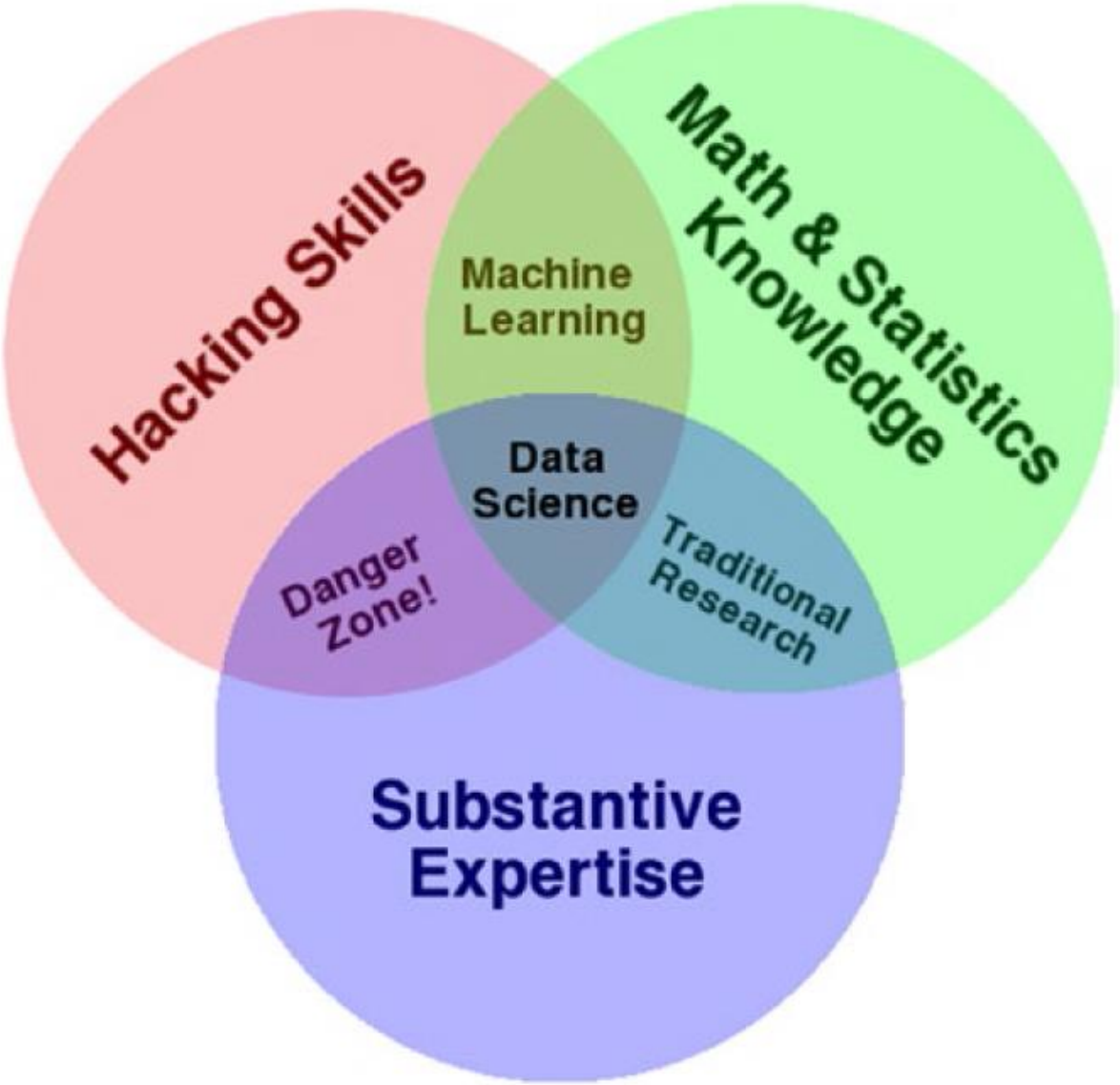
Narzędzia Sztucznej Inteligencji

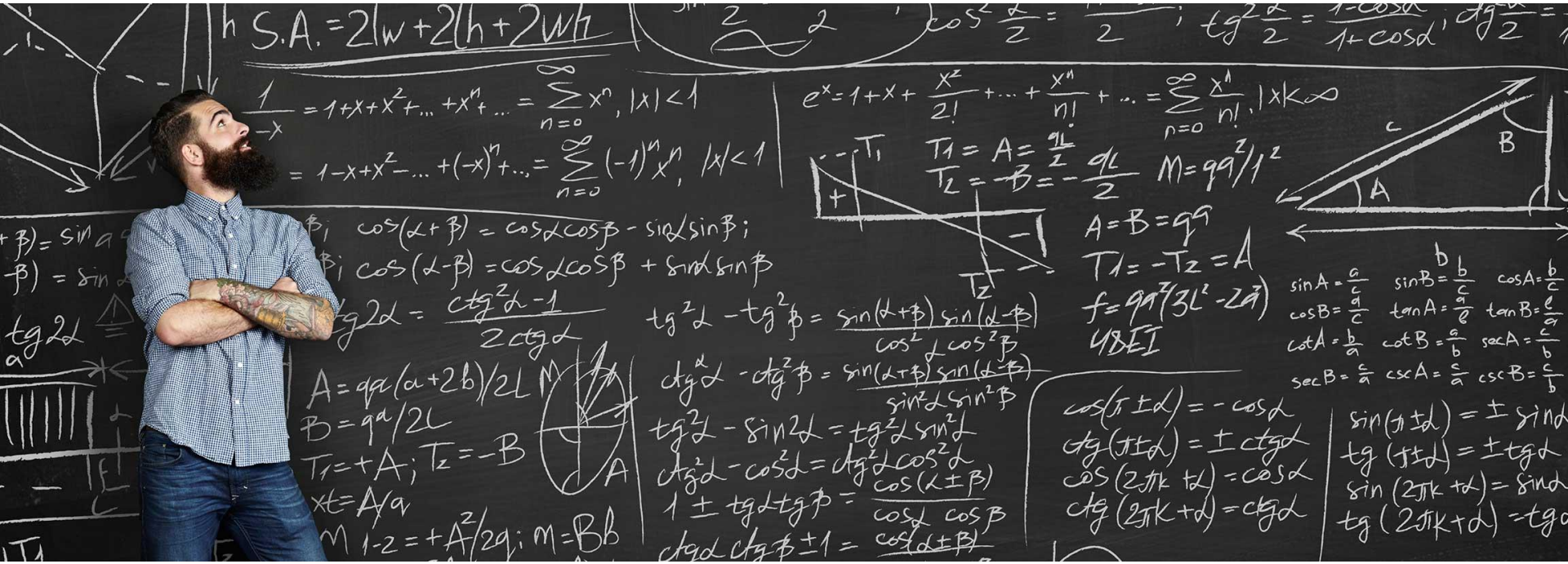
Wykład 01 09.10.2021

Czapiewski Paweł

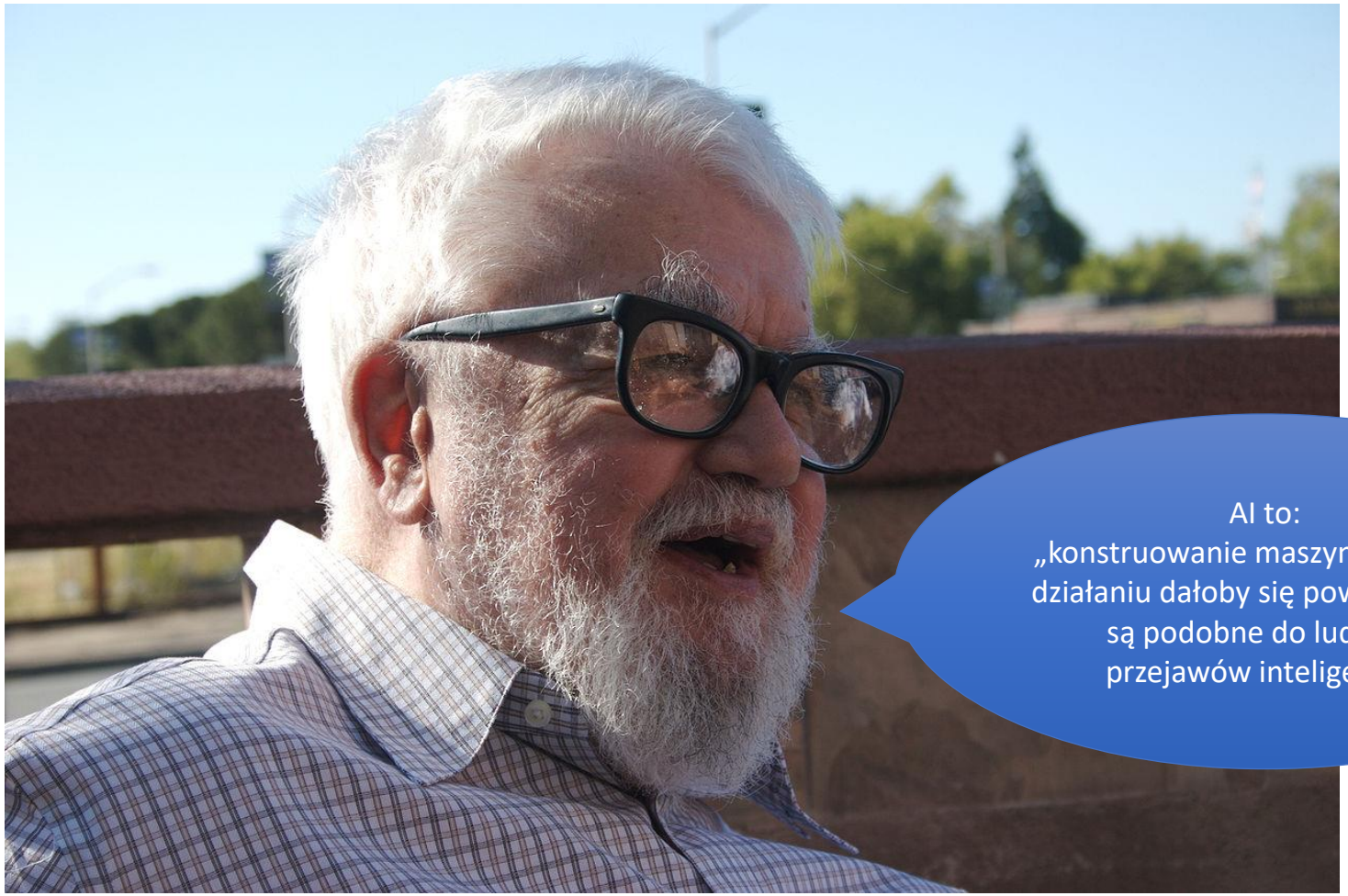
REQUIREMENTS



Source: Mastering Machine Learning with Python in Six Steps



AI Definition AI



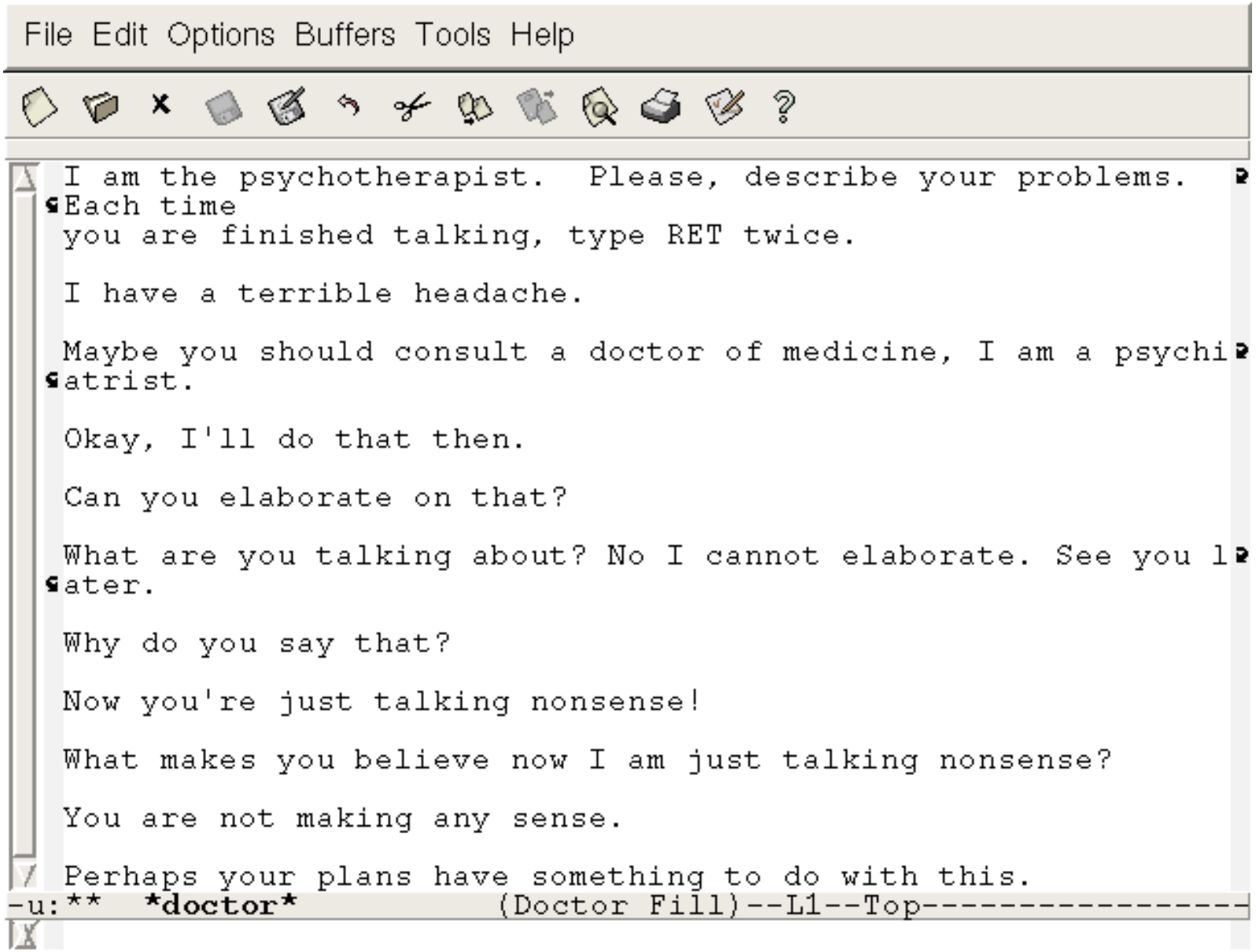
AI


DEMOTYWATORY.PL
Jeśli stworzysz grę w której będzie sztuczna inteligencja, która pokona cię w tej grze
To jesteś głupi czy mądry?

