RCAP CoSpace Autonomous Driving (Useful Functions)

A POSTERIORI

Play · Experience · Learn

Competition Timeline

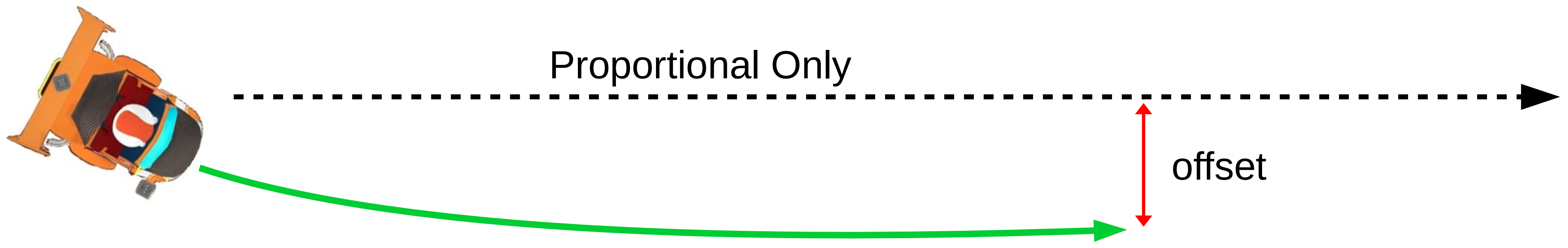
- 22 May (Team Description and Video)
  - Submit Team Description Paper & Video
  - Template will be provided by email
- 23 to 26 May (Warm-up)
  - Warm up exercises (...not graded)
  - Helps you familiarize yourself with competition procedure
- 29 May (Preliminary games) (Saturday)
  - Given a fixed time to solve challenge map
  - Do from home
  - Details to be sent via email

Competition Timeline

- 31 May (Announcement of Finalist)
  - Notified via email
- Finalists: 3 Jun (Video submission)
  - Another video. This time describing the game strategy
- Finalists: 5 - 9 Jun (Interview)
  - Interview via Zoom
- Finalists: 10 Jun (Announcement of selected students)
- Finalist: 12 Jun (Grand Finals)

Gyro Follower

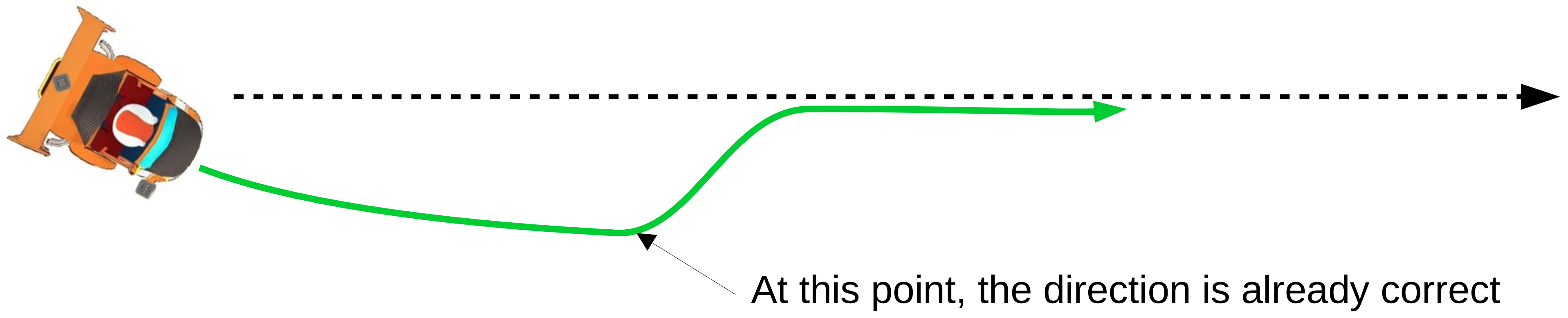
- We did this already, but…



- ...the gyro follower only corrects the heading, not the offset

Gyro Follower

- We want this...



- ...but the gyro only tells us the <u>direction</u> the robot is facing, not the <u>position</u>
- So how?

