



JÖNKÖPING UNIVERSITY

*School of Engineering*

---

# MODELLING IN PYTHON

**Peter Larsson-Green**

Jönköping University

Autumn 2018

# WHAT IS MODELLING?

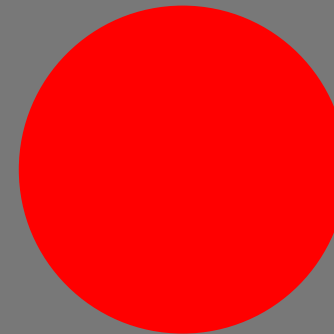
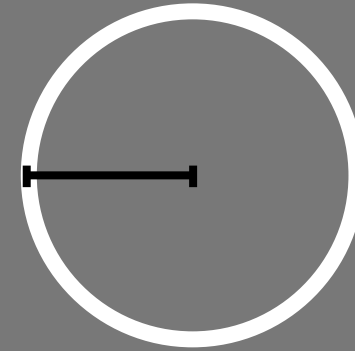
Representing data.

- Represent the age of a human?
  - Use an integer.
  - `age = 10`
- Represent the name of a human?
  - Use a string.
  - `name = "Alice"`
- Represent whether a human is dead or alive?
  - Use a boolean.
  - `alive = True`

# WHAT IS MODELLING?

Representing data.

- Represent a circle?
  - Use an integer.
  - `radius = 10`
- Represent a circle with a color?
  - Use multiple values.
  - `circle_radius = 10`
  - `circle_color = "red"`



# GROUPING DATA

Use dicts to group data together.

```
human_name = "Alice"  
human_age = 10  
human_city = "Atlanta"  
print_human(human_name, human_age, human_city)
```

```
human = {  
    'name': "Alice",  
    'age': 10,  
    'city': "Atlanta"  
}  
print_human(human)
```

# GROUPING DATA

Use dicts to group data together.

```
date_year = 2016
date_month = 3
date_day = 16
print_date(date_year, date_month, date_day)
```

```
date = {
    'year': 2016,
    'month': 3,
    'day': 16
}
print_date(date)
```

# DICTIONARIES VS LISTS

```
human = {  
    'name': "Alice",  
    'age': 10,  
    'city': "Atlanta"  
}
```

```
def print_human(human):  
    print(  
        human['name']+" comes from "+  
        human['city']+" and is "+  
        str(human['age'])+" years old."  
    )
```

```
human = [  
    "Alice",  
    10,  
    "Atlanta"  
]
```

```
def print_human(human):  
    print(  
        human[0]+" comes from "+  
        human[2]+" and is "+  
        str(human[1])+" years old."  
    )
```

# GENERAL GUIDELINES

Use a dict to represent a single entity.

```
human = {  
    'name': "Alice",  
    'age': 10,  
    'city': "Atlanta"  
}
```

```
humans = [  
    {'name': "Alice", 'age': 10, 'city': "Atlanta"},  
    {'name': "Belle", 'age': 15, 'city': "Bilbao"},  
    {'name': "Clair", 'age': 20, 'city': "Chicago"}  
]
```

Use a list to represent a collection of entities.



# PROGRAM STRUCTURE

The data

```
ages = [43, 47, 10, 7, 3]
```

Computations

```
def average(numbers):  
    return sum(numbers)/len(numbers)
```

User Interface

```
print("Average age: "+str(average(ages)))
```

# COMPLEX DATA

```
house = {  
  "city": "Jönköping",  
  "color": "yellow"  
}
```

```
room = {  
  "name": "Living Room",  
  "side-length-1": 5,  
  "side-length-2": 10  
}
```

```
house = {  
  "city": "Jönköping",  
  "color": "yellow",  
  "rooms": [  
    {"name": "Living Room", "side-length-1": 7, "side-length-2": 8},  
    {"name": "Kitchen", "side-length-1": 5, "side-length-2": 5}  
  ]  
}
```