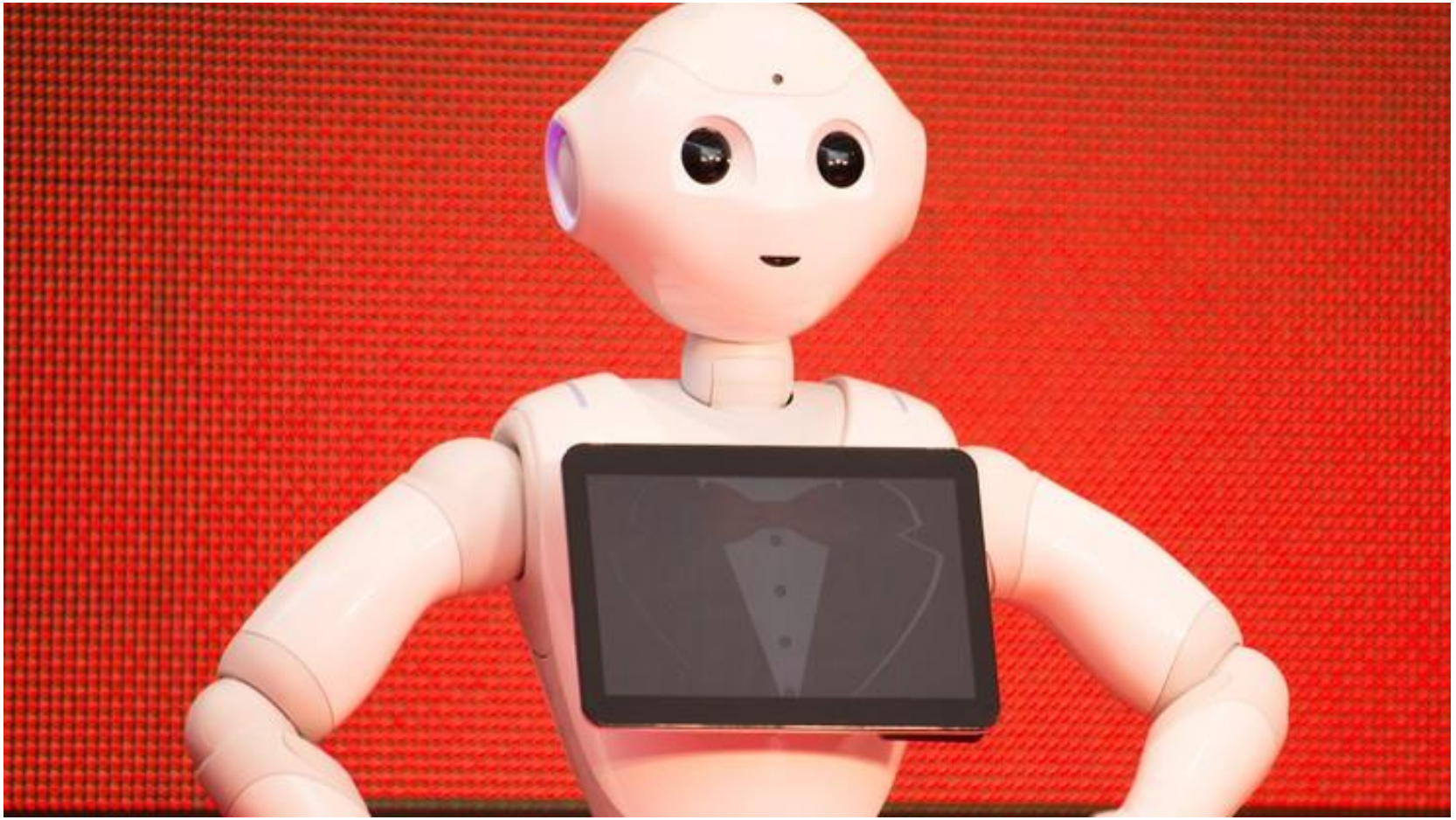
Turnig Test


Test wygląda następująco: sędzia – człowiek – prowadzi rozmowę w języku naturalnym z pozostałymi stronami. Jeśli sędzia nie jest w stanie wiarygodnie określić, czy któraś ze stron jest maszyną czy człowiekiem, wtedy mówi się, że maszyna przeszła test.

Turing Test - Elisa

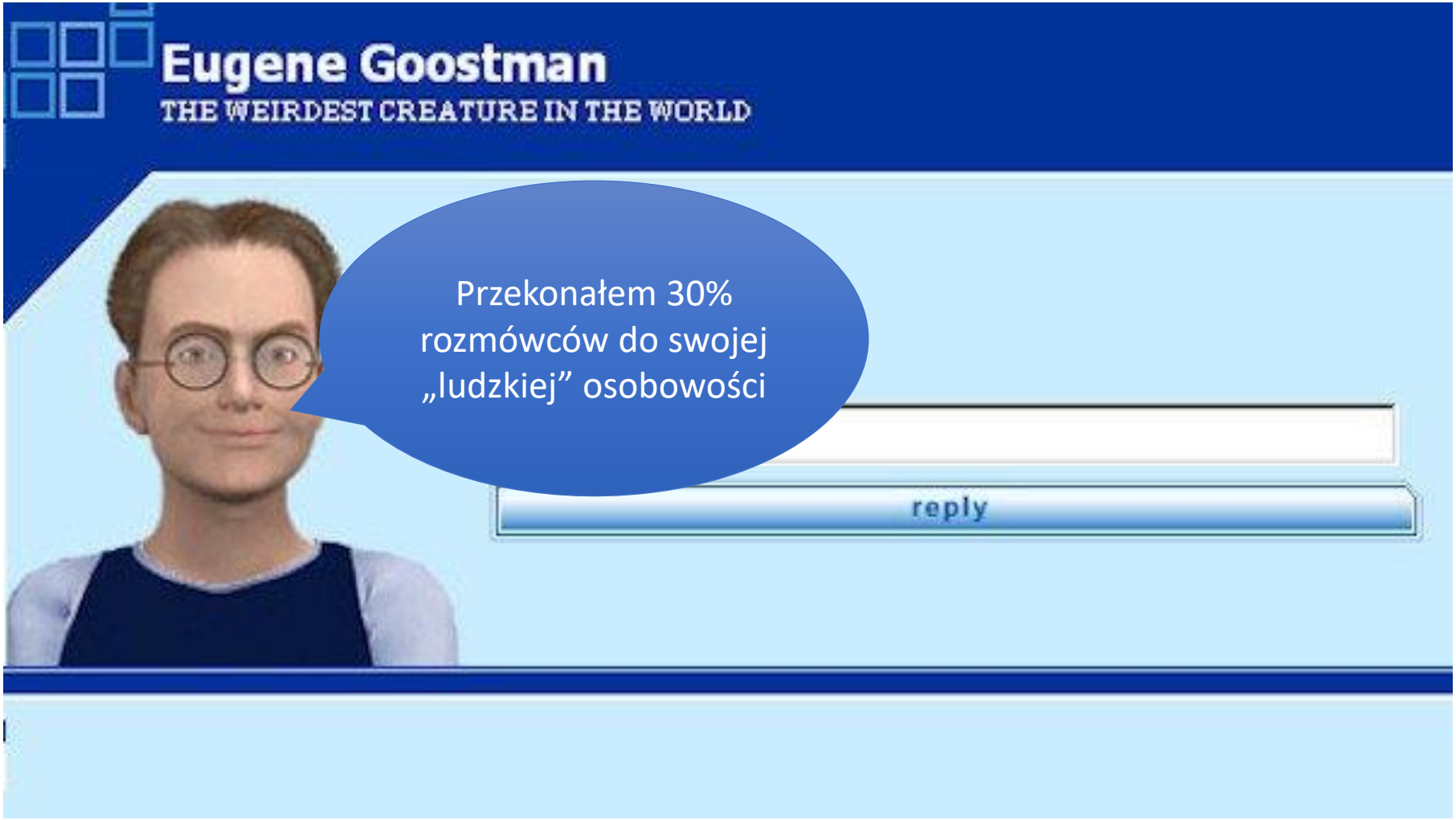


Fabio



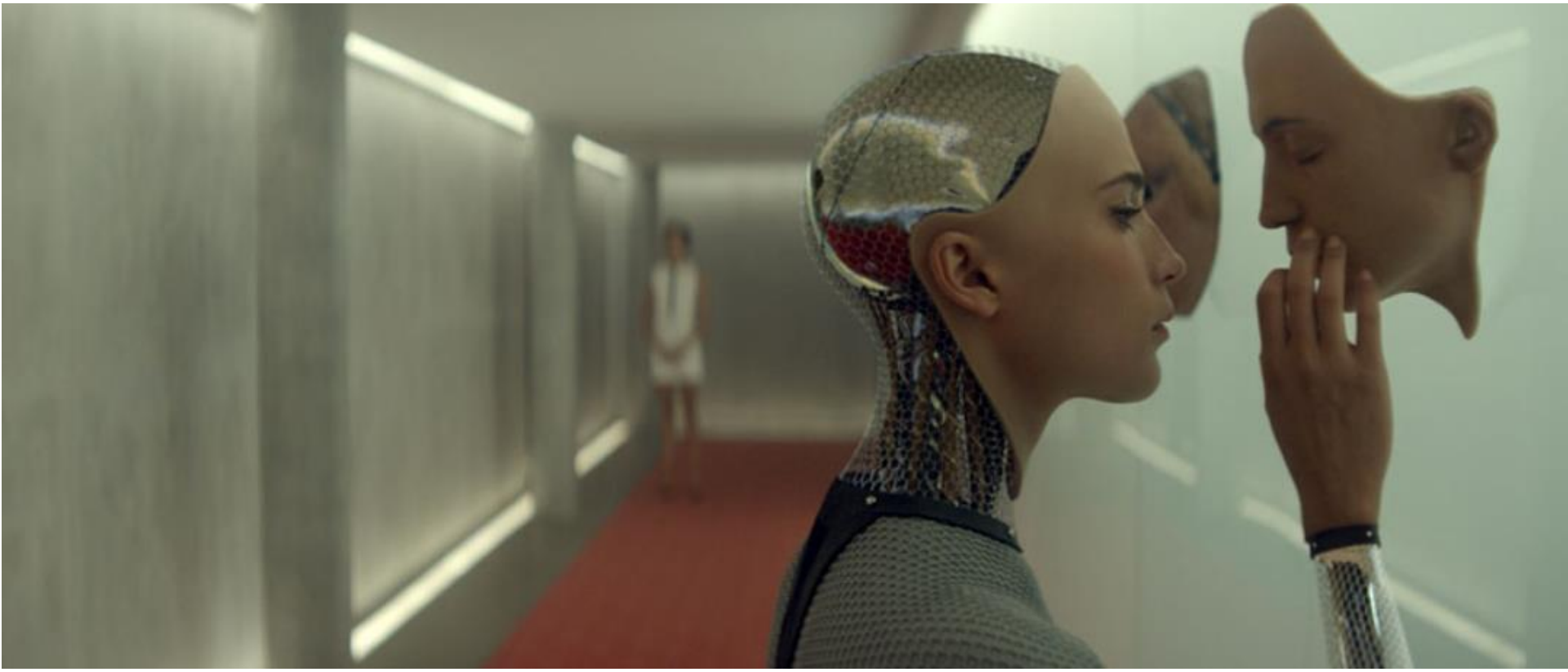
Source: http://www.polsatnews.pl/wiadomosc/2018-01-22/robot-sprzedawca-zwolniony-z-pracy-po-tygodniu-myslelismy-ze-lepiej-sobie-poradzi/?ref=technologie_i_medycyna

Turing Test



Source: https://www.tvn24.pl/wiadomosci-ze-swiata,2/komputer-po-raz-pierwszy-przeszedl-test-turinga-udawal-13-latka,437287.html

Turing Test



Source: https://www.filmweb.pl/

Trendy AI

Past

Artificial Narrow Intelligence (ANI): Machine intelligence that equals or exceeds human intelligence or efficiency at a specific task. An example is IBM's Watson, which requires close participation of subject matter or domain experts to supply data/information and evaluate its performance.

