(board,player,slot):
   return board[:slot] + player + board[slot + 1:]

Example:
Board_after_play(board,      player, slot):
Board_after_play('.O.X...X.', 'O',      2):
   return board[:slot] + player + board[slot + 1:]
               '.O'      +  'O'    +  'X...X.'
   return board[:slot] + player + board[slot + 1:]
               '.OOX...X.'
```



Practice 1: Tic-Tac-Toe

- Trained Player:

Assume that valuesO[board] is the “value” for O of position board

```
def PlayerO(board):
        boards = [Board_after_play(board,’O',x)] for x in
Empty_slots(board)]
        v = [valuesO[b] for b in boards]
        return v.index(max(v))
```

- Training:
  - Initially, all valuesO[board] are zero
  - We play an automatic game,
    - if X wins, valuesO are increased for the boards used
    - if X looses, valuesO are decreased for the boards used