Integral Control

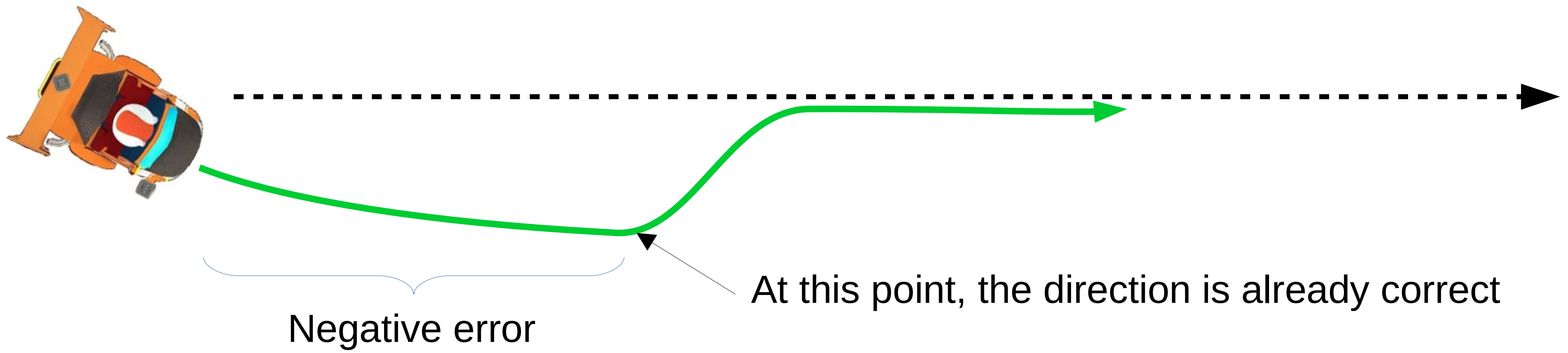
- We've previously looked at Proportional and Derivative

- Proportional looks at the error
- Derivative looks at rate of change of error
- Integral looks at the accumulated error

- How to find accumulated error?

Accumulated Error

- Error is negative for some time at the start...



- To accumulate the error, we just need to add it up...

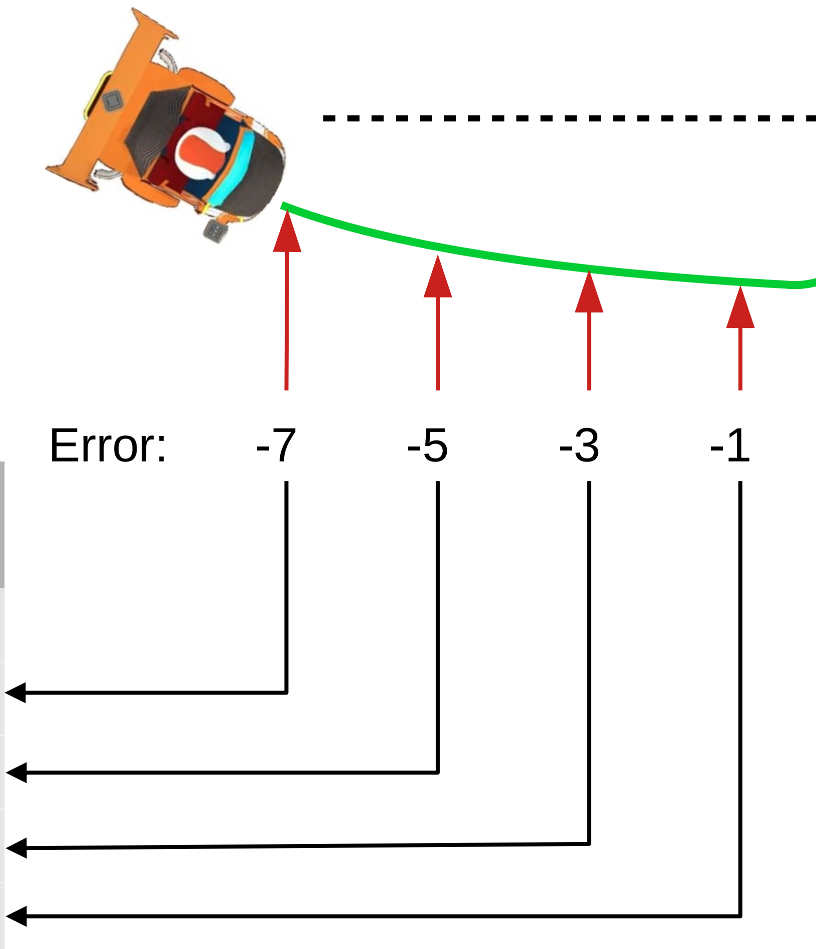
```
static int accumulated_error = 0;

// Calculate error here

accumulated_error += error;
```

static means that the variable will only be set the first time the function is executed

Add error to accumulated_error

Accumulated Error

- How it works...