Current

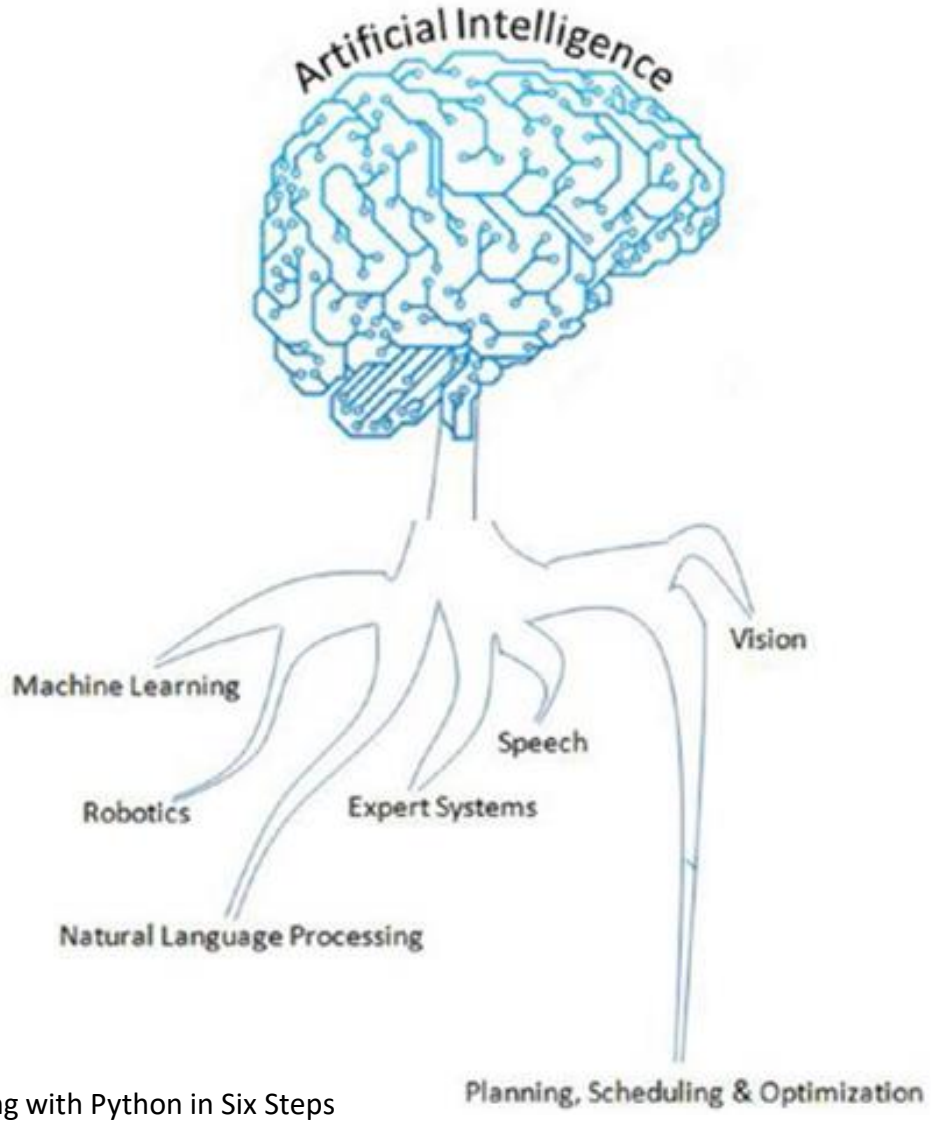
Artificial General Intelligence (AGI): A machine with the ability to apply intelligence to any problem for an area, rather than just one specific problem. Self-driving cars are a good example of this.

Future

Artificial Super Intelligence (ASI): An intellect that is much smarter than the best human brains in practically every field, general wisdom, social skills, and including scientific creativity. The key theme over here is "don't model the world, model the mind".

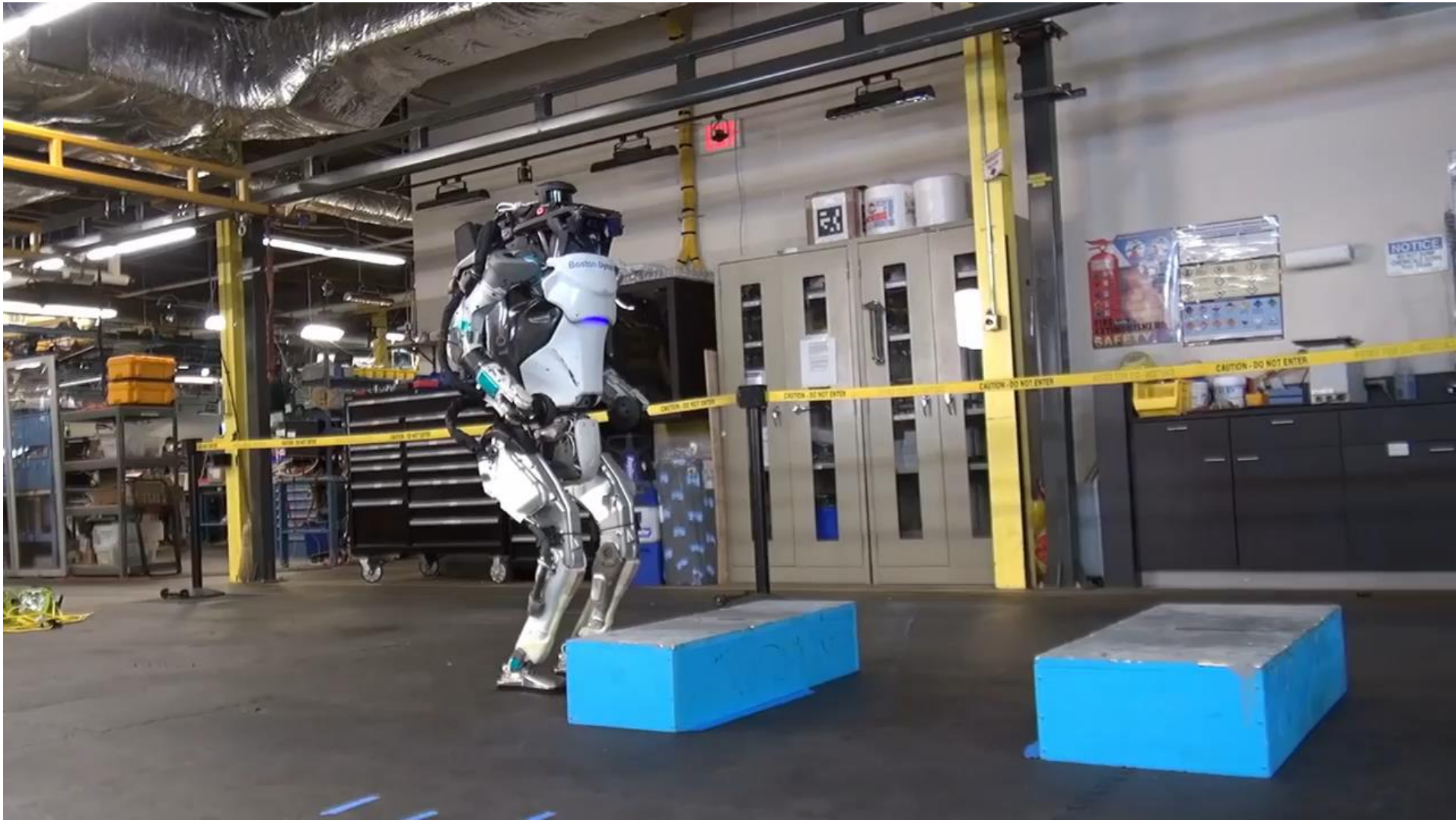
Source: Mastering Machine Learning with Python in Six Steps

Application of AI

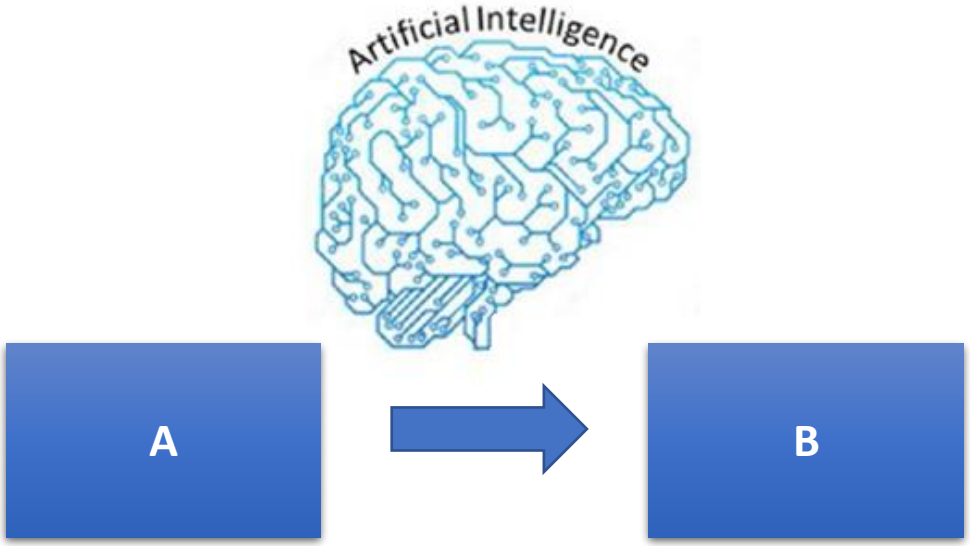


Source: Mastering Machine Learning with Python in Six Steps


Boston Dynamics

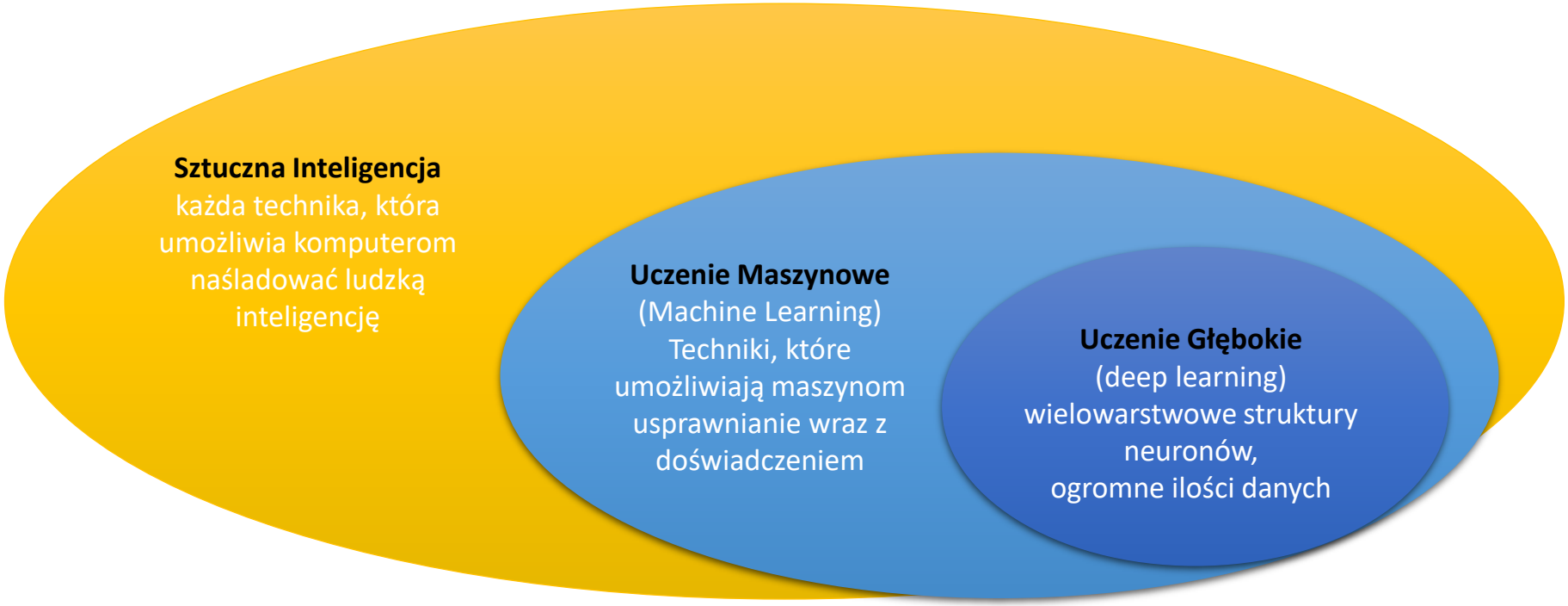

CAUTION - DO NOT ENTER
NOTICE

Most import art of AI



Source: Andrew Ng

Artificial Intelligence



Deep Blue vs Gasparow

VS

Source: https://pl.wikipedia.org/wiki/Plik:Deep_Blue.jpg , https://pl.wikipedia.org/wiki/Plik:Kasparov,_Garri.IMG_2827.JPG

