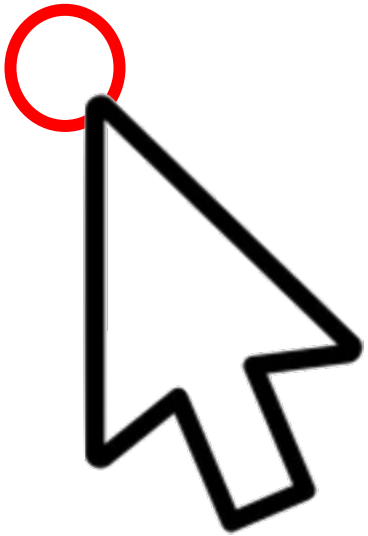
El Mouse

4 de Julio, 2022

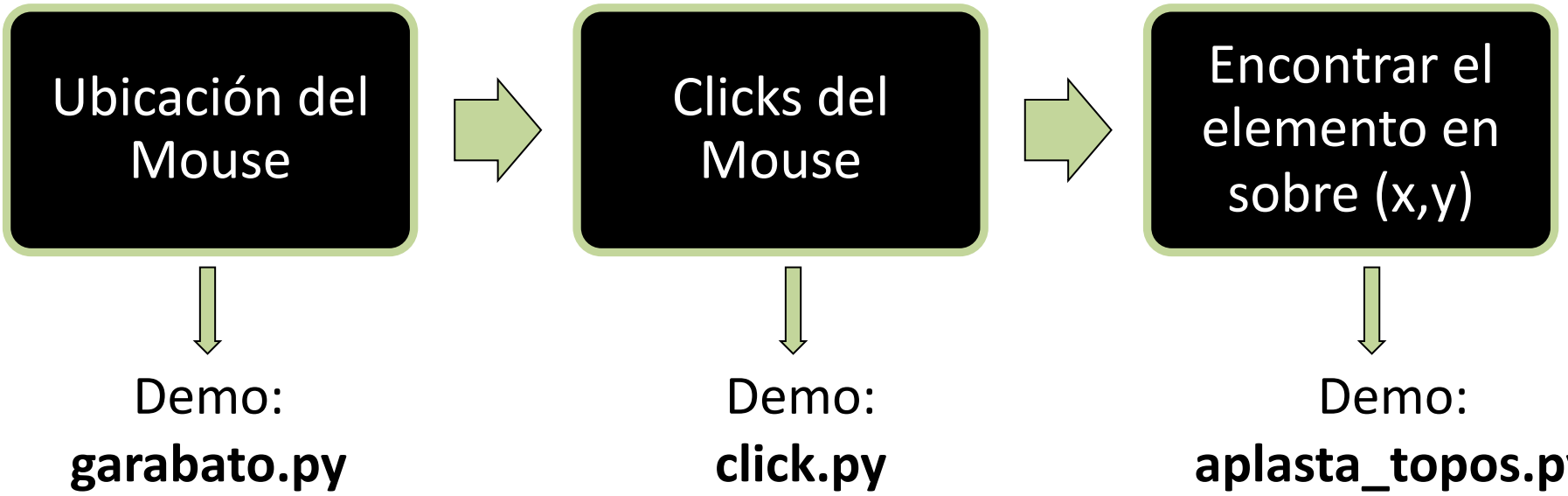
CS Bridge Clase 11

Metas de Aprendizaje

Aprender a responder a eventos del mouse en programas gráficos.



Plan de Clase



Repaso: Listas

Plan de Clase

Repaso: Listas

Lista

Definición

Una lista es una forma de mantener un registro de una **colección ordenada** de **ítems**

Colección: la lista puede **contener múltiples elementos**

Ordenada: puede referirse a los elementos por su **posición**

Los ítems de la lista se llaman "**elementos**".

Lista

Definición

Una lista es una forma de mantener un registro de una **colección ordenada** de **ítems**

La lista **ajusta dinámicamente su tamaño** a medida que se añaden o eliminan elementos

Las listas tienen **muchas funciones integ**radas para facilitar su uso

Lists

len(lista) – devuelve el tamaño de la lista

Lists

list.append(elem) – añade un elemento al final

Lists

list[i] – devuelve el elemento i

Lists

list[i] = elem – asigna *elem* al elemento i

Lists

list.insert(i, elem) – inserta *elem* en el índice i

Lists

list.remove(elem) – borra la primera instancia de elem

Lists

list.count(elem) – cuenta el número de instancias de elem

Lists

list.pop(i) – devuelve y borra el elemento i

Lists

del list[i] – borra el elemento en el indice i

Lists

list.clear() – borra todos los elementos de la lista

Lists

list.index(elem) – devuelve el índice de *elem*

Recorrer todos los Elementos

```
mi_lista = ['Leia', 'Luke', 'Han']

for i in range(len(mi_lista)):
    elemento = mi_lista[i]
    ...
```

Recorrer todos los Elementos

```
mi_lista = ['Leia', 'Luke', 'Han']

for i in range(len(mi_lista )):
    elem = mi_lista [i]
    print(elem)
```

Output:

```
Leia
Luke
Han
```

Recorrer todos los Elementos

```
mi_lista = ['Leia', 'Luke', 'Han']

for i in range(len(mi_lista )):
    elem = mi_lista [i]
    print(elem)

-----

for elem in mi_lista :
    print(elem)
```

Output:

```
Leia
Luke
Han
```

Recorrer todos los Elementos

```
mi_lista = ['Leia', 'Luke', 'Han']

for i in range(len(mi_lista )):
    elem = mi_lista [i]
    print(elem)
```

Output:

```
Leia
Luke
Han
```

```
-----

for elem in mi_lista :
    print(elem)
   # no hay variable i para utilizar
```

