Smart Home League

جزوه‌ی آموزشی درس‌به‌درس — **فرست‌استپ (دبستان)**

6 درس

نصب و اجرا

پوشه‌ی **Smart Home League** که دریافت کرده‌اید دو زیرپوشه دارد: **Windows** و **Mac**.

ویندوز

1. وارد پوشه‌ی **Windows** شوید و روی `Smart Home League.exe` دوبار کلیک کنید.
2. اگر بار اول ویندوز پیام امنیتی داد: **More info** و سپس **Run anyway** را بزنید.
3. پنجره‌ی بازی خودش باز می‌شود — همین!

مک

1. وارد پوشه‌ی **Mac** شوید و روی `Smart Home League (Mac).command` دوبار کلیک کنید.
2. اگر مک اجازه نداد: روی همان فایل **راست‌کلیک** → **Open** → **Open** (فقط بار اول لازم است).
3. اگر مک پیشنهاد نصب ابزار خط فرمان (python3) داد، **Install** را بزنید و بعد از پایان، دوباره دوبار کلیک کنید.

در خودِ برنامه: زبان از منوی ☰ قابل تغییر است؛ «کد پایه»، «هلپرها» و «حالت مسابقه» روی صفحه‌ی اول هر لیگ هستند و کتابچه‌ی کامل قوانین از دکمه‌ی «قوانین» باز می‌شود.

هلپرها — ابزارهای کدساز

در صفحه‌ی اول لیگ، دو هلپر می‌بینید. هر دو در پایان یک **فایل پایتون واقعی** می‌سازند — همان چیزی که در مسابقه اجرا می‌شود.

🤖 هلپر AI (قانون‌ساز پرسش‌وپاسخی)

1. از صفحه‌ی اول لیگ، «هلپر AI» را بزنید.
2. روی یکی از سنسورهای روی عکسِ ربات بزنید (یا بکشید) — یک «قانون» ساخته می‌شود.
3. به پرسش‌ها جواب بدهید: «نزدیک‌تر از چند سانتی‌متر؟»، «ربات کدام طرف برود؟»، «با چه سرعتی و چند ثانیه؟».
4. در صورت نیاز «حرکت دوم» را هم اضافه کنید (مثلاً: عقب بکش، بعد بچرخ).
5. در اتاقکِ شبیه‌ساز، همان لحظه نتیجه‌ی قانون‌ها را تماشا کنید.
6. پایین صفحه، پایتونِ ساخته‌شده را می‌بینید — **عددهای زرد را می‌توانید همان‌جا در خودِ کد ویرایش کنید**.
7. در پایان «▶ با همین کد بازی کن» را بزنید — مسابقه خودش شروع می‌شود. با «⬇ فایل پایتون» هم می‌توانید کد را ذخیره کنید.