Libratus

Source: https://www.engadget.com/2017/02/10/libratus-ai-poker-winner/

Neural Network

SGR-A1

Source: https://en.wikipedia.org/wiki/Samsung_SGR-A1#/media/File:SGR-A1.jpg

SUPER AEGIS II ROBOT

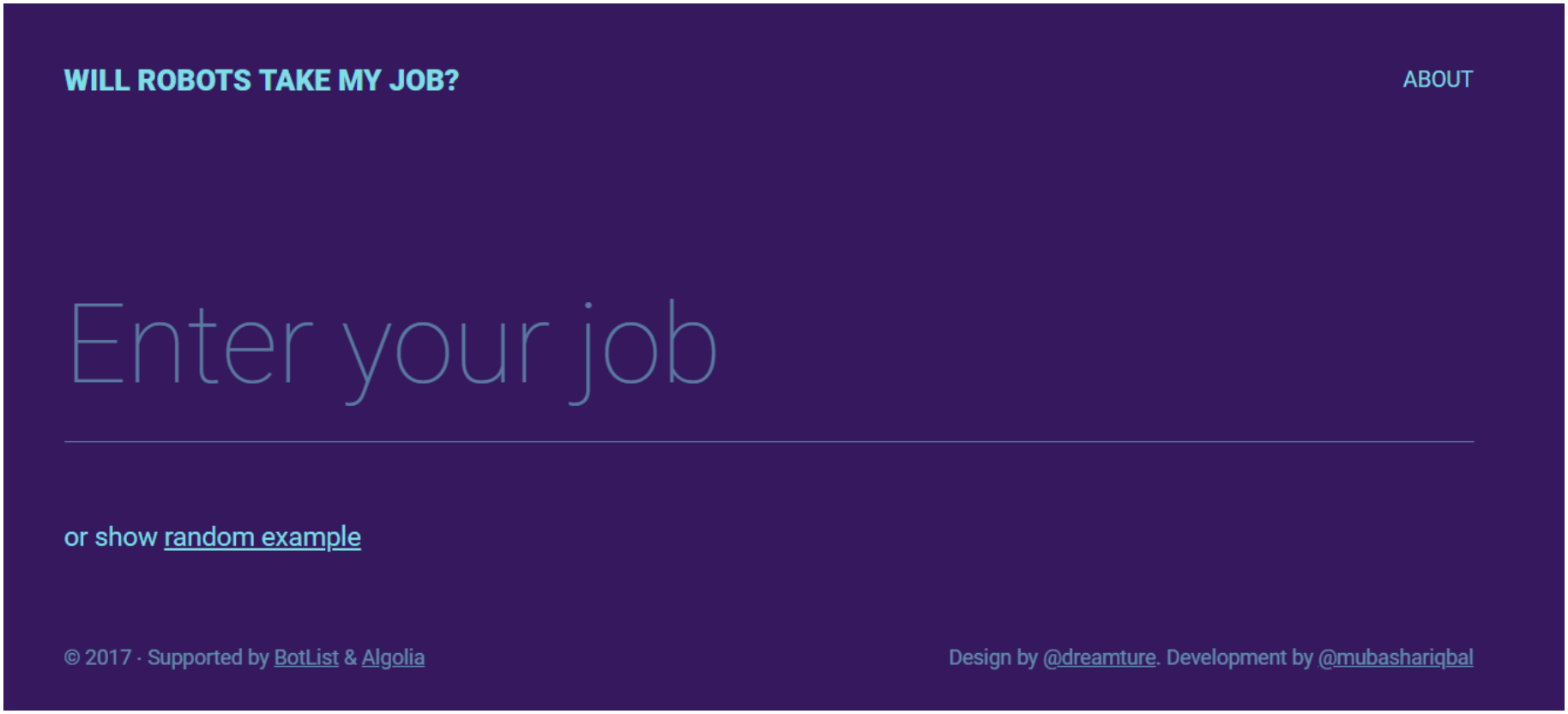
Source: http://www.dodaam.com/eng/sub2/menu2_1_4.php


DARPA

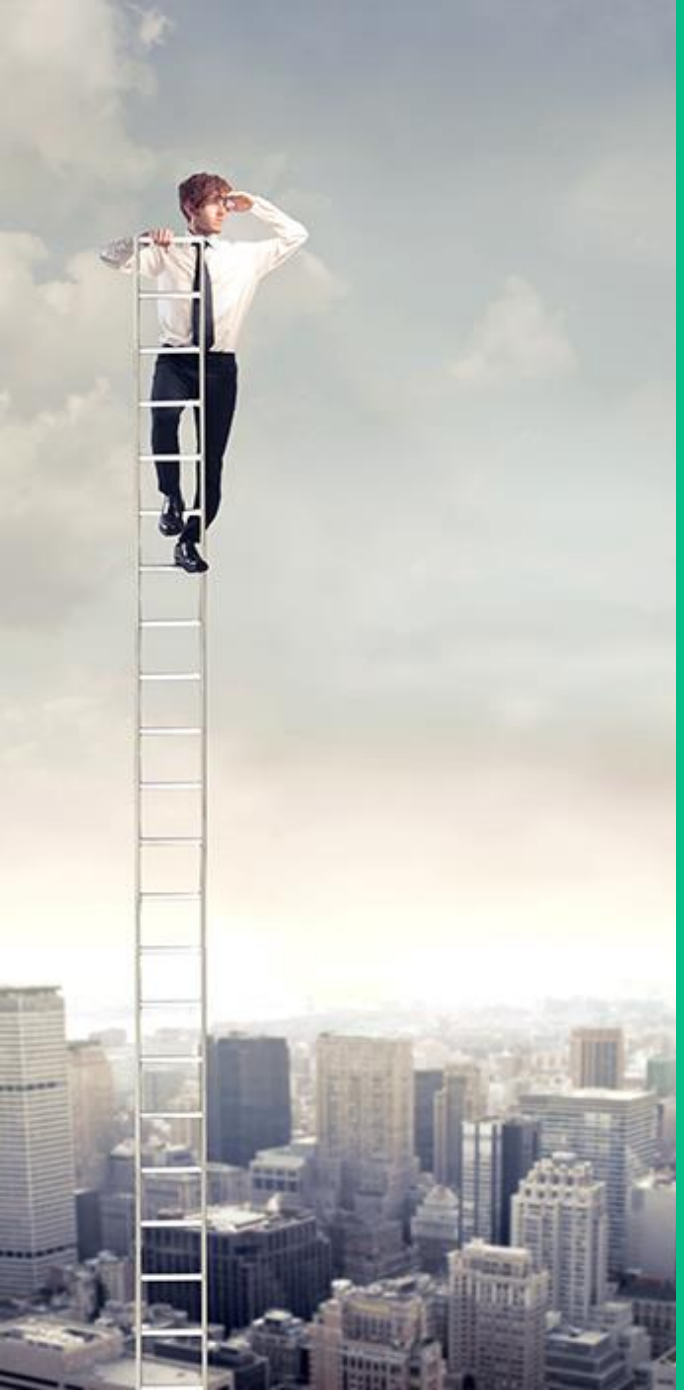

NAVY
NAVAIR

Neural Network - Andrew NG

Neural Network



https://willrobotstakemyjob.com/



Search Problems

Search

A problem can be defined formally by following components:

- The **initial state** that the agent starts in.
- A description of the **possible actions** available to the agent. Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s. We say that each of these actions is applicable in s.
- A description of what each action does; the formal name for this is the **transition model**, specified by a function RESULT(s,a) that returns the state that results from doing action a in state s. We also use the term successor to refer to any state reachable from a given state by a single action

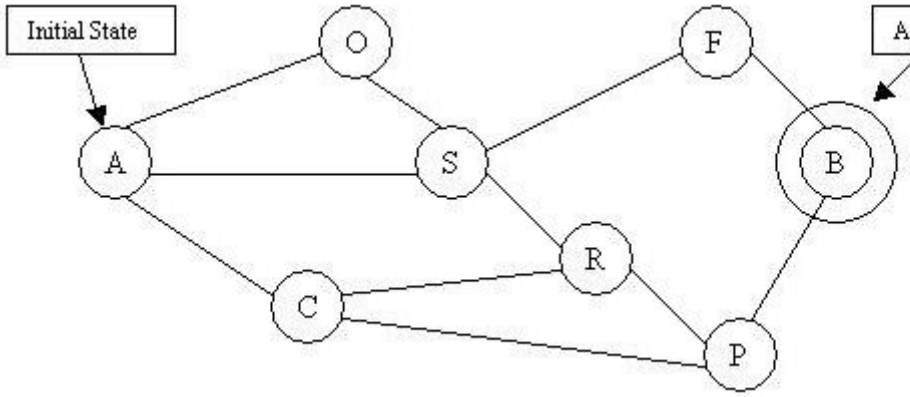
SourceL https://mytechpulse.com/solved-windows-search-working-windows-7-8-8-1/

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Search

A problem can be defined formally by following components:

- Together, the initial state, actions, and transition model implicitly define the **state space** of the problem—the set of all states reachable from the initial state by any sequence of actions. The state space forms a directed network or graph in which the nodes are states and the links between nodes are actions.
- The **goal test**, which determines whether a given state is a goal state. Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state isone ofthem.



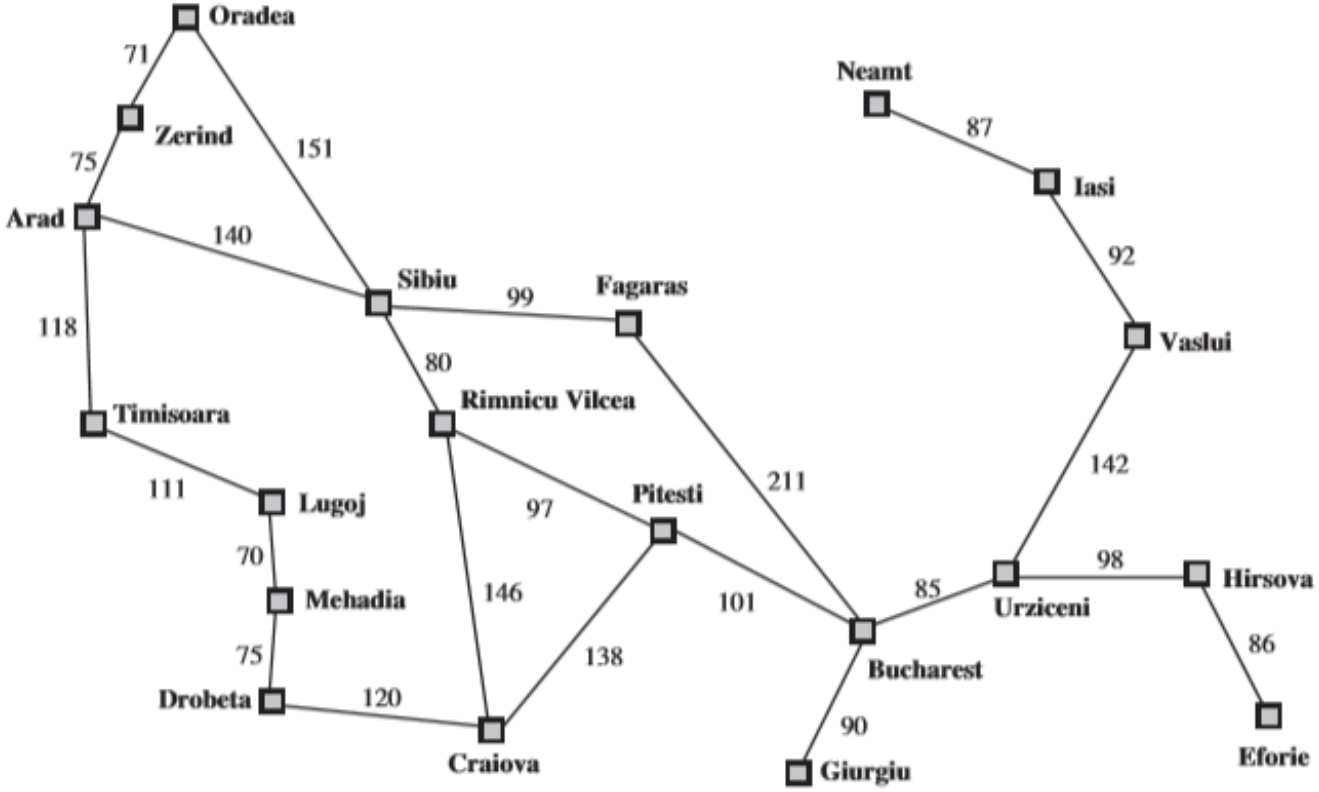
Source: http://cseweb.ucsd.edu/~dturnbul/CSE150/DiscussionNotes/Week1/CSE150_Discussion_Week1.html

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Search

A problem can be defined formally by following components:

- A **path cost** function that assigns a numeric cost to each path. The problem-solving agent chooses a cost function that reflects its own **performance measure**. The step cost of taking action *a* in state *s* to reach state *s'* is denoted by *c(s,a,s')*.