Recorrer todos los Elementos

mi_lista:

```
Leia
Luke
Han
```

```
for elem in mi_lista:
    print(elem)
```

Iteración 1 del for:

```
elem = 'Leia'
```

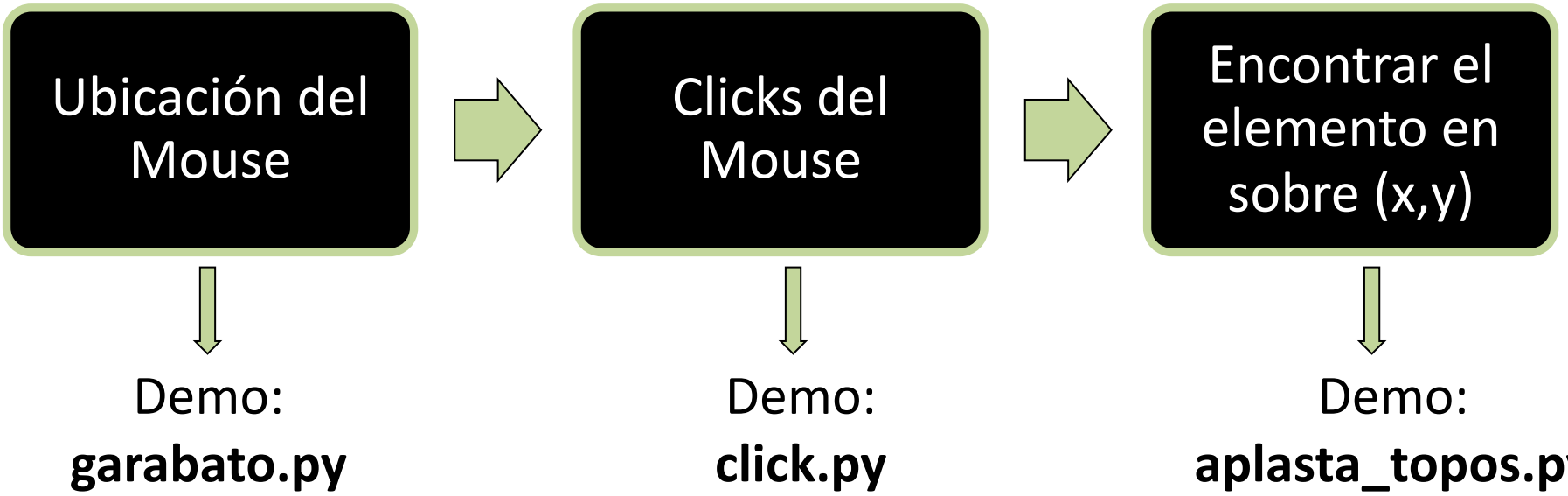
Iteración 2 del for:

```
elem = 'Luke'
```

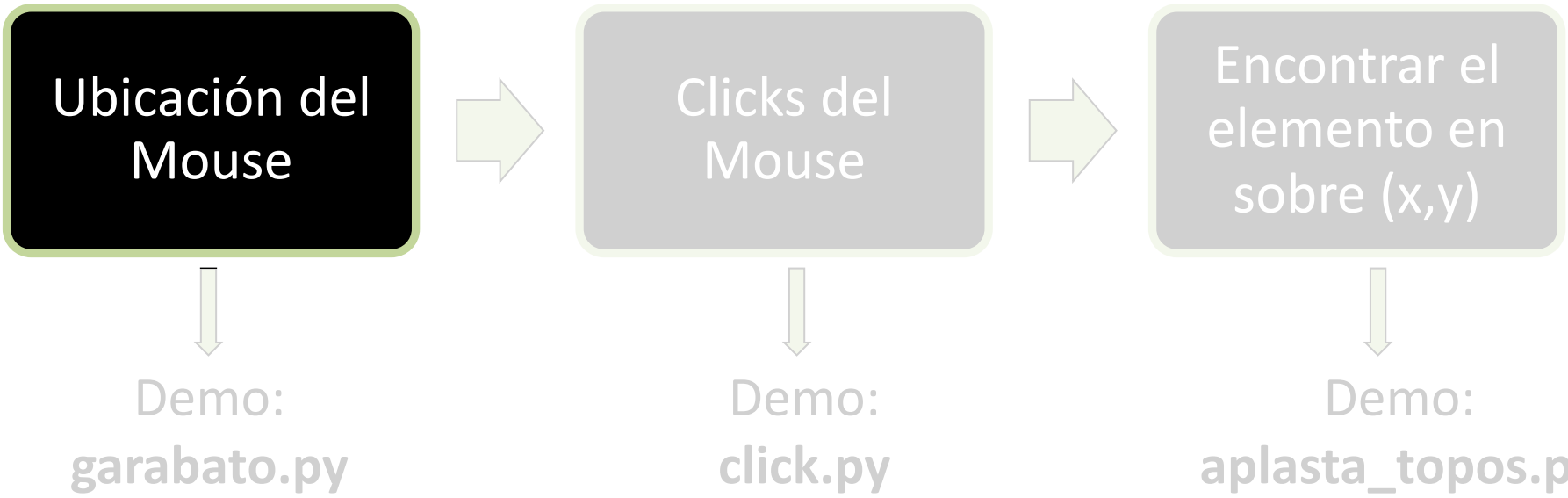
Iteración 3 del for:

```
elem = 'Han'
```

Plan de Clase

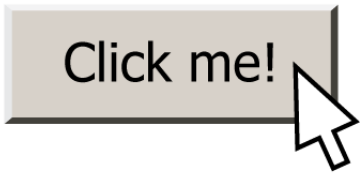


Plan de Clase



Respondiendo al Mouse

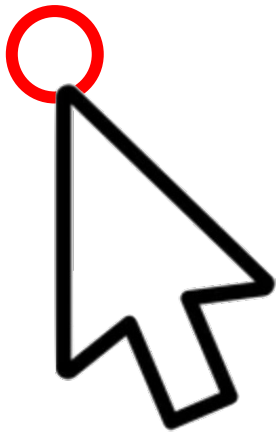
evento: algún estímulo externo al que tiene que responder tu programa



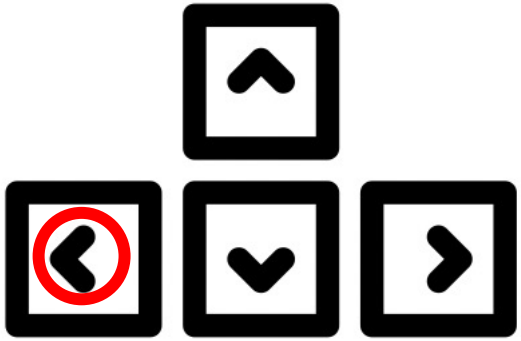
Respondiendo al Mouse

evento: algún estímulo externo al que tiene que responder tu programa

Ejemplos:

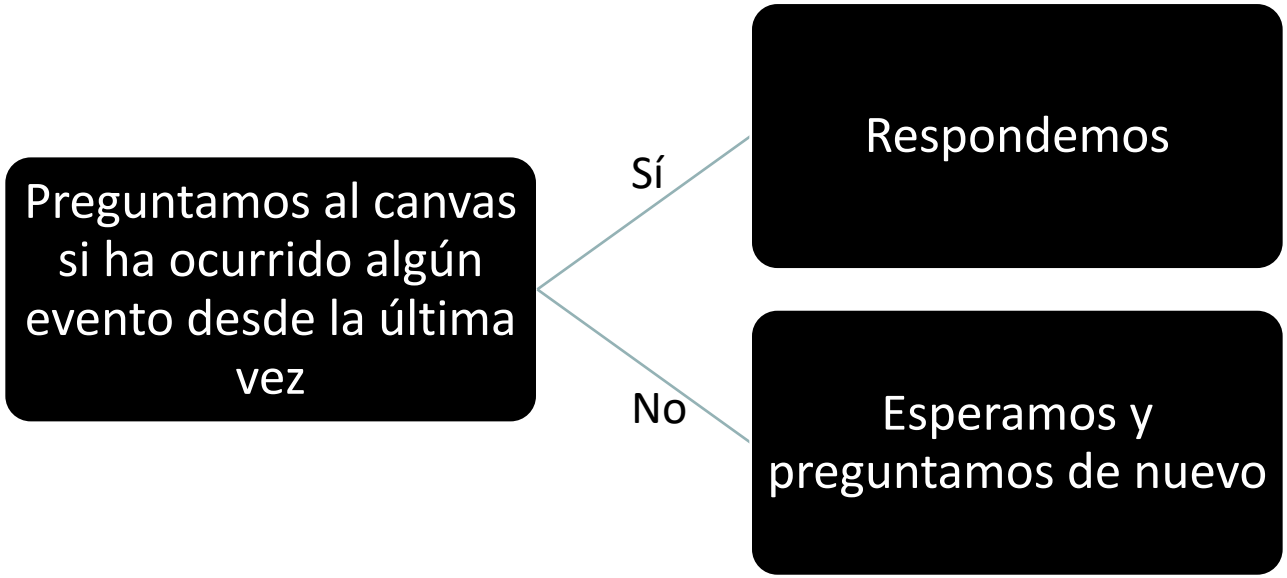


Click del mouse



Presión de una tecla

Eventos




```
while True:
      # Encargarse de cualquier evento
      # ...
      canvas.update()
```