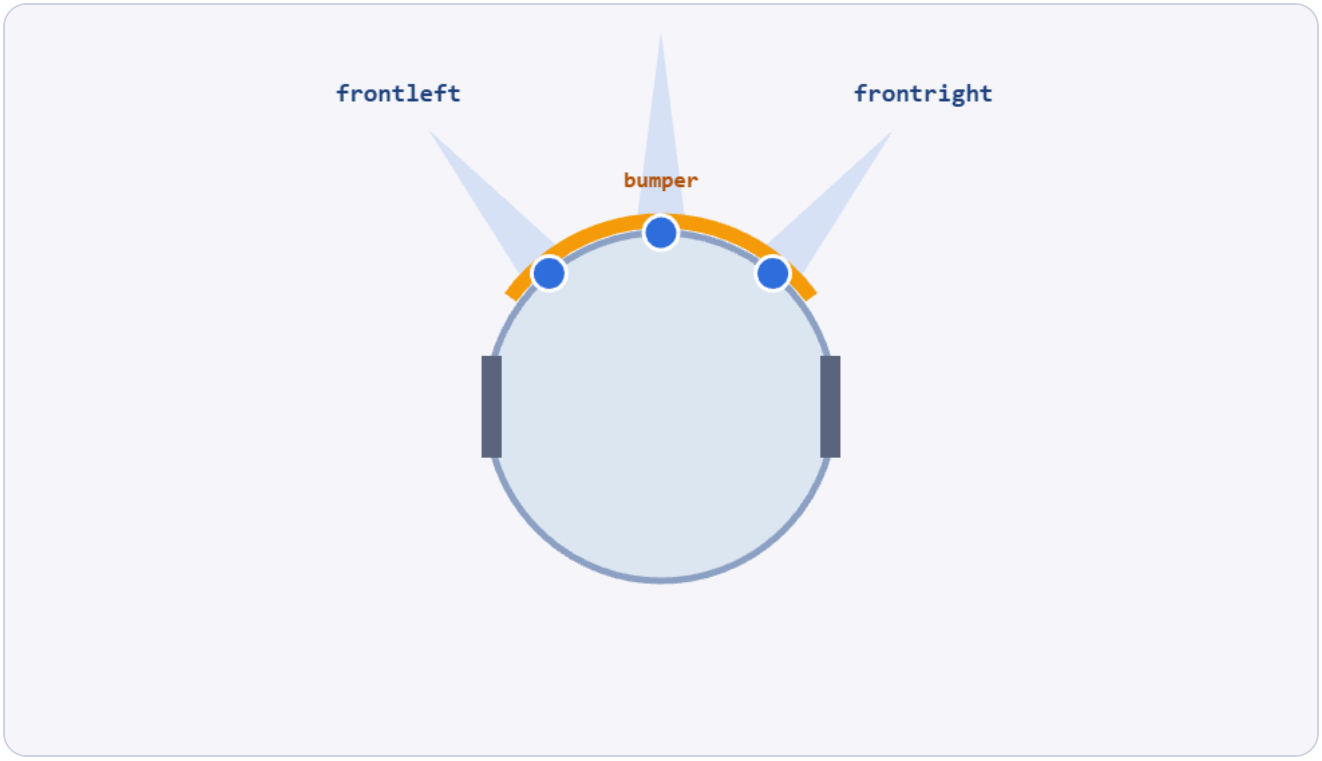
🧩 هلپر بلاکی (پازلی)

1. از صفحه‌ی اول، «هلپر بلاکی» را بزنید.

2. از ستون بلاک‌ها، بلوکِ «اگر …» (سنسور) و بلوکِ حرکت را انتخاب کنید — بلوک‌ها مثل پازل به هم چفت می‌شوند.
3. «+ قانون جدید» بالای صفحه قانون تازه می‌سازد؛ بعد از انداختن هر شرط، جعبه‌ی چشمک‌زن می‌پرسد: «شرط دیگر یا دستور؟».
4. هلپر بلاکی و هلپر AI **یک پرونده‌ی قوانین مشترک** دارند — هر جا ساختید، در دیگری هم قابل ادامه است؛ حتی می‌توانید یک فایل پایتون را در بلاکی «لود» کنید تا دوباره بلوک شود.
5. در پایان ▶ را بزنید تا همان کد در مسابقه اجرا شود.

سنسورهای ربات در یک نگاه

- **سه سنسور فاصله‌ی آلتراسونیک (US)** — `front` روبه‌جلو، `frontleft` جلو سمتِ چپ، `frontright` جلو سمتِ راست؛ بر حسب سانتی‌متر، عدد کوچک‌تر یعنی نزدیک‌تر.
- **بامپر (سپر لمسی)** — `bumperfront` برخورد با جلو و `bumperback` برخورد از پشت (۰ یا ۱).
- **سنسور رنگ** — `color` : رنگِ کفِ درست جلوی ربات (`purple` ، `green` ، `white` ، …).