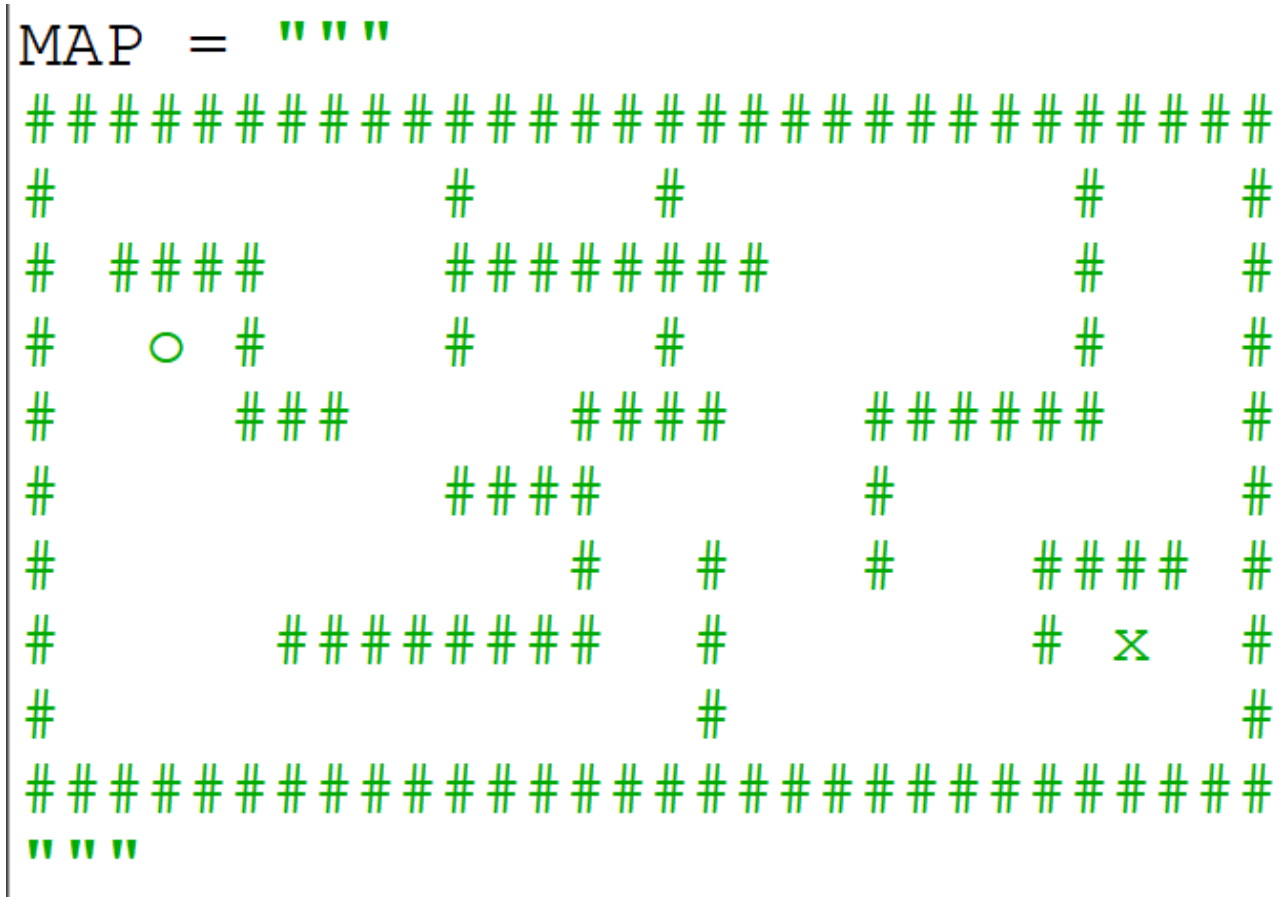


A simplified road map of part of Romania

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

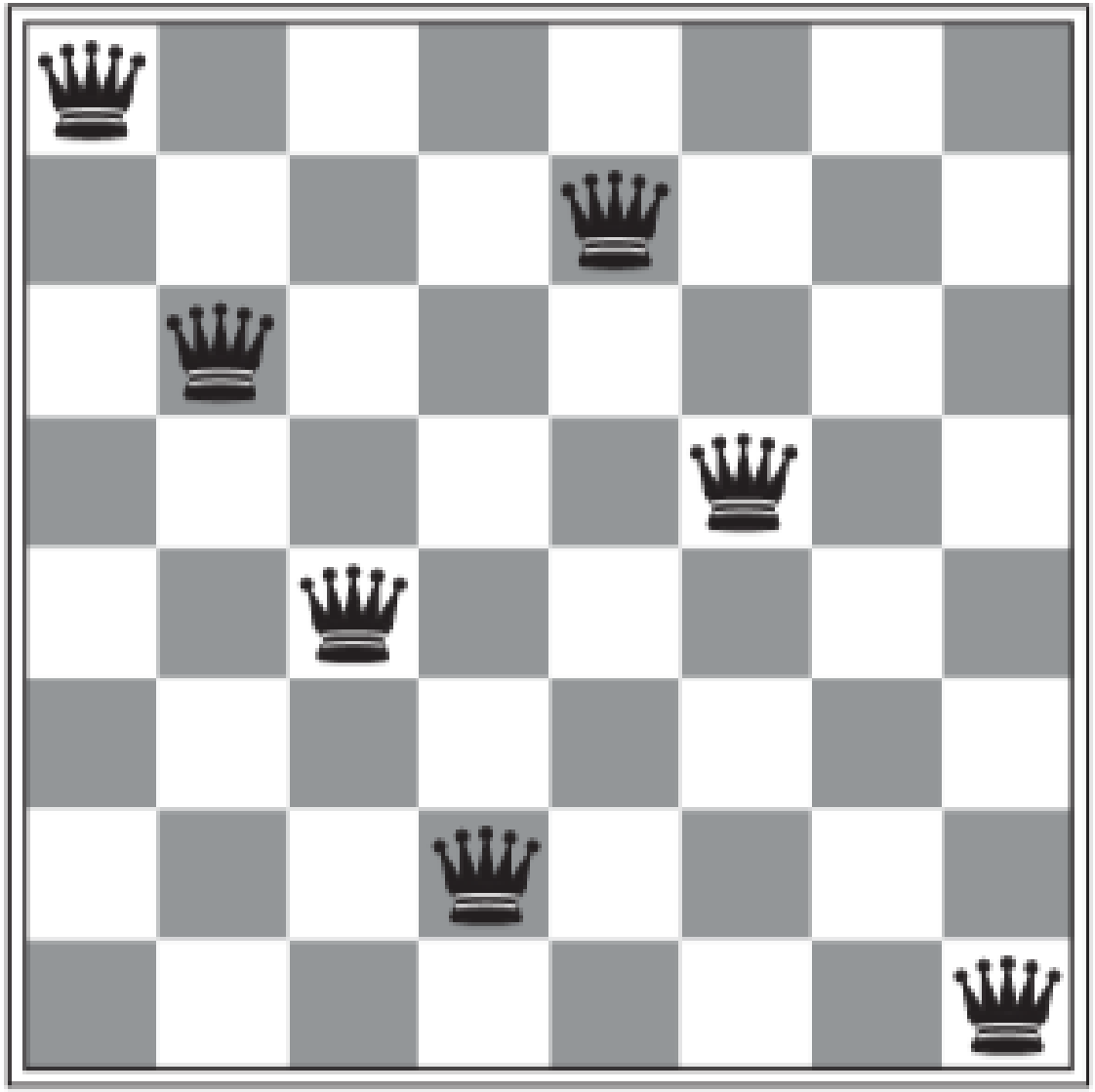
Search Problem - Demo

from simpleai.search import SearchProblem, astar



Source: https://github.com/simpleai-team/simpleai/blob/master/samples/search/game_walk.py

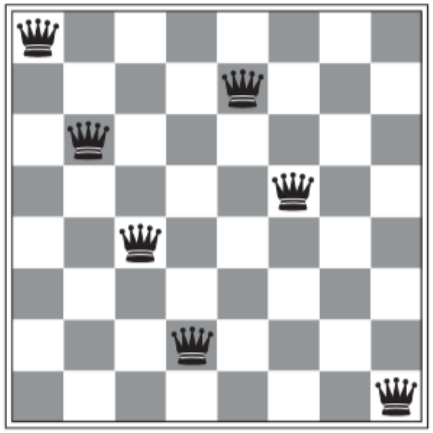
Search 8-queens problem



Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Search 8-queens problem

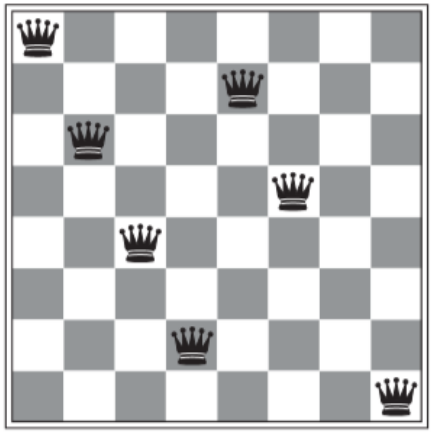
- **States**: Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state:** No queens on the board.
- **Actions:** Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attackedsequences to investigate. A better formulation would prohibit placing a queen in any square that is already attacked:
- **States**: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- **Actions**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.



Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Search 8-queens problem

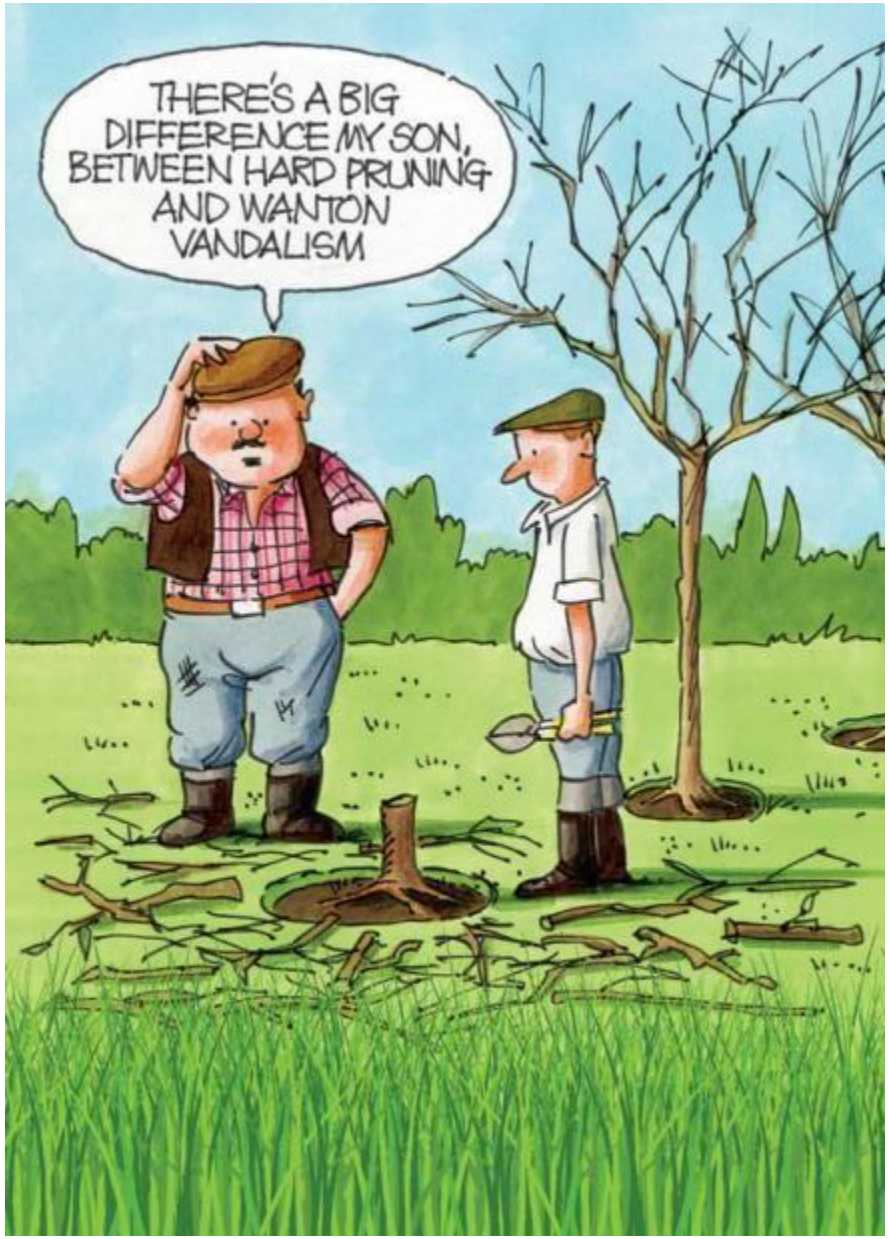
- **States**: Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state:** No queens on the board.
- **Actions:** Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attackedsequences to investigate. A better formulation would prohibit placing a queen in any square that is already attacked:
- **States**: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- **Actions**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.



Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Aversarial Search – Search Tree

Pruning allows us to ignore portions of the search tree that make no difference to the final choice, and heuristic evaluation functions allow us to approximate the true utility of a state **without** doing a **complete search**.