Ubicación del Mouse

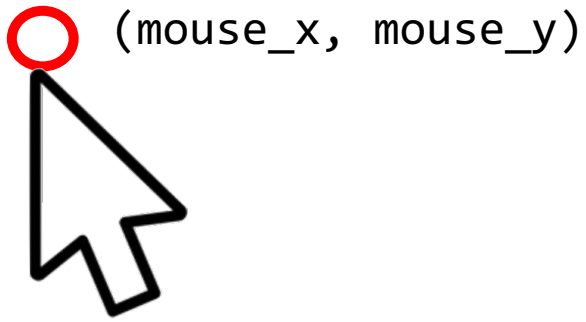
En cualquier momento, le podemos pedir al canvas la ubicación del mouse en ese momento

```
mouse_x = canvas.get_mouse_x()
mouse_y = canvas.get_mouse_y()
```

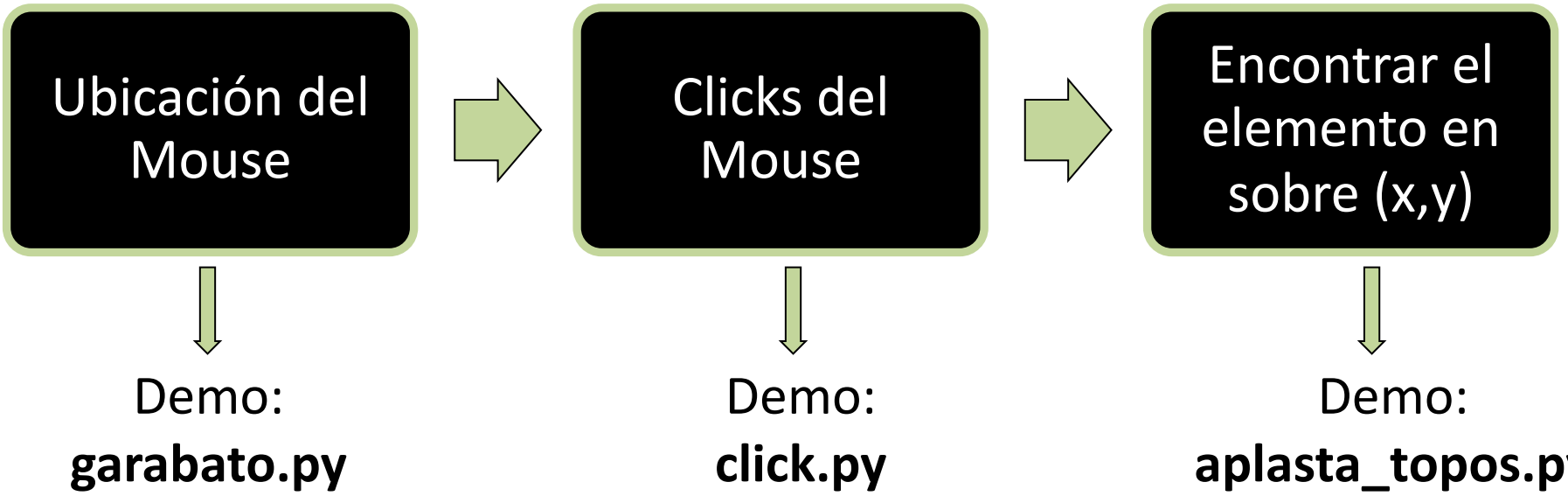
Ubicación del Mouse

En cualquier momento, le podemos pedir al canvas la ubicación del mouse en ese momento

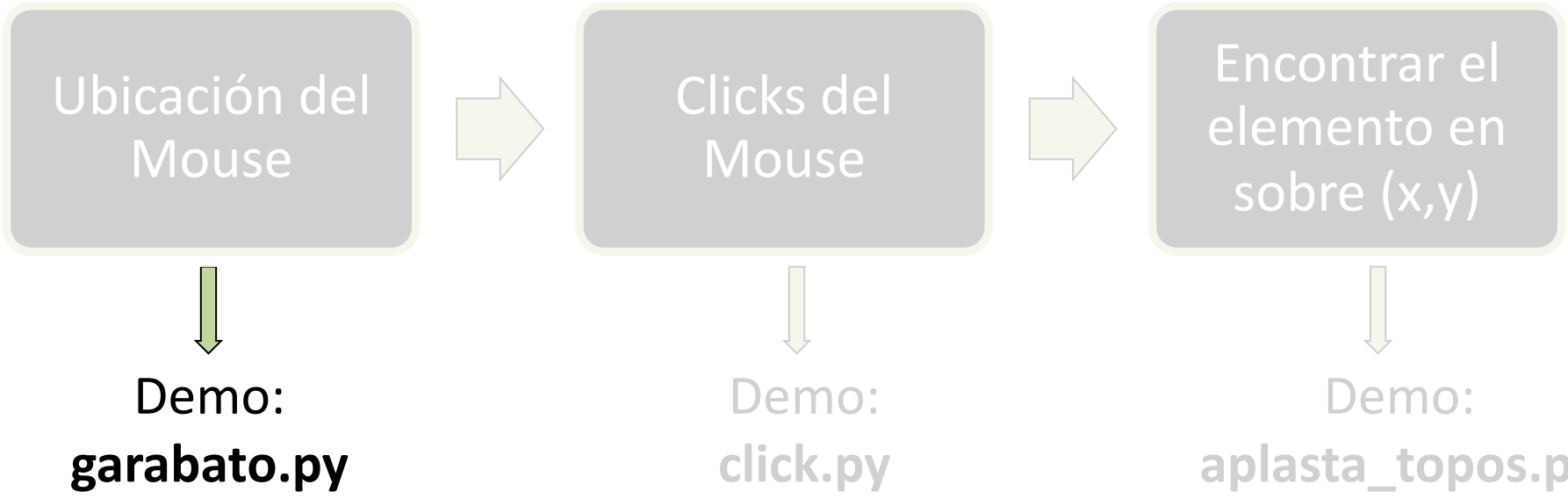
```
mouse_x = canvas.get_mouse_x()
mouse_y = canvas.get_mouse_y()
```



