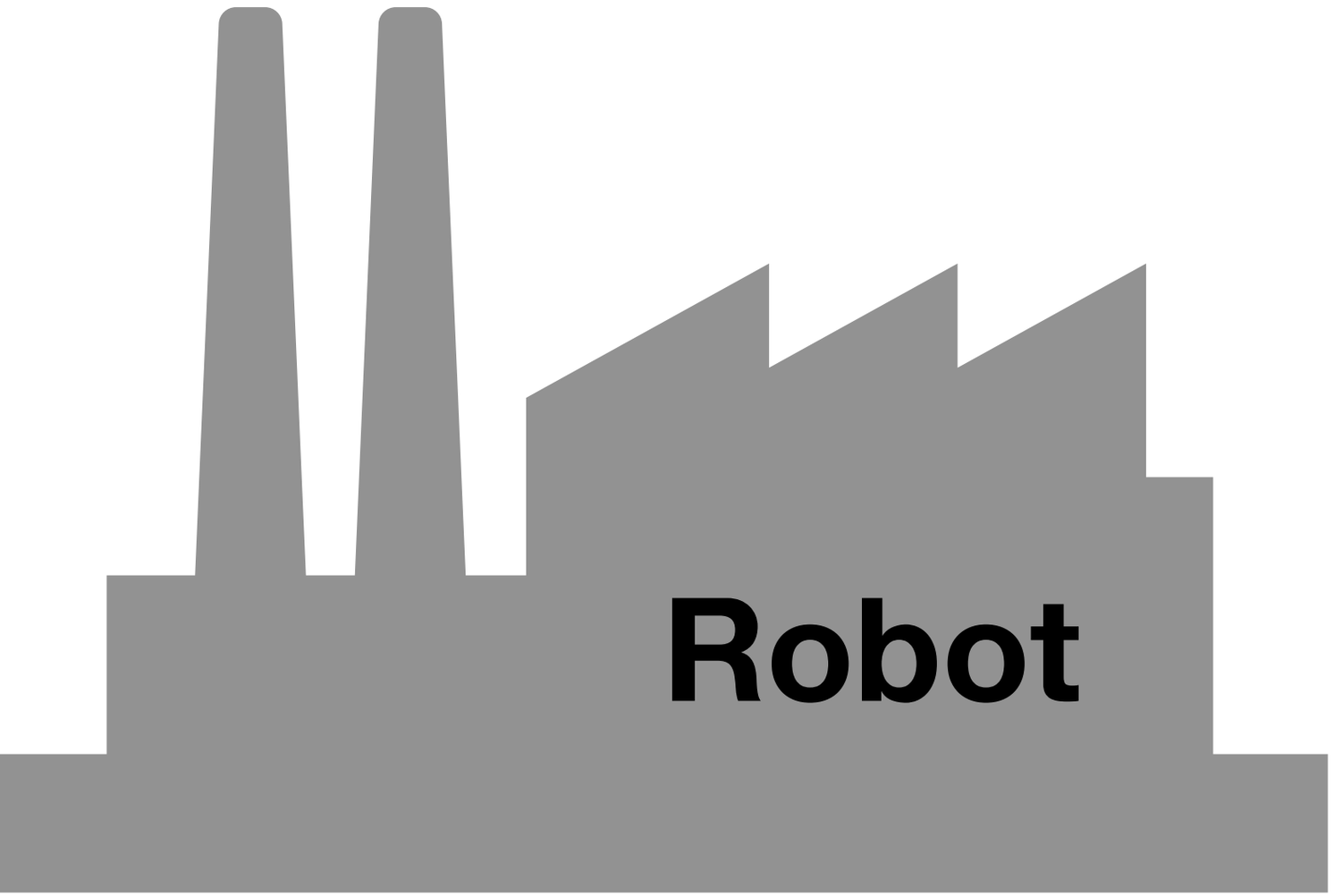
SC101

Week 2

```
r1 = Robot(183, 70, color='orange')
r2 = Robot(190, 80, color='red')
r3 = Robot(160, 50)
```