سه سنسور فاصله‌ی آلتراسونیک (US): front جلوی ربات، frontleft جلو-چپ، frontright جلو-راست — و حلقه‌ی نارنجی، بامپر (سپر لمسی) است

پیش از شروع

نکته: برنامه‌ی ربات ۱۰ بار در ثانیه از بالا تا پایین اجرا می‌شود؛ مقداری که در `wheelleft` و `wheelright` می‌گذارید تا اجرای بعدی می‌ماند. هر درس یک برنامه‌ی کامل و قابل‌اجراست — آن را در بخش «آموزش» داخل برنامه اجرا کنید یا در ادیتور مسابقه بگذارید.

درس ۱ — سنسورهای فاصله (Ultrasonic)

ربات FS سه سنسور فاصله‌ی آلتراسونیک (Ultrasonic Sensor — به‌اختصار US) رو به جلو دارد که حکم سنسورِ فاصله‌ی ربات را دارند و فاصله را به سانتی‌متر گزارش می‌کنند: **front** جلوی ربات، **frontleft** جلو سمتِ چپ و **frontright** جلو سمتِ راست. عدد کوچک‌تر یعنی نزدیک‌تر (بیشینه ۲۰۰). این ربات دارد به سمت قفسه می‌راند — به آن بگویید دیوار که نزدیک شد چه کند، بعد ▶ را بزنید و در پنل پایین عددِ front را تماشا کنید که آب می‌رود.

```
wheelleft = 25
wheelright = 25

if front < 50:        # دیوار نزدیکتر از ۵۰ سانتی‌متر
    wheelleft = 0     # -> بایست
    wheelright = 0
```

نتیجه‌ی مورد انتظار: ربات به سمت شرق می‌راند و حدود نیم متر مانده به قفسه می‌ایستد. به‌جای 50 عدد 80 یا 25 بگذارید و دوباره اجرا کنید — زودتر یا دیرتر می‌ایستد.

درس ۲ — جلو-چپ و جلو-راست را جدا ببینید

سنسورِ وسط (front) همه‌چیز را نمی‌بیند! **frontleft** مایل به چپ و **frontright** مایل به راست نگاه می‌کنند. با این دو می‌توانید به‌جای ایستادن، «فرمان بدهید»: هرطرف که مانع دید، به طرفِ دیگر قوس بزنید.

```
wheelleft = 25
wheelright = 25

if front < 55:          # روبه‌رو بسته است
    wheelleft = 18      # -> چرخش درجا به راست
    wheelright = -18
elif frontleft < 45:    # مانع سمت چپ
    wheelleft = 25      # -> قوس به راست
    wheelright = 8
elif frontright < 45:   # مانع سمت راست
    wheelleft = 8       # -> قوس به چپ
    wheelright = 25
```

نتیجه‌ی مورد انتظار: ربات بدون توقف در خانه می‌چرخد و از کنار مبل‌ها قوس می‌گیرد. عددهای 45 را بزرگ‌تر کنید (مثلاً 70) — محتاط‌تر می‌شود ولی کاشیِ کمتری می‌گیرد. همین موازنه، قلبِ مسابقه است!

درس ۳ – سنسور رنگ

زیر دماغه‌ی ربات یک سنسور رنگ است که کفِ درست جلوی ربات را می‌خواند. با اسم‌ها مقایسه‌اش کنید: **white** (کف کثیف – امتیاز!)، **red / blue** (قبلاً تمیز شده)، **green** (فرش بزرگ)، **purple** (فرش کوچک)، **black** (دیوار). این ربات رو به فرش سبز ایستاده – کاری کنید لحظه‌ای که سبز دید، واکنش نشان دهد.

```
wheelleft = 25
wheelright = 25

if color == green:     # فرش سبز درست جلوست
    wheelleft = 25     # -> بچرخ و ازش دور شو
    wheelright = -25
```

نتیجه‌ی مورد انتظار: ربات به سمت شمال می‌راند و همان لحظه که در پنل، color سبز شد، از فرش رو می‌گرداند. حالا -25 و 25 را جابه‌جا کنید تا به چپ بچرخد. دقت کنید: چرخش فقط تا وقتی ادامه دارد که سبز جلوی سنسور است – برای حرکتِ ماندگار، درس بعدی را ببینید!