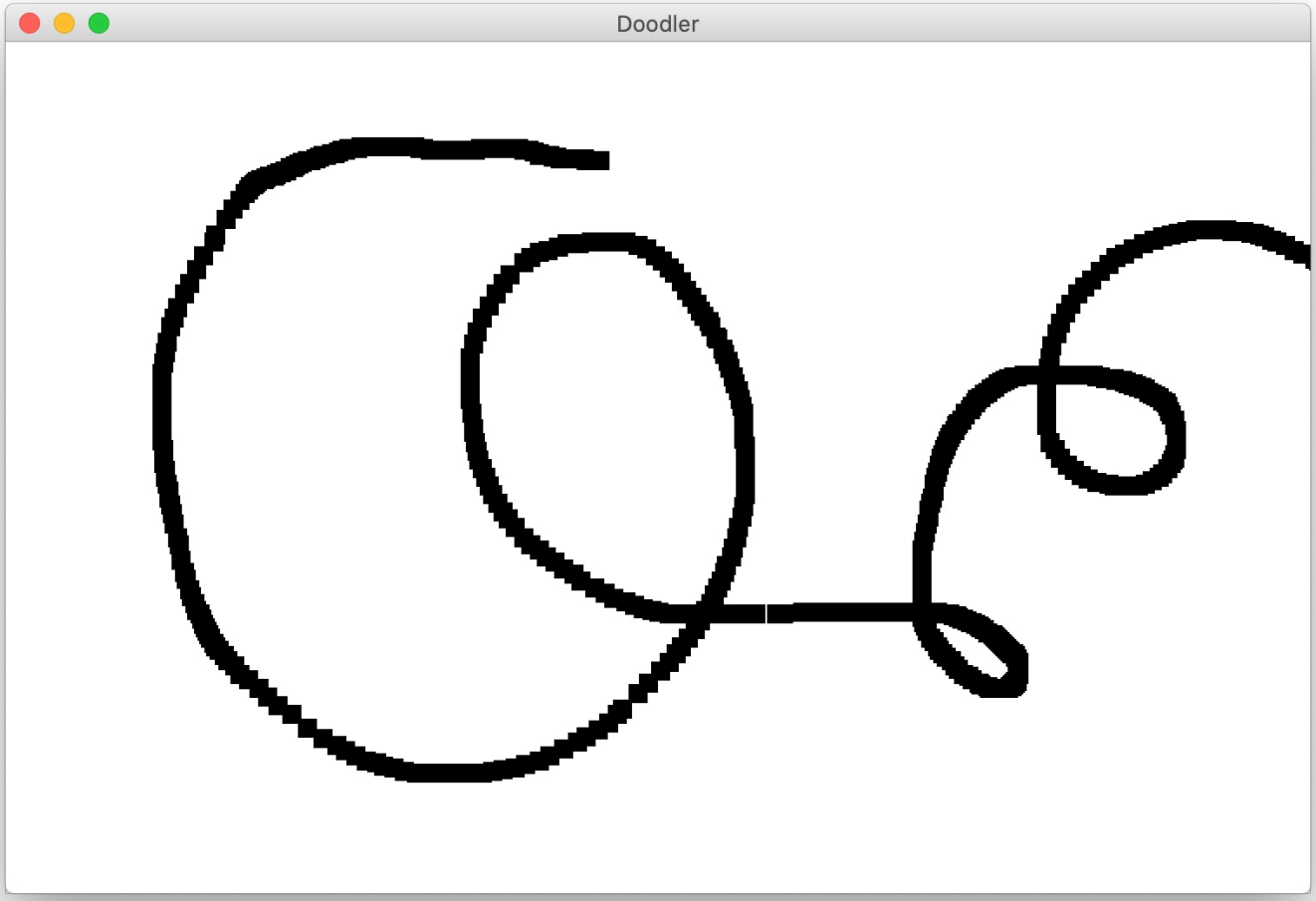
Plan de Clase



Plan de Clase



Garabato



Garabato

```
TAMAÑO_CUADRADO = 10
...

while True:
    # Pide la ubicación del mouse
    mouse_x = canvas.get_mouse_x()
    mouse_y = canvas.get_mouse_y()

    # Crea un rectángulo negro en esa localización
    rect = canvas.create_rectangle(mouse_x, mouse_y,
                                   mouse_x + TAMAÑO_CUADRADO,
                                   mouse_y + TAMAÑO_CUADRADO)
    canvas.set_color(rect, 'black')

    canvas.update()
```

Garabato

```
TAMAÑO_CUADRADO = 10
...

while True:
    # Pide la ubicación del mouse
    mouse_x = canvas.get_mouse_x()
    mouse_y = canvas.get_mouse_y()

    # Create a black rectangle at this location
    rect = canvas.create_rectangle(mouse_x, mouse_y,
                                   mouse_x + TAMAÑO_CUADRADO,
                                   mouse_y + TAMAÑO_CUADRADO
 canvas.set_color(rect, 'black')

    canvas.update()
```

Garabato

```
TAMAÑO_CUADRADO = 10
...

while True:
    # Get the mouse location
    mouse_x = canvas.get_mouse_x()
    mouse_y = canvas.get_mouse_y()

   # Crea un rectángulo negro en esa localización
  rect = canvas.create_rectangle(mouse_x, mouse_y,
                                 mouse_x + TAMAÑO_CUADRADO,
                                 mouse_y + TAMAÑO_CUADRADO)
    canvas.set_color(rect, 'black')

    canvas.update()
```

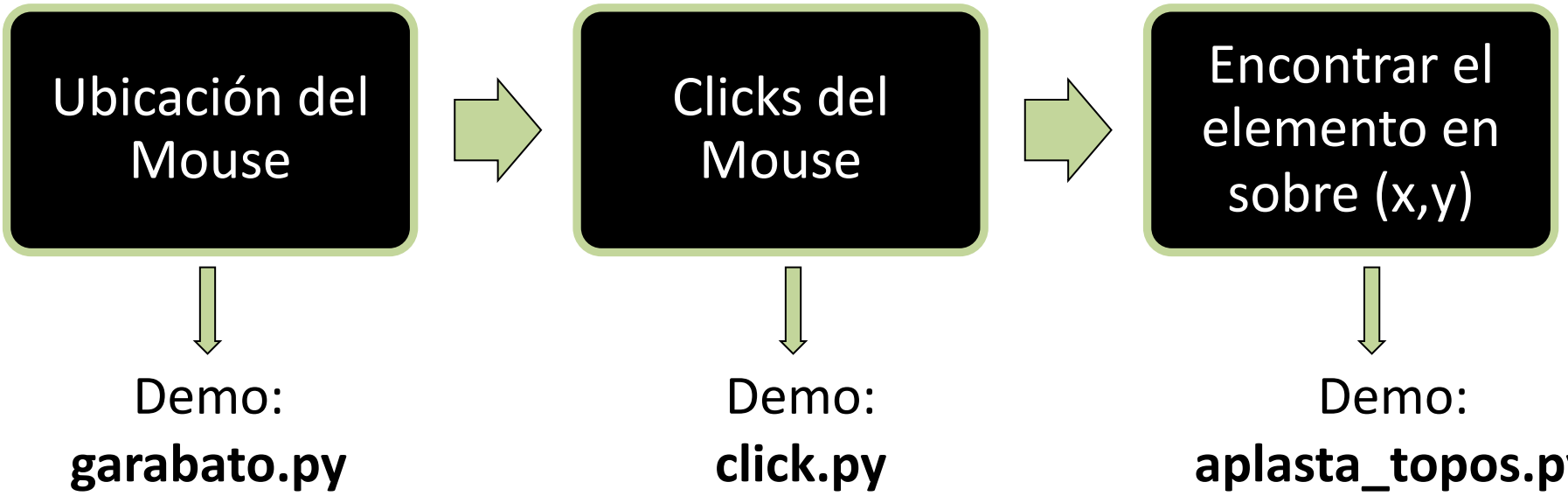
Garabato

```
TAMAÑO_CUADRADO = 10
...

while True:
    # Pide la ubicación del mouse
     mouse_x = canvas.get_mouse_x()
    mouse_y = canvas.get_mouse_y()

    # Crea un rectángulo negro en esa localización
    rect = canvas.create_rectangle(mouse_x, mouse_y,
                                   mouse_x + TAMAÑO_CUADRADO,
                                   mouse_y + TAMAÑO_CUADRADO)
    canvas.set_color(rect, 'black')

    canvas.update()
```

