Checking qualifications

Demonstrate your knowledge of:

- the principles of working with events;
- keyboard events from pygame;
- lists and their methods.

Checking qualifications

What is an event?

What is event subscription ?

Give some examples.

Checking qualifications

An event

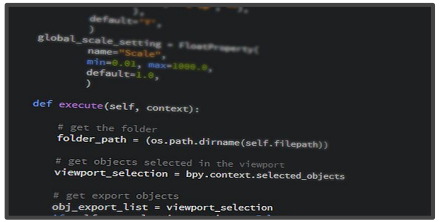
is information prepared by the execution system about what is happening in the "external environment".

<u>The external environment</u> refers to
any **equipment** connected to a computer.

Did something **happen**?

<u>Execution system:</u>
"An **event** has happened!"
(Prepares information about it).

<u>Running program</u>



Checking qualifications

Event subscription is

a request from the program asking which event is important to it.

An event handler is

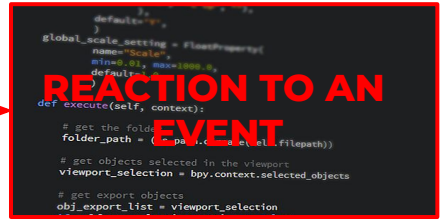
an algorithm that describes the reaction to an event.

<u>The external environment</u> refers to
any **equipment** connected to a computer.

Did something **happen**?

<u>Execution system:</u>
"An **event** has happened!"
(Prepares information about it).

<u>Running program:</u>
"This is an **important event** for me!
I should **react.**"



Checking qualifications

What events do you <u>already know how to handle</u> using pygame?

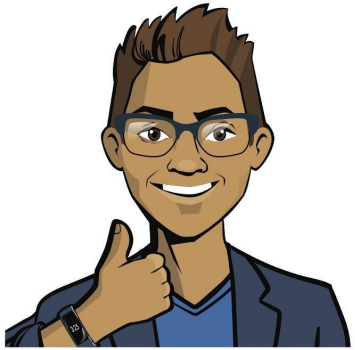
***<u>Which command</u>* allows you to get all the events happening in the program at a given moment?**

Checking qualifications

Handling keyboard events in pygame

With the help of pygame, we are able to respond to keystrokes.

Command	*Purpose*
`pygame.event.get()`	Get a set of events happening during a given frame of the loop.
`event.type / event.key`	Event type / Event associated with the key



Checking qualifications

How do you handle a keyboard event? (For example, pressing the W key.)

Checking qualifications

Handling keyboard events

Let's take a look at the commands for handling keyboard events.

Command	*Purpose*
`pygame.event.get()`	A set of events that occur during a given frame of the loop
`event.type / event.key`	Event type / Event associated with the key

```
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            Action
```

"If there is an event with a key pressed down among the current events and this key is W, then perform the action."

Checking qualifications

What is a list?

How do we get a list item by number?

How can you check the occurrence of a certain element in the list?

What other methods of working with lists do you know?

Checking qualifications

A list is a structure for storing different types of data in a well-ordered manner

Example. A list of results from a tournament of the online game "Space Shooter".

```
results = [181, 176, 160, 178, 171, 179, 165]
```

```
print('Best result:', results[0])
```

→ Best result: 181

Get a list item by its number (index)

Checking qualifications

Working with lists

Command	*Purpose*
?	Declare an empty list
?	Add an item to the end of the list
?	Search for the occurrence of an element in the list (returns True or False)
?	Sort the list in lexicographic order (by ascending numbers and letters of the alphabet)
?	Iterate through the `results` list items. “For each item (`result`) of the list (`results`), execute Command1, Command2”
?	Determine the length of the `results` list

Checking qualifications

Working with lists

Command	*Purpose*
`participants = list()`	Declare an empty list
`participants.append('Smith')`	Add an item to the end of the list
‘Johnson’ `in participants`	Search for the occurrence of an element in the list (returns True or False)
`participants.sort()`	Sort the list in lexicographic order (by ascending numbers and letters of the alphabet)
`for result in results:` Command1 Command2	Iterate through the `results` list items. “For each item (`result`) of the list (`results`), execute Command1, Command2”
`len(results)`	Determine the length of the `results` list

Checking qualifications

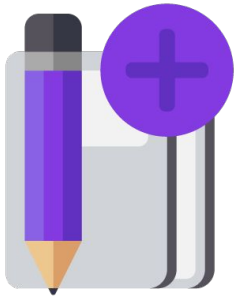
Qualifications confirmed!

