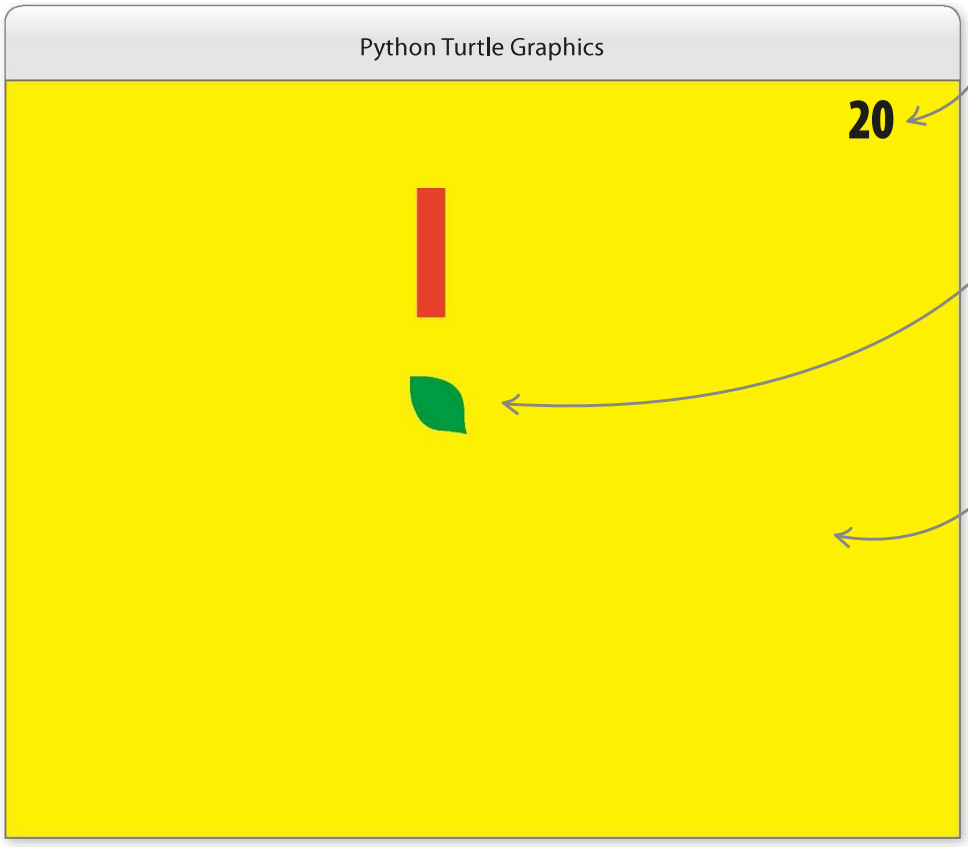
Caterpillar

If all this coding has worked up your appetite, you're not alone—the star of this project is a hungry caterpillar. Using Python's `turtle` module, you'll find out how to animate game characters and control them on screen with the keyboard.

What happens

You use the four arrow keys to steer a caterpillar around the screen and make it "eat" leaves. Each leaf gives you a point, but it also makes the caterpillar bigger and faster, making the game harder. Keep the caterpillar inside the game window, or the game's over!

