Lego Robotics Camp

Day 1: Programming basics

Welcome!

- Camp overview
  - Goals
    - Computational thinking
    - Python programming
    - Robotics with Lego Mindstorms EV3 kits
  - Programming, Lego building, discussions, and unplugged activities
  - Work hard and have fun!

Daily schedule

- **9:00-9:15** Arrival
- **9:15-10:15** Computational thinking unplugged activity
- **10:15-10:30** Snack and mask break
- **10:30-12:00** Daily lesson and lab overview
- **12:00 - 1:00** Lunch and mask break
- **1:00-2:30** Lab (programming activity)
- **2:30-3:30** Wrap-up and free time (soccer, swimming, etc)
- **3:30** Show & tell and dismissal

Rules

- Treat each other with respect
- Don’t be disruptive
- No such thing as a stupid question
- Be patient (with each other and with me!)
- Be kind to your robots ☺
- Don’t give up
- Wear your masks when indoors

Today's Plan

- Today we will:
  - Discuss how computers solve problems
  - Design algorithms for simple, everyday tasks
  - Get to know our robots
  - Gain experience with our programming environments
  - Learn about variables

Computational Thinking

- Think like a computer scientist!
- Four pillars of CT:
  - Decomposition – break big problems up into small pieces
  - Pattern recognition – look for similarities within a problem
  - Abstraction – ignore unimportant information and focus on stuff that matters
  - Algorithms – develop step-by-step rules for solving the problem
- We will use these four pillars to solve problems with our robots!

Peanut Butter and Jelly!

- Make an *algorithm* for making a PB&J sandwich
- What is an algorithm?
  - A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer
  - Algorithms provide computers with a successive guide to completing actions

Peanut Butter and Jelly!

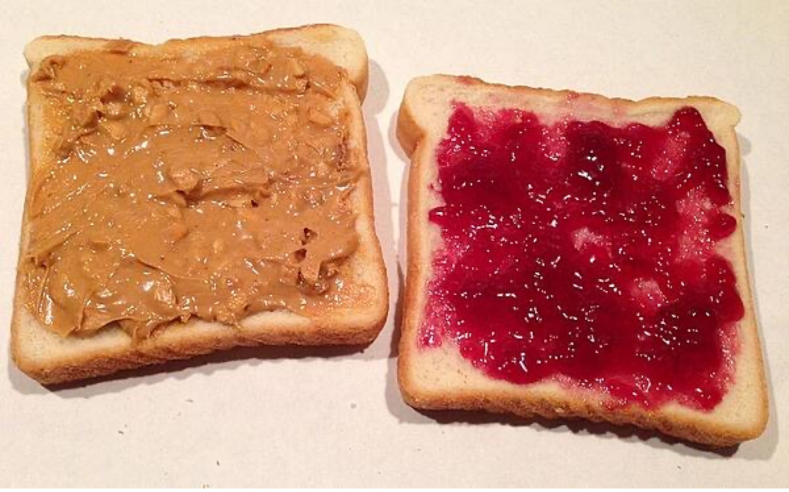
- Make an *algorithm* for making a PB&J sandwich
- Supplies:
  - Peanut butter
  - Jelly
  - Loaf of bread
  - Two knives
  - Plate
- Work with a partner! Write down your steps. Be specific!

Peanut Butter and Jelly!

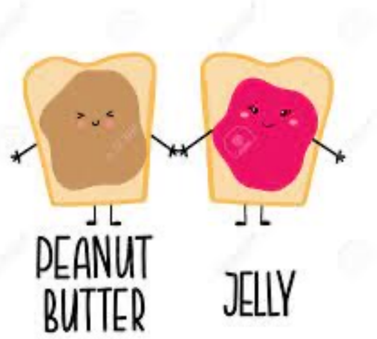
What Happened?

- It’s easy to overlook crucial information and details!
- As a programmer, you need to develop **specific** directions to help the computer solve your problems.

- Try again!



Peanut Butter and Jelly!



1. Take a slice of bread.
2. Open the jar of peanut butter by twisting the lid counter-clockwise
3. Pick up a knife by the handle
4. Insert the knife into the jar of peanut butter
5. Withdraw the knife from the jar of peanut butter and run it across the slice of bread
6. Take a second slice of bread
7. Repeat steps 2-5 with the second slice of bread and the jar of jelly
8. Press the two slices of bread together such that the peanut butter and jelly meet

Lessons Learned

- Remember our four pillars:
  - Decomposition – break big problems up into small pieces
  - Pattern recognition – look for similarities within a problem
  - Abstraction – ignore unimportant information and focus on stuff that matters
  - Algorithms – develop step-by-step rules for solving the problem
- Computers are really not that smart!
- But they are VERY good at following directions. They only do EXACTLY what you tell them to do.
- We must provide specific instructions for solving problems

Robot Basics

- Helps to know what basic actions we can use to instruct our robot
- What actions might we want our robot to perform?

