Module 5. Lesson 2. OOP. Events

Qualifications

Demonstrate your knowledge to begin working on the tasks.

Show that you are ready to brainstorm!

Qualifications

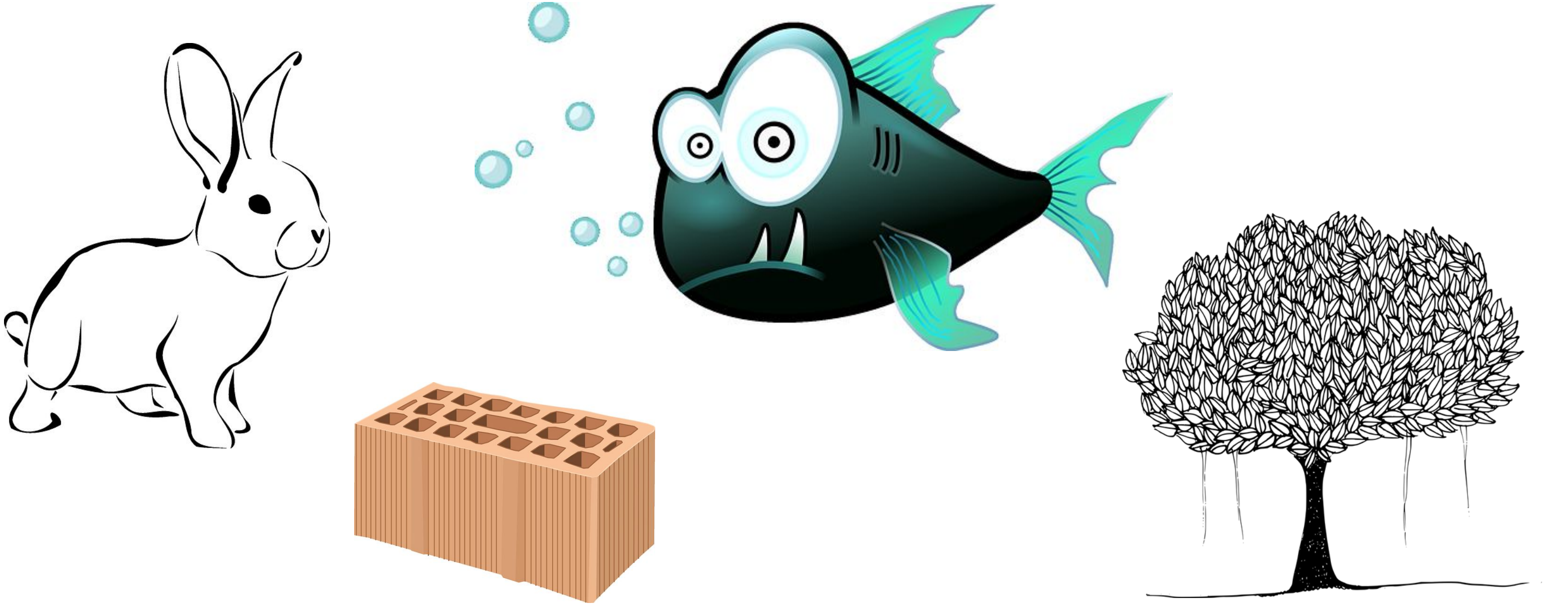
What is an object ?

Give examples of objects.

An object

is a set of data and actions that is convenient to perceive as a whole.

Real world objects:



Qualifications

What are object properties and methods ?

How can we access them programmatically ?

Qualifications

Each of these objects stores information about itself and knows how to perform some actions.

In the English language:

Rabbit, run!

In the programming language:

```
Rabbit.run()
```

Qualifications

Each of these objects stores information about itself and knows how to perform some actions.

An object is said to have properties and be controlled by methods.

Properties	*Methods*
rabbit.speed = 50	rabbit.run()
turtle.speed = 1	turtle.walk()
fish.speed = 30	fish.swim()

Variable placed inside the object.

Function placed inside the object.

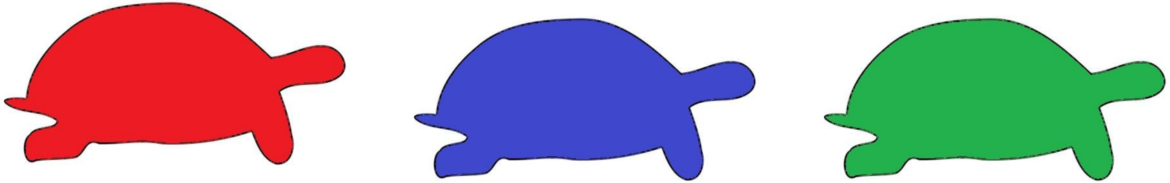
Qualifications

In your own words, explain what the essence of the object-oriented approach to programming is.

Qualifications

Object-Oriented Programming

is an approach based on <u>creating</u> objects and <u>controlling</u> them.



Objects aren’t always turtles!
We just only know how to work with them so far.

Qualifications

Qualifications confirmed!

Great, you are ready to brainstorm and work on your tasks!

Qualifications

Brainstorming:

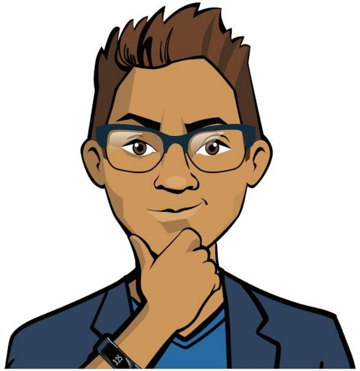
Events and handling them

Switching between algorithms

It can be very difficult to control the switching between program algorithms with a conditional statement.

Fortunately, systems have already been written for analyzing user actions and calling the appropriate algorithms!