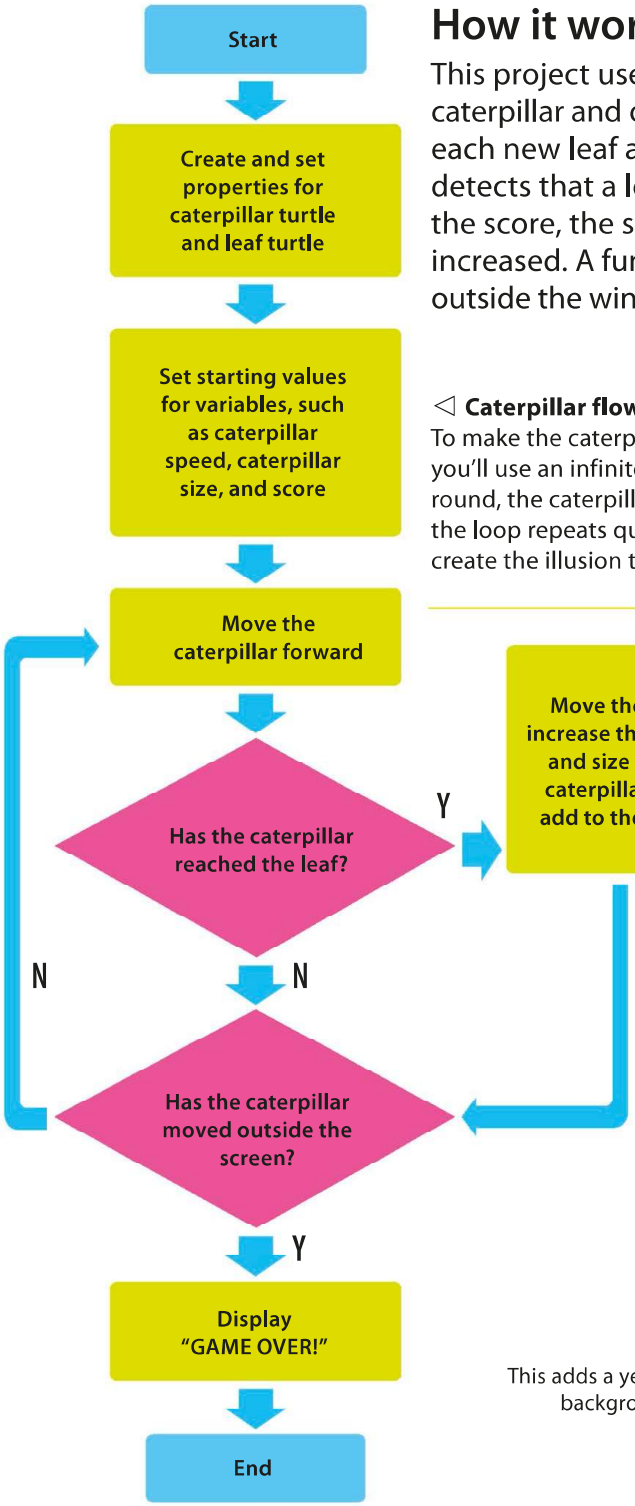


Your score is displayed in the top-right corner of the game window.

The leaf disappears when eaten, and a new leaf then appears elsewhere.

To start the game, the player has to click on the screen first and then press the space bar.

◁ **Increasing difficulty**
The more leaves the caterpillar eats, the harder the game becomes. As the caterpillar gets longer and faster, your reactions have to speed up too; otherwise, your caterpillar will zoom off the screen.



How it works

This project uses two main turtles: one to draw the caterpillar and one to draw the leaves. The code places each new leaf at a random location. When the program detects that a leaf has been eaten, the variables storing the score, the speed of the caterpillar, and its length are increased. A function figures out if the caterpillar has moved outside the window, which would signal the end of the game.

◁ **Caterpillar flowchart**
To make the caterpillar move across the screen you'll use an infinite loop. Each time the loop goes round, the caterpillar moves forward slightly. When the loop repeats quickly, these small movements create the illusion that your caterpillar is crawling.

First steps

For such a fun game, the code is surprisingly straightforward. You'll start by setting up the turtles, before moving on to the main game loop and finally the keyboard controls.

1 Getting started
Open IDLE and create a new file. Save it as "caterpillar.py".

2 Import the modules
Add these two `import` statements to tell Python that you need the `turtle` and `random` modules. The third line sets the background color for the game window.

This adds a yellow background.

```
import random
import turtle as t

t.bgcolor('yellow')
```

3 Create a caterpillar turtle

Now create the turtle that will become your caterpillar. Add the code shown here. It creates the turtle and sets its color, shape, and speed. The function `caterpillar.penup()` disables the turtle's pen, allowing you to move the turtle around the screen without drawing a line along the way.

```
caterpillar = t.Turtle()
caterpillar.shape('square')
caterpillar.color('red')
caterpillar.speed(0)
caterpillar.penup()
caterpillar.hideturtle()
```

Create a new turtle for the caterpillar.

We don't want the turtle to move before the game starts.

This command hides the turtle.

4 Create a leaf turtle

Below the code for Step 3, type these lines to set up the second turtle, which will draw the leaves. The code uses a list of six coordinate pairs to draw a leaf shape. Once you tell the turtle about this shape, it can reuse the details to draw more leaves. A call to `hideturtle` here makes this turtle invisible on the screen.

This turtle will draw the leaves.

The coordinates for leaf shape

```
leaf = t.Turtle()
leaf_shape = ((0, 0), (14, 2), (18, 6), (20, 20), \
              (6, 18), (2, 14))
t.register_shape('leaf', leaf_shape)
leaf.shape('leaf')
leaf.color('green')
leaf.penup()
leaf.hideturtle()
leaf.speed(0)
```

Use a backslash character if you need to split a long line of code over two lines.