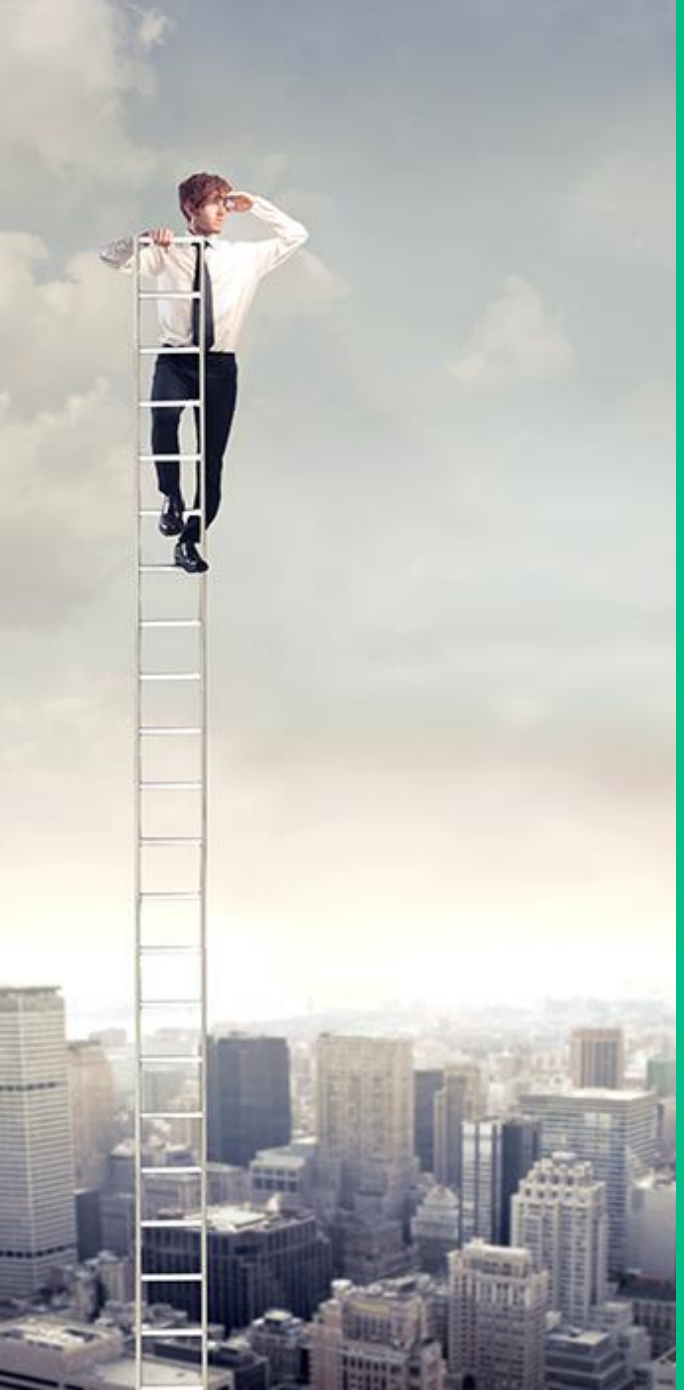
Source: https://justseed.com/products/peartreegw2

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Alfa Beta Prunning

```
/**
 * <p>Title: GT Chess</p>
 * <p>Description: Simple Chess Program</p>
 * <p>Copyright: Copyright (c) 2005</p>
 * <p>fun project</p>
 * @author H. Eck
 * @version 0.01
 */

// class to do an alpha-beta search from the given position
//

/*  nice pseudo-code from Wikipedia :

 evaluate (node, alpha, beta)
    if node is a leaf
        return the heuristic value of node
    if node is a minimizing node
        for each child of node
            beta = min (beta, evaluate (child, alpha, beta))
            if beta <= alpha
                return alpha
        return beta
    if node is a maximizing node
        for each child of node
            alpha = max (alpha, evaluate (child, alpha, beta))
            if beta <= alpha
                return beta
        return alpha

 */
```

Source: http://web.archive.org/web/20070430102811/http://www.computerchess.us/gtchess/



Aversarial Search

Aversarial Search

Mathematical game theory, a branch of economics, views any **multiagent environment** as a game, provided that the impact of each agent on the others is “significant,” regardless of whether the agents are cooperative or competitive.

In AI, the most common games are of a rather specialized kind—what game theorists call **deterministic**, **turn-taking**, two-player**, zero-sum games** of **perfect information** (such as chess). In our terminology, this means deterministic, fully observable environments in which two agents act alternately and in which the utility values at the end of the game are always equal and opposite.

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

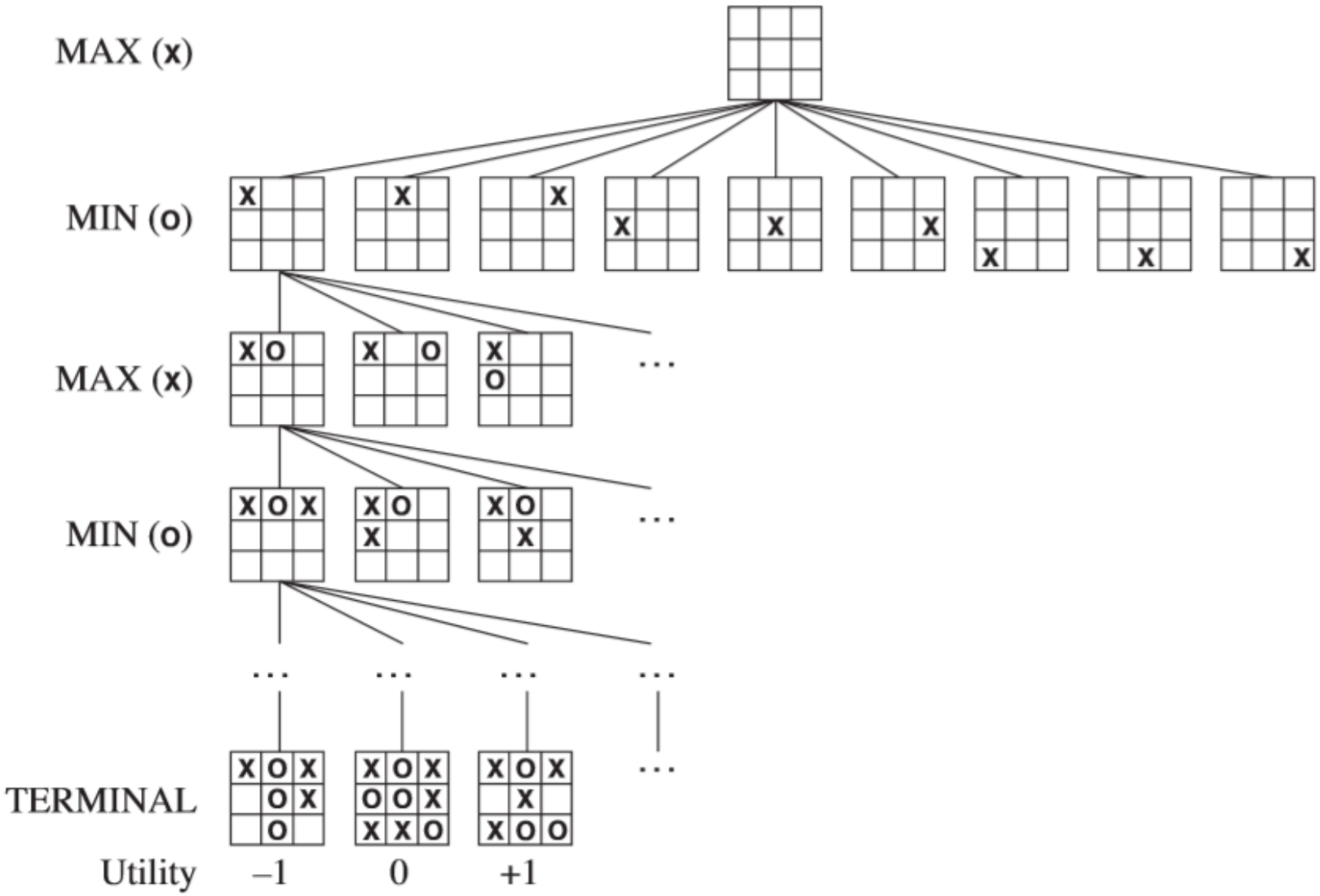
Aversarial Search

A game can be formally defined as a kind of search problem with the following elements:

- S_0: The **initial state**, which specifies how the game is set up at the start.
- PLAYER(s): Defines which player has the move in a state.
- ACTIONS(s): Returns the set of legal moves in a state.
- RESULT(s, a): The **transition model**, which defines the result of a move.
- TERMINAL-TEST(s): A **terminal test**, which is true when the game is over and false otherwise. States where the game has ended are called **terminal states**.
- UTILITY(s, p): A **utility function** (also called an objective function or payoff function), defines the final numeric value for a game that ends in terminal state s for a player p. In chess, the outcome is a win, loss, or draw, with values $+1$, 0, or $\frac{1}{2}$. Some games have a wider variety of possible outcomes; the payoffs in backgammon range from 0 to $+192$. A **zero-sum game** is (confusingly) defined as one where the total payoff to all players is the same for every instance of the game. Chess is zero-sum because every game has payoff of either $0 + 1$, $1 + 0$ or $\frac{1}{2} + \frac{1}{2}$. "Constant-sum" would have been a better term, but zero-sum is traditional and makes sense if you imagine each player is charged an entry fee of $\frac{1}{2}$.

Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

Aversarial Search – Search Tree

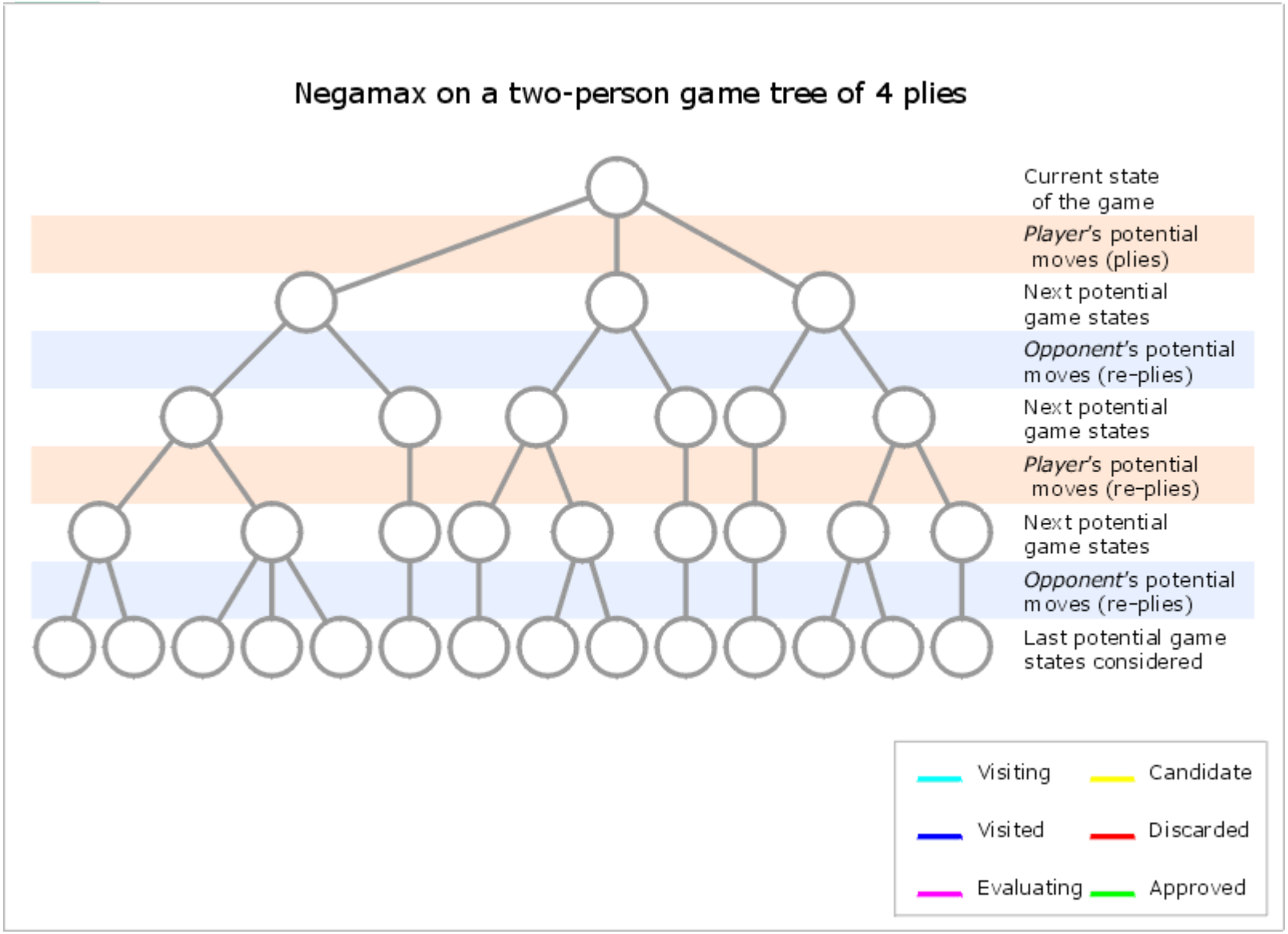


Source: Artificial Intelligence A Modern Approach, Third Edition, S. Russel, P. Norving

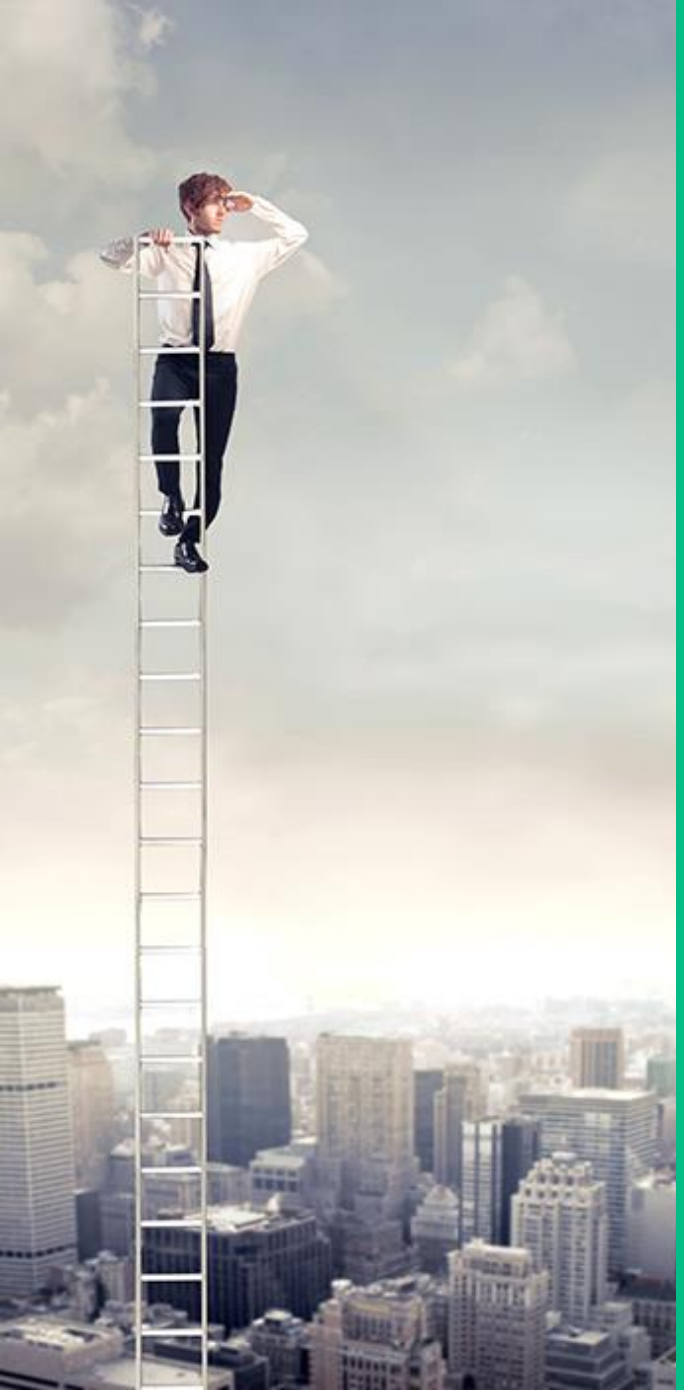
„Jak do tej pory widzę, nie mogłoby być żadnej teorii gier… bez tej teorii… Myślałem, że nic nie było warte publikowania, aż *Teoria* ***Minimax*** została udowodniona”

Source: https://pl.wikipedia.org/wiki/John_von_Neumann#/media/Plik:JohnvonNeumann-LosAlamos.jpg

Source: Von Neumann, J: *Zur Theorie der Gesellschaftsspiele* Math. Annalen. **100** (1928) 295–320.