درس ۴ — سپرِ دو نیمه

حلقه‌ی دور ربات لمس را حس می‌کند و می‌گوید کدام نیمه خورده: **bumperfront** یعنی با جلو کوبیدم، **bumperback** یعنی از پشت خوردم. نکته‌ی مهم FS: اگر از پشت ضربه خوردید، دنده‌عقب فقط شما را محکم‌تر می‌چسباند — راه نجات **جلو** است! این ربات کور می‌راند؛ فقط با سپر نجاتش بدهید.

```
wheelleft = 25
wheelright = 25

if timer > 0:            # هنوز مشغول مانورم
    timer -= 1
    if state == 2:
        wheelleft = -25  # دنده‌عقب کج
        wheelright = -8
    else:
        wheelleft = 25   # جلو + چرخش
        wheelright = -10

elif bumperfront == 1:   # با جلو کوبیدم
    state = 2            # -> عقب بکش
    timer = 7

elif bumperback == 1:    # از پشت خوردم
    state = 1            # -> فرار به جلو!
    timer = 6
```

نتیجه‌ی مورد انتظار: ربات به چیزی می‌کوبد، bumperfront در پنل ۱ می‌شود و ربات با قوس عقب می‌کشد. اگر روزی از پشت گیر کرد، به جای له‌شدن، جلو می‌رود. timer = 7 را بزرگ‌تر کنید تا مانور طولانی‌تر شود.

درس ۵ — تایمر: حرکتی که ادامه دارد

مقدار دادن به چرخ‌ها فقط **یک قدم** (۰٫۱ ثانیه) دوام دارد — قدم بعد، کدِ شما دوباره از بالا اجرا می‌شود! برای حرکتِ ماندگار: عددی در **timer** بگذارید و بلوک «if timer > 0:» بالای منطق، همان حرکت را قدم‌به‌قدم ادامه می‌دهد. ۱۰ قدم = ۱ ثانیه.

```
wheelleft = 25
wheelright = 25

if timer > 0:           # هنوز دنده‌عقبم
    timer -= 1
    wheelleft = -25
    wheelright = -18    # عقبِ کج

elif color == green:    # فرش سبز!
    timer = 10          # قدم = ۱ ثانیه عقب ۱۰
    wheelleft = -25
    wheelright = -18
```

نتیجه‌ی مورد انتظار: به فرش که رسید، دقیقاً ۱ ثانیه‌ی کامل عقب می‌کشد — حتی وقتی دیگر سبز جلوی سنسور نیست. timer = 10 را ۳۰ کنید و تفاوت را ببینید. عدد timer را هم می‌توانید زنده در پنل پایین تماشا کنیدید.

درس ۶ — حرکت‌های آسان

ساده‌ترین راه: یک دستور، کلِ حرکت — و عددش **ثانیه‌ی واقعی** است. **backward(2)** یعنی دقیقاً ۲ ثانیه دنده‌عقب. **forward(1)**، **turnright(1)**، **turnleft(1)** و **stop(1)** هم هستند. تا حرکتی تمام نشده، خط‌های چرخِ شما نادیده گرفته می‌شوند و بعدش کدِ عادیِ شما دوباره فرمان می‌گیرد.

```
wheelleft = 25
wheelright = 25

if color == green:      # فرش سبز جلوست
    backward(2)         # دو ثانیه‌ی واقعی عقب

elif bumper == 1:       # خوردم به چیزی
    backward(1)

elif front < 50:        # دیوار نزدیک است
    turnright(1)        # یک ثانیه چرخش به راست
```

نتیجه‌ی مورد انتظار: به فرش که رسید backward(2) دو ثانیه‌ی کامل عقب می‌بَرَد؛ نزدیک دیوار turnright(1) می‌چرخاند. عددها را عوض کنید: backward(5) یا turnright(0.5) — نیم هم می‌شود، عدد همیشه ثانیه است!