This line tells the turtle about the leaf shape.

5 Add some text

Now set up two more turtles to add text to the game. One will display a message before the action starts, telling players to press the space bar to begin. The other will write the score in the corner of the window. Add these lines after the leaf turtle code.

You'll need to know later if the game has started.

```
game_started = False
text_turtle = t.Turtle()
text_turtle.write('Press SPACE to start', align='center',\
                  font=('Arial', 16, 'bold'))
text_turtle.hideturtle()

score_turtle = t.Turtle()
score_turtle.hideturtle()
score_turtle.speed(0)
```

This line draws some text on the screen.

This hides the turtle but not the text.

Add a turtle to write the score.

The turtle needs to stay where it is, so that it can update the score.

Main loop

Your turtles are now set up and ready to go. Let's write the code that makes the game come to life.

6 Placeholder functions

You can put off defining a function until later by using the `pass` keyword. Under the code for the turtles, add the following placeholders for functions that you'll fill with code in later steps.

```
def outside_window():
    pass

def game_over():
    pass

def display_score(current_score):
    pass

def place_leaf():
    pass
```

To get a basic version of the program running sooner, you can use placeholders for functions that you'll finish coding later.

EXPERT TIPS

Pass

In Python, if you're not yet sure what code you want inside a function, you can just type in the `pass` keyword and then come back to it later. It's a bit like passing on a question in a quiz.

7 Game starter

After the four placeholder functions comes the `start_game()` function, which sets up some variables and prepares the screen before the main animation loop begins. You'll add the code for the main loop, which forms the rest of this function, in the next step.

```
def start_game():
    global game_started
    if game_started:
        return
    game_started = True

    score = 0
    text_turtle.clear()

    caterpillar_speed = 2
    caterpillar_length = 3
    caterpillar.shapesize(1, caterpillar_length, 1)
    caterpillar.showturtle()
    display_score(score)
    place_leaf()
```

If the game has already started, the return command makes the function quit so it doesn't run a second time.

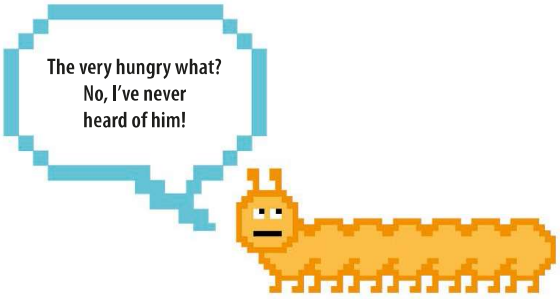
Clear the text from the screen.

The turtle stretches into a caterpillar shape.

This line places the first leaf on the screen.

8 Get moving

The main loop moves the caterpillar forward slightly, before performing two checks. It first checks if the caterpillar has reached the leaf. If the leaf has been eaten, the score increases, a new leaf gets drawn, and the caterpillar gets longer and faster. The loop then checks if the caterpillar has left the window—if so, the game's over. Add the main loop below the code you typed in Step 7.




```
place_leaf()


while True:
    caterpillar.forward(caterpillar_speed)
    if caterpillar.distance(leaf) < 20:
        place_leaf()
        caterpillar_length = caterpillar_length + 1
        caterpillar.shapesize(1, caterpillar_length, 1)
        caterpillar_speed = caterpillar_speed + 1
        score = score + 10
        display_score(score)
    if outside_window():
        game_over()
        break
```

The caterpillar eats the leaf when it's less than 20 pixels away.

The current leaf has been eaten, so add a new leaf.

This will make the caterpillar grow longer.

9 Bind and listen

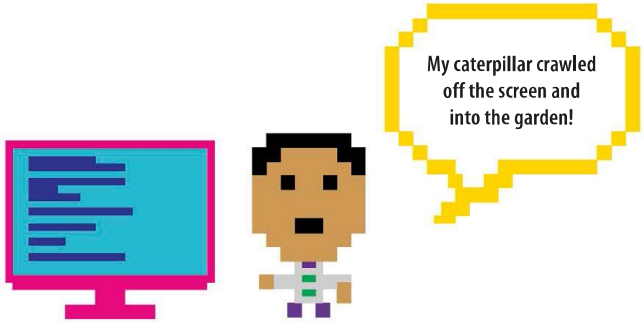
Now put these lines below the function you've just created. The `onkey()` function binds the space bar to `start_game()`, so you can delay the start until the player presses space. The `listen()` function allows the program to receive signals from the keyboard.

```
t.onkey(start_game, 'space')
t.listen()
t.mainloop()
```

When you press the space bar, the game begins.