Let's study how they work.



A program execution system

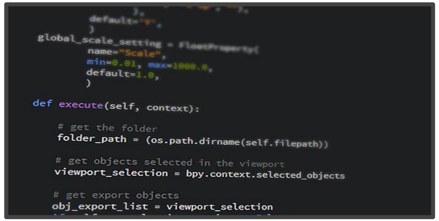
is a system built into the computer that runs various algorithms and automatically switches between them, analyzing the "outside world" of the program.

The outside world

is any **equipment** connected to the computer.

Execution system

Running program



```
        default='Y',
        )
global_scale_setting = FloatProperty(
        name="Scale",
        min=0.01, max=1000.0,
        default=1.0,
        )

def execute(self, context):

    # get the folder
    folder_path = (os.path.dirname(self.filepath))

    # get objects selected in the viewport
    viewport_selection = bpy.context.selected_objects

    # get export objects
    obj_export_list = viewport_selection
```

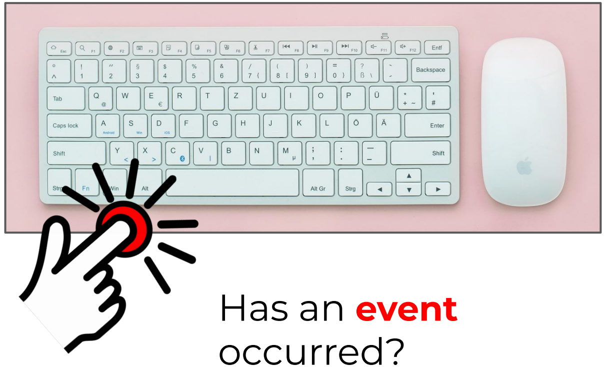
Brainstorming

An event

is information prepared by the execution system about what is happening in the "outside world."

The outside world

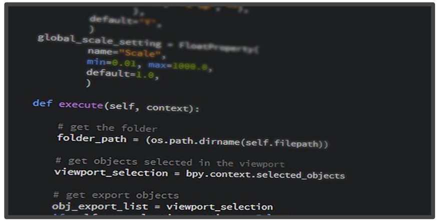
is any **equipment** connected to the computer.



Has an **event** occurred?

Execution system

Running program



```
            default='Y',
            )
    global_scale_setting = FloatProperty(
            name="Scale",
            min=0.01, max=1000.0,
            default=1.0,
            )

    def execute(self, context):

        # get the folder
        folder_path = (os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
```

Brainstorming

An event

is information prepared by the execution system about what is happening in the "outside world."

The outside world

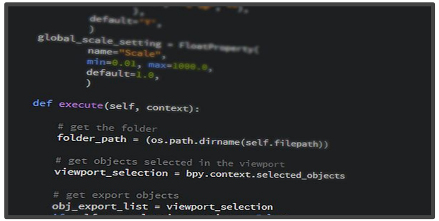
is any **equipment** connected to the computer.

Has an **event** occurred?

Execution system:

"An **event** has occurred!"
(*Prepares information about it*).

Running program



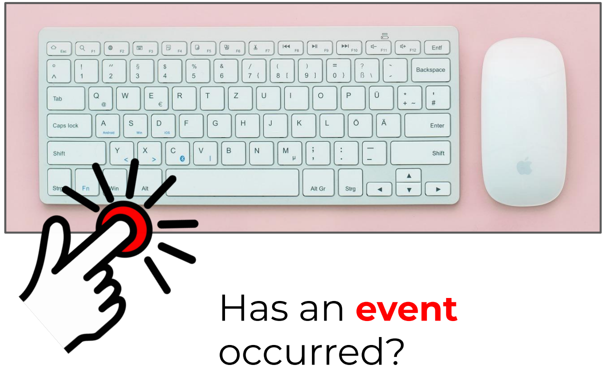
```
                ),
        default='Y',
        )
    global_scale_setting = FloatProperty(
            name="Scale",
            min=0.01, max=1000.0,
            default=1.0,
            )

def execute(self, context):

    # get the folder
    folder_path = (os.path.dirname(self.filepath))

    # get objects selected in the viewport
    viewport_selection = bpy.context.selected_objects

    # get export objects
    obj_export_list = viewport_selection
```

Brainstorming

A subscription to an event

is a request of the program about which event is important to it.

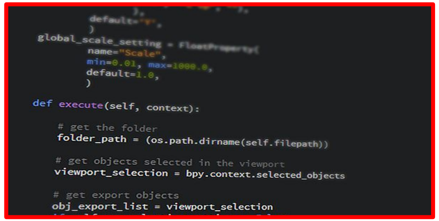
The outside world
is any **equipment** connected to the computer.



Has an **event** occurred?

Execution system:
"An **event** has occurred!"
(*Prepares information about it*).

Running program:
"This is an **important event** to me!
I have to **react**."



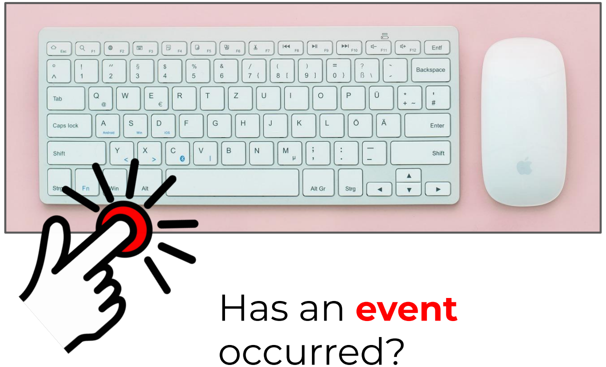
```
                    default='Y',
                    )
    global_scale_setting = FloatProperty(
                    name="Scale",
                    min=0.01, max=1000.0,
                    default=1.0,
                    )

    def execute(self, context):

        # get the folder
        folder_path = (os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
```

Brainstorming

The event handler

is an algorithm that describes the reaction to an event.

