Name:	Date:

Lists and Animations

Film Projection

Some time ago, films were long tapes wound onto large reels. The tapes were run through a projector. Thanks to a powerful lamp the film was projected onto a screen.

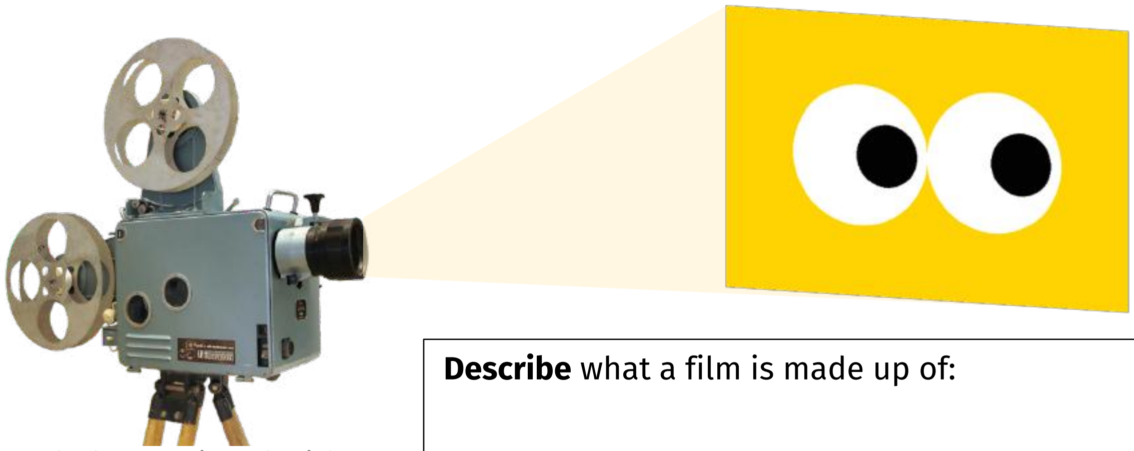


Photo by konstantin rotkevich from Pixabay

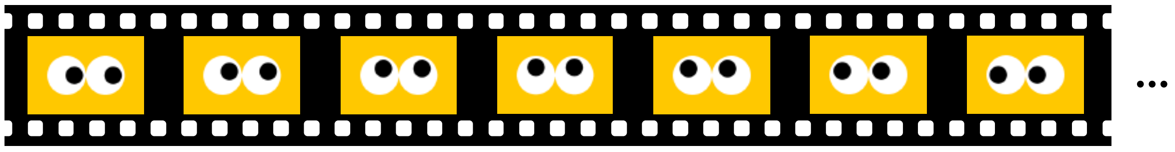
Describe what a film is made up of:

Here is a film:

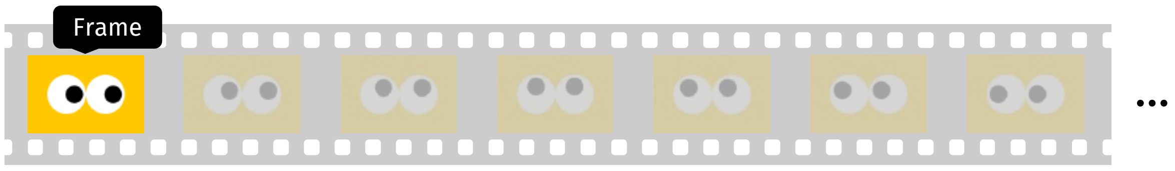
Photo by Denise Jans on Unsplash

The Cinematographic Film

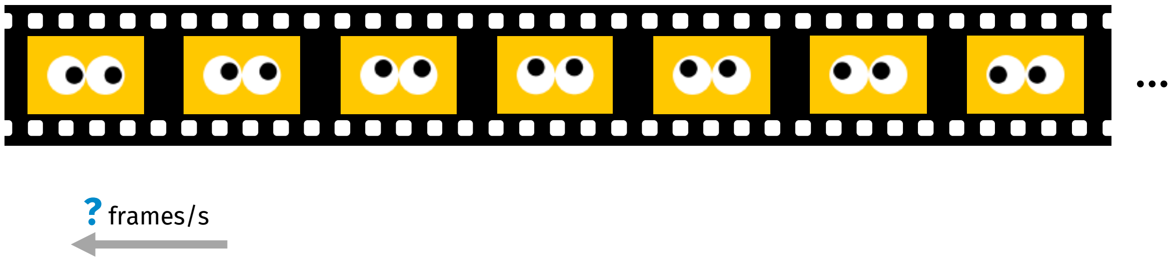
Here is a short tape representing rolling eyes:



The tape is composed of many frames:



When projecting a film, the tape moves within the projector, from frame to frame, stopping on each for a short time:



If the film moved through the projector stopping at each frame for 10 seconds, it would have a speed of 1 frame/10 s, which corresponds to 0.1 frame/s. If we had 10 seconds to watch each frame, we would be able to distinguish individual images from each other; the problem is that the result would be closer to a slide presentation than a movie.

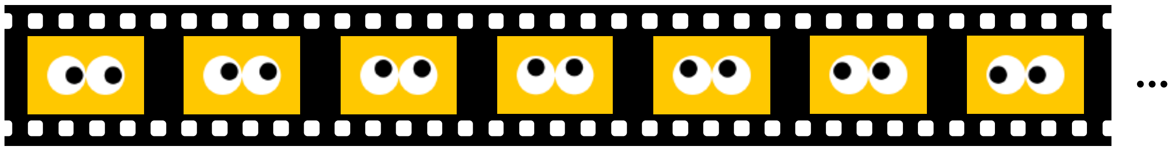
If, on the other hand, the film moved fast enough, we would not be able to distinguish individual images, and so we would get a projected film.

Estimate: how many frames per second do we need to perceive it fluidly?

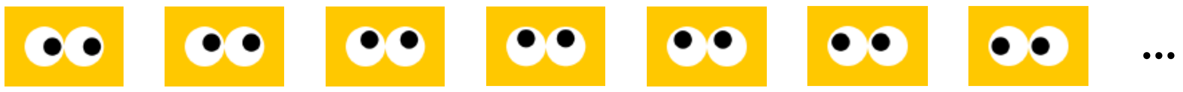
How many frames per second are shown on a gaming computer to provide an even smoother experience?

Composing a Graphic: Rolling Eyes

What are the similarities and differences in the various frames of our film?

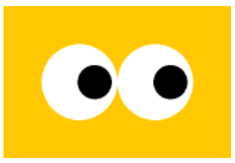
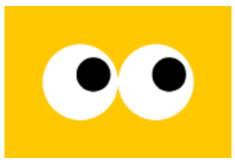
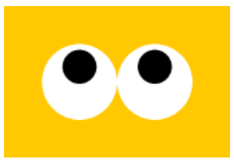


Now let's focus only on the frames, removing the black border:



Similarities	Differences

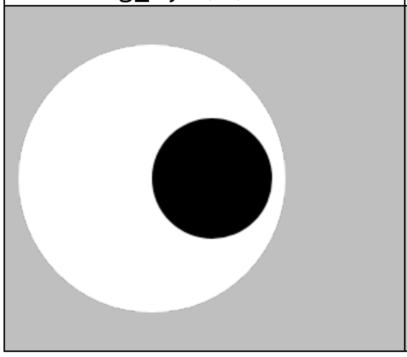
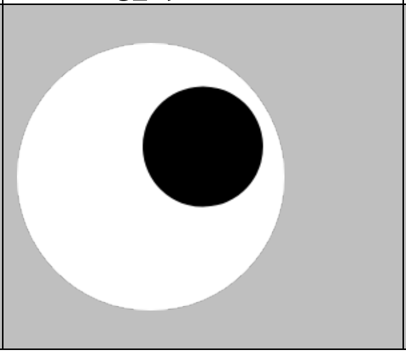
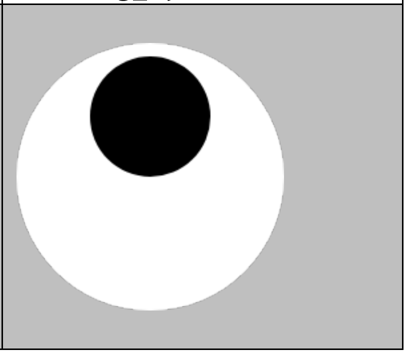
Let's create a function that can produce a pair of rolling eyes, given the angle of rotation. Here are examples:

`framed_eyes(0)`	`framed_eyes(30)`	`framed_eyes(90)`
		

Describe how you would decompose the first graphic (where the angle is 0):

We decompose the graphic into a yellow background and two eyes.
Each eye consists of a white sclera and a black pupil.