Robot Basics

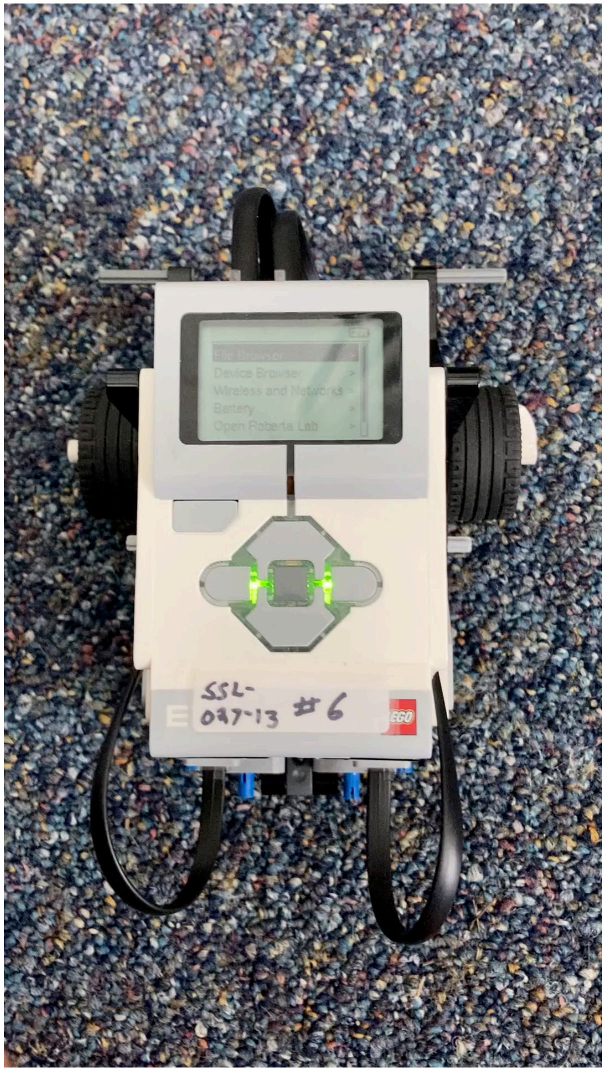
- Helps to know what basic actions we can use to instruct our robot
- What actions might we want our robot to perform?
  - Sound: beep, speak?
  - Display: lights on/off, show image
  - Movement: go forward, backward, turn/rotate, stop
  - Advanced actions: react to “sensed environment” in some way (requires **sensors** for light, sound, temperature, touch, etc)

Robot Basics

- Today we will begin learning how to perform very basic actions with our robots
- For the rest of the week, we’ll use these basic actions to solve problems

Let’s Meet Our Robots!

- FYI: My robot looks a little different than yours
- Let’s start with some simple examples with basic movement



(A video, just in case my robot misbehaves…)

Run basic_movement

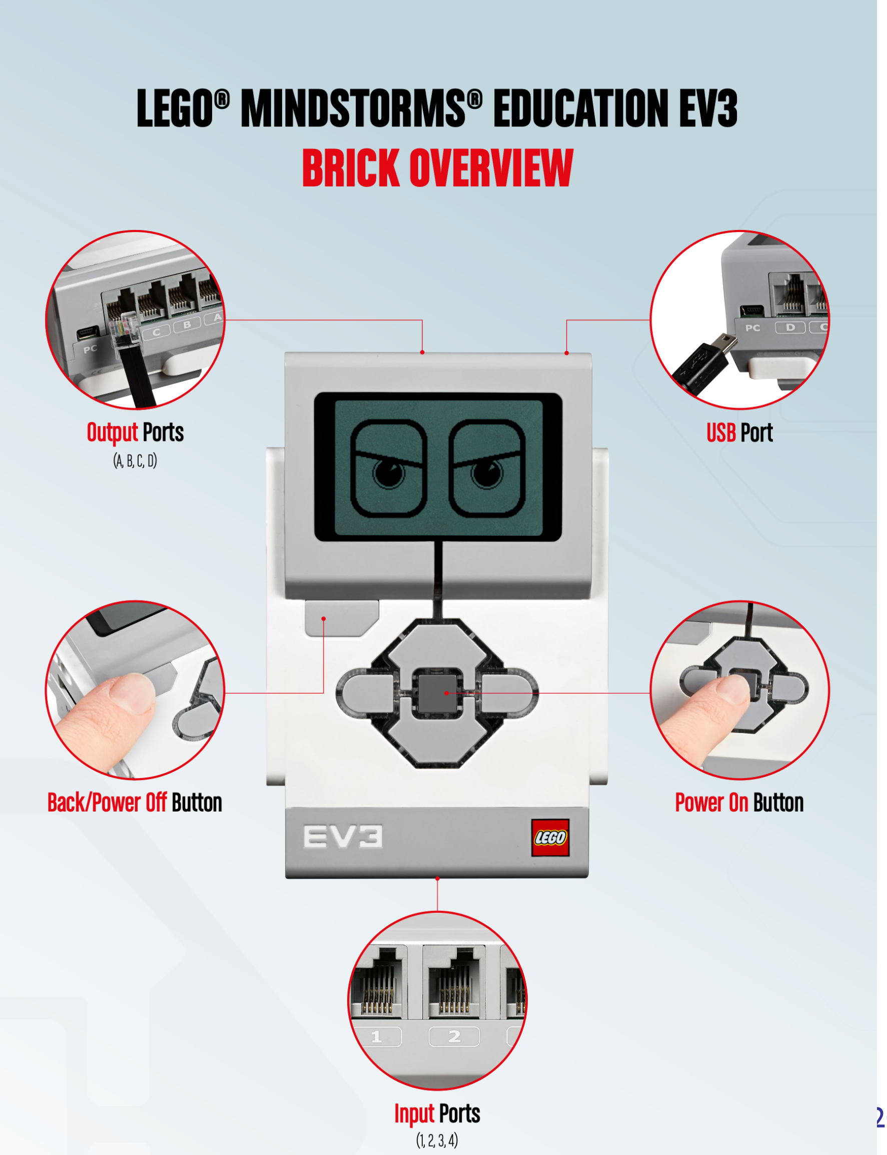
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, Image, ImageFile