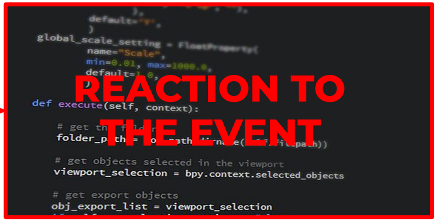
The outside world
is any **equipment** connected to the computer.



Has an **event** occurred?

Execution system:
"An **event** has occurred!"
(*Prepares information about it*).

Running program:
"This is an **important event** to me!
I have to **react**."

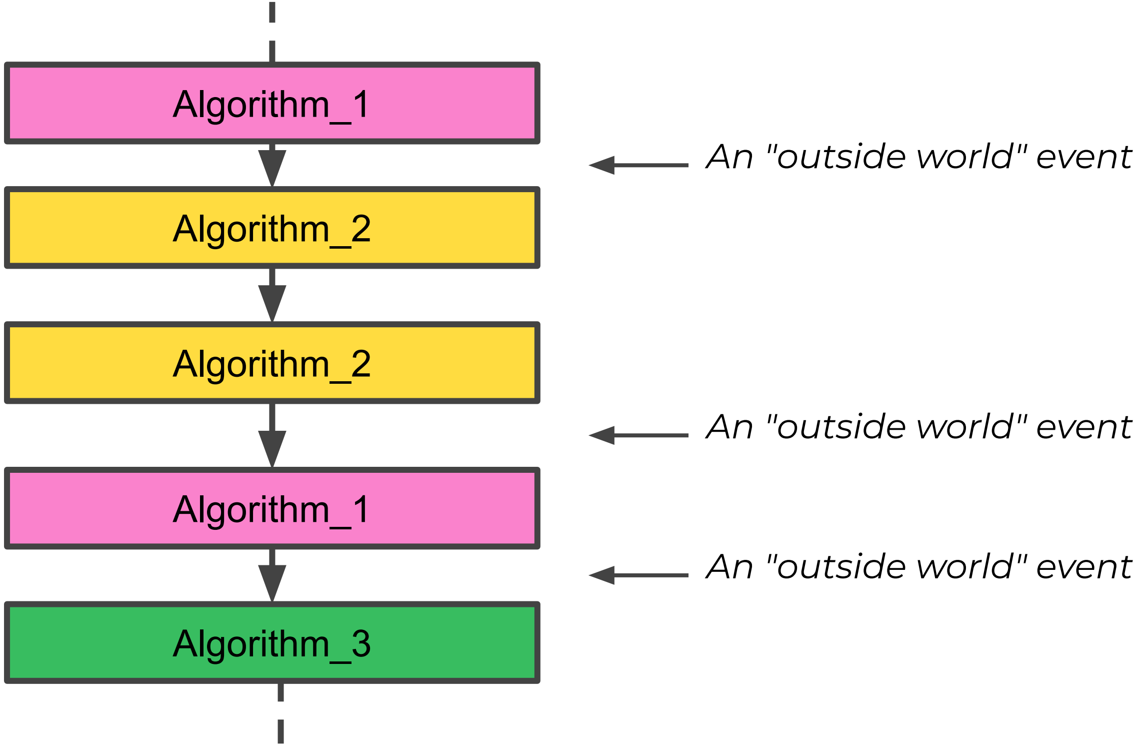


```
                default='Y',
            )
    global_scale_setting = FloatProperty(
            name="Scale",
            min=0.01, max=1000.0,
            default=1.0,
            )

    def execute(self, context):

        # get the folder
        folder_path = os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
```

Brainstorming

Program execution system

Thus, switching between different algorithms can occur depending on external events.



Brainstorming

With a single click on the turtle event, we can create two different prototypes of the game!

Catch the Turtle - prototype 1.

Brainstorming

With a single click on the turtle event, we can create two different prototypes of the game!

Catch the Turtle - prototype 2.

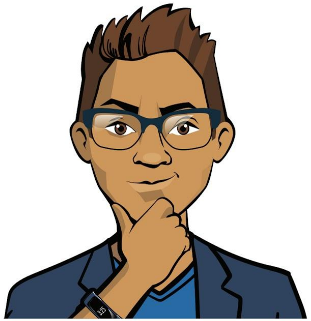
Creating prototype 1:

First, let’s look at the task <u>without</u> the "click on the turtle" event.

How do we display the turtle **in a random location** when starting the game?