Executed code	Value of accumulated_error
`static accumulated_error = 0;`	0
`accumulated_error += error;`	-7
`accumulated_error += error;`	-12
`accumulated_error += error;`	-15
`accumulated_error += error;`	-16

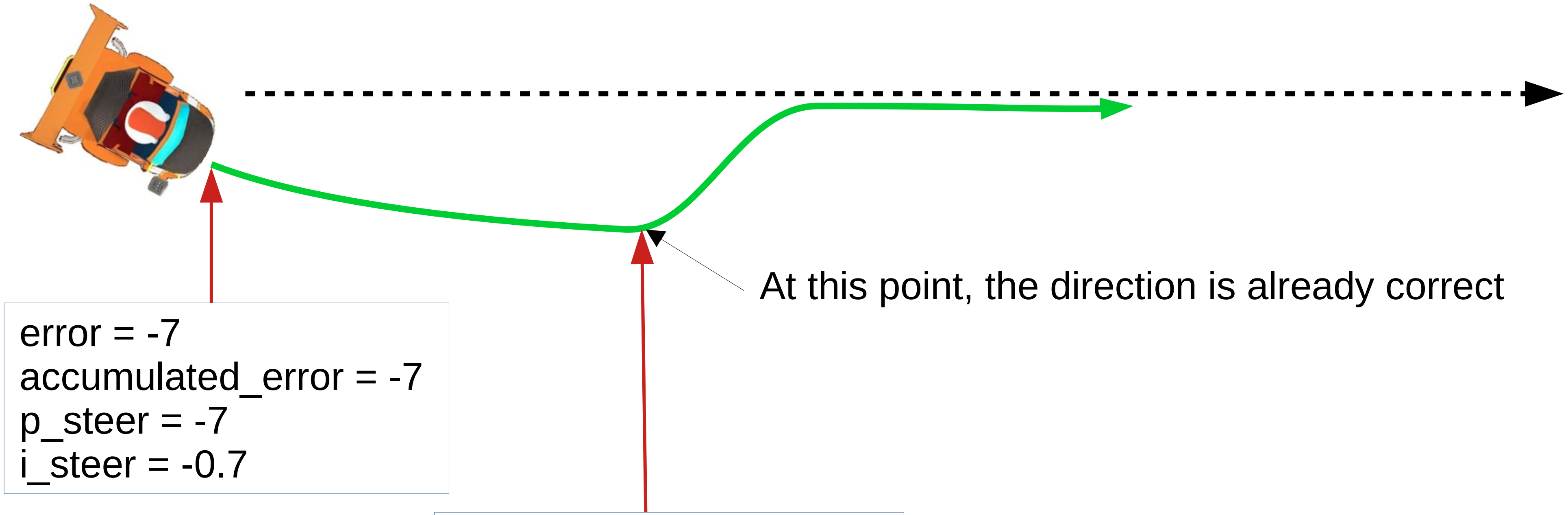
Integral Control

1) Calculate the <u>error</u>
   (i_error = whatYouHave – whatYouWant)
   (i_error = accumulated_error – 0)
   // We want accumulated_error to be 0
   (i_error = accumulated_error)

2) Calculate the <u>correction</u>
   (i_correction = i_error * i_gain)
   (i_steer = i_error * 0.1)
   // Accumulated error can be very large, so keep the gain small

3) Combine with Proportional control and apply the correction
   move_steering(speed, p_steer + i_steer)