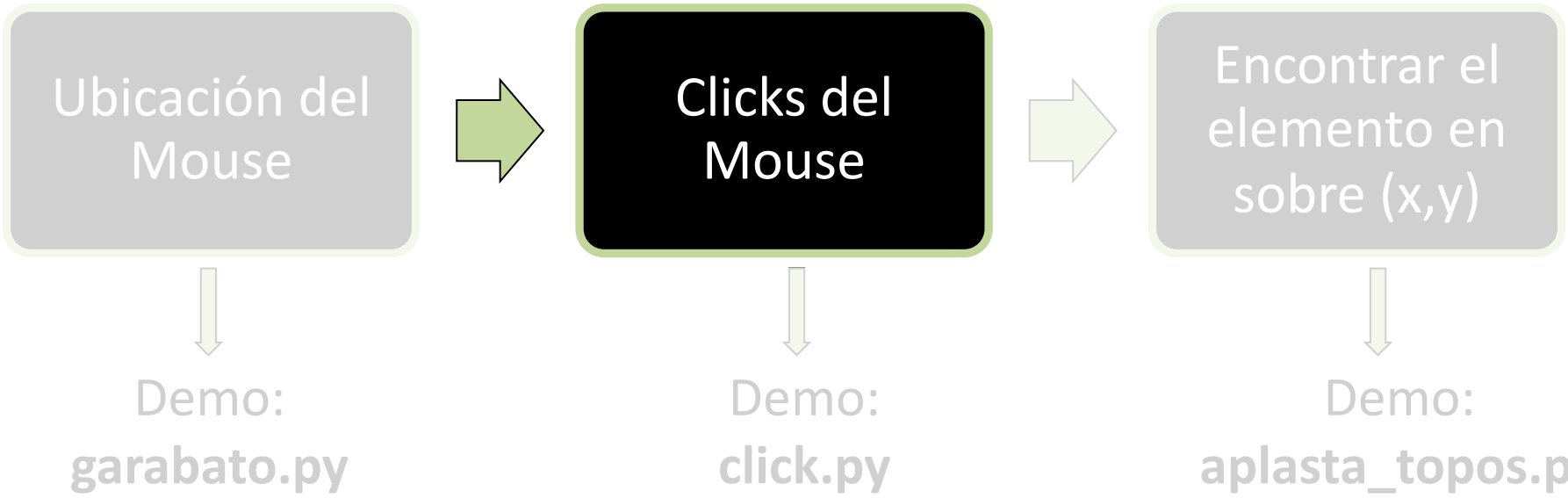
Garabato

```
TAMAÑO_CUADRADO= 10
...

while True:
    # Pide la ubicación del mouse
     mouse_x = canvas.get_mouse_x()
    mouse_y = canvas.get_mouse_y()

    # Crea un rectángulo negro en esa localización
    rect = canvas.create_rectangle(mouse_x, mouse_y,
                                   mouse_x + TAMAÑO_CUADRADO,
                                   mouse_y + TAMAÑO_CUADRADO)
    canvas.set_color(rect, 'black')

    canvas.update()
```

Plan de Clase



Plan de Clase



Clicks del Mouse

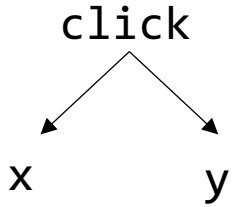
En cualquier momento, le podemos pedir al canvas una lista de los clicks del mouse que hayan tenido lugar, desde la última vez que preguntamos

```
clicks = canvas.get_new_mouse_clicks()
```

Clicks del Mouse

Cada elemento en la lista tiene una coordenada **x** y **y** que indica donde tuvo lugar el click

```
clicks = canvas.get_new_mouse_clicks()

for click in clicks:
    print(click.x, click.y)
```



Recorrer todos los Elementos

mi_lista:

```
click1
click2
click3
```

```
for click in clicks:
    print(click.x, click.y)
```

Iteración 1 del for:

```
click = click1
click.x = click1.x
click.y = click1.y
```

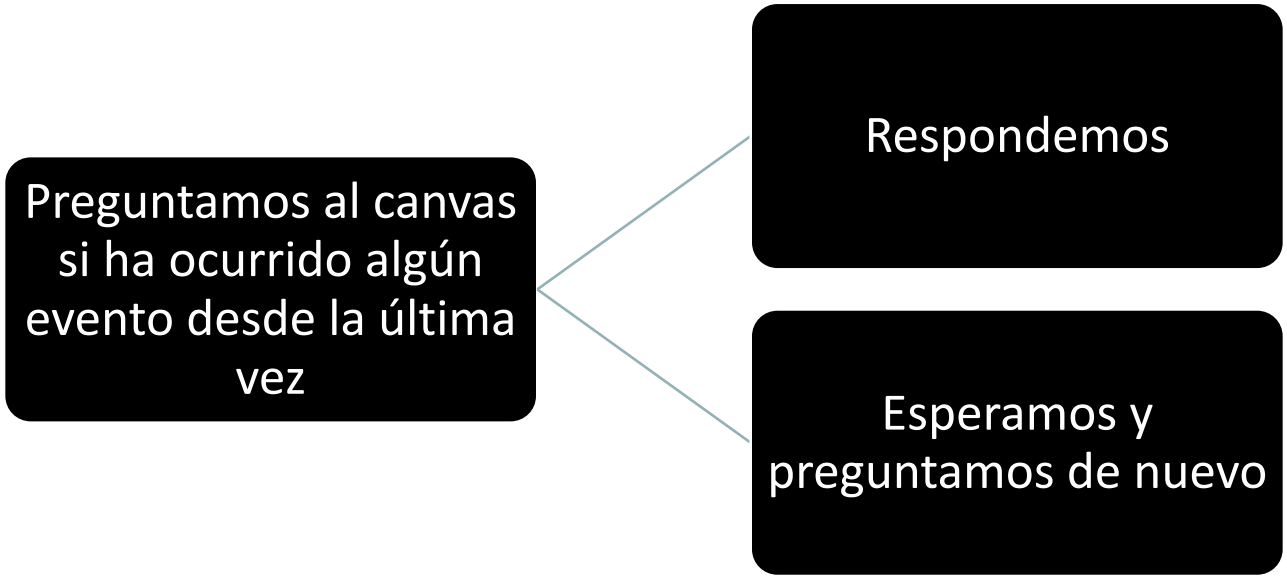
Iteración 2 del for:

```
click = click2
click.x = click2.x
click.y = click2.y
```