Great, you are ready to brainstorm and complete your work task!

Checking qualifications

Brainstorm:

Random displaying of a label



Random display of a label

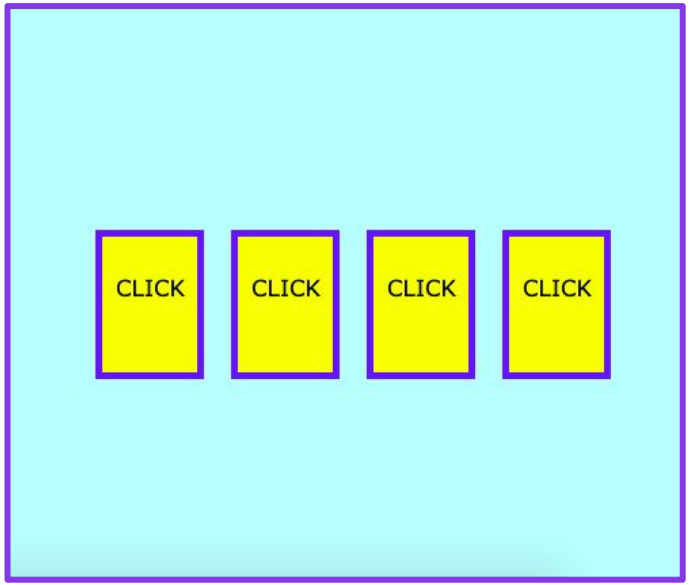
The result is that "CLICK" appears on a random card for a fraction of a second, and then disappears and appears again, but on another card.

- [ ] Program the display of the inscription CLICK on a random card.
- [ ] Limit CLICK display time (for example, 0.5 sec).

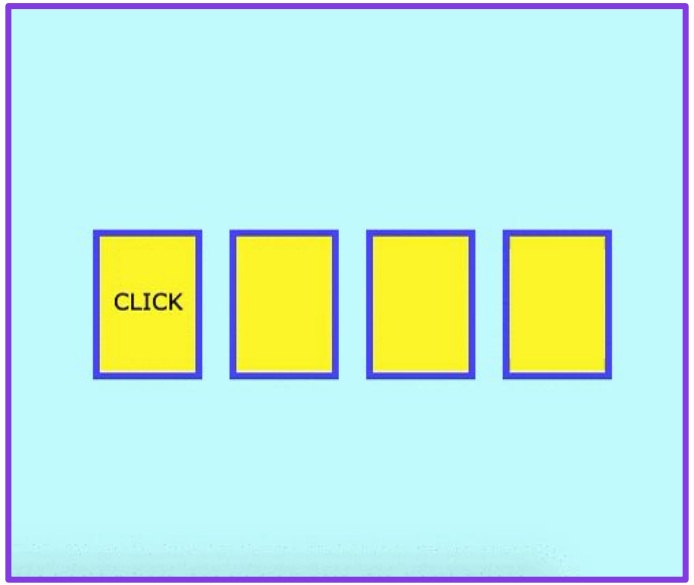
Brain storm

Random display of a label

The result is that "CLICK" appears on a random card for a fraction of a second, and then disappears and appears again, but on another card.



Now



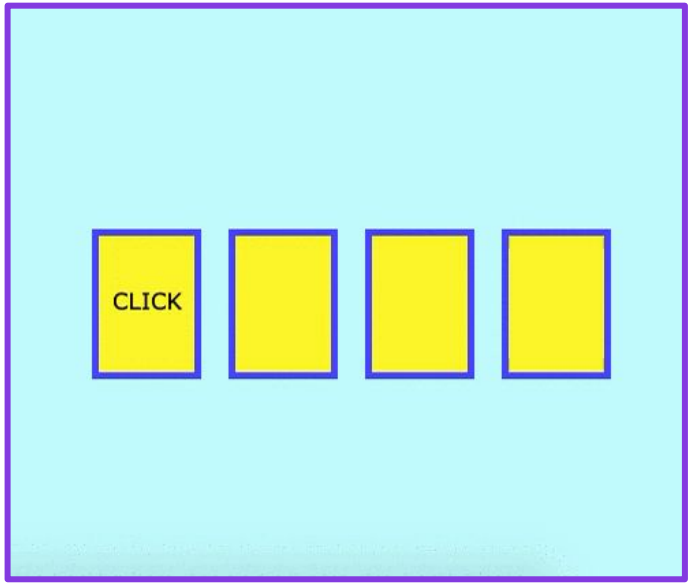
We will program this in the first half of the working day