# COMPLEX DATA

```
get_total_area(house)
```

```
def get_total_area(house):  
    area = 0  
    for room in house["rooms"]:  
        area += room["side-length-1"]*room["side-length-2"]  
    return area
```

```
house = {  
    "city": "Jönköping",  
    "color": "yellow",  
    "rooms": [  
        {"name": "Living Room", "side-length-1": 7, "side-length-2": 8},  
        {"name": "Kitchen", "side-length-1": 5, "side-length-2": 5}  
    ]  
}
```

# COMPLEX DATA

```
folder = {  
  "name": "images",  
  "files": [  
    {"name": "me.jpeg", "size": 2048},  
    {"name": "you.png", "size": 4096},  
    {"name": "dad.jpeg", "size": 4096}  
  ]  
}
```

```
folder = {  
  "name": "images"  
}
```

```
file = {  
  "name": "me.jpeg",  
  "size": 2048  
}
```

# COMPLEX DATA

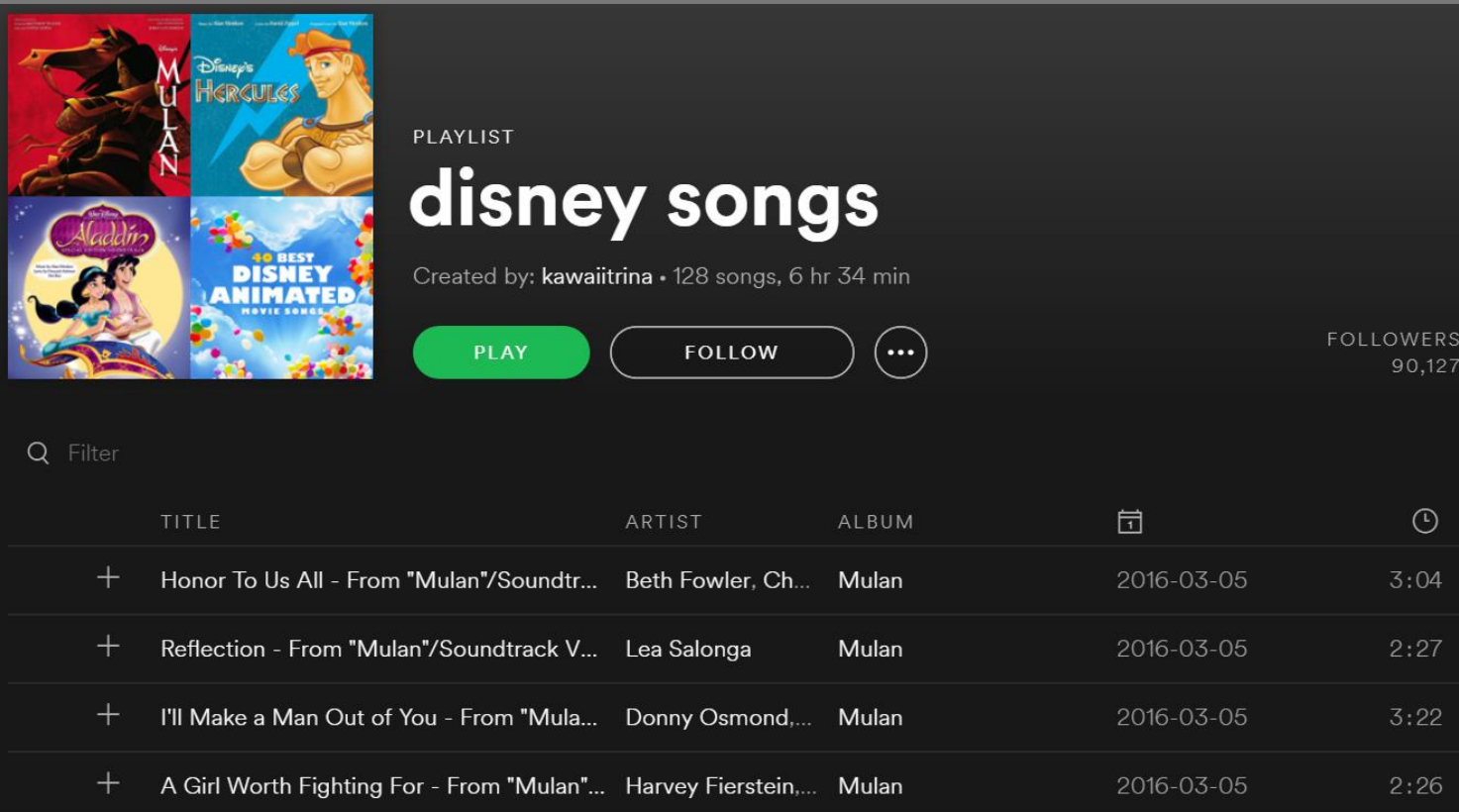
```
get_files_with_extension(folder, ".jpeg")
```

```
folder = {  
    "name": "images",  
    "files": [  
        {"name": "me.jpeg", "size": 2048},  
        {"name": "you.png", "size": 4096},  
        {"name": "dad.jpeg", "size": 4096}  
    ]  
}
```

```
def get_files_with_extension(folder, ext):  
    files = []  
    for file in folder["files"]:  
        if file["name"].endswith(ext):  
            files.append(file)  
    return files
```