Iteración 3 del for:

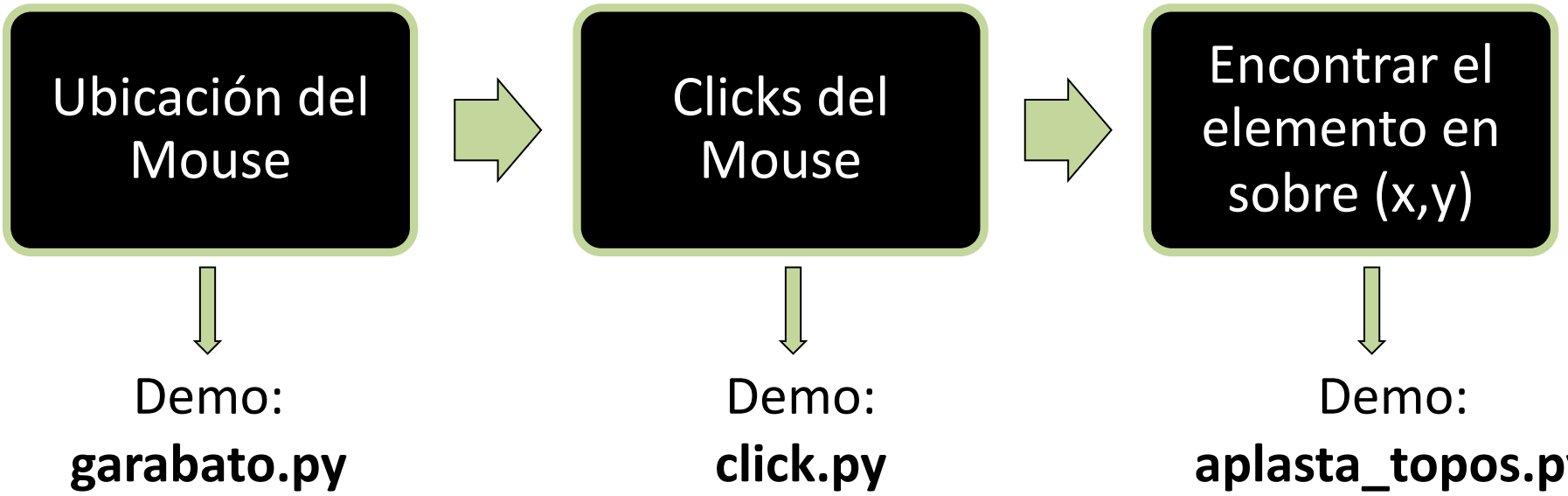
```
click = click3
click.x = click1.x
click.y = click1.y
```

Eventos

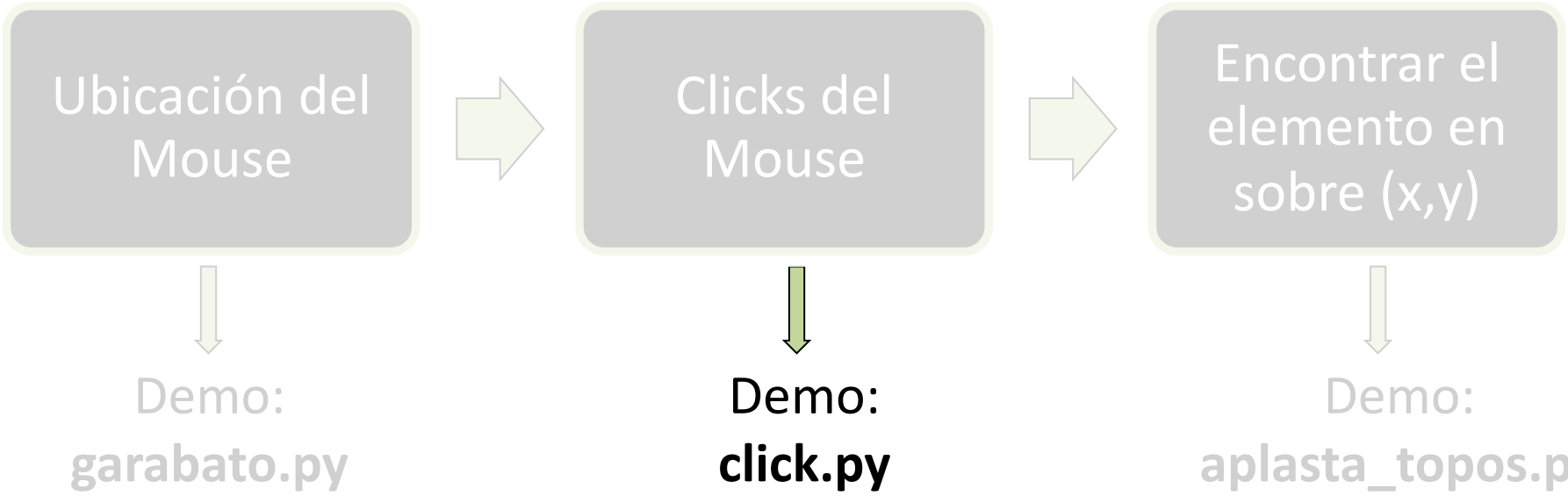



```
while True:
    # Nos encargamos de clicks del mouse
    # ...
    canvas.update()
```