r3

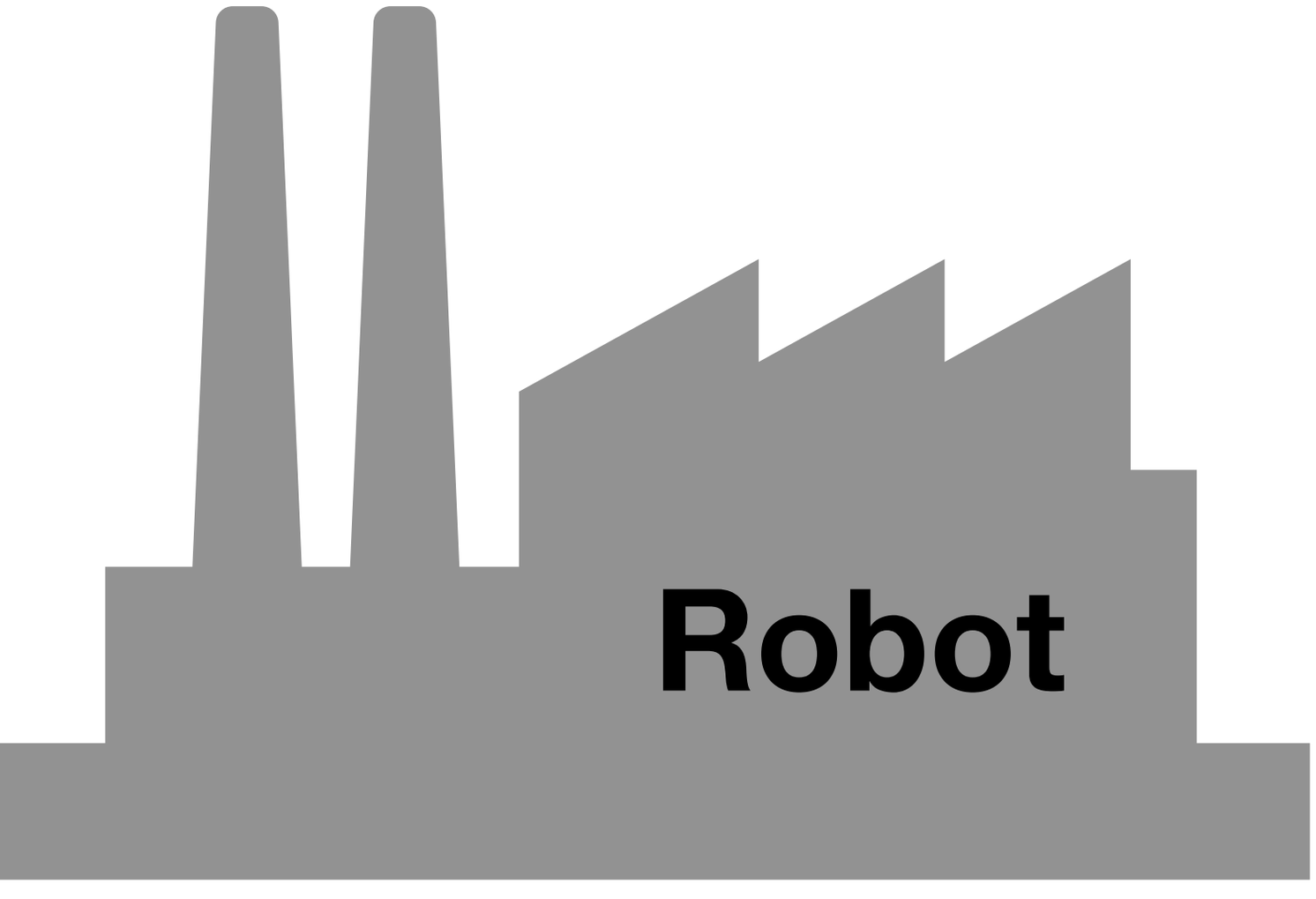
r1

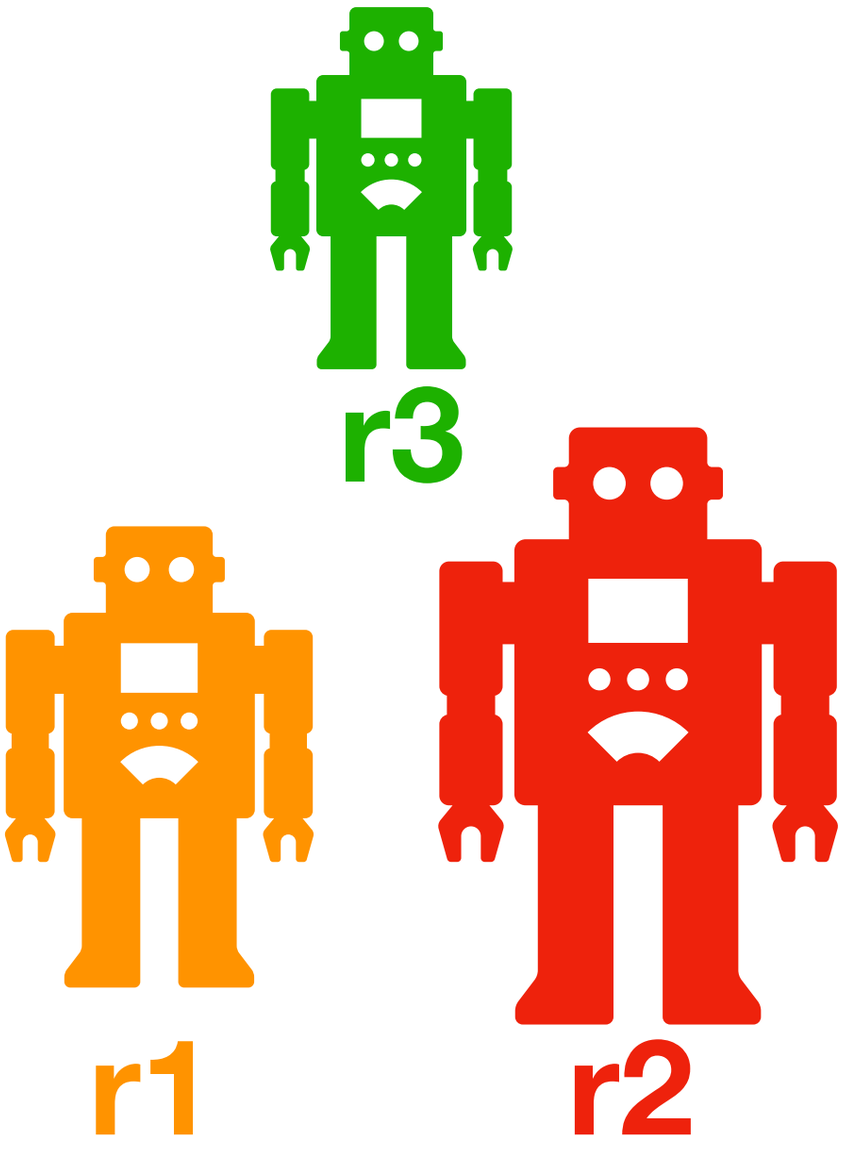
r2

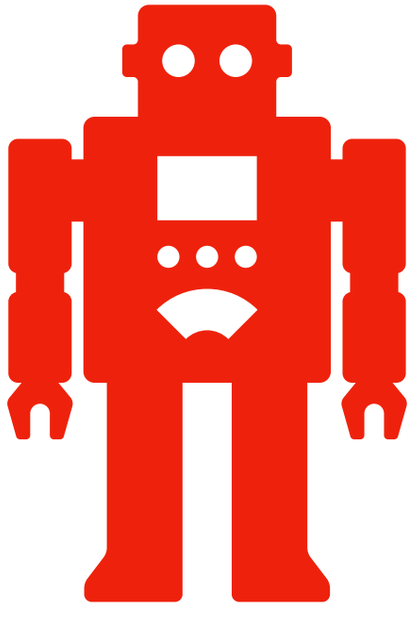
r1 = Robot(183, 70, color='orange')

r2 = Robot(190, 80, color='red')

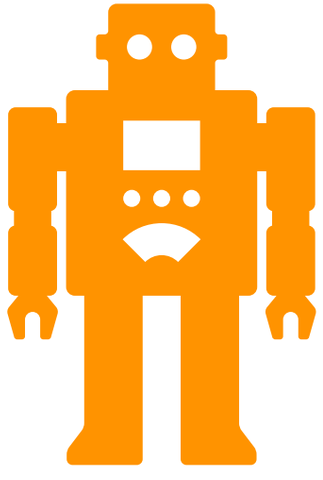
r3 = Robot(weight=50, height=160)