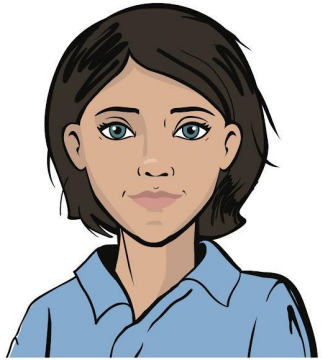
Brain storm

How do we display CLICK?

This part of the program's behavior can be programmed in two steps:

- ➔ The appearance of an inscription on a random card.
- ➔ Displaying the label for a certain time. Then it disappears.



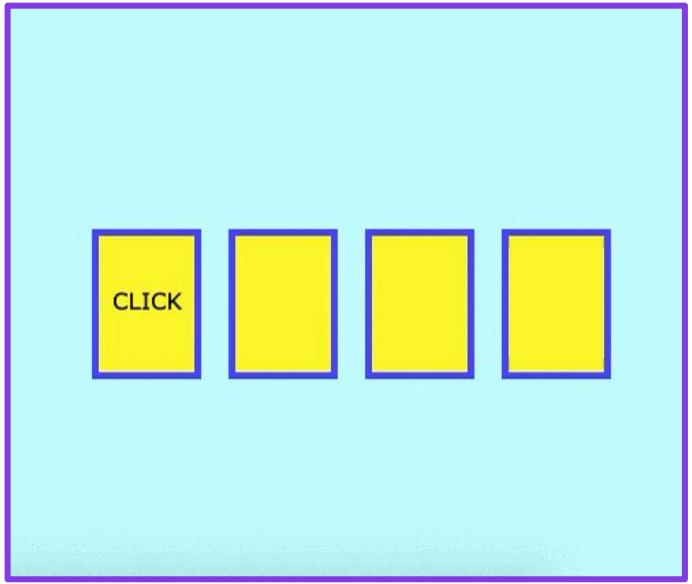


Brain storm

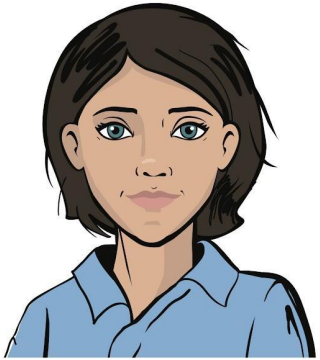
How do we display CLICK?

This part of the program's behavior can be programmed in two steps:

- ➔ The appearance of an inscription on a random card.
- ➔ Displaying the label for a certain time. Then it disappears.