Building an eye is not easy: the pupil must be “pushed” to the side and rotated.
We start by creating a function called `rolling_eye` to produce a single eye.

`rolling_eye(0)`	`rolling_eye(30)`	`rolling_eye(90)`
		

Define two graphics, `pupil` and `sclera`, assuming the sclera is white and 200 in diameter. The pupil is black and 90 in diameter.
Then, **use Tamaro Cards to design** the `rolling_eye` function. The function must have one parameter, the rotation angle. Within the function, use the definitions of the two graphics `pupil` and `sclera`.

Write the Python code for the `rolling_eye(rotation_angle)` function you just programmed with Tamaro Cards:

```
pupil =
sclera =
def
```

We create a new function `framed_eyes(rotation_angle)` to draw an entire frame with Tamaro Cards. The yellow background rectangle has the dimensions 600 and 400. Remember to call the `rolling_eye` function to produce an eye.

Draw the results of the following function calls:

`framed_eyes(270)`	`framed_eyes(720)`

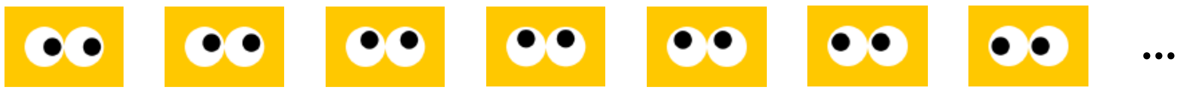
Now translate into Python code:

```
def framed_eyes(rotation_angle):
```

What will happen if you call the function `framed_eyes(-180)`?

Representing a Film with a List of Graphics

To represent a film, we can represent the various frames as a **list**.



A frame is just a graphic. Please note: all frames in a film **must be the same size**!

We defined a `framed_eyes` function to produce a frame.
Call the function seven times to produce the sequence you see above:

```
frame0 = framed_eyes(
frame1 = framed_eyes(
```

In the PyTamaro library with `show_graphic` a graphic is shown as output.

How to display an animation?
PyTamaro has a function for such purpose: `show_animation`!

In an animation there are many graphics, not just one. Some animations are very long and may consist of thousands of graphics. Others, however, are very short, with only a few graphics.

The `show_animation` function does not show only one, two, or three graphics. This function displays a **list** of graphics. But how do we create a list of graphics?

In the Python language we can create a list by listing elements, separated by a comma and in square brackets [..., ..., ..., ...].

Here are some examples of lists:

A list containing `True` and `False`	A list of the first three natural numbers	A list of the angles in a circle that are a multiple of 90	An empty list
`[True, False]`	`[0, 1, 2]`	`[0, 90, 180, 270]`	`[]`

Create a list of the seven frames:

Using the previously defined constants:	Not using the previously defined constants, but by calling the function `framed_eyes` each time:

`[`
`  frame0,`
`  frame1,`
`  frame2,`
`  frame3,`
`  frame4,`
`  frame5,`
`  frame6`
`]`
	`[`

We are ready to display our animation!
Call the `show_animation` function and pass a list of graphics (the frames we would like to see) as an argument.
Complete the code:

```
show_animation(





)
```

If you run this code, along with the `rolling_eye` function and all the other necessary functions on the PyTamaro site, you should be able to see an animated GIF of-the rolling eyes! ("*Occhi Roteanti*" activity).

You could create a longer list, rotating the angle in smaller steps (instead of 30° steps, for example 5° steps). This would produce an animation with slower, smoother eye rotation.