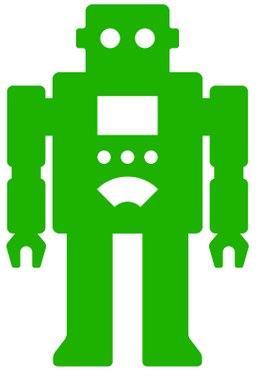




r2

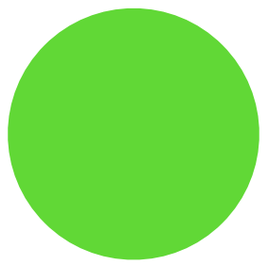


r1



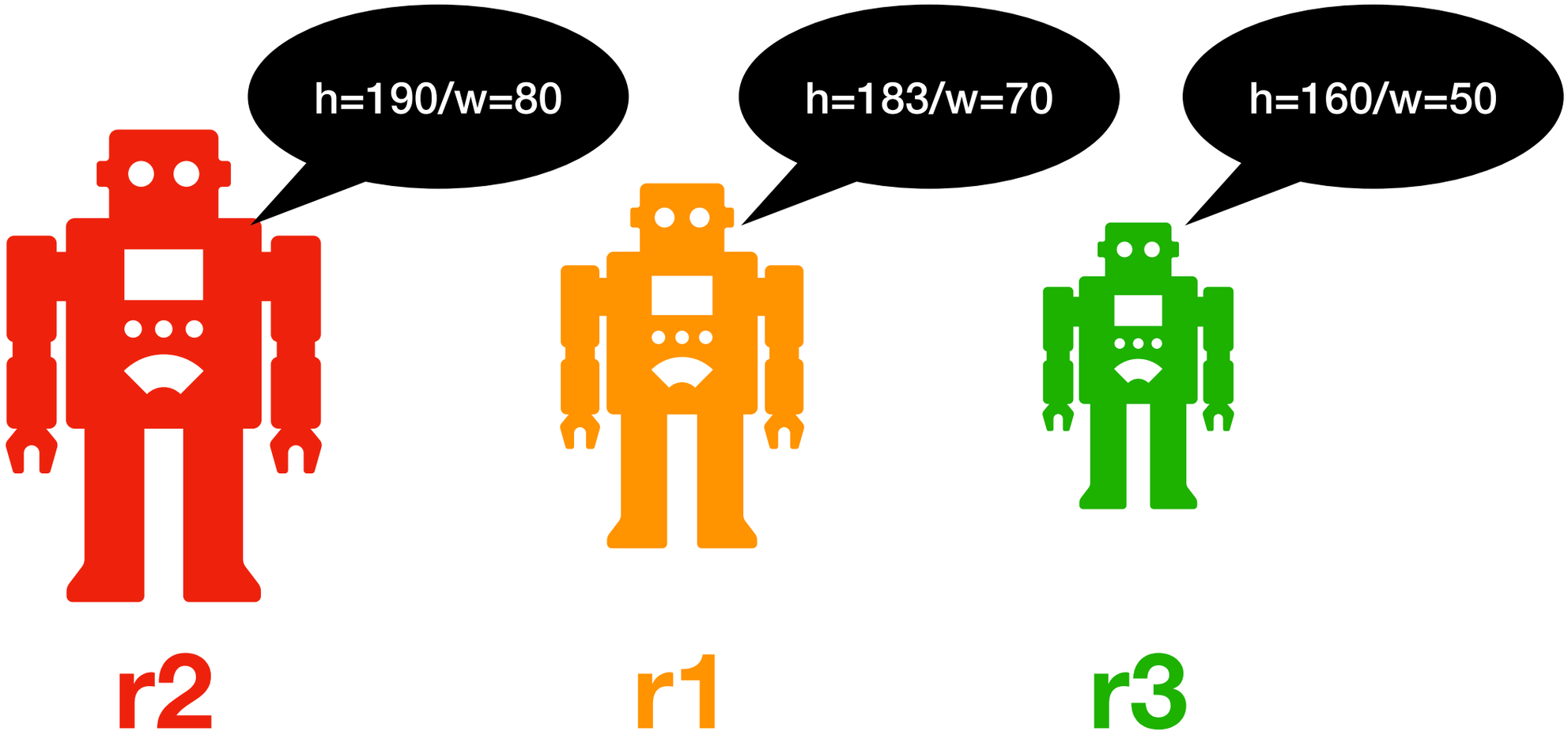
r3

oval = r3.give_me_a_ball(50)



oval = r1.give_me_a_ball(10)

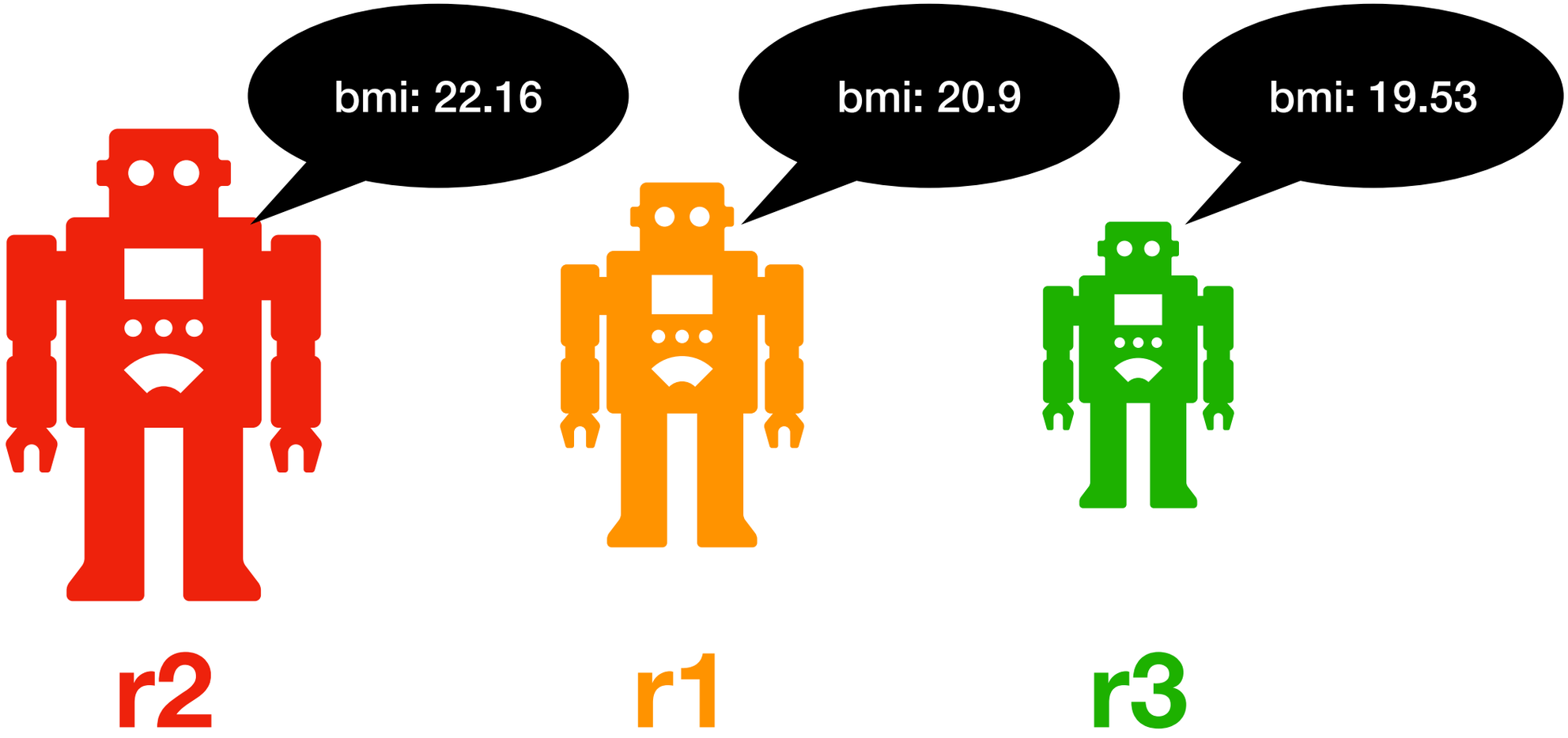
oval = r2.give_me_a_ball(20)



r3.self_introduce( )

r1.self_introduce( )