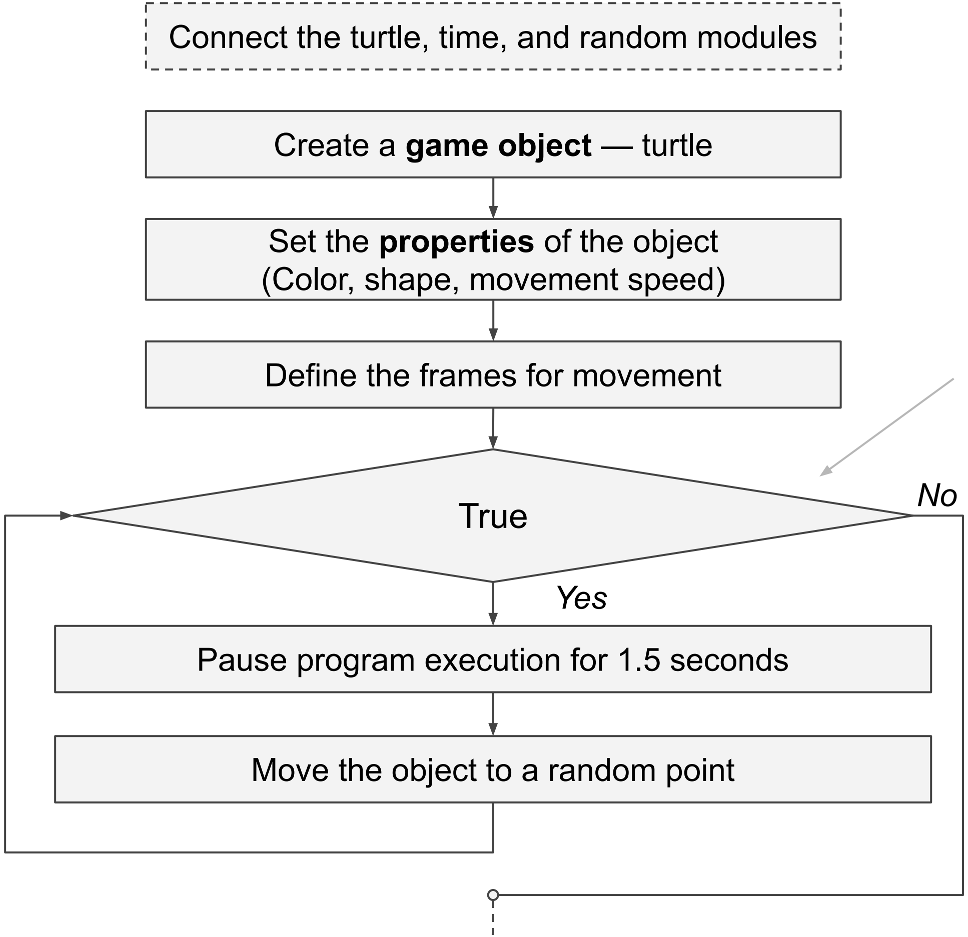
How do we leave it there **for 1.5 seconds**?

How can we then move it **to another random location**?



Brainstorming

Let's look at the task without the "click on the turtle" event.


Connect the turtle, time, and random modules
Create a **game object** — turtle
Set the **properties** of the object (Color, shape, movement speed)
Define the frames for movement
True
No
Yes
Pause program execution for 1.5 seconds
Move the object to a random point

For the first step, a loop with an always true condition is sufficient.

Brainstorming

```python
from turtle import *
from time import sleep
from random import randint

t = Turtle()
t.color('red')
t.penup()
t.shape('turtle')
t.speed(100)

w = 200
h = 200

def rand_move():
    t.goto(randint(-w, w), randint(-h, h))

while True:
    sleep(1.5)
    rand_move()
```

For the draft version of the game, a loop with an always true condition is sufficient.

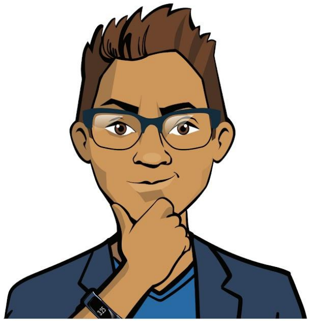
Brainstorming

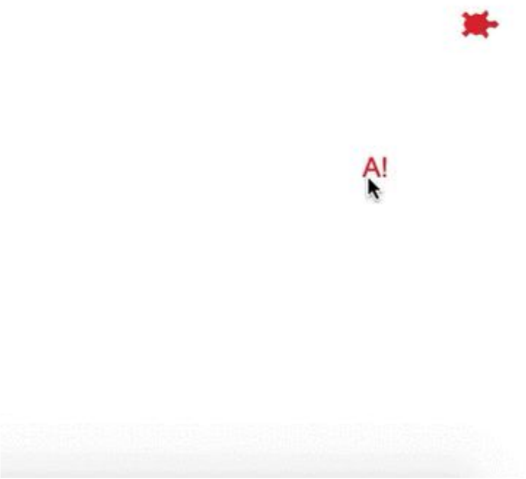
Let’s continue!

Let’s look at the task with the "click on a turtle" event.

We will program that when you click on the object, it will display: "A!"

What tools do we not have to program this?





A!

Brainstorming

Turtle: new command

Command	*Purpose*
`write('A!',font)`	Write the text in the specified font.
`font=('Arial', 14, 'normal')`	The font is specified in this sequence: "name, size, style."

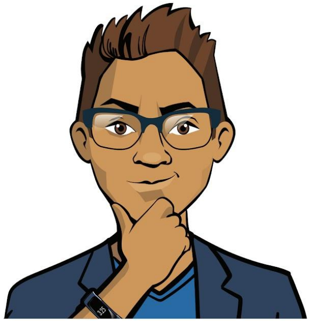
Example:

```
t.write('A!', font=('Arial', 14, 'normal'))
```

Brainstorming

Handling the "click on the turtle" event

- ➔ The reaction to the event should come immediately after the user clicks on the turtle. ***How do we subscribe to the turtle click?***
- ➔ In response to a click on the turtle, the message "A!" should be displayed, and the click itself should be counted (the game ends after three hits). ***How do we create a handler function with those commands?***