# MODELING EXAMPLES

A playlist on Spotify.



The screenshot shows a Spotify playlist interface. At the top left, there are four album covers: Mulan, Disney's Hercules, Aladdin, and a collection of 40 Best Disney Animated Movie Songs. The playlist title is "disney songs" in white text on a dark background. Below the title, it says "Created by: kawaiiitrina • 128 songs, 6 hr 34 min". There are buttons for "PLAY" (green), "FOLLOW", and a menu icon. To the right, it says "FOLLOWERS 90,127". Below the buttons is a search bar with "Filter" and a list of songs. The list has columns for a plus sign, title, artist, album, date, and duration.

	TITLE	ARTIST	ALBUM	📅	🕒
+	Honor To Us All - From "Mulan"/Soundtr...	Beth Fowler, Ch...	Mulan	2016-03-05	3:04
+	Reflection - From "Mulan"/Soundtrack V...	Lea Salonga	Mulan	2016-03-05	2:27
+	I'll Make a Man Out of You - From "Mula...	Donny Osmond,...	Mulan	2016-03-05	3:22
+	A Girl Worth Fighting For - From "Mulan"...	Harvey Fierstein,...	Mulan	2016-03-05	2:26

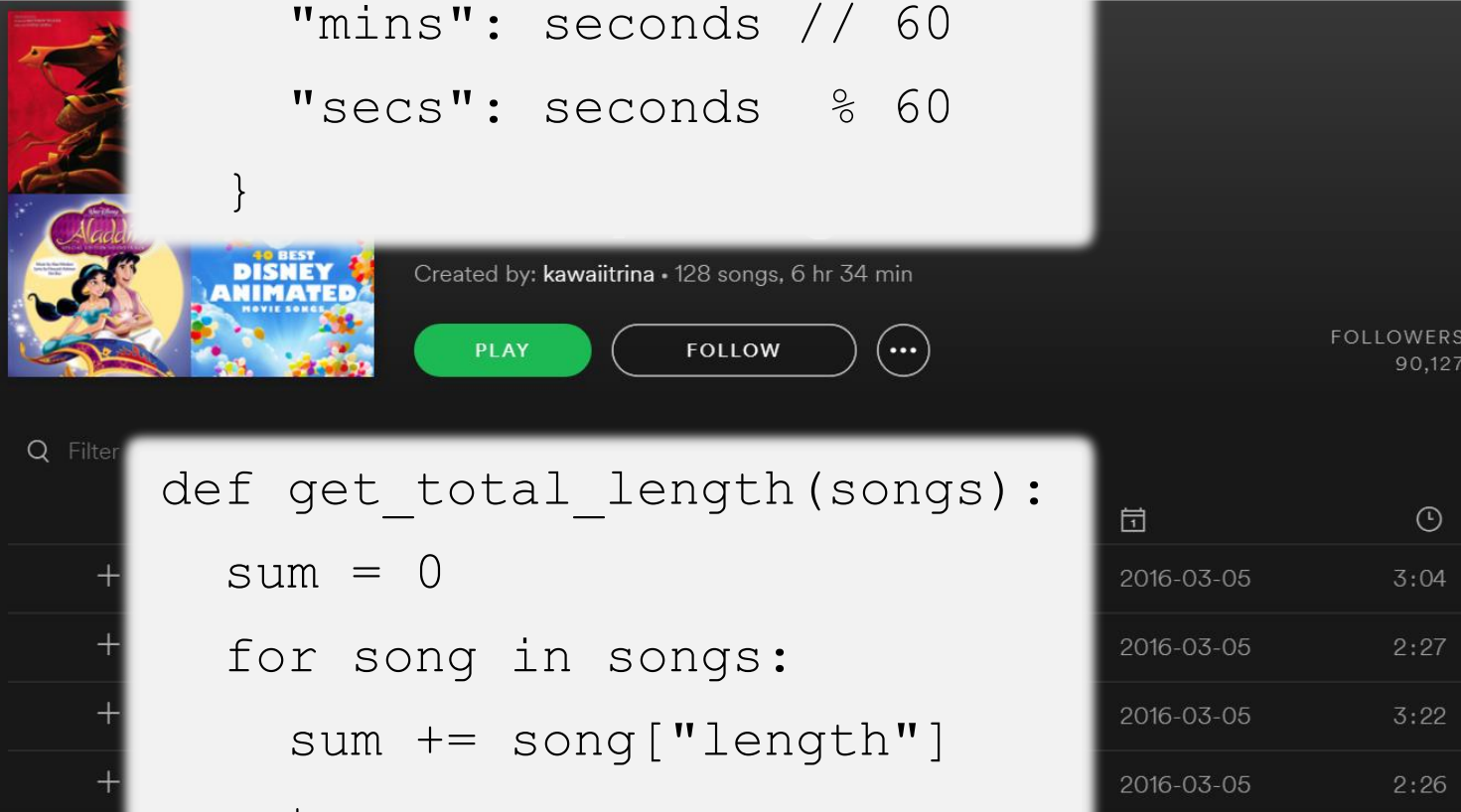
```
songs = [  
  {"title": "Honor To...",  
   "length": 184},  
  {"title": "Reflecti...",  
   "length": 147},  
  {"title": "I'll Mak...",  
   "length": 202},  
  {"title": "A Girl W...",  
   "length": 146},  
  # ...  
]
```

# MODELING EXAMPLES

```
def get_in_units(seconds):  
    return {  
        "mins": seconds // 60  
        "secs": seconds % 60  
    }
```

```
songs = [  
    {"title": "Honor To...",  
     "length": 184},  
    {"title": "Reflecti...",  
     "length": 147},  
    {"title": "I'll Mak...",  
     "length": 202},  
    {"title": "A Girl W...",  
     "length": 146},  
    # ...  
]
```

```
def get_total_length(songs):  
    sum = 0  
    for song in songs:  
        sum += song["length"]  
    return sum
```