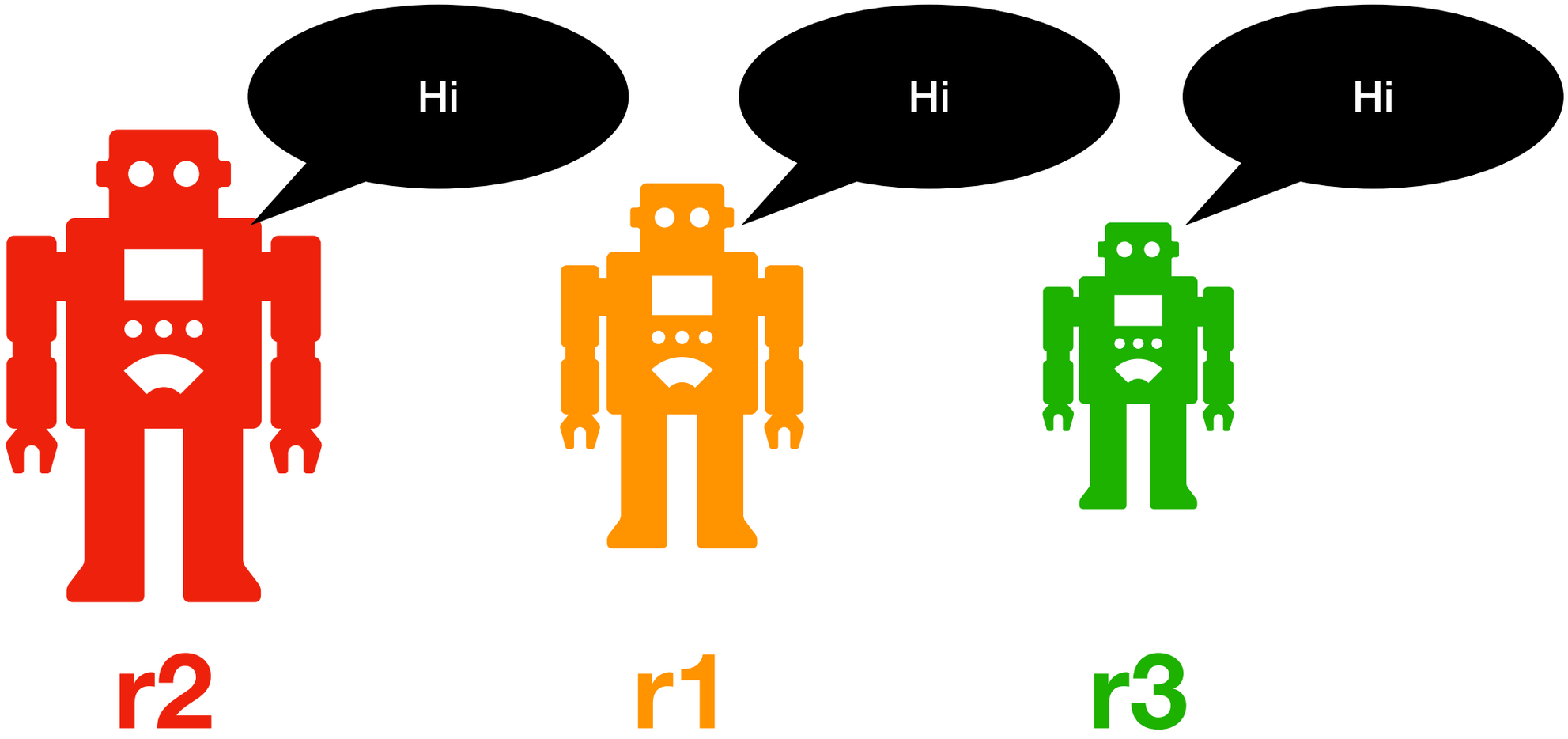
r2.self_introduce( )



r3.bmi( )

r1.bmi( )

r2.bmi( )



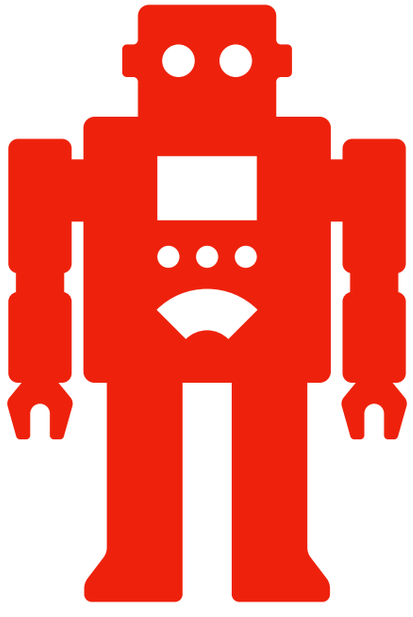
r3.say_hi( )

r1.say_hi( )

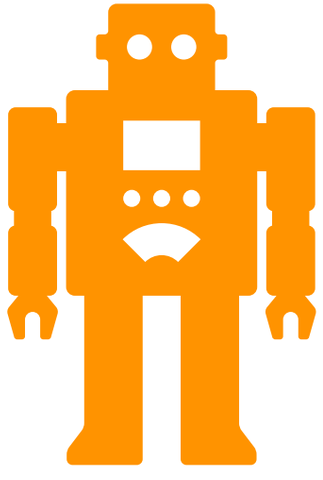
r2.say_hi( )

Let’s code it up!