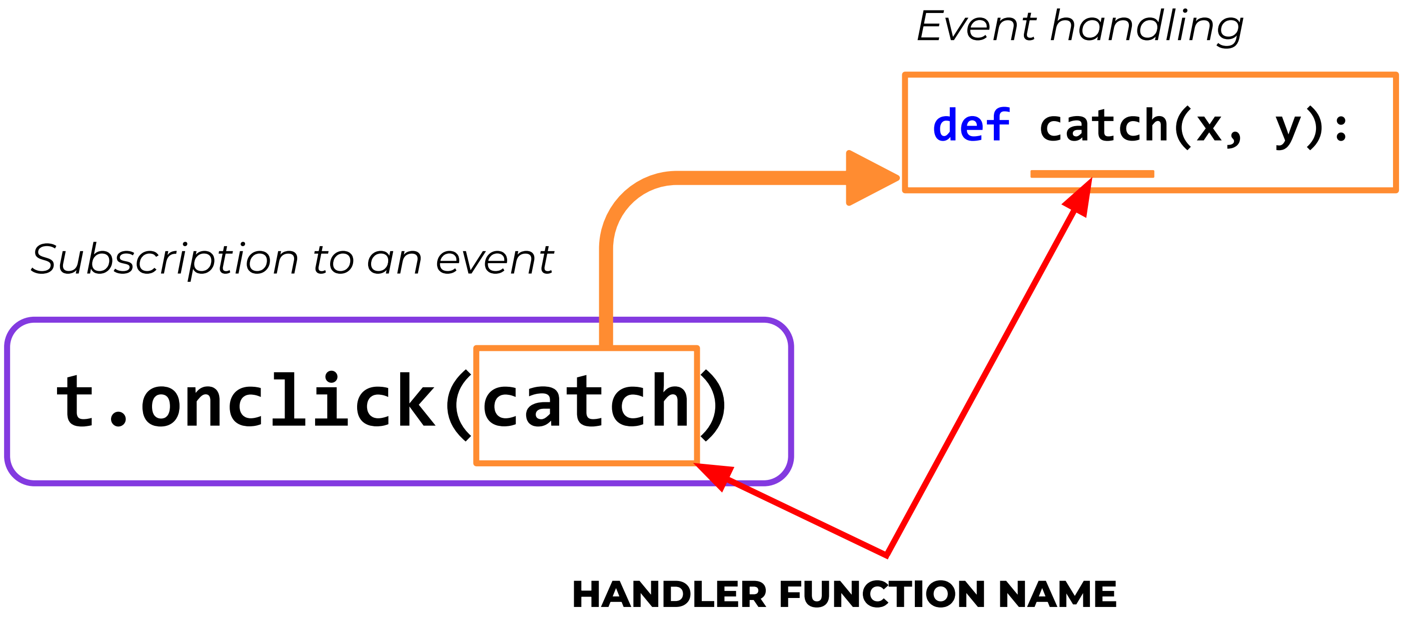


Brainstorming

Handling the "click on the turtle" event

To handle a click on the object, we'll create a **catch()** function, whose parameters will be the coordinates of the "caught" turtle.

The location of the click is sent by the execution system **by subscribing to the event**.



Event handling

```
def catch(x, y):
```

Subscription to an event

```
t.onclick(catch)
```

HANDLER FUNCTION NAME

Brainstorming

Connecting modules

Creating a game object

For the draft version of the game, a loop with an always true condition is sufficient.

```python
def rand_move():
    t.goto(randint(-w, w), randint(-h, h))

def catch(x, y):
    t.write('A!', font=('Arial', 14, 'normal'))
    rand_move()

t.onclick(catch)

while True:
    sleep(1.5)
    rand_move()
```

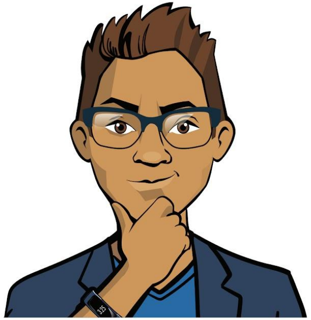
Handling the "click on the turtle" event with the catch() function.

Brainstorming

How do we program winning?

After three clicks on the turtle, the caption "WOW!" should be displayed, and the game should end (the turtle disappears).

How do we count the clicks on the object and get out of the loop in time?



Brainstorming

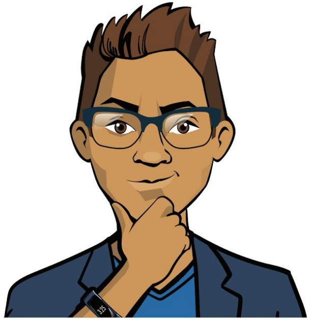
Counting clicks

The **onclick()** command calls the **catch()** handler function, sending it the coordinates of the clicked location.

Click counting is also easy to do in catch().

However, if you set the counter as an ordinary points variable, you will need to send it as an argument to change it in the function.

This is not possible because onclick() requires a handler with two arguments!



Brainstorming

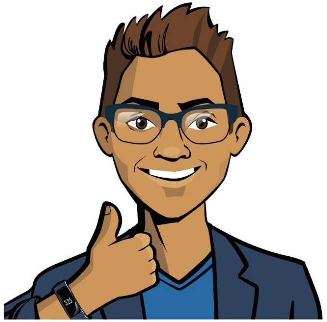
Creating a new property of an object

In Python, you don't only have to work with ready-filled objects, you can also complement them with new features!

Creating a new property is similar to creating a variable:

Object.property = value

Objects have a different scope than functions, so there is no problem when changing their values!



Brainstorming

Connecting modules

```
t = Turtle()
```

Setting the appearance of the object

```
t.points = 0
```

Declaring the rand_move() function

```
def catch(x, y):
  t.write('A!', font=('Arial', 14, 'normal'))
  t.points +=  1
  rand_move()

t.onclick(catch)

while t.points < 3:
  sleep(1.5)
  rand_move()
t.write('WOW!', font=('Arial', 16, 'bold'))
t.hideturtle()
```

Let's set the click counter as a new property of the **t.points** turtle object!

When starting the game **t.points = 0**.

On each click **t.points += 1**.

The game continues as long as **t.points < 3**.