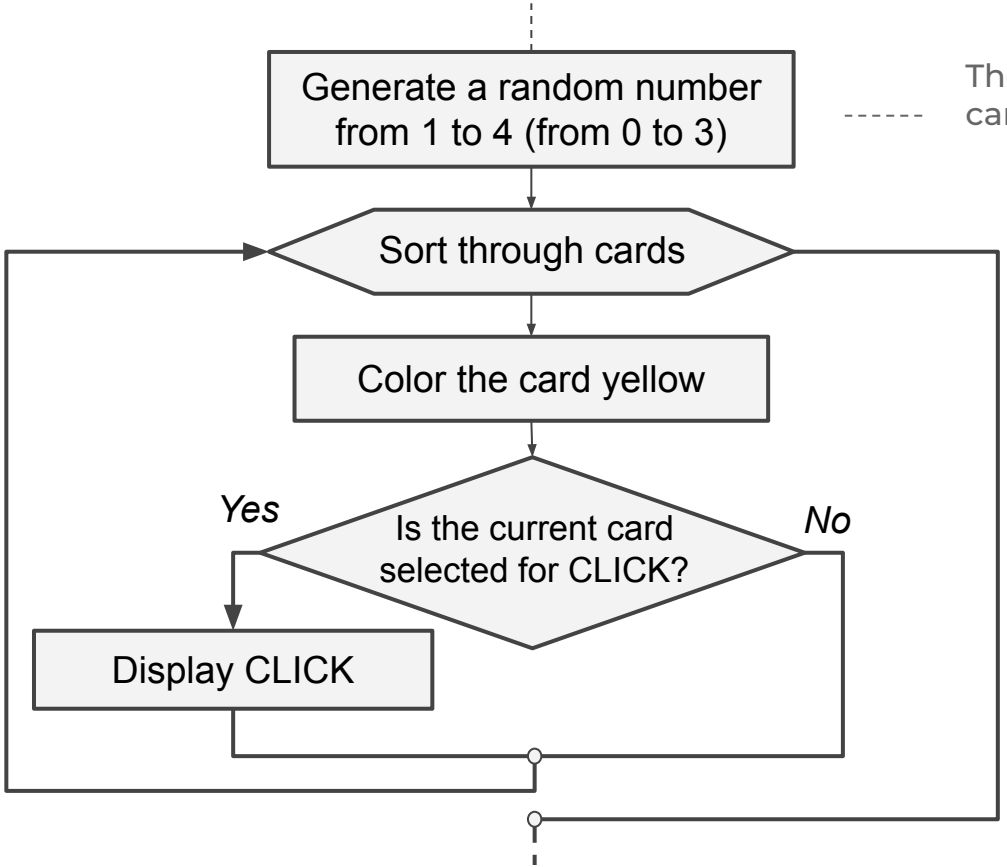
How can you solve this task?



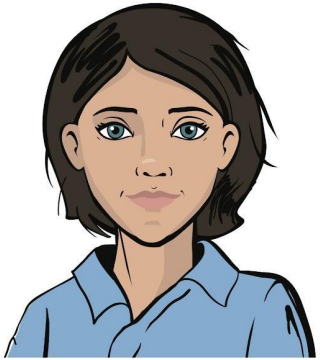
Brain storm

1. The appearance of an inscription on a random card

Important: Before displaying the CLICK, the card must not have any text on it.



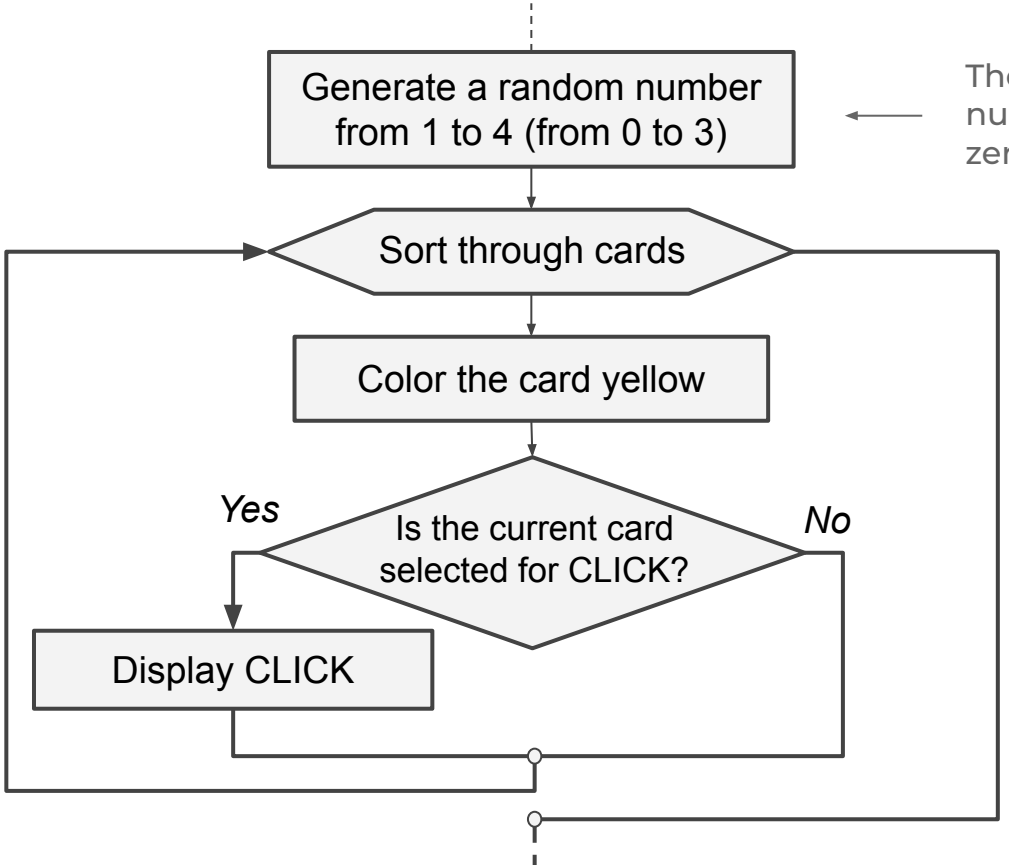
The number of a random card to display the label.