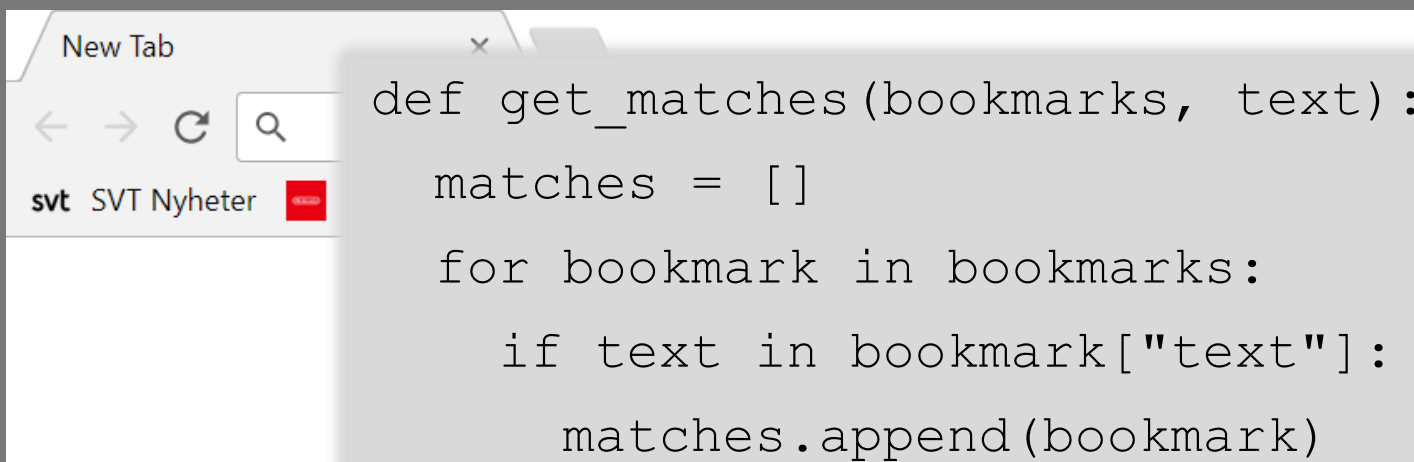


The screenshot shows a Spotify playlist titled "40 BEST DISNEY ANIMATED MOVIE SONGS" created by "kawaiiitrina". It features 128 songs with a total duration of 6 hours and 34 minutes. The interface includes a "PLAY" button, a "FOLLOW" button, and a "FOLLOWERS" count of 90,127. A search filter is visible at the top left. Below the buttons, a list of songs is shown with columns for a date icon, the date "2016-03-05", and a clock icon with the song duration. The visible durations are 3:04, 2:27, 3:22, and 2:26.