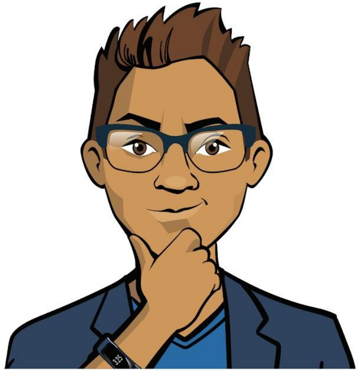
Brainstorming

The task :

Program a **Catch the Turtle prototype**.

Use the click on the turtle handling in the program.

Use the *documentation* if necessary.



Brainstorming

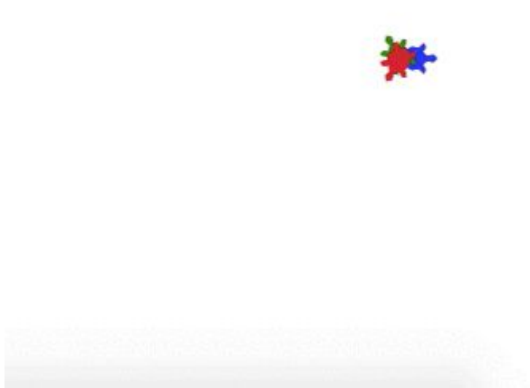
Brainstorming:

Events and handling them

Terms of Reference

The "Catch the Turtle" game *(prototype 2)*. When you start the game, three turtles appear at the same point, then they start moving in different directions. The direction of an object’s movement can be changed by clicking on it.

The task — prevent the turtles from "running away" off the screen.



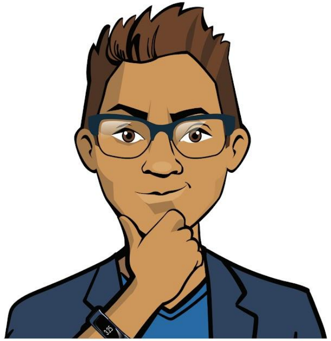
Brainstorming

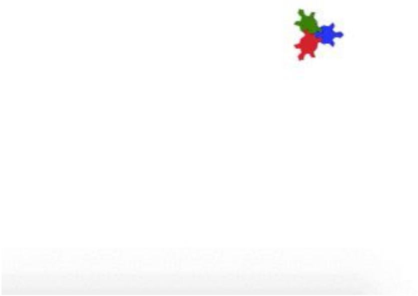
The "Catch the Turtle" game

Let’s look at a simplified task without events:

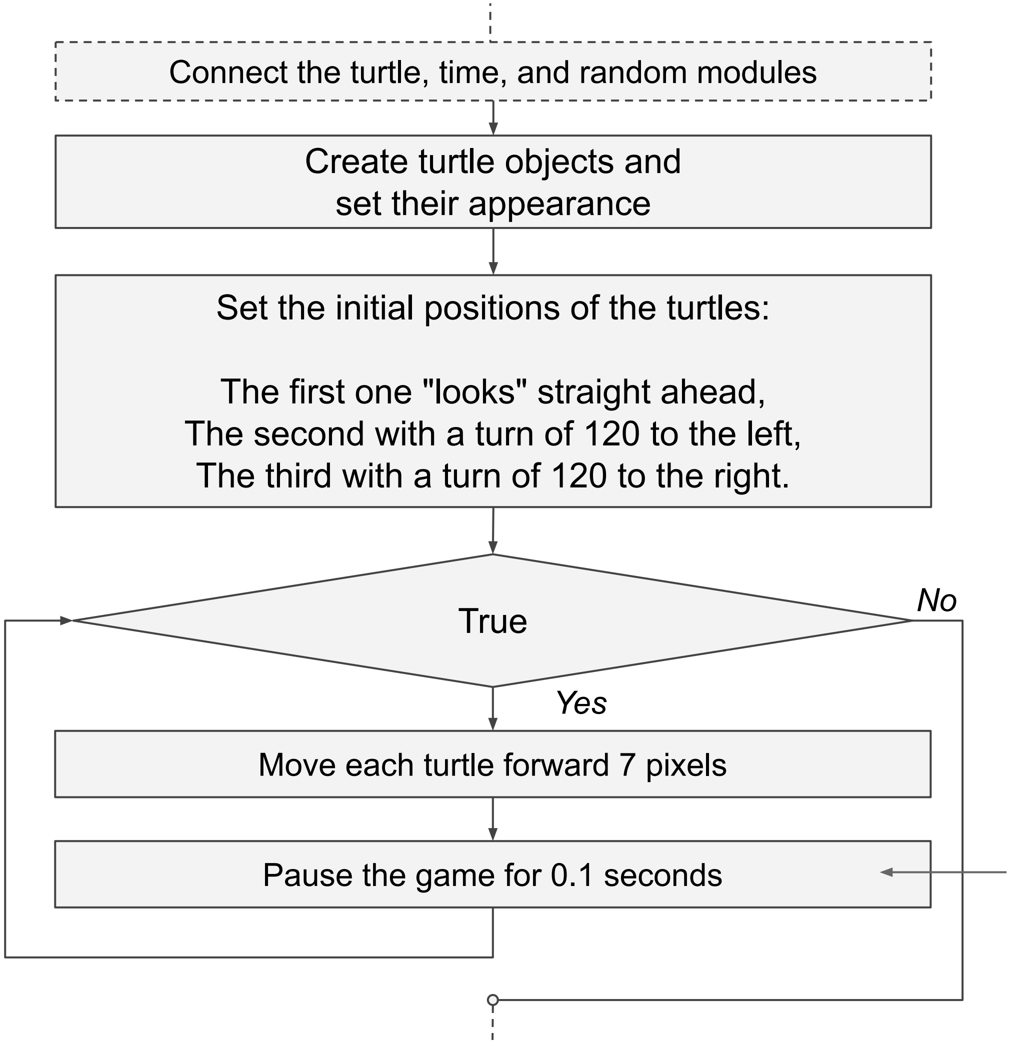
> when the program starts, the turtles appear at the same point and then "creep away" in different directions.