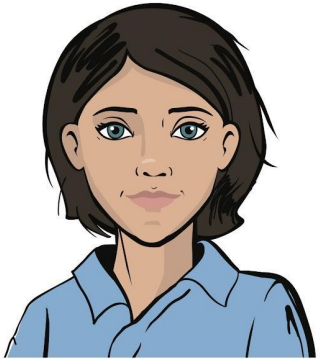
Brain storm

1. The appearance of an inscription on a random card

Important: Before displaying the CLICK, the card must not have any text on it.



The list items are numbered starting at zero, be careful!



Brain storm

1. The appearance of an inscription on a random card

Some useful commands:

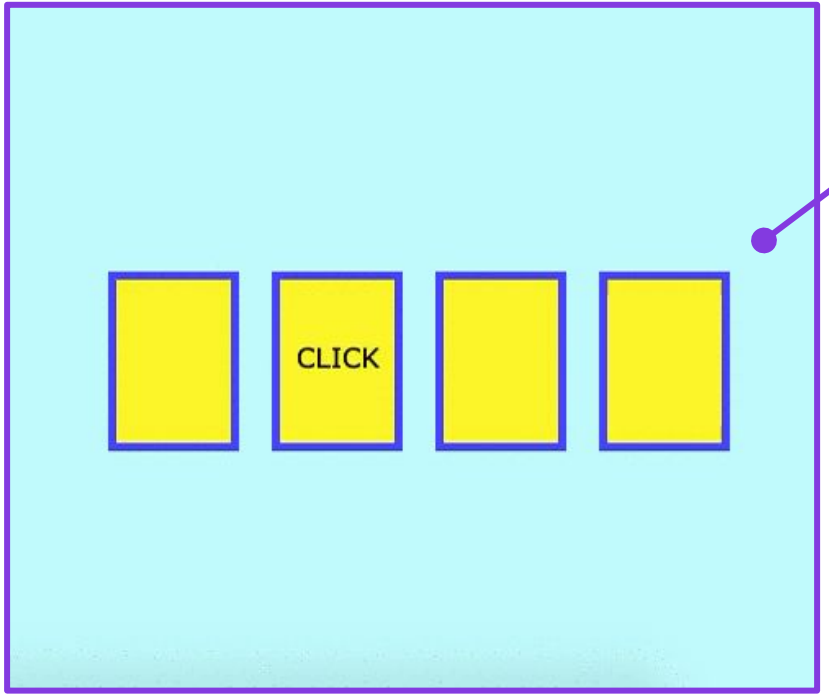
Command	*Purpose*
`number = randint(1, total)`	Get a random number from 1 to total
`for i in range(n):` `  command`	Execute the command for each i from the list from 0 to n
`for card in cards:` `  command`	Execute a command for each card from the list of cards
`cards[i].method()`	Apply the method to the i-th card from the list

Brain storm

2. Display the label for a certain amount of time

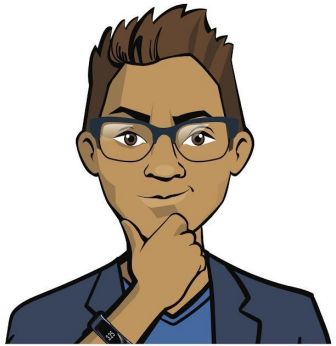
Without additional time control, the label will disappear and appear at every step of the game loop.

