



Trinity College Dublin  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

pytch\_



## Lesson 6

Starting “catch the apple” game

Developed by:

pytch.team

<https://pytch.org/>

<https://pytch.scss.tcd.ie/>

# Pytch movements with sense method

You can use one of Pytch's sensing methods:

```
pytch.key_pressed(key_name)
```

Report **True** or **False** according to whether the key called **key\_name** is currently pressed.

- **Example of Pytch movement with sensing method:**

```
while True:
```

```
    if pytch.key_pressed("a"):  
        self.change_x(-3)
```

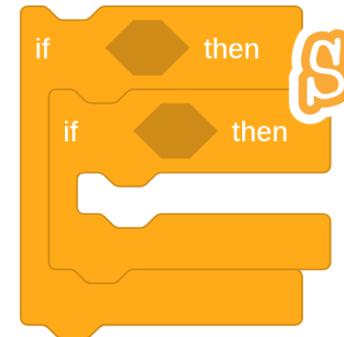
Forever, Pytch will check if the key "a" is pressed, and when it is pressed Pytch will move the Sprite to the left.



# Python nested conditions

You can use an “**if statement**” **inside another if statement** to check more than one condition before the execution of some code. This is called **nesting**.

```
if <condition_1>:  
    if <condition_2>:  
        code_to_run_if_both_conditions_true()
```



- **Example in Pytch:**

```
>>> if pytch.key_pressed("ArrowRight") :  
    if self.touching(Apple) :  
        self.say_for_seconds("An apple!", 2.0)  
        self.set_size(1)
```



# Python conditions: operators

The 'if' condition can use operators like:

- > (greater than): the condition is true if the first value is greater than the second.
- < (smaller than): the condition is true if the first value is smaller than the second.
- == (equal to): the condition is true if the first value is exactly equal to the second

```
if a > 0:
```

When the value of "a" is 1 or 2 or 3 etc, the condition is True

When the value of "a" is 0, -1 or -2 or -3 etc, the condition is False

## Example in Pytch

Each Sprite in Pytch has two built-in variables, `self.x_position` and `self.y_position`, that tell you where on the stage that Sprite is.

```
>>> if self.x_position > 0:
    self.say_for_seconds("I'm on right side of the screen!", 2.0)

if self.y_position < 0:
    self.say_for_seconds("I'm in the bottom half of the screen!", 2.0)
```



# Work in pairs – Worksheet 1

- What does this code do?
- Write your answers on worksheet 1

when green flag clicked

```
1 self.go_to_xy(0, -145)
2 while True:
3     if pytorch.key_pressed("ArrowLeft"):
4         if self.x_position > -190:
5             self.change_x(-2)
6
```



A Scratch sprite named 'Bowl' is shown. It features a brown bowl icon on a light gray background, with the word 'Bowl' written in white on a dark blue base.

when green flag clicked

```
1 self.go_to_xy(0, 200)
2 while True:
3     self.change_y(-3)
```



A Scratch sprite named 'Apple' is shown. It features a red apple icon on a light gray background, with the word 'Apple' written in white on a dark blue base.

# Try it out

- Follow this link to get a Pytch project that you can run
- Run the program
- Do the Bowl and the Apple Sprites do *exactly* what you thought they would do?
- If not:
  - Look at the differences
  - Correct your answer on worksheet 1

<https://pytch.org/app/lesson/sbys/6>



# Questions to do in pairs – Worksheet 2

1. Why do we use `if self.x_position > -190:` in the Bowl? What happens to the bowl movement if we don't do this check (try removing it)? What happen if we replace the value -190 with different numbers (for example, -100)?
2. What happens if you change the numbers in `self.change_x` (Bowl) and `self.change_y` (Apple)? What happen if you write `self.change_y` instead of `self.change_x` in Bowl?
3. What happen if you change the `while True` in the Apple's script to `while self.y_position > 0`? Why?
4. If you swap the order of the if statements in the Bowl like below what changes (and why)?

```
while True:  
    if self.x_position > -190  
        if pytch.key_pressed("ArrowLeft") :  
            self.change_x(-2)
```

<https://pytch.org/app/lesson/sbys/6>



# Tasks – Worksheet 3

Work in pairs on these activities:

1. Bowl: Add rightwards movement to the bowl, so that using the left and right arrows you can move the bowl left and right. Make sure the bowl can't move off the stage.
2. Apple: Increase the game difficulty by making the apple drop from a random x position.
  - NB: remember to select a good range for the values (not outside the stage)
3. Apple: Make the apple movement faster after it reaches the bottom half of the stage.

## Extension

Finished early? Challenge: How do we make the apple disappear when the bowl collects it?



- You can change the Apple code within the `while True` loop
- You can use the Touching sense introduced in the previous lesson: `self.touching(Object)`

When the Apple reaches the bottom of the screen it vanishes: can you add a script so that pressing a key (for example “d”) drops the apple again? What happens if you press “d” while the apple is already dropping? Can you think of a solution to this (this is quite hard!)?



# Recap

Today we have

1. Learned about another way to move Sprites in Pytch, and think about waiting for events or checking in our own code
2. Learned about *nesting* conditions in Python
3. Learned about some *operators* for doing calculations

In the next lesson we will finish our “Catch the apple” game, and finish our Python journey (for now?)