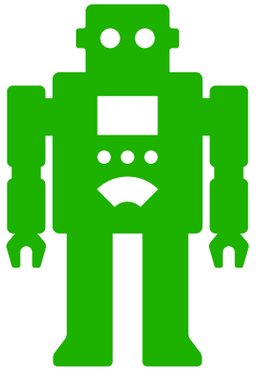
give_me_a_ball(50)



r2

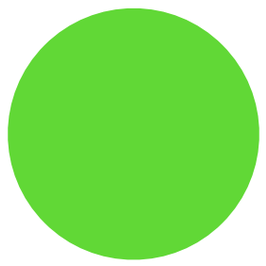


r1



r3

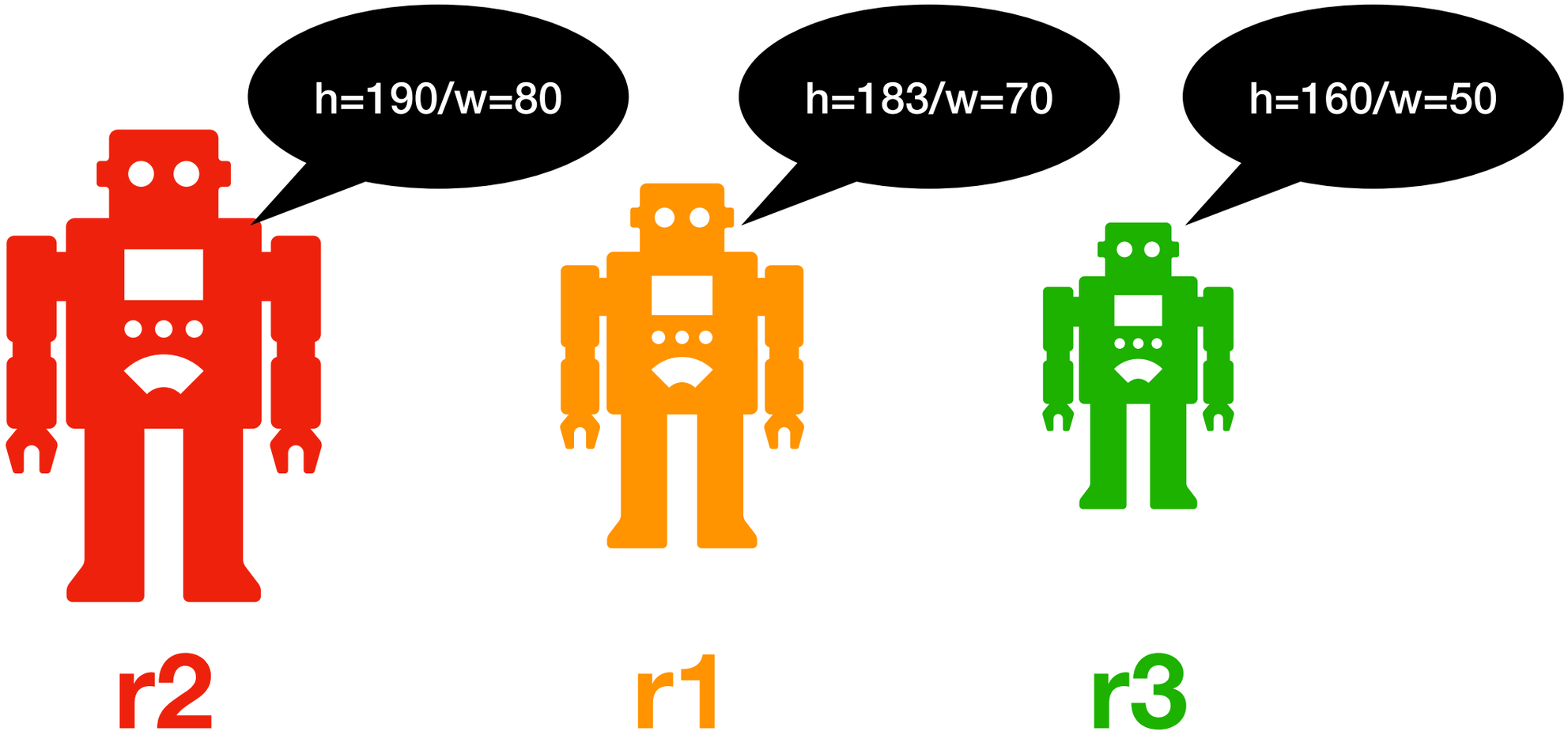
oval = r3.give_me_a_ball(50)



oval = r1.give_me_a_ball(10)

oval = r2.give_me_a_ball(20)

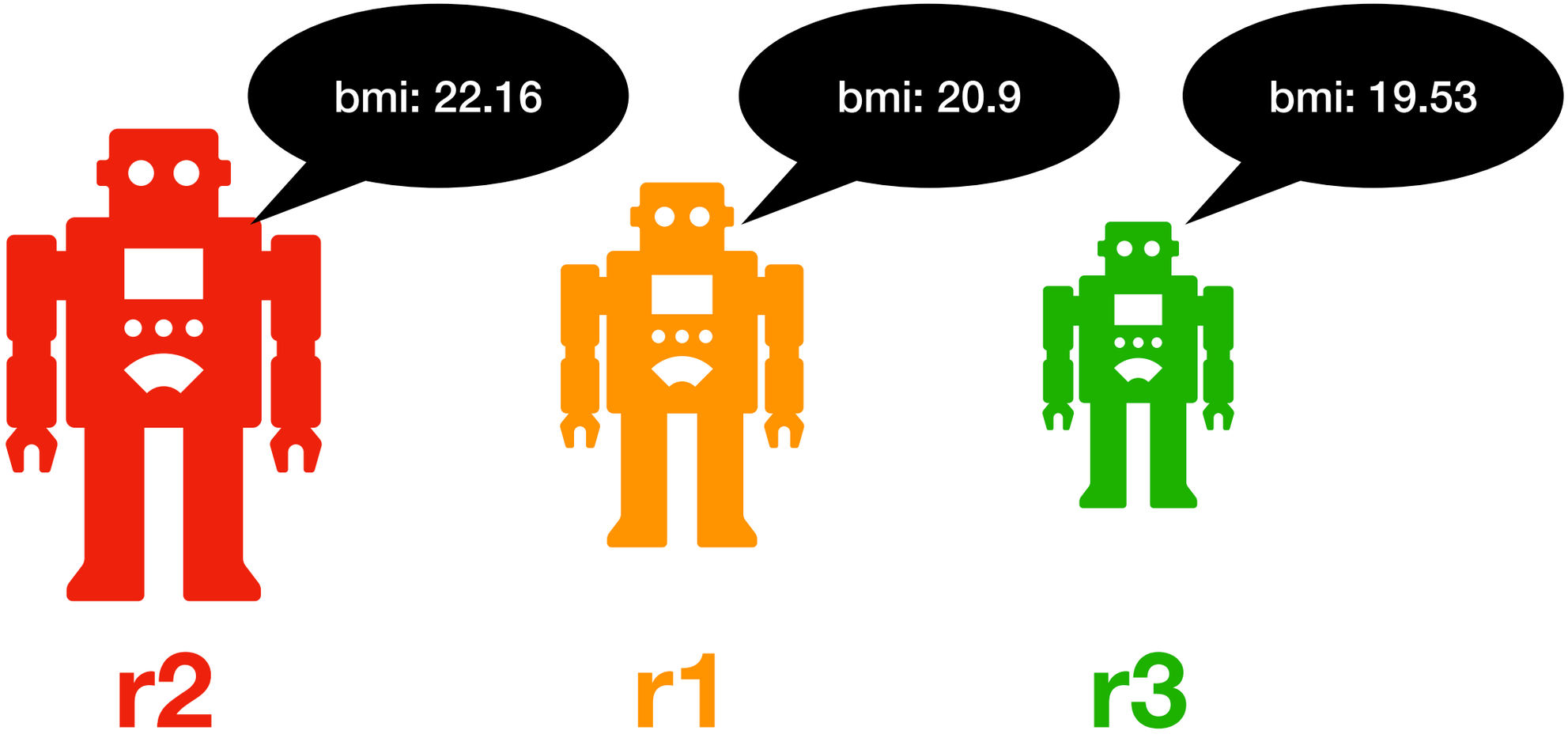
Python prints



r3.self_introduce( )

r1.self_introduce( )

r2.self_introduce( )