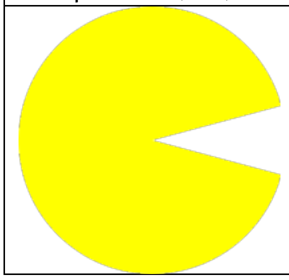
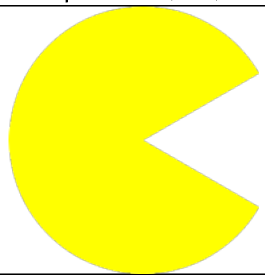
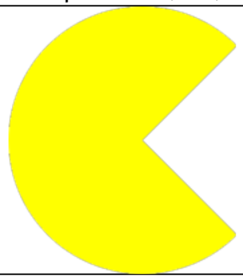
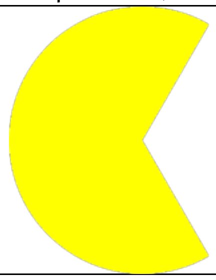
You can download your animation as a GIF and save it to your computer!

You can use Python and PyTamaro to produce lots of custom GIFs...

Animated Pac-Man

Let's try animating Pac-Man: we create a small film (an animation) with Pac-Man opening and closing his mouth.

In the previous unit we created a function called `pacman` that produces Pac-Man with a variable mouth opening:

`pacman(30)`	`pacman(60)`	`pacman(90)`	`pacman(120)`
			

Note: Each frame in a movie must be the **same size**! But Pac-Man graphics have different dimensions. In particular, they have different widths!

To get around this problem, **define** a function that creates a Pac-Man overlaid on a fixed base with equal size for all frames.
Use a 200x200 square as the background (exactly equal to the diameter of Pac-Man).

```
def pacman_frame(mouth_angle):
  return
```

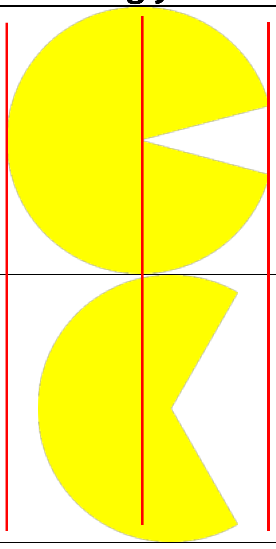
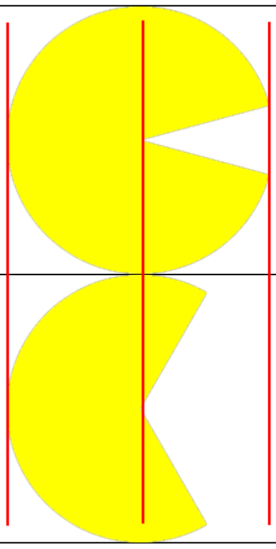
Now call `show_animation` to animate the four Pac-Man frames shown above. ("*Animazione Pac-Man*" activity)

```

```

Looking at the animation, it appears that Pac-Man moves as he opens his mouth.

You probably used `overlay` to create the frame.
The `overlay` function places the Pac-Man graphic in the center of the background, but in doing so the center point of the circular sector does not remain fixed in the center of the background:

Ugly	Nice
	

Rewrite your `pacman_frame` function so that it consistently aligns Pac-Man's head against the background:

```
def pacman_frame(mouth_angle):
  return
```

Animation Speed

The basic speed setting of a film in `show_animation` is 25 frames/s.

1000 ms / 25 frames

......   ms / 1 frame

The `show_animation` function has a second “secret” parameter, where you can specify how long you want to view each frame.
If this “secret” parameter is not specified, `show_animation` shows each frame for .... ms.

Write a call to `show_animation` that produces the film at double speed:

If you want a smoother animation (as in video games on the PC), you can set the speed to 60 frames/s (1 frame every 16.7 milliseconds or so).
Try varying the speed on the PyTamaro site and see what happens.

Warning: if you want Pac-Man to open his mouth with the same speed, you have to create twice as many frames.

The Animated Traffic Light

What are the four phases of a traffic light in Switzerland?

green			

Let's program an animation of a traffic light that goes through these four phases.

Use the `traffic_light` function created in Unit 7. It has three parameters: the first determines the color of the first light at the top, the second determines the color of the middle light, and the third determines the color of the light at the bottom. For simplicity, it uses the colors black, yellow, green and red.
Create a list of four graphics corresponding to the four phases of the traffic light. We give this list a name (`frames`), so that we can use this name whenever we need it. Finally, call `show_animation`.

```
frames = [









]

show_animazione(
```

Now, try the "*Animazione Semaforo*" activity on the PyTamaro web site!

To have a slow animation, how many milliseconds per frame would be needed?