# Create your objects here.
ev3 = EV3Brick()

# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the drive base.
robot = DriveBase(left_motor, right_motor,
wheel_diameter=56, axle_track=114)
```

```
# Set eyes
ev3.screen.load_image(Image(ImageFile.NEUTRAL))

# Go forward and backwards for one meter.
robot.straight(500)
ev3.speaker.beep()

robot.straight(-500)
ev3.speaker.beep()

# Turn clockwise by 360 degrees and back again.
robot.turn(360)
ev3.speaker.beep()

robot.turn(-360)
ev3.speaker.say(“hello campers”)
```

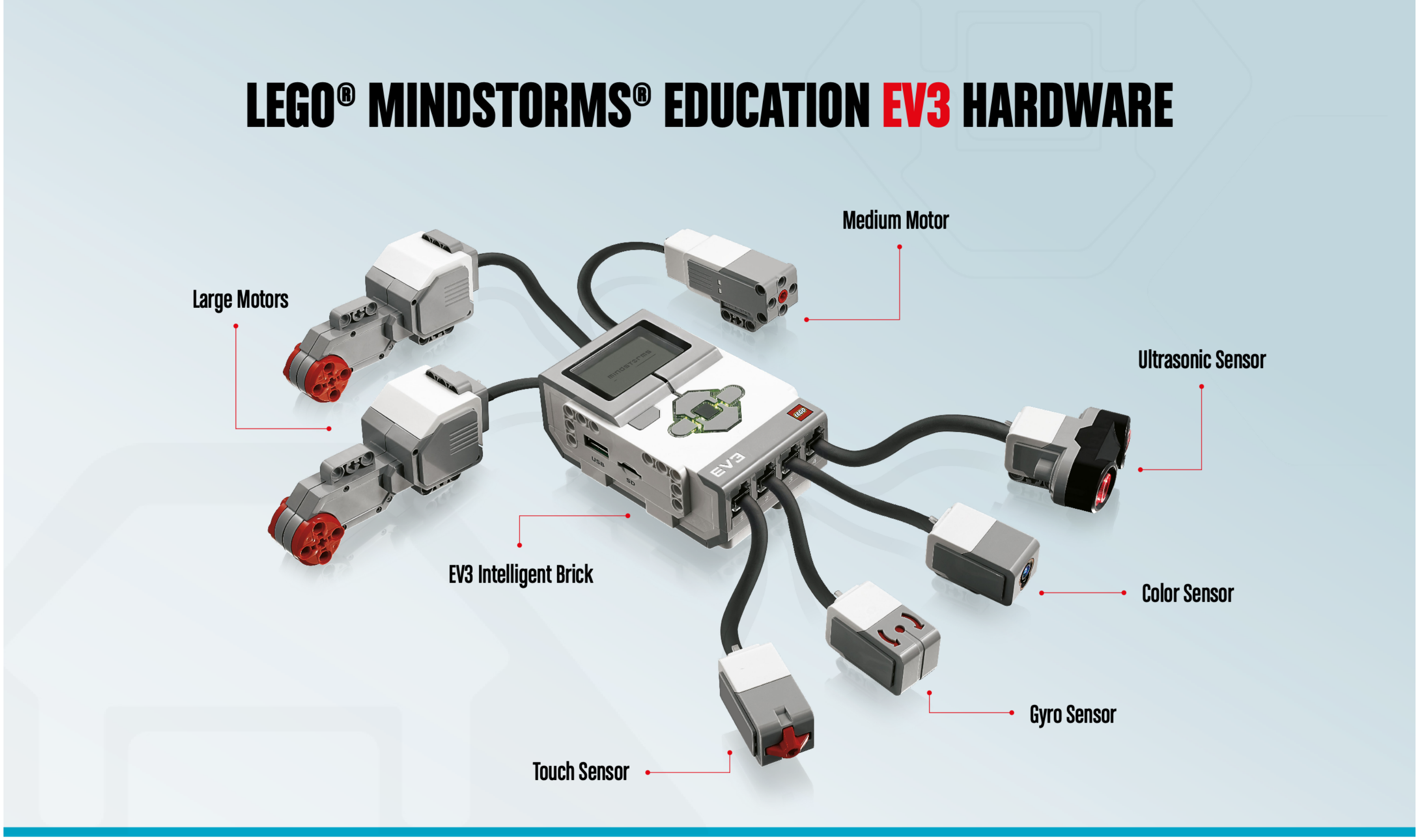
BREAK

Getting Started

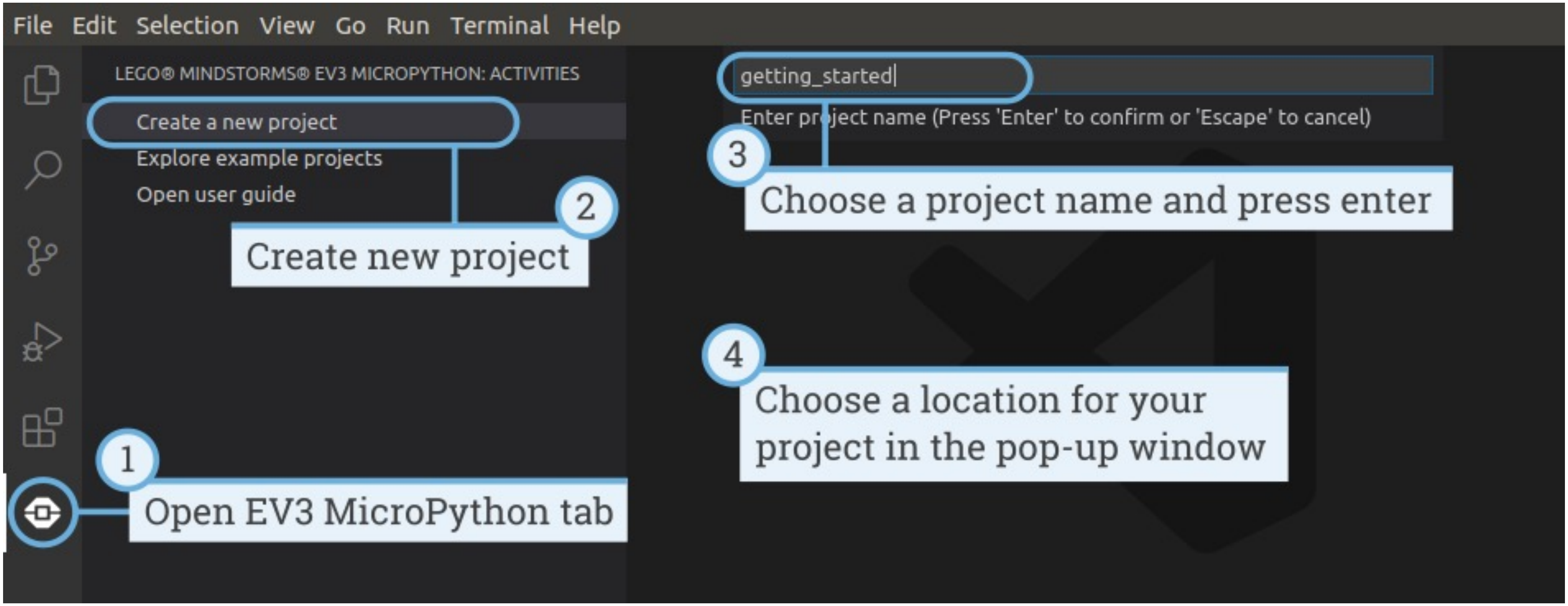
- Before we build, let's go over some basic info
- We'll start with the EV3 brick!



Hardware


LEGO® MINDSTORMS® EDUCATION EV3 HARDWARE
Medium Motor
Large Motors
Ultrasonic Sensor
EV3 Intelligent Brick
Color Sensor
Gyro Sensor
Touch Sensor

Visual Studio: Create New Project



File Edit Selection View Go Run Terminal Help