10 Test your code

Run the program. If your code is correct, you should see the caterpillar moving after you press the space bar. Eventually, it should crawl off the screen. If the program doesn't work, check your code carefully for bugs.

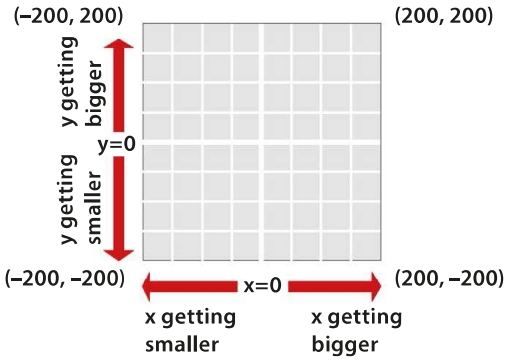


Filling in the blanks

It's time to replace `pass` in the placeholder functions with actual code. After adding the code for each function, run the game to see what difference it makes.

11 Stay inside

Fill the `outside_window()` function with this code. First it calculates the position of each wall. Then it asks the caterpillar for its current position. By comparing the caterpillar's coordinates with the coordinates of the walls, it can tell whether the caterpillar has left the window. Run the program to check the function works—the caterpillar should stop when it reaches the edge.




```
def outside_window():
    left_wall = -t.window_width() / 2
    right_wall = t.window_width() / 2
    top_wall = t.window_height() / 2
    bottom_wall = -t.window_height() / 2
    (x, y) = caterpillar.pos()
    outside = \
            x< left_wall or \
            x> right_wall or \
            y< bottom_wall or \
            y> top_wall
    return outside
```

This function returns two values (a "tuple").

If any of the four conditions above is True, then `outside` is True.

◁ How it works

The center of the window has the coordinates (0, 0). Since the window is 400 wide, the right wall is half the width from the center, which is 200. The code gets the left wall's position by subtracting half the width from 0. In other words, 0–200, which is –200. It finds the position of the top and bottom walls by a similar method.

12 GAME OVER!

When the caterpillar has left the screen, display a message to tell the player the game has ended. Fill in the `game_over()` function with this code. When called, the function will hide the caterpillar and leaf, and write "GAME OVER!" on the screen.


```
def game_over():
    caterpillar.color('yellow')
    leaf.color('yellow')
    t.penup()
    t.hideturtle()
    t.write('GAME OVER!', align='center', font=('Arial', 30, 'normal'))
```

The text should be centered.

13 Show the score

The function `display_score()` instructs the score turtle to rewrite the score, putting the latest total on the screen. This function is called whenever the caterpillar reaches a leaf.

```
def display_score(current_score):
    score_turtle.clear()
    score_turtle.penup()
    x = (t.window_width() / 2) - 50
    y = (t.window_height() / 2) - 50
    score_turtle.setpos(x, y)
    score_turtle.write(str(current_score), align='right', \
                       font=('Arial', 40, 'bold'))
```

50 pixels from the right

50 pixels from the top

14 A new leaf

When a leaf is reached, the function `place_leaf()` is called to move the leaf to a new, random location. It chooses two random numbers between –200 and 200. These numbers become the x and y coordinates for the next leaf.

```
def place_leaf():
    leaf.ht()
    leaf.setx(random.randint(-200, 200))
    leaf.sety(random.randint(-200, 200))
    leaf.st()
```

`ht` is short for hideturtle.

Chooses random coordinates to move the leaf.

`st` is short for showturtle.

15 Turning the caterpillar

Next, to connect the keyboard keys to the caterpillar, add four new direction functions after the `start_game()` function. To make this game a little trickier, the caterpillar can only make 90-degree turns. As a result, each function first checks to see which way the caterpillar is moving before altering its course. If the caterpillar's going the wrong way, the function uses `setheading()` to make it face the right direction.

```
            game_over()
            break

def move_up():
    if caterpillar.heading() == 0 or caterpillar.heading() == 180:
        caterpillar.setheading(90)

def move_down():
    if caterpillar.heading() == 0 or caterpillar.heading() == 180:
        caterpillar.setheading(270)

def move_left():
    if caterpillar.heading() == 90 or caterpillar.heading() == 270:
        caterpillar.setheading(180)

def move_right():
    if caterpillar.heading() == 90 or caterpillar.heading() == 270:
        caterpillar.setheading(0)
```

Check if the caterpillar is heading left or right.