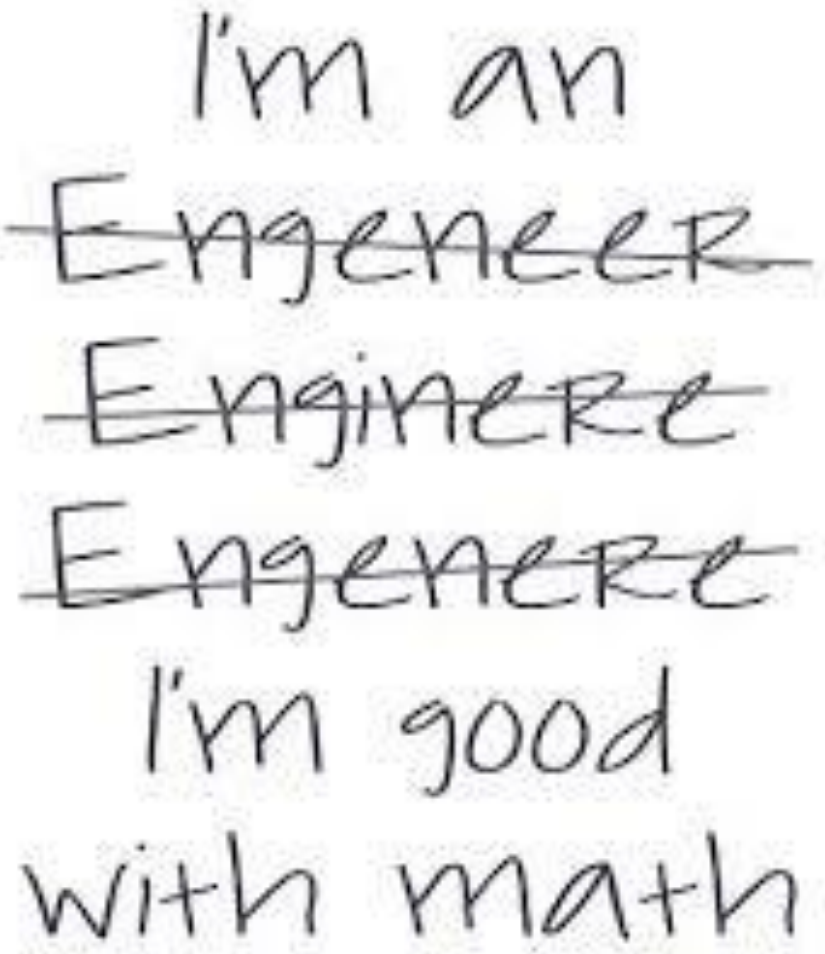
Source: https://en.wikipedia.org/wiki/File:Plain_Negamax.gif



Python Recap

Interacitve console:

>>>


I'm an
~~Engeneer~~
~~Enginere~~
~~Engeneere~~
I'm good
with math

CALCULATOR

Operator	Description	Example	Result
+	addition	5 + 8	13
-	subtraction	90 - 10	80
*	multiplication	4 * 7	28
/	floating point division	7 / 2	3.5
//	integer (truncating) division	7 // 2	3
%	modulus (remainder)	7 % 3	1
**	exponentiation	3 ** 4	81

Source: "Introducing Python. Modern Computing In Simple Packages", B. Lubanovic

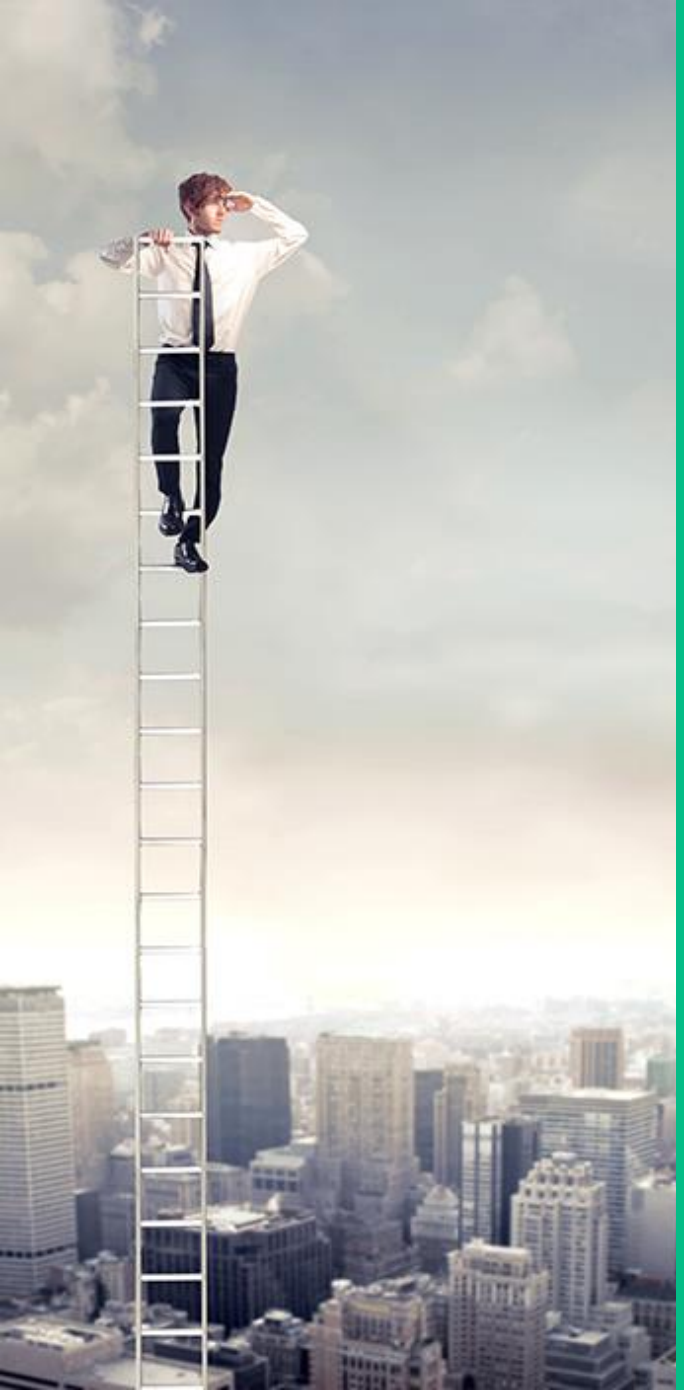
INTRODUCTION TO PYTHON

„Always look on the bright side of life”

CODE COMMENTS

```
# this is the first comment spam = 1
# and this is the second comment
# ... and now a third!


text = "# This is not a comment."
```


REAL Programmers
DON'T COMMENT
their CODE.
If it was
HARD TO WRITE
it should be
HARD
to UNDERSTAND
MemeCenter.com
memecenter.com
MemeCenter



Basic data types

Variables example

```
people = 12
tax = 12.5 / 100
price = 100.50
print (price*tax)
```

STRINGS

Besides numbers, Python can also manipulate strings, which can be expressed in several ways. They can be enclosed in single quotes (’...’) or double quotes ("...") with the same result

```
Text_variable = "my text example"
```

s = ’First line.\nSecond line.’

```
>>> spam = 'Say hi to Bob\'s mother.'
```

Escape Characters

Escape character	Prints as
\'	Single quote
\"	Double quote
\t	Tab
\n	Newline (line break)
\\	Backslash

INPUT

The syntax for input() is

```
input([prompt])
```

where prompt is the string we wish to display on the screen. It is optional.

```
>>> num = input('Enter a number: ')
Enter a number: 10
>>> num
'10'
```

CONVERSION

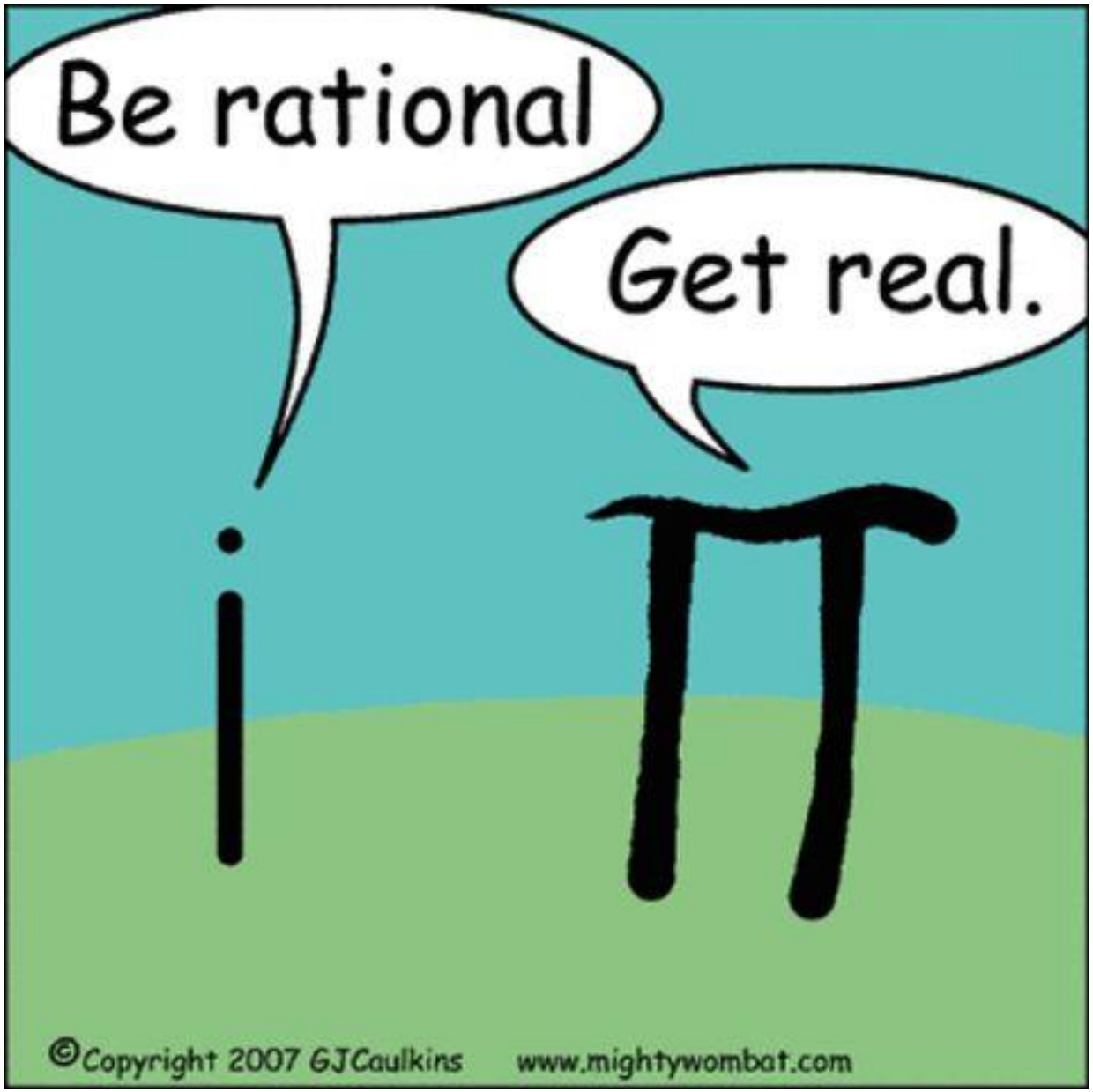
Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use **int()** or **float()** functions.

```
>>> int('10')
10
>>> float('10')
10.0
```