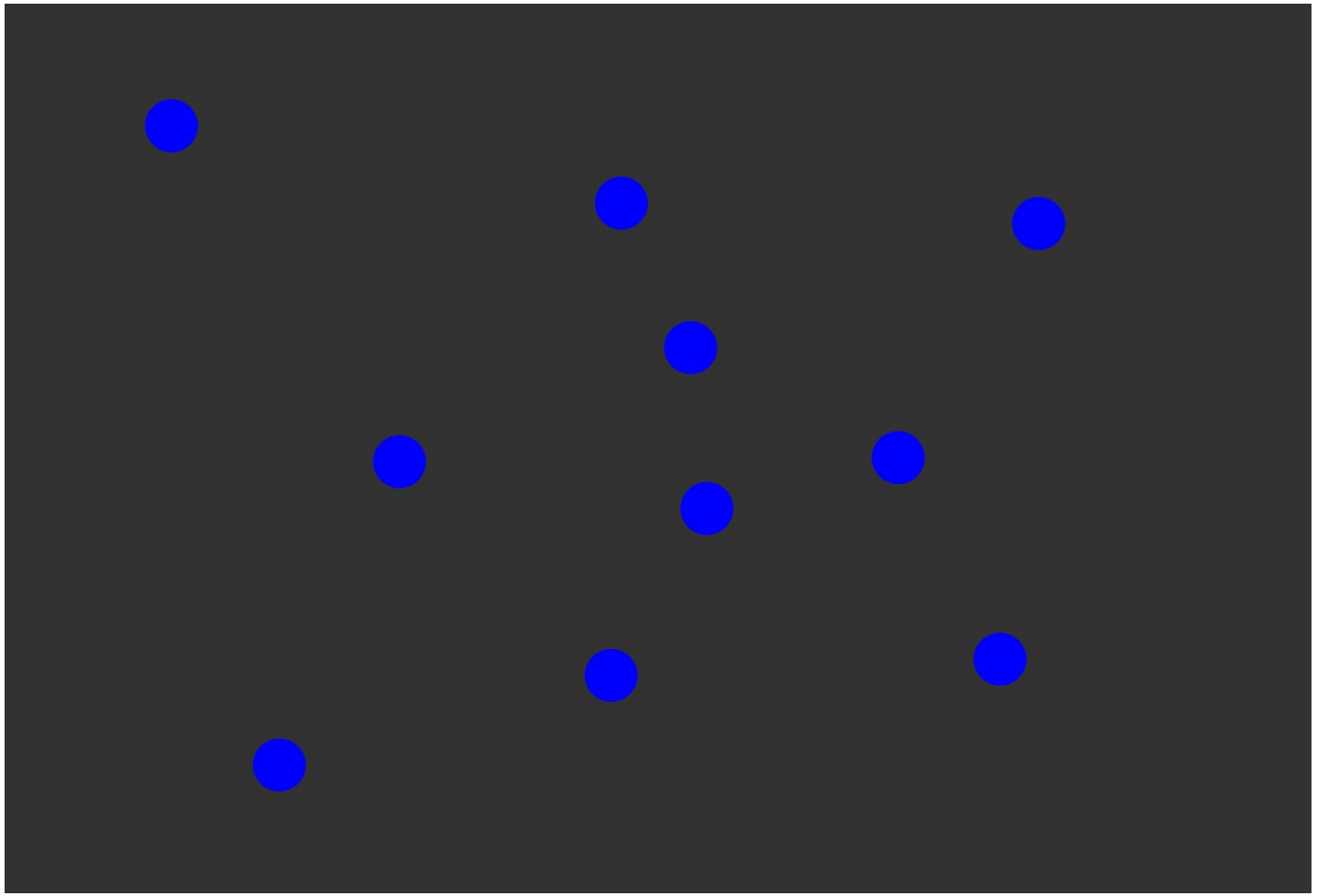
Plan de Clase



Plan de Clase



clicks



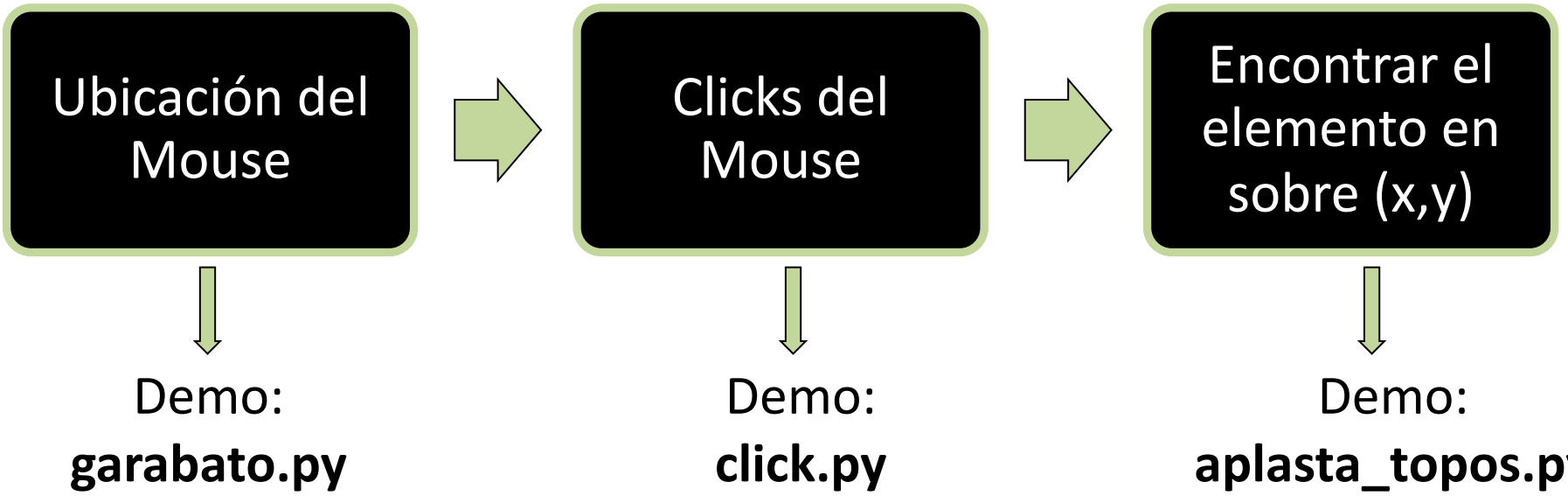
clicks.py

```
while True:
    clicks = canvas.get_new_mouse_clicks()

    # Añade un círculo cada vez que haya un click
    for click in clicks:
        circle = canvas.create_oval(click.x, click.y,
                                    click.x + CIRCLE_SIZE,
                                    click.y + CIRCLE_SIZE)
        canvas.set_color(circle, 'blue')
    canvas.update()
```