How do we program that?





Brainstorming



Connect the turtle, time, and random modules

Create turtle objects and
set their appearance

Set the initial positions of the turtles:

The first one "looks" straight ahead,
The second with a turn of 120 to the left,
The third with a turn of 120 to the right.

True

No

Yes

Move each turtle forward 7 pixels

Pause the game for 0.1 seconds

This creates the effect of turtle steps.

Brainstorming

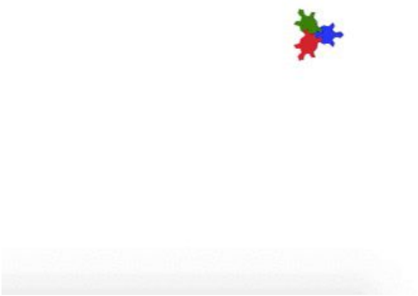
Connecting modules

```
w = 200

h = 200


t1 = Turtle()

t1.color('blue')

t1.width(5)

t1.shape('turtle')
```

Creating t2 and t3 the same way

```
while True:

    t1.forward(7)

    t2.forward(7)

    t3.forward(7)

    sleep(0.1)


exitonclick()
```

For the draft version of the game, a loop with an always true condition is sufficient.



Brainstorming

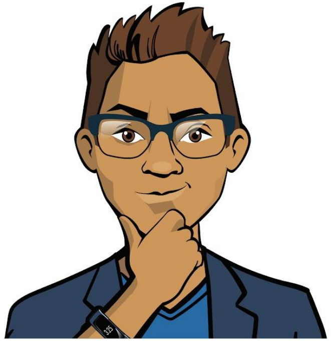
The "Catch the Turtle" game

Adding "click on the turtle" event handling (without leaving the game)**:**

When you click on the turtle, it moves to a new location on the screen and changes direction.

How do we subscribe to the "click on a turtle" event if there are three turtles?

How do we handle the event by moving the turtle?



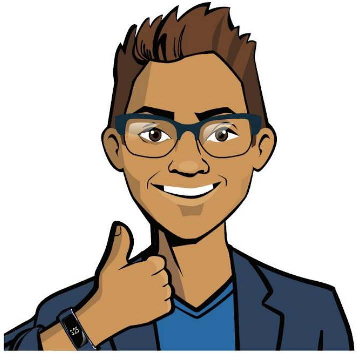
Brainstorming

Handling the "click on the turtle" event

In the second prototype, we have **three** turtles.

Clicking on the object is accompanied by the **movement** of **that** particular **object**.

That means there are **three different events** in the game: "click on the turtle 1," "click on the turtle 2," and "click on the turtle 3."



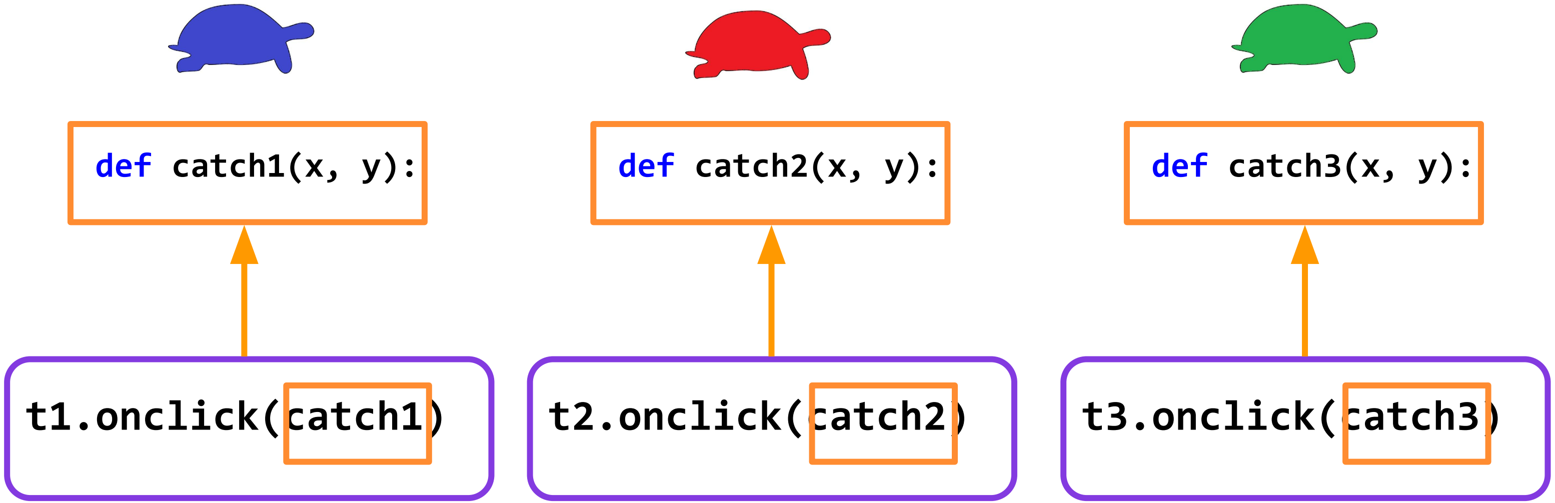
Brainstorming

Handling the "click on the turtle" event

In the second prototype, we have **three** turtles.

Clicking on the object is accompanied by the **movement** of **that** particular **object**.