BINARY, OCTAL, HEXADECIMAL

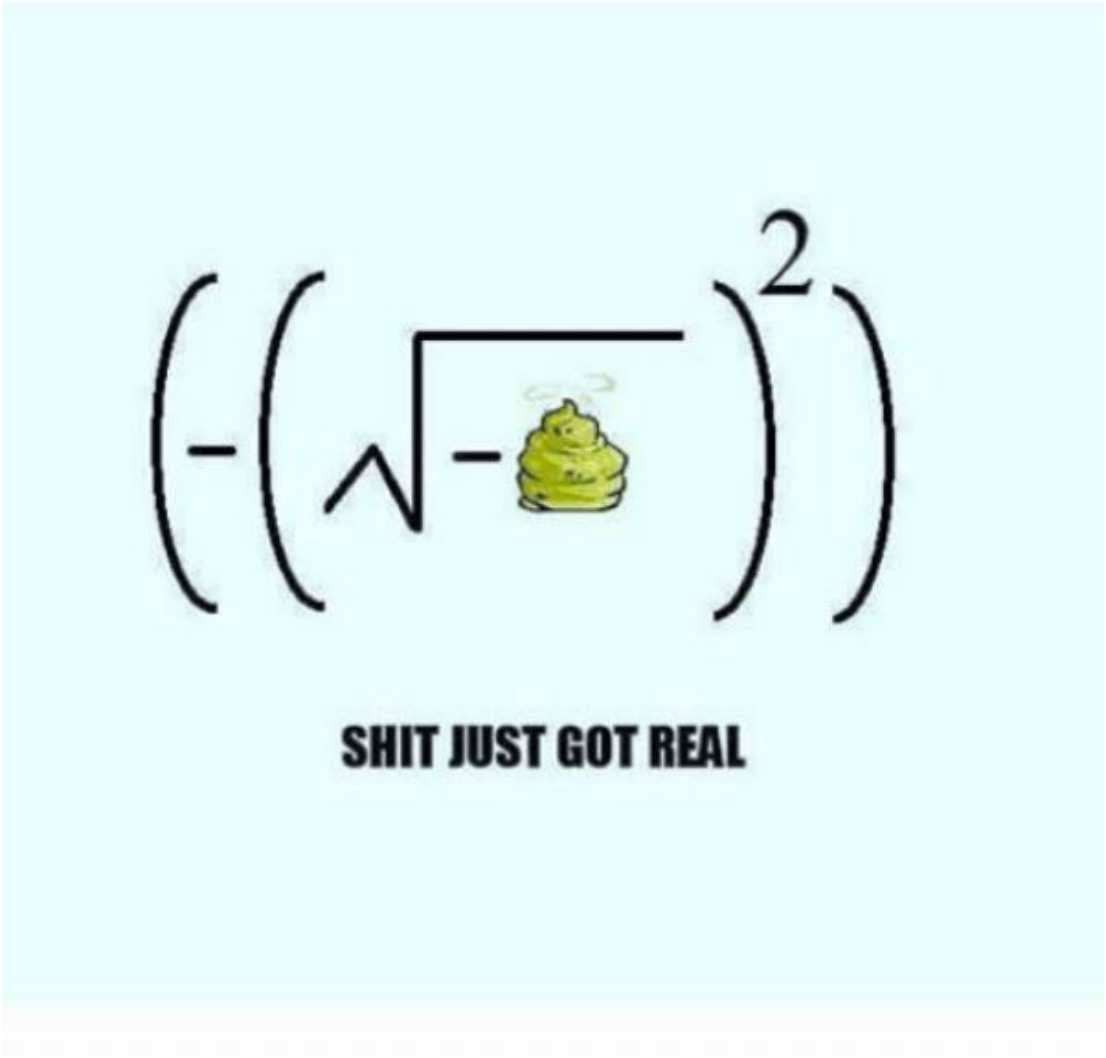
```
number = 0b10      #binary number (0, 1)
print (number)     #2

number = 0o10      #octal number (0 - 7)
print(number)      #8

number = 0x10      #hexadecimal numbers (0-9; A-F)
print(number)      #16


#bin()    - converts number to binary
#oct()    - converts number to octal
#hex()    - converts number to hexadecimal

print(bin(100))       #0b1100100
```

COMPLEX NUMBERS



COMPLEX NUMBERS

```
#example of complex numbers
x = 1 + 1j
y = 2 + 3j

print (x + y)
print (x * y)
```



#engineeringmemes #memes #funny #mechanicalengineering #civilengineering #engineering

CONCATENATE

If you want to concatenate variables or a variable and a literal, use +

```
text = "textA" + "textB"
```

INDEXING

```
word = "Python"
```

Indices may also be negative numbers, to start counting from the right:

```
word[-1]  # last character’n’
word[-2]  # second-last character’o’
word[-6]  #?
```

SLICING

In addition to indexing, slicing is also supported. While indexing is used to obtain individual characters, slicing allows you to obtain substring:

```
word[0:2]
# characters from position 0 (included) to 2
(excluded)
'Py'
 word[2:5]
# characters from position 2 (included) to 5
(excluded)
'tho'
```

Note how the start is always included, and the end always excluded.

STRINGS

Python strings cannot be changed — they are immutable. Therefore, assigning to an indexed position in the string results in an error:

word[0] = ’J’ #ERROR!

EMBEDDING VALUES IN STRINGS

If you want to display a message using the contents of a variable, you can embed values in a string using %s, which is like a marker for a value that you want to add later.

```
>>> myscore = 1000
>>> message = 'I scored %s points'
>>> print(message % myscore)
I scored 1000 points
```

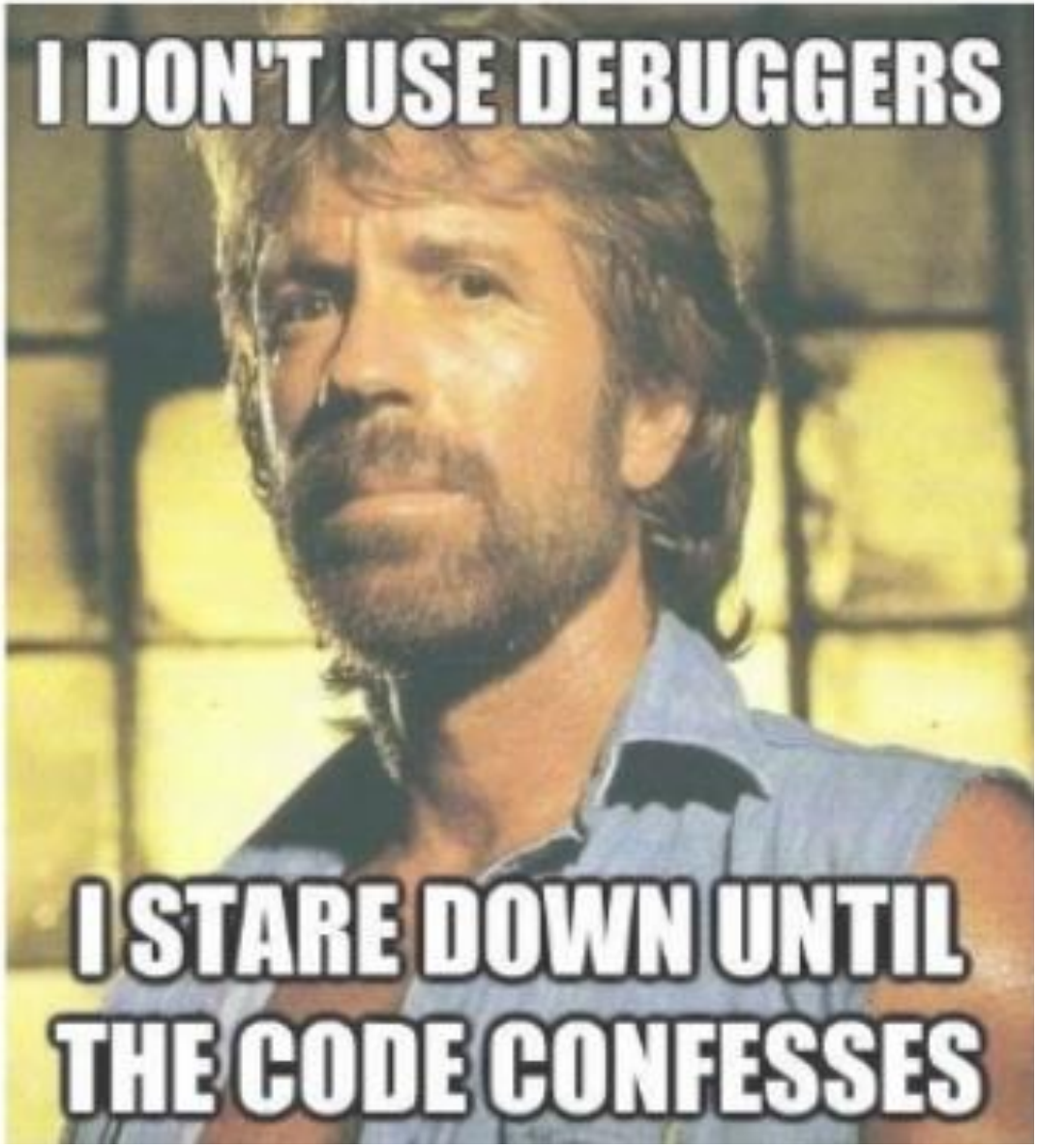
```
>>> joke_text = '%s: a device for finding furniture in the dark'
>>> bodypart1 = 'Knee'
>>> bodypart2 = 'Shin'
>>> print(joke_text % bodypart1)
Knee: a device for finding furniture in the dark
>>> print(joke_text % bodypart2)
Shin: a device for finding furniture in the dark
```

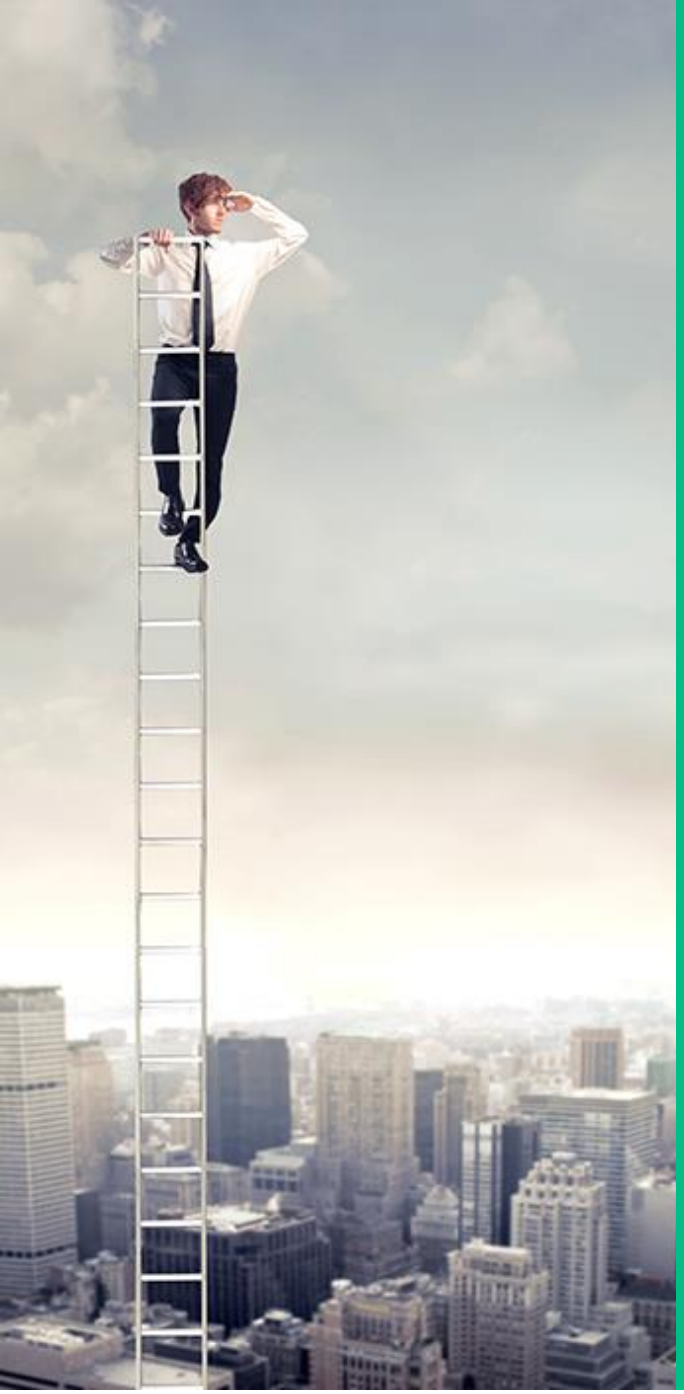
You can also use more than one placeholder in a string, like this:

```
>>> nums = 'What did the number %s say to the number %s? Nice belt!!'
>>> print(nums % (0, 8))
What did the number 0 say to the number 8? Nice belt!!
```

Source: J.R. Briggs, „Python for Kids”

DEBUGGER





Conditions

COMPARISON OPERATORS

The while loop executes as long as the condition (here: b < 10) remains true. In Python, like in C, any non-zero integer value is true; zero is false. The condition may also be a string or list value, in fact any sequence; anything with a non-zero length is true, empty sequences are false.
The standard comparison operators are:

```
<   (less than),
>   (greater than),
==  (equal to),
<=  (less than or equal to),
>=  (greater than or equal to)
!=  (not equal to)
```


fun = 0
if booze:
fun +=
else:
fun -=

IF STATEMENTS

Perhaps the most well-known statement type is the **if statement**. For example:

```
x = int(input("Please enter an integer: "))
if x < 0:
     print('Negative changed to zero')
elif x == 0:
     print('Zero')
elif x == 1:
     print('Single')
else:
     print('More')
```

LOGICAL OPERATOR

`and`	Determines whether both operands are true.	True and True is True True and False is False False and True is False False and False is False
`or`	Determines when one of two operands is true.	True or True is True True or False is True False or True is True False or False is False
`not`	Negates the truth value of a single operand. A true value becomes false and a false value becomes true.	not True is False not False is True