LEGO® MINDSTORMS® EV3 MICROPYTHON: ACTIVITIES
Create a new project
Explore example projects
Open user guide
getting_started

Enter project name (Press 'Enter' to confirm or 'Escape' to cancel)

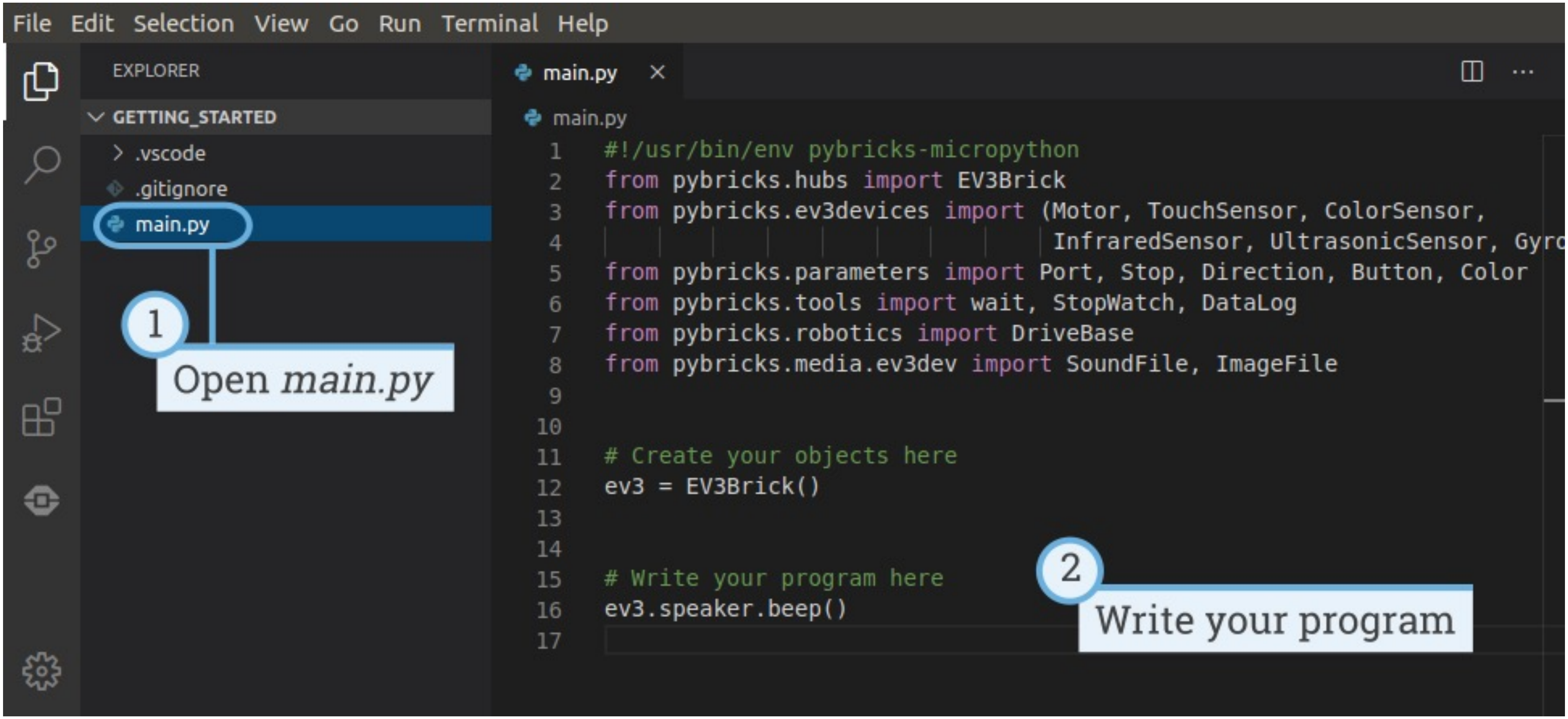
1 Open EV3 MicroPython tab

2 Create new project

3 Choose a project name and press enter

4 Choose a location for your project in the pop-up window

Visual Studio: Writing Code

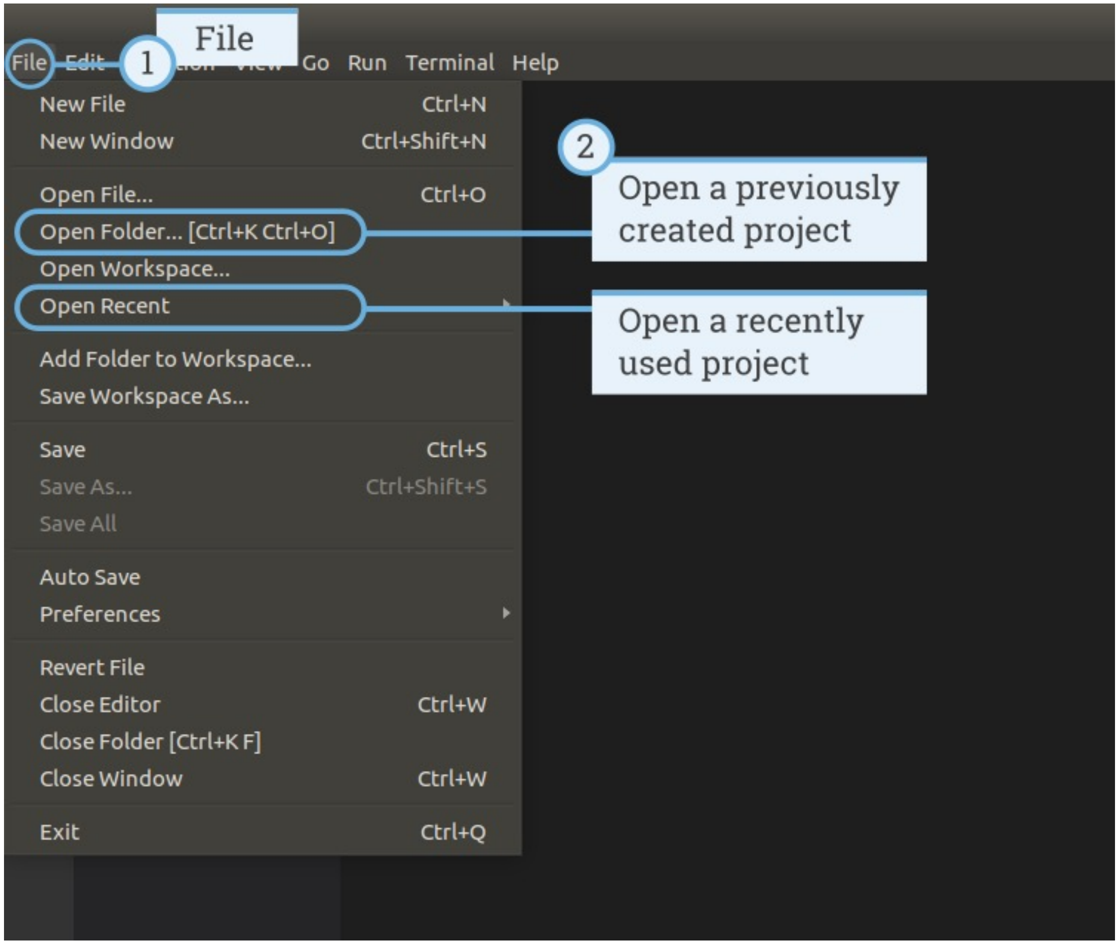


1 Open *main.py*