A heading of 270 sends the caterpillar down the screen.

16 Listening for presses

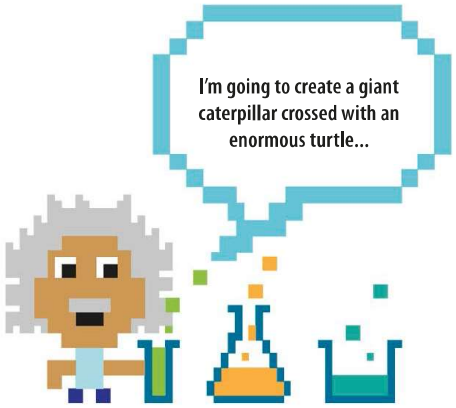
Finally, use `onkey()` to link the direction functions to the keyboard keys. Add these lines after the `onkey()` call you made in Step 9. With the steering code in place, the game's complete. Have fun playing and finding out your highest score!

```
t.onkey(start_game, 'space')
t.onkey(move_up, 'Up')
t.onkey(move_right, 'Right')
t.onkey(move_down, 'Down')
t.onkey(move_left, 'Left')
t.listen()
```

Call the `move_up` function when the "up" key is pressed.

Hacks and tweaks

Now that your caterpillar game is working, it won't be too difficult to modify it or even introduce a helper or rival caterpillar!



Make it a two-player game

By creating a second caterpillar turtle with separate keyboard controls, you and a friend can work together to make the caterpillar eat even more leaves!

1 Create a new caterpillar

First you'll need to add a new caterpillar. Type these lines near the top of your program, below the code that creates the first caterpillar.

```
caterpillar2 = t.Turtle()
caterpillar2.color('blue')
caterpillar2.shape('square')
caterpillar2.penup()
caterpillar2.speed(0)
caterpillar2.hideturtle()
```

2 Add a parameter

To reuse the `outside_window()` function for both caterpillars, add a parameter to it. Now you can tell it which caterpillar you want it to check on.

```
def outside_window(caterpillar):
```

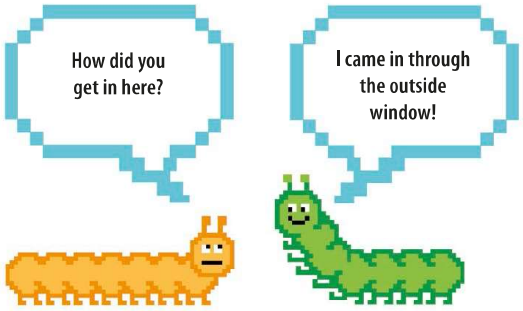
3 Hide caterpillar2

When the `game_over()` function is called, it hides the first caterpillar. Let's add a line to hide the second caterpillar as well.

```
def game_over():
    caterpillar.color('yellow')
    caterpillar2.color('yellow')
    leaf.color('yellow')
```

4 Change the main function

You'll need to add code for caterpillar2 to the main `start_game()` function. First set its starting shape and make it face the opposite direction from the first caterpillar. Then add it to the `while` loop to make it move, and add a check to the `if` statement so it can eat the leaves. You'll also need to add a line to make it grow. Finally, edit the call to the `outside_window()` function in your second `if` statement to see if the game is over.




```
score = 0
text_turtle.clear()

caterpillar_speed = 2
caterpillar_length = 3
caterpillar.shapesize(1, caterpillar_length, 1)
caterpillar.showturtle()
caterpillar2.shapesize(1, caterpillar_length, 1)
caterpillar2.setheading(180)
caterpillar2.showturtle()
display_score(score)
place_leaf()

while True:
    caterpillar.forward(caterpillar_speed)
    caterpillar2.forward(caterpillar_speed)
    if caterpillar.distance(leaf) < 20 or leaf.distance(caterpillar2) < 20:
        place_leaf()
        caterpillar_length = caterpillar_length + 1
        caterpillar.shapesize(1, caterpillar_length, 1)
        caterpillar2.shapesize(1, caterpillar_length, 1)
        caterpillar_speed = caterpillar_speed + 1
        score = score + 10
        display_score(score)
    if outside_window(caterpillar) or outside_window(caterpillar2):
        game_over()
```

This sets caterpillar2's starting shape.

Caterpillar2 starts heading left.

Each time the program loops, caterpillar2 moves forward.

This checks if caterpillar2 has eaten the leaf.

Caterpillar2 gets longer.

Has caterpillar2 left the screen?