The user will not have enough time to "catch" it!



The frames of the game loop change too fast....

How can we fix this problem?

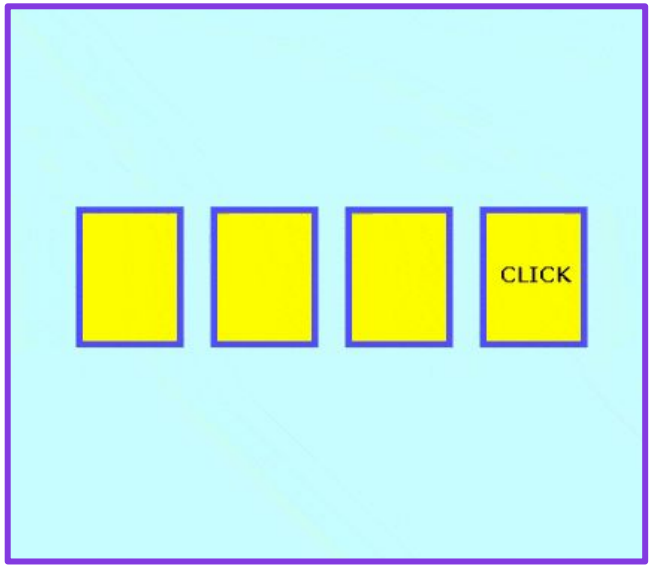


Brain storm

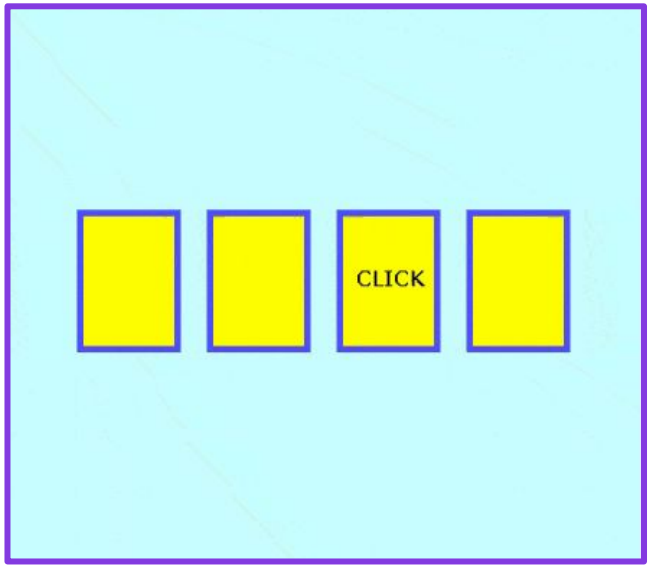
2. Display the label for a certain amount of time

<u>Possible solution:</u> choose how many frames of the game loop the CLICK label should be displayed for.

When these frames pass, the label "shifts" to another card.



The label is displayed for 15 frames.