```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, Gyro
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# Create your objects here
ev3 = EV3Brick()


# Write your program here
ev3.speaker.beep()
```

2 Write your program

Visual Studio: Open Existing Project



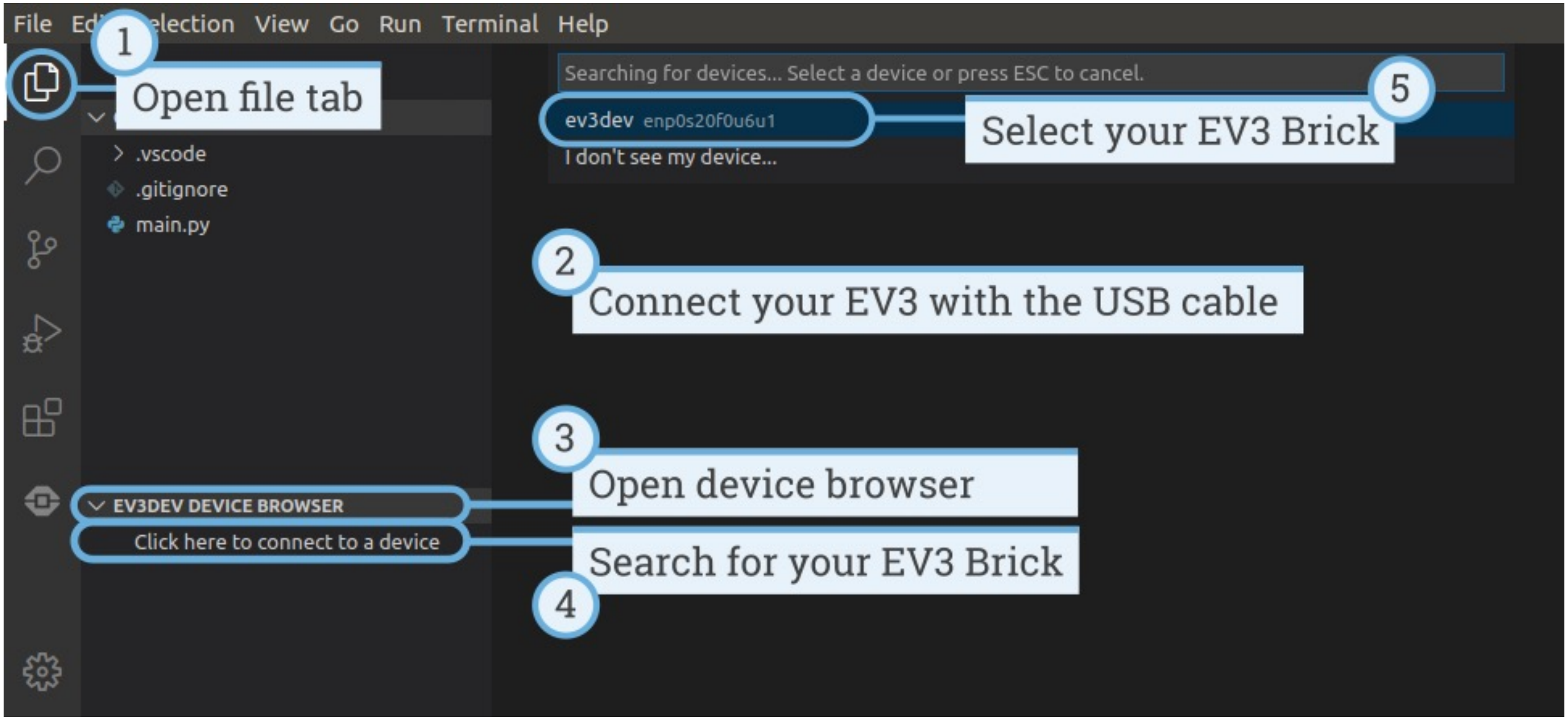
1 File

2 Open a previously created project

Open a recently used project

Visual Studio: Connecting the Brick

- Turn EV3 brick on first



1 Open file tab

2 Connect your EV3 with the USB cable

3 Open device browser

EV3DEV DEVICE BROWSER