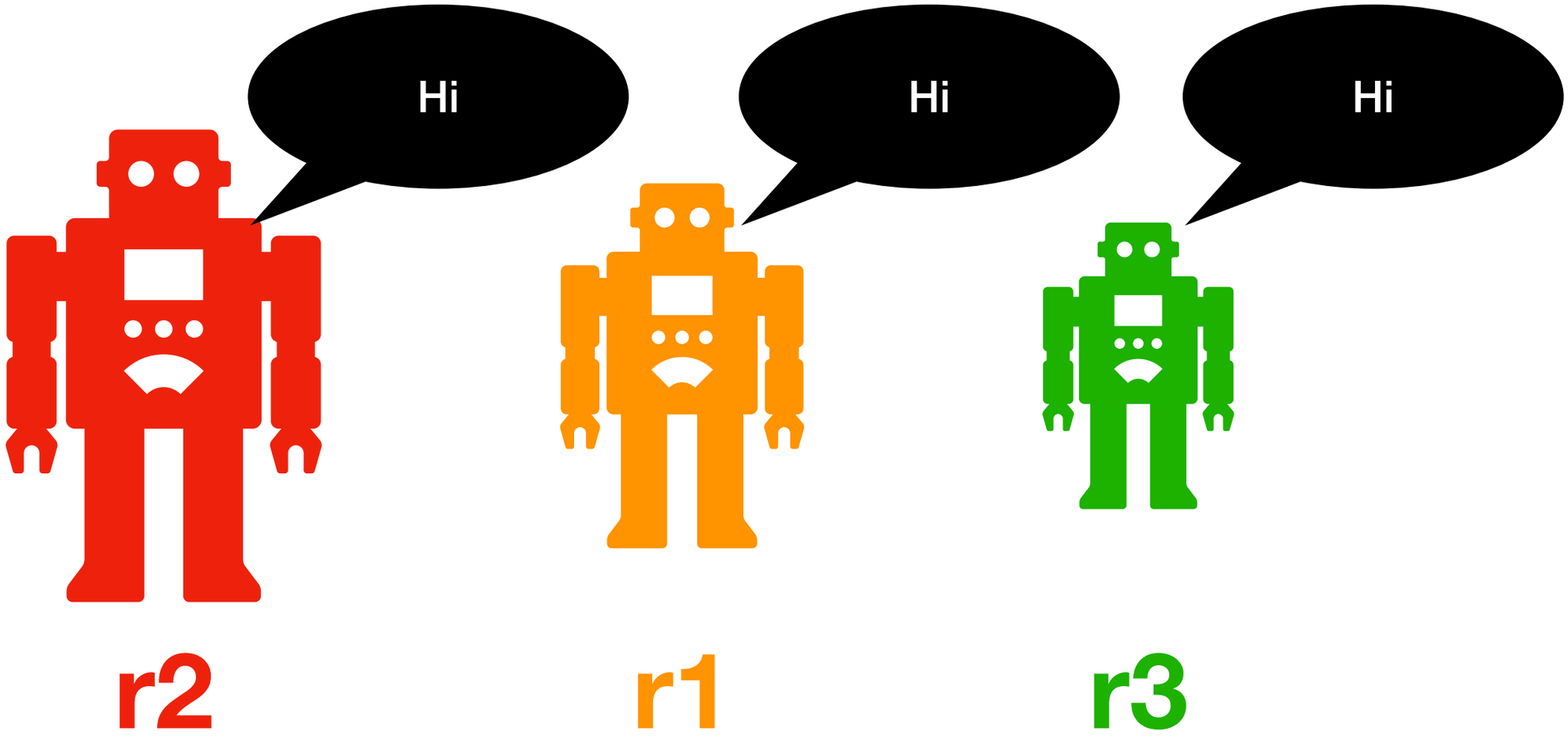


r3.bmi( )

r1.bmi( )

r2.bmi( )

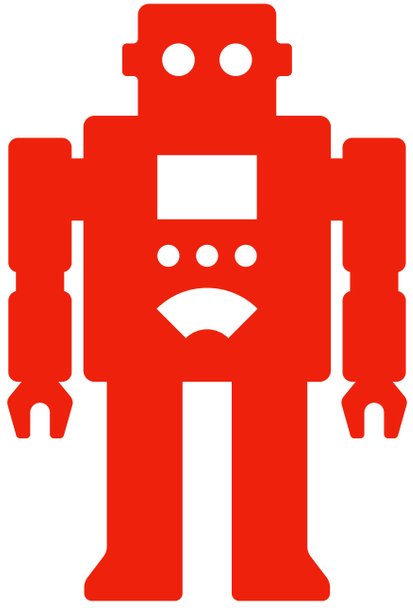
static method



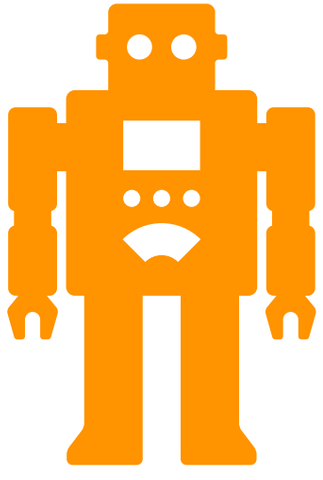
r3.say_hi( )

r1.say_hi( )

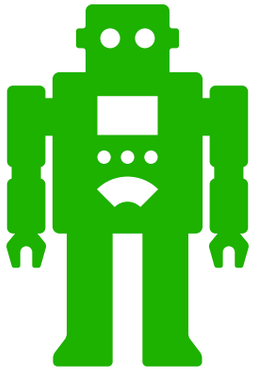
r2.say_hi( )