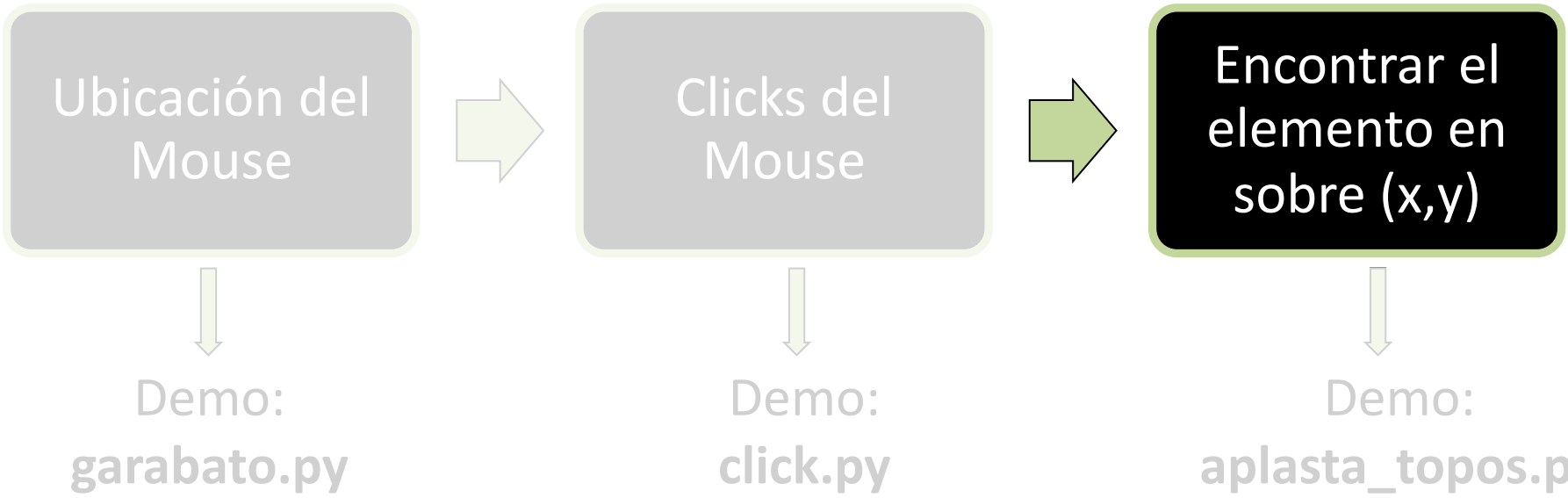
clicks.py

```
while True:
    clicks = canvas.get_new_mouse_clicks()

    # Add a circle each time the user clicks
    for click in clicks:
        circle = canvas.create_oval(click.x, click.y,
                                    click.x + CIRCLE_SIZE,
                                    click.y + CIRCLE_SIZE)
        canvas.set_color(circle, 'blue')
    canvas.update()
```

Plan de Clase



Plan de Clase



find_element_at

find_element_at devuelve el objeto en la posición x, y

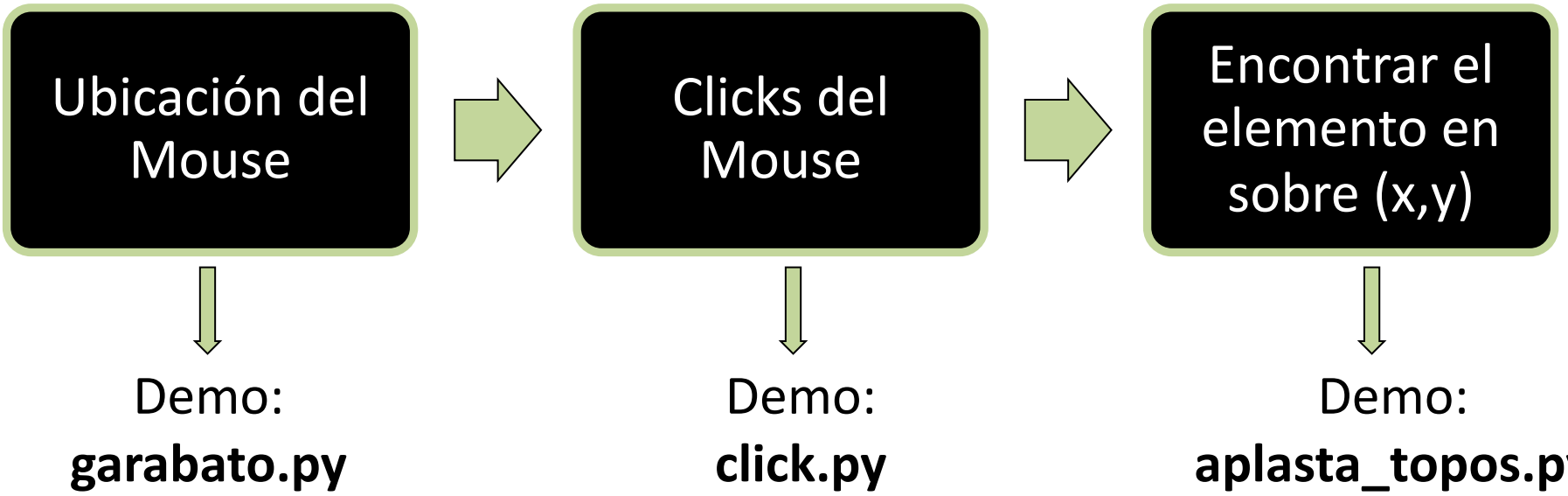
```
objeto_aquí = canvas.find_element_at(x, y)
```

find_element_at

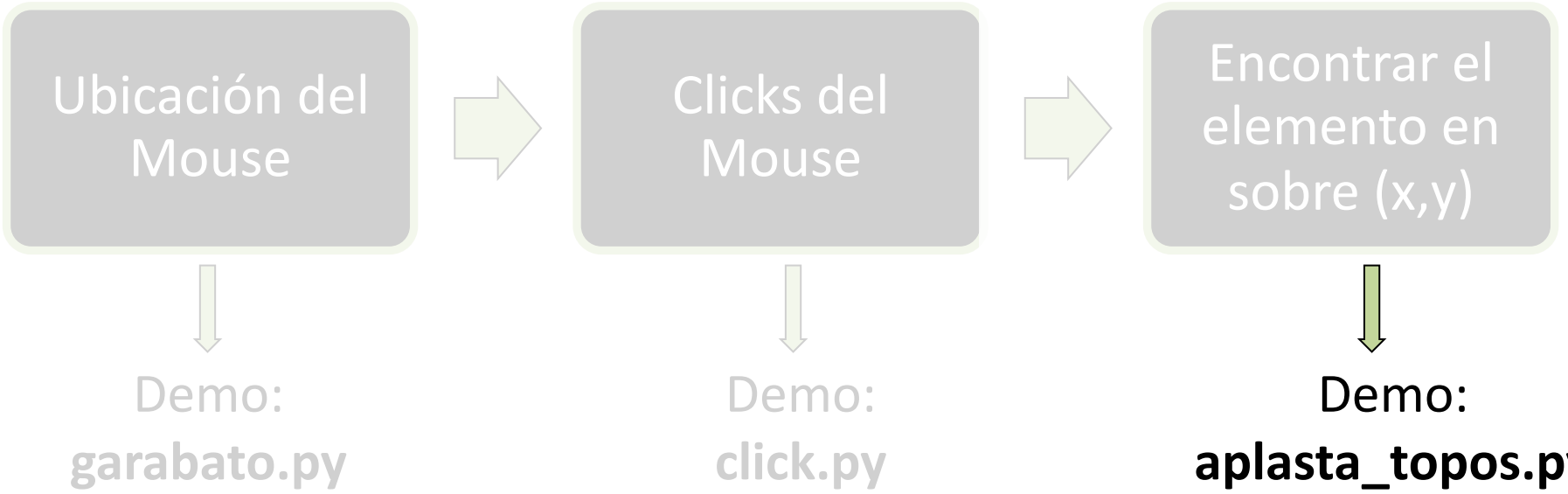
find_element_at devuelve el objeto en la posición x, y

```
object_here = canvas.find_element_at(x, y)
if objeto_aquí:
      // haz algo con el objeto
else:
      // no hagas nada aquí
```

Plan de Clase



Plan de Clase



Práctica: aplasta un topo

Aplasta un Topo

- Los topos salen de manera aleatoria en la pantalla
- Si el usuario hace click sobre uno, este desaparece y gana un punto