4 Search for your EV3 Brick

Click here to connect to a device

5 Select your EV3 Brick

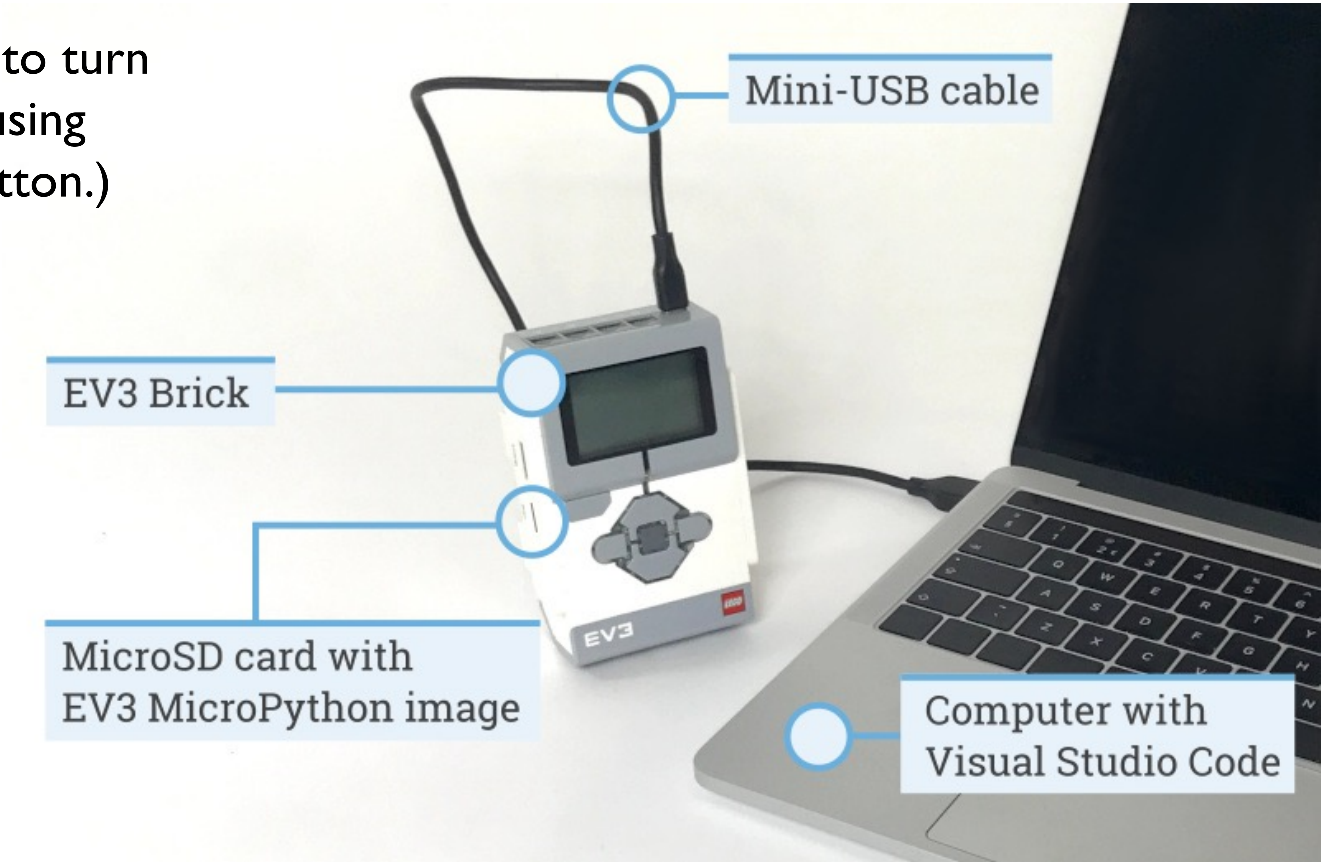
Searching for devices... Select a device or press ESC to cancel.

ev3dev enp0s20f0u6u1

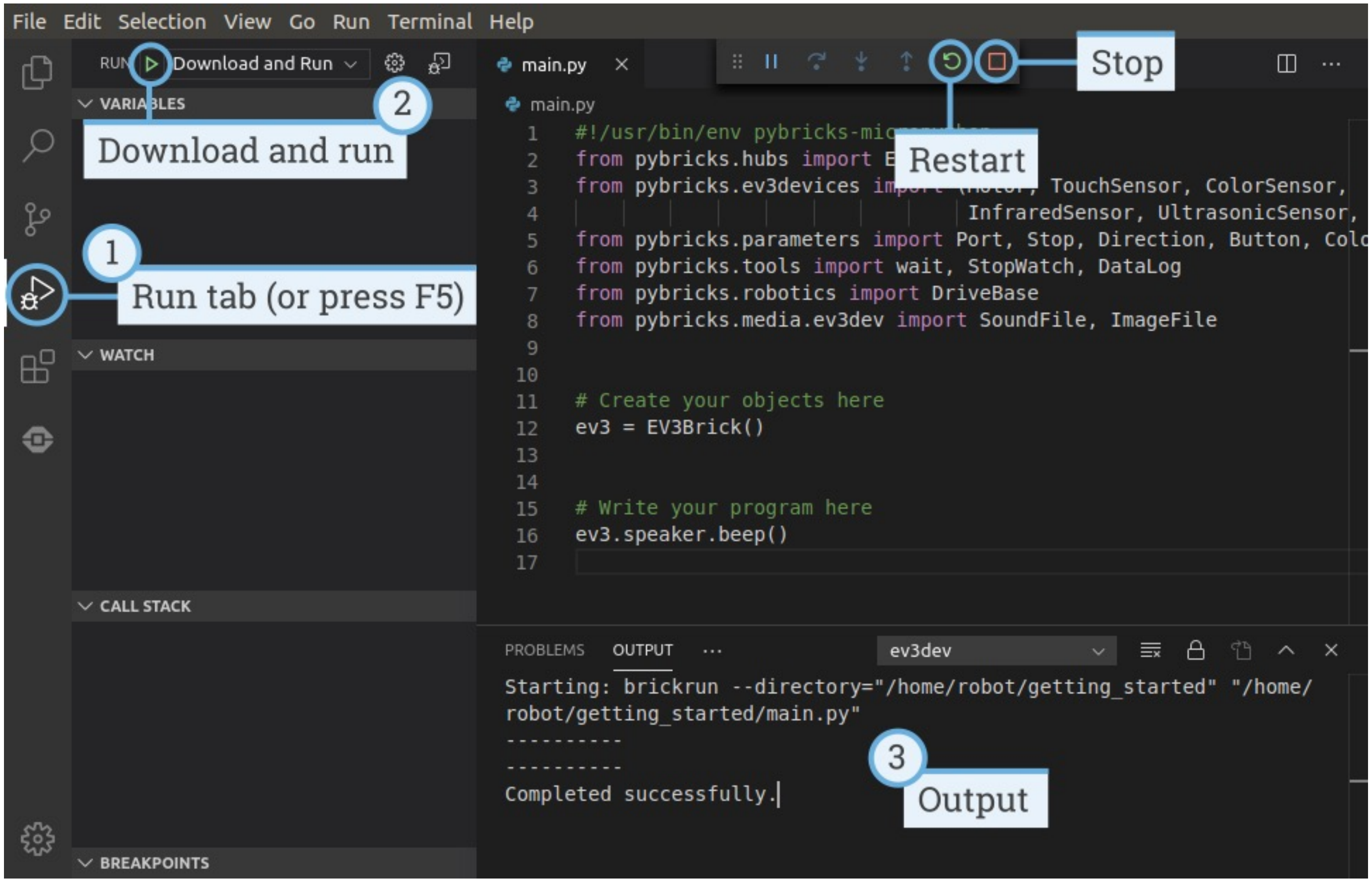
I don't see my device...

Connecting to Brick

(Don’t forget to turn on the brick using the center button.)


Mini-USB cable
EV3 Brick
MicroSD card with EV3 MicroPython image
Computer with Visual Studio Code

Visual Studio: Running Program



1 Run tab (or press F5)

2 Download and run

3 Output

Restart

Stop