That means there are **three different events** in the game: "click on the turtle 1," "click on the turtle 2," and "click on the turtle 3."



```
def catch1(x, y):
t1.onclick(catch1)

def catch2(x, y):
t2.onclick(catch2)

def catch3(x, y):
t3.onclick(catch3)
```

Brainstorming

Connecting modules

Creating t1, t2, and t3

```
def catch1(x, y):
  t1.penup()
  t1.goto(randint(-100,100),randint(-100,100))
  t1.pendown()
  t1.left(randint(0, 180))
```

Similarly for catch2(), catch3()

```
t1.onclick(catch1)
```

Similarly, subscription to the event for t2 and t3

```
while True:
   t1.forward(7)
```

Similarly, moving t2 and t3

For the draft version of the game, a loop with an always true condition is sufficient.

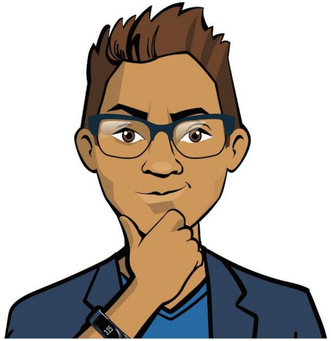
Brainstorming

The "Catch the Turtle" game

Let's add a condition for exiting the game:

at least one turtle has left the screen.

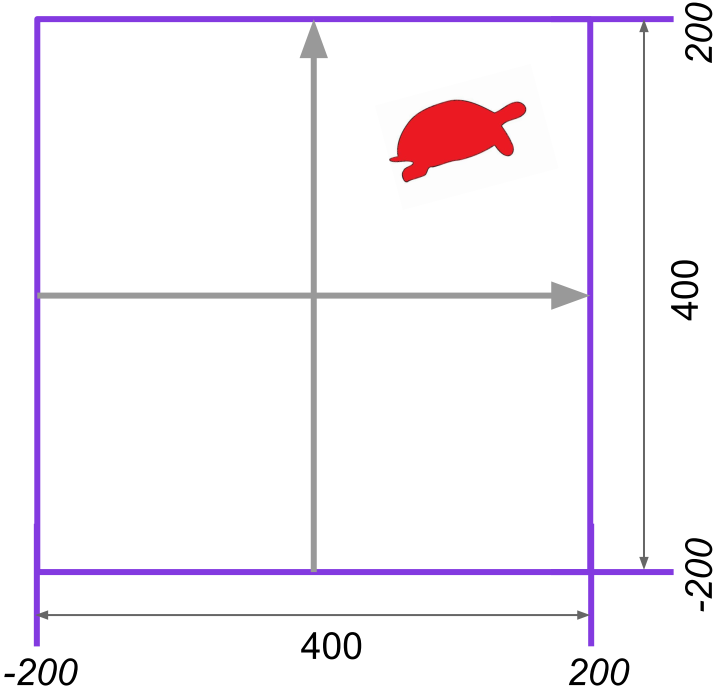
What boolean expression for the turtle to go off-screen should you add to the while loop?



Brainstorming

The "Catch the Turtle" game

Let's add a condition for exiting the game:
at least one turtle has left the screen.



Game ending conditions for one turtle:

- the X coordinate, <u>taken without a sign</u>, is greater than 200;
- the Y coordinate, <u>taken without a sign</u>, is greater than 200.

Brainstorming

The "Catch the Turtle" game

Let's describe the gameFinished() function, which returns True if the game is over and False if the game is still going:

```
def gameFinished(t1, t2, t3):
    t1_outside = abs(t1.xcor()) > w or abs(t1.ycor()) > h
    t2_outside = abs(t2.xcor()) > w or abs(t2.ycor()) > h
    t3_outside = abs(t3.xcor()) > w or abs(t3.ycor()) > h
    isOutside = t1_outside or t2_outside or t3_outside
    return isOutside
```

Command	*Purpose*
unsigned_number = abs(number)	A function that discards the sign of a number (before -5, after 5).
coord_x = t.xcor(), coord_y = t.ycor()	Functions that return the turtle's current X and Y coordinates.

Brainstorming

The "Catch the Turtle" game

Let's describe the gameFinished() function, which returns True if the game is over and False if the game is still going:

```
def gameFinished(t1, t2, t3):
    t1_outside = abs(t1.xcor()) > w or abs(t1.ycor()) > h
    t2_outside = abs(t2.xcor()) > w or abs(t2.ycor()) > h
    t3_outside = abs(t3.xcor()) > w or abs(t3.ycor()) > h
    isOutside = t1_outside or t2_outside or t3_outside
    return isOutside
```

`t1_outside` is **True** if the turtle has left the screen

If at least one turtle has left the screen, the variable **`isOutside`** = **`True`**.

↓

The game loop stops working!

Brainstorming

Connecting modules

Creating t1, t2, and t3

Event handler functions catch1(), catch2(), catch3()

Subscriptions to events "Click on the turtle 1, 2, 3"

GameFinished() function, which determines if one of the turtles has left the screen

```
while gameFinished(t1, t2, t3) != True:
  t1.forward(7)
  t2.forward(7)
  t3.forward(7)
  sleep(0.1)
```

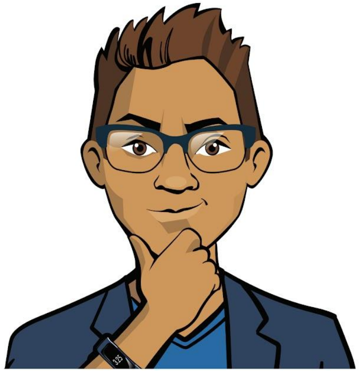
Brainstorming

The task :

Program your **second Catch the Turtle prototype**.

Use click handling on different turtles in the program.

Use the *documentation* if necessary.



Brainstorming