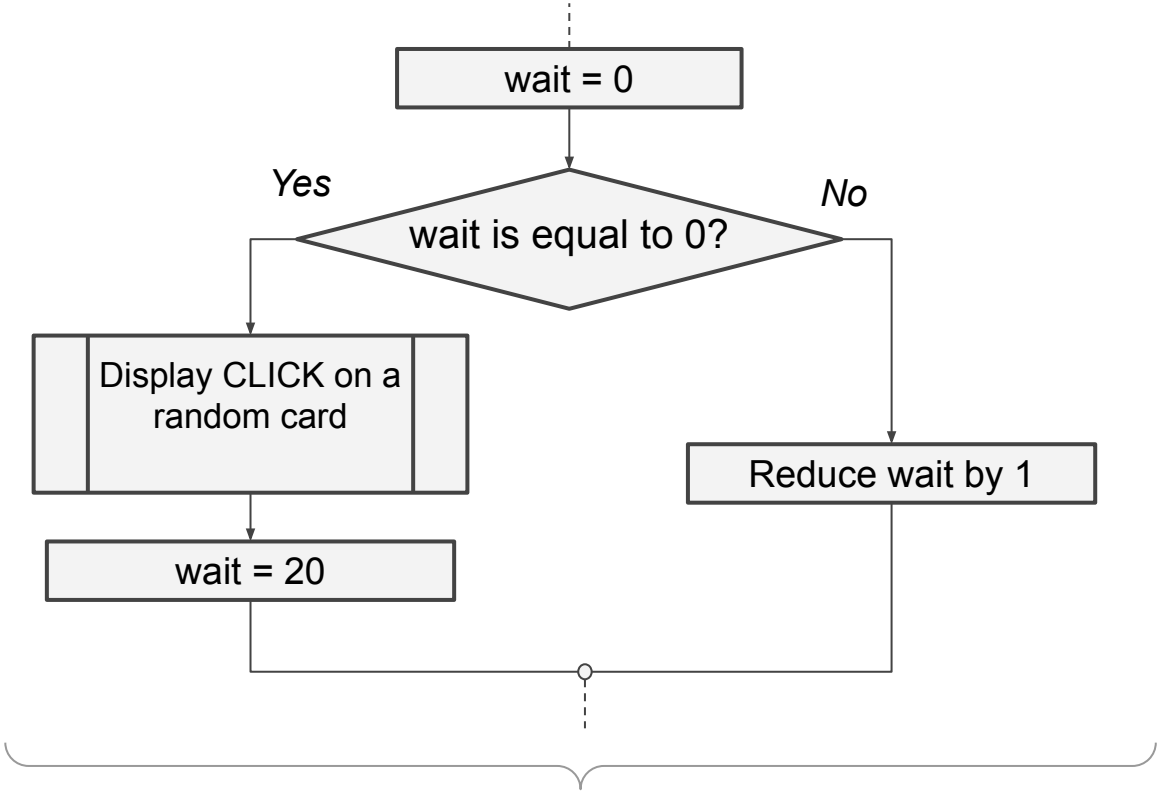


The label is displayed for 40 frames.

Brain storm

2. Display the label for a certain amount of time

We just need to enter a variable - the wait counter to count the frames.



Everything happens in the game loop

Brain storm

New game loop:

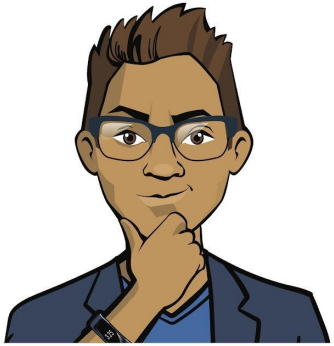
The result is that "CLICK" appears on a random card for a fraction of a second, and then disappears and appears again, but on another card.

```
wait = 0
while True:
  if wait == 0:
      wait = 20
      click = randint(1, num_cards)
      for i in range(num_cards):
          cards[i].color(YELLOW)
          if (i + 1) == click:
              cards[i].draw(10, 40)
          else:
              cards[i].fill()
  else:
      wait -= 1
```

Your tasks:

1. Program the display of the CLICK label on a random card.
2. Choose the optimal number of frames for the label's display time.

If you have time left over, start programming text objects for the statistics counters.



Brain storm

Handling mouse events.

With the help of pygame, we are able to respond to mouse clicks.

Command	*Purpose*
`cur_events = pygame.event.get()`	Get a set of events happening during a given frame of the loop.
`event_type = event.type` `button_type = event.button`	Event type (Mouse? Keyboard?) Type of mouse event (Which button is pressed?)
`x, y = event.pos`	Returns the coordinates of the point where the event occurred

Brain storm

Handling mouse events.

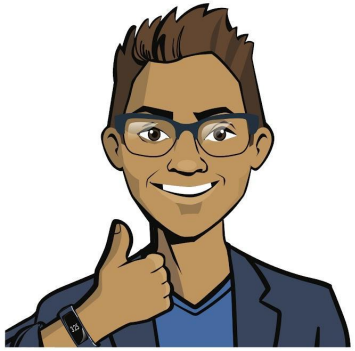
With the help of pygame, we are able to respond to mouse clicks.

```
for event in pygame.event.get():

        if event.type == pygame.MOUSEBUTTONDOWN:

            x, y = event.pos
```

Viewing current events...

If there was a mouse click...

Get the coordinates of the click location!