Robot Behavior



error = -7
accumulated_error = -7
p_steer = -7
i_steer = -0.7

At this point, the direction is already correct

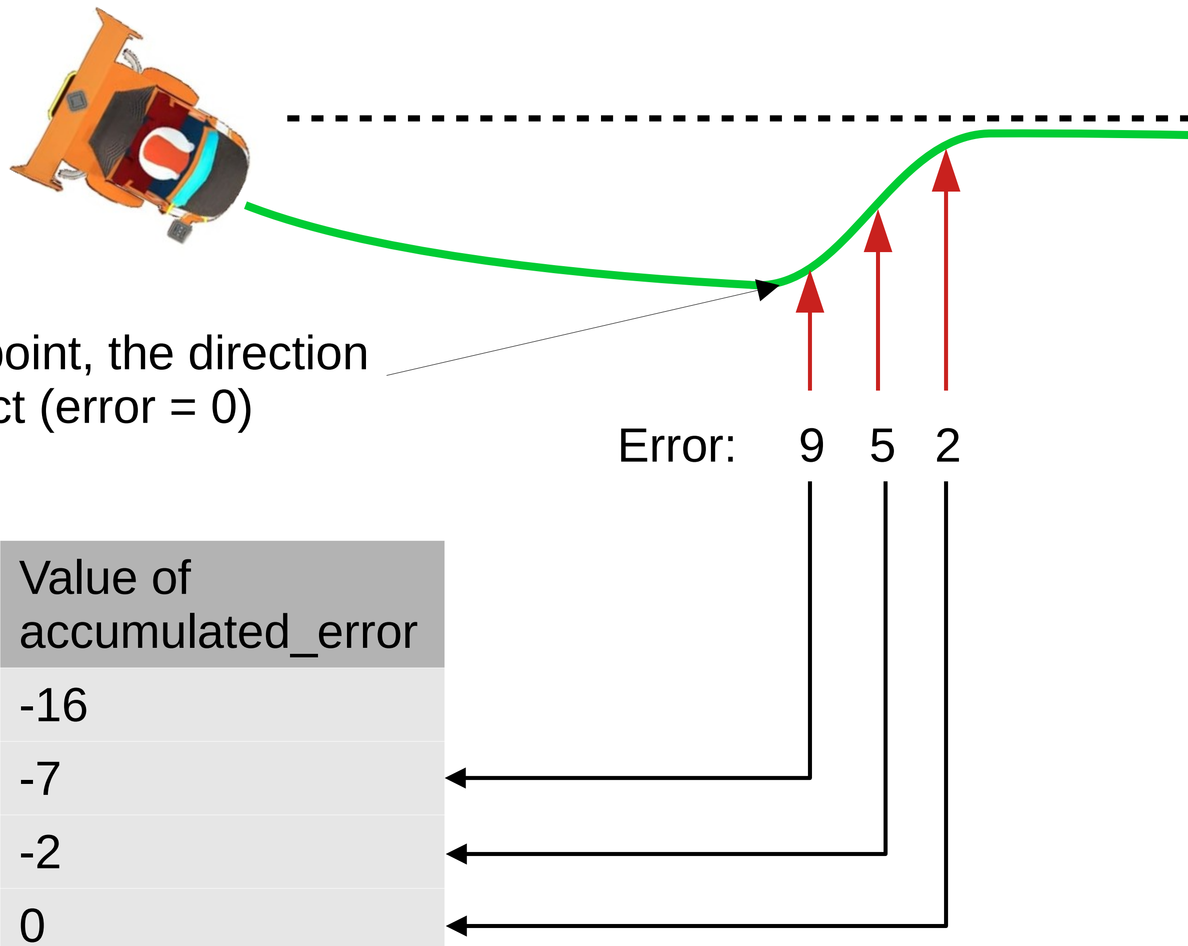
error = 0
accumulated_error = -16
p_steer = 0
i_steer = -1.6

Direction is correct now, so p_steer is 0 (go straight)...

...but accumulated_error is not zero yet, so the i_steer will make the robot continue to turn

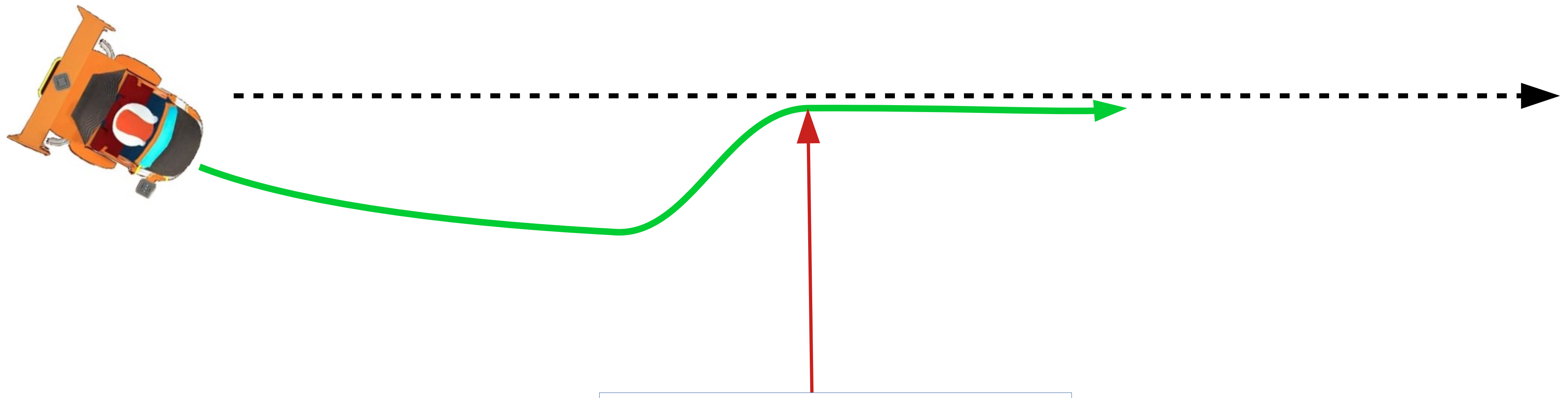
Accumulated Error

- accumulated_error will reduce to zero...