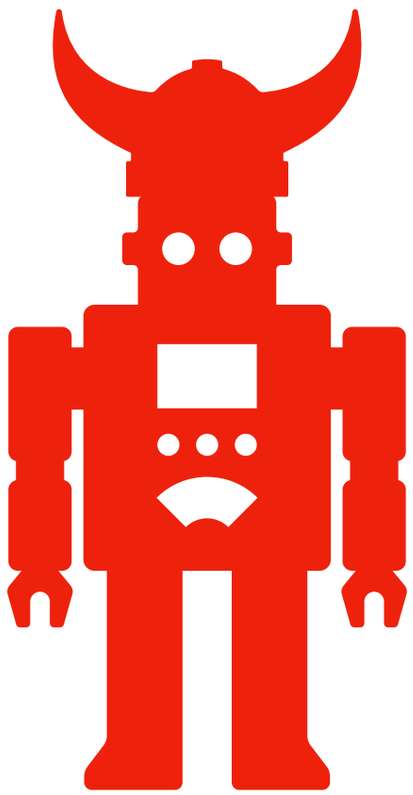
r2



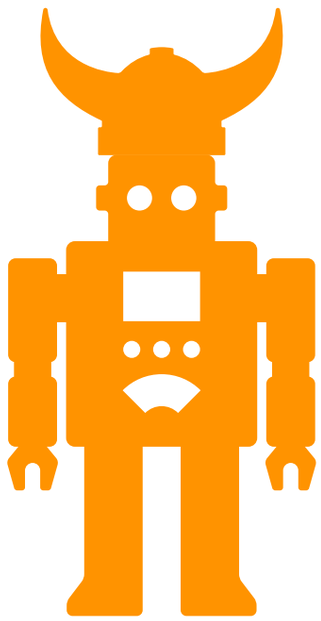
r1



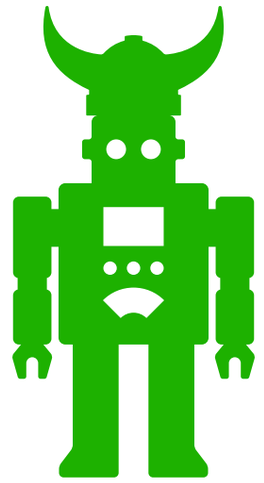
r3



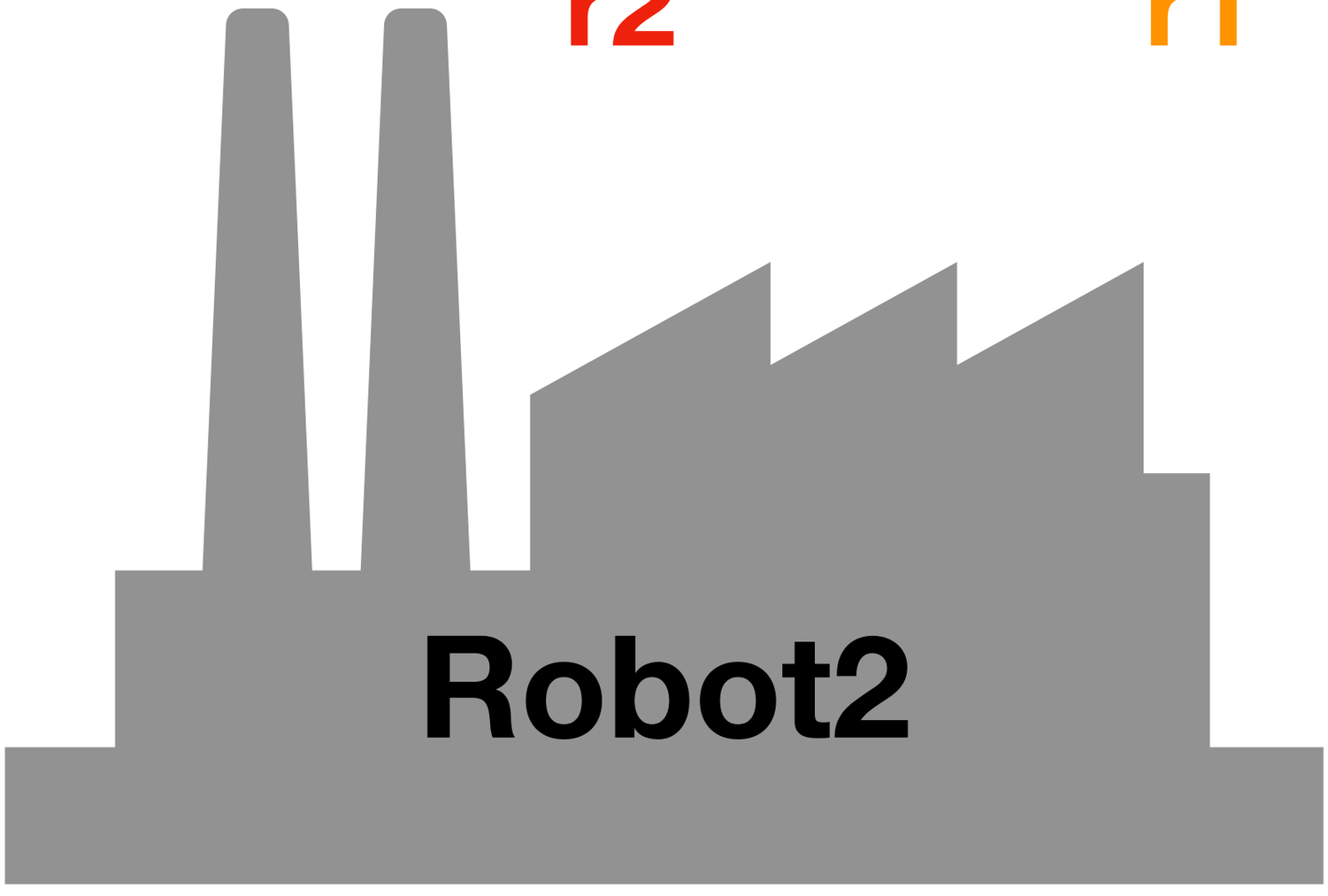
r2

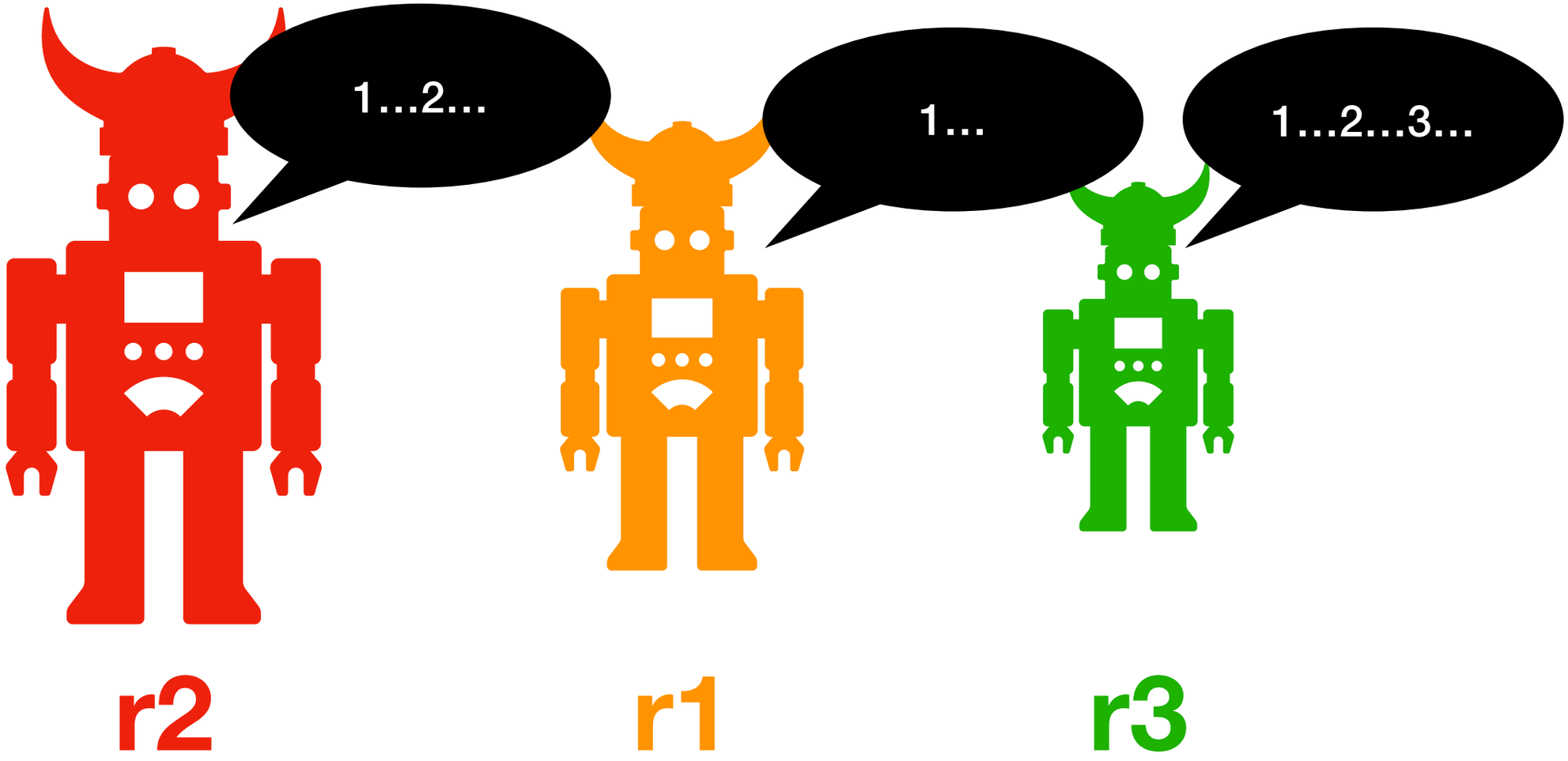


r1



r3





r3.start_count( )

r1.start_count( )

r2.start_count( )


PayPal

zone.py & zone_graphics.py

```python
from campy.gui.events.timer import pause
from zonegraphics_demo import ZoneGraphics

FRAME_RATE = 1000 / 120 # 120 frames per second.
NUM_LIVES = 3


def main():
    """
    This program plays a Python game 'zone'
    A ball will be bouncing around the GWindow
    Players must defend the zone indicated by black
    line at the middle of the GWindow by clicking on
    the bouncing ball
    """
    graphics = ZoneGraphics()
    lives = NUM_LIVES
    while True:
        if graphics.ball_in_zone():  # terminating condition
            lives -= 1
            if lives > 0:
                graphics.reset_ball()
            else:
                break                # stop looping if we've lost 3 lives

        graphics.move_ball()
        graphics.handle_wall_collisions()
        pause(FRAME_RATE)
```

```
class ZoneGraphics:

    def __init__(self, window_width=WINDOW_WIDTH, window_height=WINDOW_HEIGHT,
                 zone_width=ZONE_WIDTH, zone_height=ZONE_HEIGHT, ball_radius=BALL_RADIUS):
        # Create window
        self.window = GWindow(width=window_width, height=window_height, title='Zone Game')

        # Create zone
        self.zone = GRect(width=zone_width, height=zone_height, x=(window_width - zone_width) / 2,
                          y=(window_height - zone_height) / 2)
        self.zone.color = 'blue'
        self.window.add(self.zone)

        # Create ball and initialize velocity/position
        self.ball = GOval(width=ball_radius*2, height=ball_radius*2)
        self.ball.filled = True
```

```
        self.dx = 0
        self.dy = 0

        self.reset_ball()

        # Initialize mouse listeners
        onmouseclicked(self.handle_click)

    def set_ball_position(self):
        """