Date	Duration
2016-03-05	3:04
2016-03-05	2:27
2016-03-05	3:22
2016-03-05	2:26

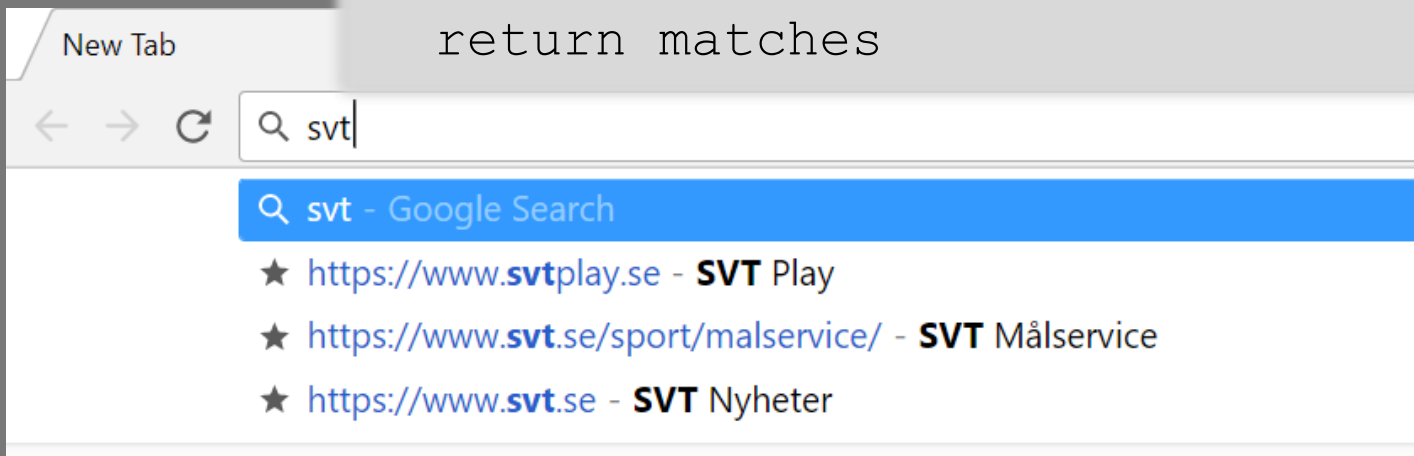
# MODELING EXAMPLES

## Bookmarks in a web browser.



A screenshot of a web browser interface. The address bar shows 'svt SVT Nyheter' with a red icon. Below the address bar, a search bar contains the text 'svt'. A dropdown menu is visible, showing search results for 'svt'.

```
def get_matches(bookmarks, text):  
    matches = []  
    for bookmark in bookmarks:  
        if text in bookmark["text"]:  
            matches.append(bookmark)  
    return matches
```



A screenshot of a web browser interface. The address bar shows 'svt'. Below the address bar, a search bar contains the text 'svt'. A dropdown menu is visible, showing search results for 'svt'.

```
def get_matches(bookmarks, text):  
    matches = []  
    for bookmark in bookmarks:  
        if text in bookmark["text"]:  
            matches.append(bookmark)  
    return matches
```

```
bookmarks = [  
    {"text": "SVT Nyhet...",  
     "url": "https://www..."},  
    {"text": "Nintendo",  
     "url": "https://nin..."},  
    {"text": "SVT Play",  
     "url": "https://www..."},  
    {"text": "Jönköping...",  
     "url": "https://ju.se"},  
    {"text": "SVT Målser...",  
     "url": "https://www..."}  
]
```