Executed code	Value of accumulated_error
`// At start`	-16
`accumulated_error += error;`	-7
`accumulated_error += error;`	-2
`accumulated_error += error;`	0

Robot Behavior



error = 0
accumulated_error = 0
p_steer = 0
i_steer = 0

Both error and accumulated_error
are zero, so the the robot go straight

PID Control

- Combines Proportional, Integral, Derivative
- Widely used, eg...
  - Aircon temperature control
  - Airplane auto-pilot
  - Robots
- Using all 3, means having 3 gains to tune
  - Can be difficult, so only use what you need

PID controller

From Wikipedia, the free encyclopedia

A **proportional-integral-derivative controller** (
variety of other applications requiring continuously
process variable (PV) and applies a correction base

In practical terms it automatically applies an accura

Code?

- Nope. That’s for you to figure out.
- I’ve already covered all the tricky bits.
- You won’t learn if you’re just copying code.

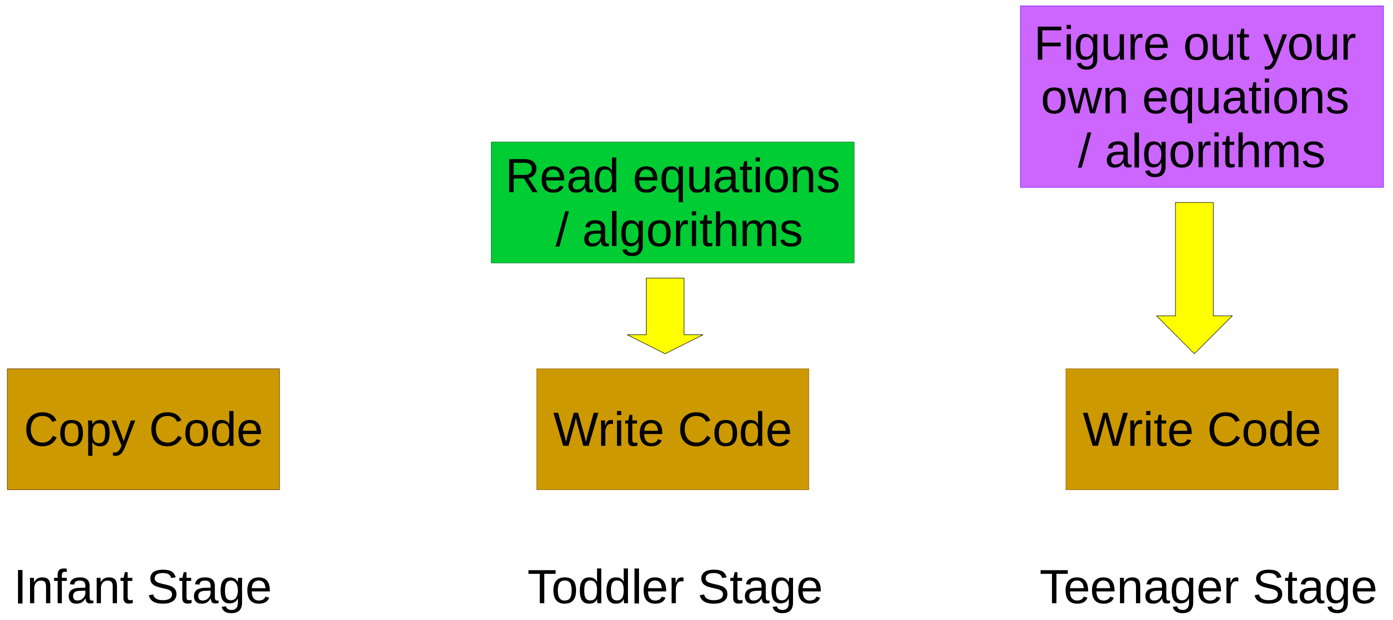


Figure out your own equations / algorithms
Read equations / algorithms
Copy Code
Write Code
Write Code
Infant Stage
Toddler Stage
Teenager Stage

Copyright