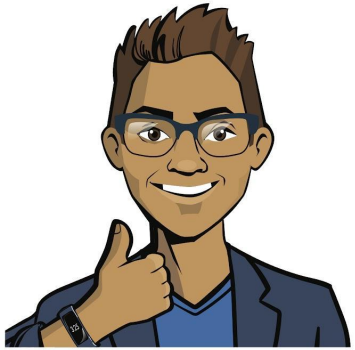
Brain storm

Handling mouse events.

With the help of pygame, we are able to respond to mouse clicks.

```
for event in pygame.event.get():
    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
        x, y = event.pos
```

The condition can be made stricter and require a click with the left (first) mouse button!



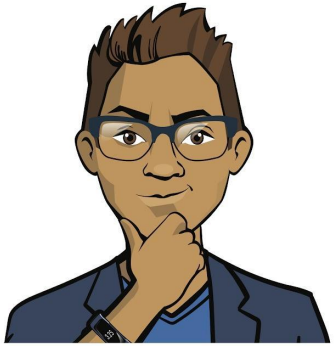
Brain storm

Handling mouse events.

With the help of pygame, we are able to respond to mouse clicks.

```
for event in pygame.event.get():
    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
        x, y = event.pos
```

As a result, we will find out the coordinates of the click.
How do we find out <u>if the click touched the card</u>?



Brain storm

Handling mouse events.

With the help of pygame, we are able to respond to mouse clicks.

```
for event in pygame.event.get():
    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
        x, y = event.pos
```

As a result, we will find out the coordinates of the click.
How do we find out <u>if the click touched the card</u>?

<u>One way</u> is to create a new method in the Area class, which determines whether a point with coordinates (x, y) has touched a rectangle.

Brain storm

The collidepoint() method

To find out if the card was clicked on, let's add a new method to the Area class. We will need the command:

Command	*Purpose*
`res = rect.collidepoint(x, y)`	A method of the Rect class that determines whether a point (x, y) has touched (or gone inside) a rect-type object

Remember, rect is responsible for coordinates, and image is responsible for appearance!

Brain storm

The collidepoint() method

To find out if the card was clicked on, let's add a new method to the Area class. We will need a new command.

Command	*Purpose*
`res = rect.collidepoint(x, y)`	A method of the Rect class that determines whether a point (x, y) has touched (or gone inside) a rect-type object

```
def collidepoint(self, x, y):
        return self.rect.collidepoint(x, y)
```

The collidepoint() method will just refer to the similar rect() method.

Brain storm

Handling mouse events.

After clicking the mouse, you need to find the card it occurred on.

If the card has CLICK written on it, then you need to color it green.

```
for i in range(num_cards):
    if cards[i].collidepoint(x,y):
        if i + 1 == click:
            cards[i].color(GREEN)
```

For each card, we check...

Was there a click on it?

If there was a click and the card number matches the card number with the label, then we color it green!

`click` is a random card number for displaying the phrase CLICK

Brain storm

Handling mouse events.

After clicking the mouse, you need to find the card it occurred on.

If the card has CLICK written on it, then you need to color it green.

```
for i in range(num_cards):
    if cards[i].collidepoint(x,y):
        if i + 1 == click:
            cards[i].color(GREEN)
        else:
            #...
```

For each card, we check...

Was there a click on it?

If yes, and the card number matches the card number with the label, then we color it green!

Otherwise, it should be colored red.

By the way, coloring it for one frame of the game loop is enough!
Our eyes will detect it.

Brain storm

Program flowchart:

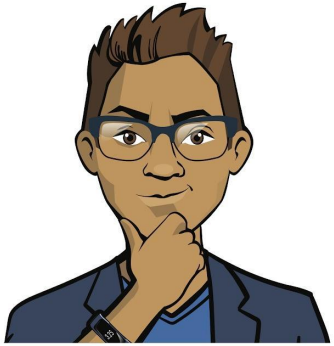
- Connect Pygame modules
- Create a background and a timer
- *Create the **Area** class*
- *Create the **Label class***
- Create a set of cards
- Game loop:
  - Display all cards
  - Variable display of the word CLICK on the cards
  - Handling clicks on cards
  - Frame rate ~40 FPS

Brain storm

Your task:

➔ Program the handling of a left mouse click on the card.

If you have time left over, start programming text objects for the statistics counters.



Brain storm