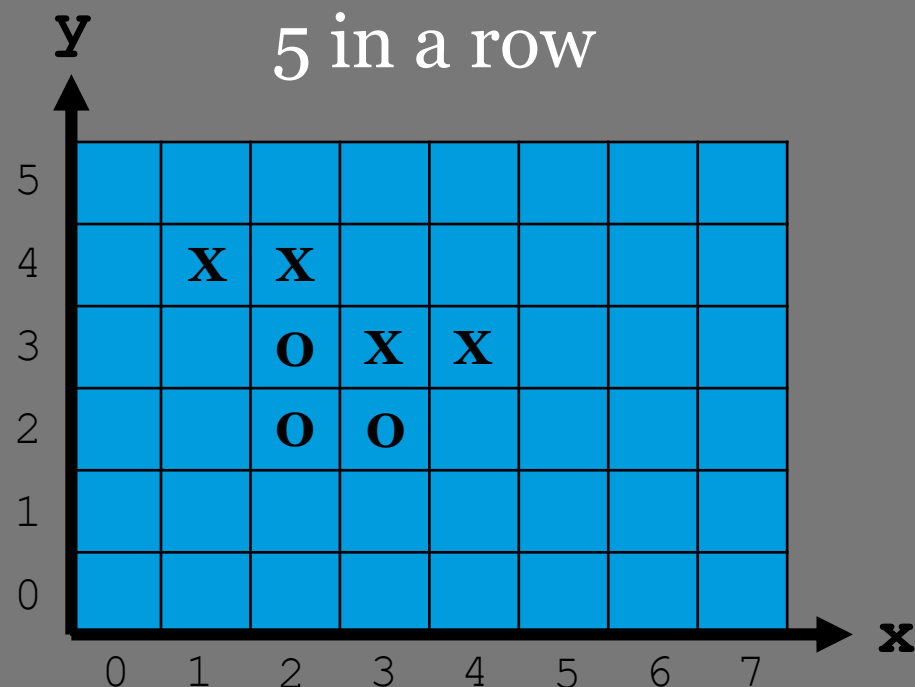


# MODELLING

5 in a row

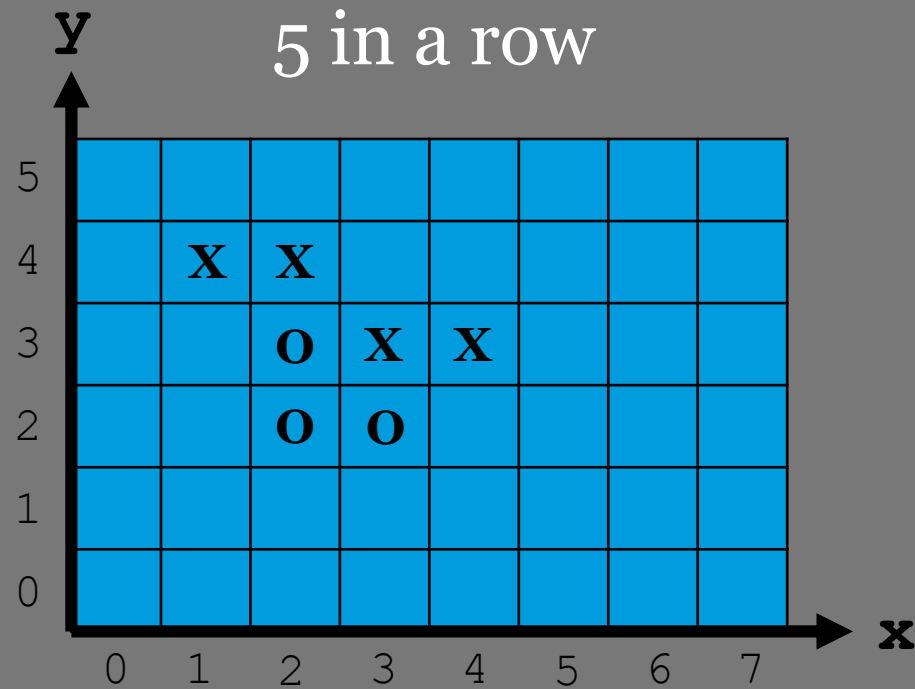
	X	X					
		O	X	X			
		O	O				

# MODELLING



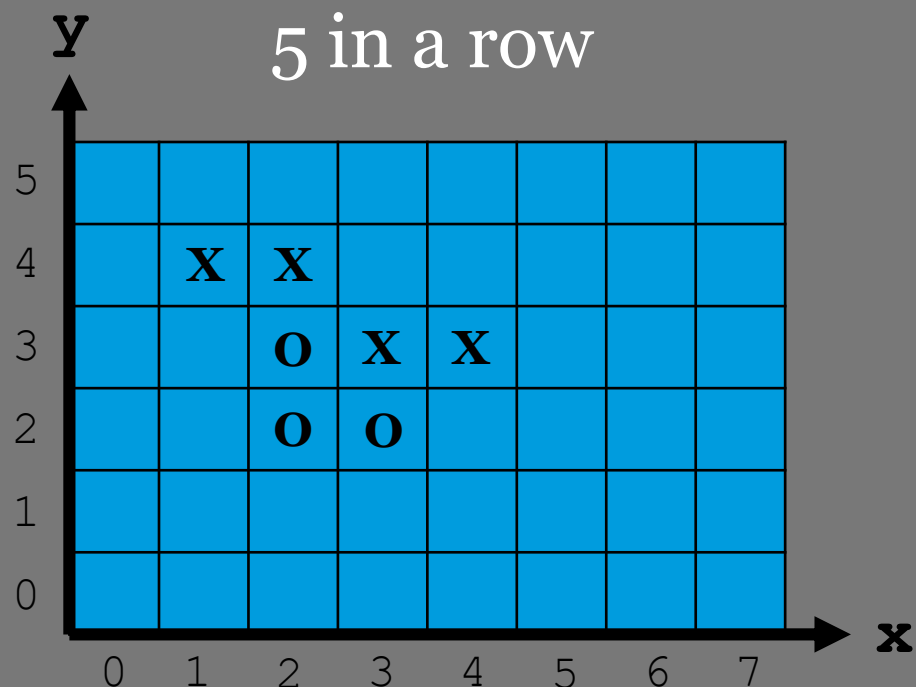
```
game = {  
  'width': 8,  
  'height': 6,  
  'moves': [  
    {'x': 1, 'y': 4, 'player': 'X'},  
    {'x': 2, 'y': 3, 'player': 'O'},  
    {'x': 2, 'y': 4, 'player': 'X'},  
    {'x': 2, 'y': 2, 'player': 'O'},  
    {'x': 3, 'y': 3, 'player': 'X'},  
    {'x': 3, 'y': 2, 'player': 'O'},  
    {'x': 4, 'y': 3, 'player': 'X'}  
  ]  
}
```