        Sets the ball position to a random x, y where ball contained in window.
        """
        self.ball.x = random.randint(0, self.window.width - self.ball.width)
        self.ball.y = random.randint(0, self.window.height - self.ball.height)
```

```
def set_ball_velocity(self):
    """
    Sets ball x velocity to random negative or positive number.
    Sets ball y velocity to random positive number.
    """
    self.dx = random.randint(0, MAX_SPEED)
    if random.random() > 0.5:
        self.dx = -self.dx
    self.dy = random.randint(MIN_Y_SPEED, MAX_SPEED)
    if random.random() > 0.5:
        self.dy = -self.dy

def reset_ball(self):
    """
    Sets the ball in a new position and new velocity. Displays in window.
    """
    self.set_ball_position()
    while self.ball_in_zone():
        self.set_ball_position()
    self.set_ball_velocity()
    self.window.add(self.ball)

def move_ball(self):
    """
    Moves ball by the change in x and change in y stored in ZoneGraphics class.
    """
    self.ball.move(self.dx, self.dy)
```

```
def handle_wall_collisions(self):
    """
    Updates dx and dy depending on whether or not ball has hit a wall.
    """
    if self.ball.x <= 0 or self.ball.x >= self.window.width - self.ball.width:
        self.dx = -self.dx
    if self.ball.y <= 0 or self.ball.y >= self.window.height - self.ball.height:
        self.dy = -self.dy

def ball_in_zone(self):
    """
    Returns whether or not the ball is completely contained within zone.
    """
    zone_left_side = self.zone.x
    zone_right_side = self.zone.x + self.zone.width
    ball_x_in_zone = zone_left_side <= self.ball.x <= zone_right_side - self.ball.width

    zone_top = self.zone.y
    zone_bottom = self.zone.y + self.zone.height
    ball_y_in_zone = zone_top <= self.ball.y <= zone_bottom - self.ball.height

    return ball_x_in_zone and ball_y_in_zone

def handle_click(self, event):
    """
    Resets the ball if the ball was clicked.

    Input:
        event (GMouseEvent): mouse clicked event
    """
    obj = self.window.get_object_at(event.x, event.y)
    if self.ball == obj:
        self.reset_ball()
```