# MODELLING



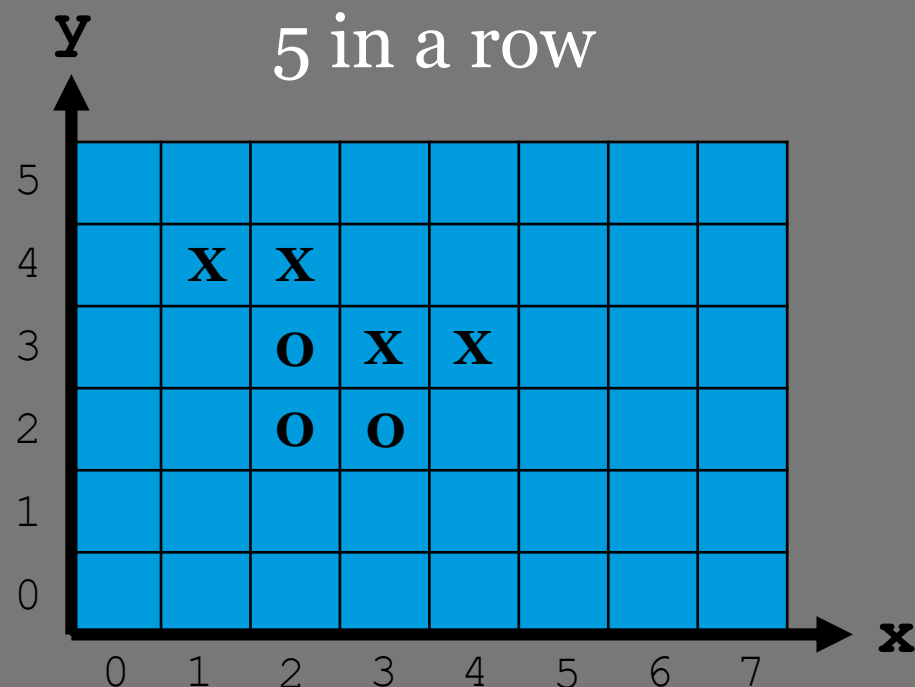
```
game = {  
    'width': 8,  
    'height': 6,  
    'x-moves': [  
        {'x': 1, 'y': 4},  
        {'x': 2, 'y': 4},  
        {'x': 3, 'y': 3},  
        {'x': 4, 'y': 3}  
    ],  
    'o-moves': [  
        {'x': 2, 'y': 3},  
        {'x': 2, 'y': 2},  
        {'x': 3, 'y': 2}  
    ]  
}
```

# MODELLING



```
game = {  
  'width': 8,  
  'height': 6,  
  'board': [  
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],  
    [' ', 'x', 'x', ' ', ' ', ' ', ' ', ' '],  
    [' ', ' ', 'O', 'X', 'X', ' ', ' ', ' '],  
    [' ', ' ', 'O', 'O', ' ', ' ', ' ', ' '],  
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],  
    [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']  
  ]  
}
```

# MODELLING



```
game = {  
    'width': 8,  
    'height': 6,  
    'board': {  
        (1, 4): 'X',  
        (2, 3): 'O',  
        (2, 4): 'X',  
        (2, 2): 'O',  
        (3, 3): 'X',  
        (3, 2): 'O',  
        (4, 3): 'X'  
    }  
}
```