```
#!/usr/bin/env pybricks-mic
from pybricks.hubs import E
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor,
from pybricks.parameters import Port, Stop, Direction, Button, Col
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# Create your objects here
ev3 = EV3Brick()


# Write your program here
ev3.speaker.beep()
```

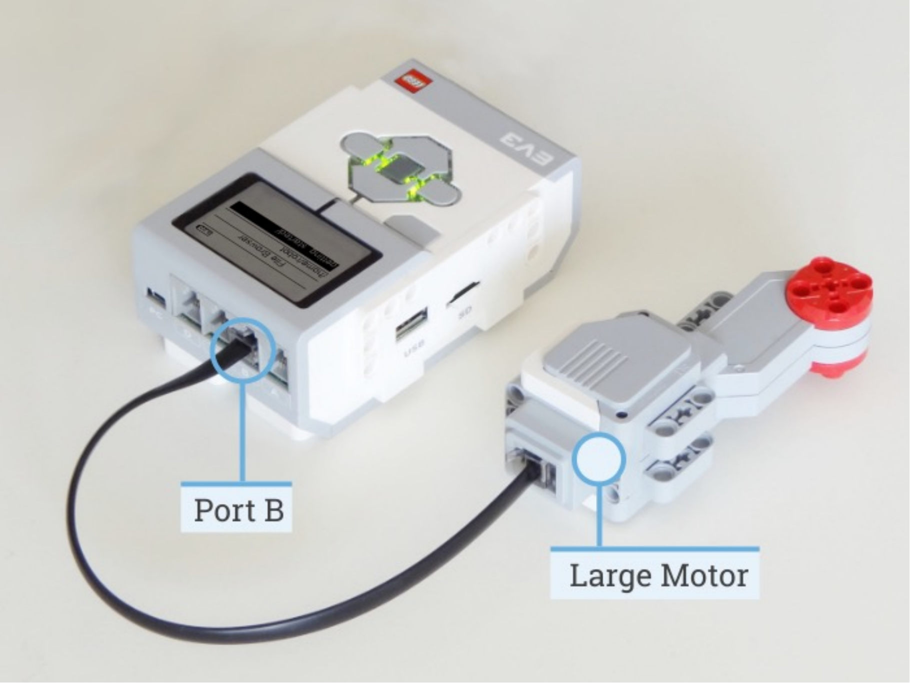
```
Starting: brickrun --directory="/home/robot/getting_started" "/home/
robot/getting_started/main.py"
----------
----------
Completed successfully.
```

Demo: Moving Motors

1. First just run the default program
2. Next plug a large motor into Port B
3. (We'll keep it plugged into our computer for now)
4. Add these 2 lines to the bottom of our program and run it again

```
test_motor = Motor(Port.B)
test_motor.run_time(500, 5000)
```

5. What happens?



Our first (closer) look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()


# Write your program here
# Play a sound.
ev3.speaker.beep()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

Our first (closer) look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

Our first look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

This tells our robot where to find the programming libraries we are using. You can ignore this for now!

Our first look at Python!

```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

Lines that start with # are called comments. They aren't part of the program but are very useful and important for making our programs easy to understand.

Our first look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

- This gives a name to our brick. In this case, the name is **ev3**. For the rest of the program, every time we say “ev3” we know we are talking about our brick.
- ev3 is called a **variable**.
- A variable is just a name for referring to an object. Try changing it!

Our first look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

- **test_motor** is the name we are giving to the Motor attached to Port.B.
- **test_motor** is another **variable**.
- If we had more than one motor, we would name them separately.

Our first look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

Here we are telling our brick to perform some action. In particular, we are telling ev3 to use it's speaker to play a beep sound.

Our first look at Python!

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Initialize the EV3 Brick.
ev3 = EV3Brick()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Write your program here

# Play a sound.
ev3.speaker.beep()

# Run the motor 500 degrees per second, for 5000 ms = 5 seconds
test_motor.run_time(500, 5000)
```

Here we are telling our motor to perform some action. In particular, we are telling it to run_time, or run for 5 seconds. The numbers control how fast it spins and for how long.

Revisiting basic_movement

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, Image, ImageFile

# Create your objects here.
ev3 = EV3Brick()

# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the drive base.
robot = DriveBase(left_motor, right_motor,
        wheel_diameter=56, axle_track=114)
```

```
# Set eyes
ev3.screen.load_image(Image(ImageFile.NEUTRAL))

# Go forward and backwards for one meter.
robot.straight(500)
ev3.speaker.beep()

robot.straight(-500)
ev3.speaker.beep()

# Turn clockwise by 360 degrees and back again.
robot.turn(360)
ev3.speaker.beep()

robot.turn(-360)
ev3.speaker.say(“hello campers”)
```

Revisiting basic_movement

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, Image, ImageFile

# Create your objects here.
ev3 = EV3Brick()

# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the drive base.
robot = DriveBase(left_motor, right_motor,
        wheel_diameter=56, axle_track=114)
```

```
# Set eyes
ev3.screen.load_image(Image(ImageFile.NEUTRAL))

# Go forward and backwards for one meter.
robot.straight(500)
ev3.speaker.beep()

robot.straight(-500)
ev3.speaker.beep()

# Turn clockwise by 360 degrees and back again.
robot.turn(360)
ev3.speaker.beep()
```

Declare and initialize all variables. Give names to the important parts of our robots so we can control them later.

Revisiting basic_movement

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, Image, ImageFile

# Create your objects here.
ev3 = EV3Brick()

# I
le
ri

#
ro
```

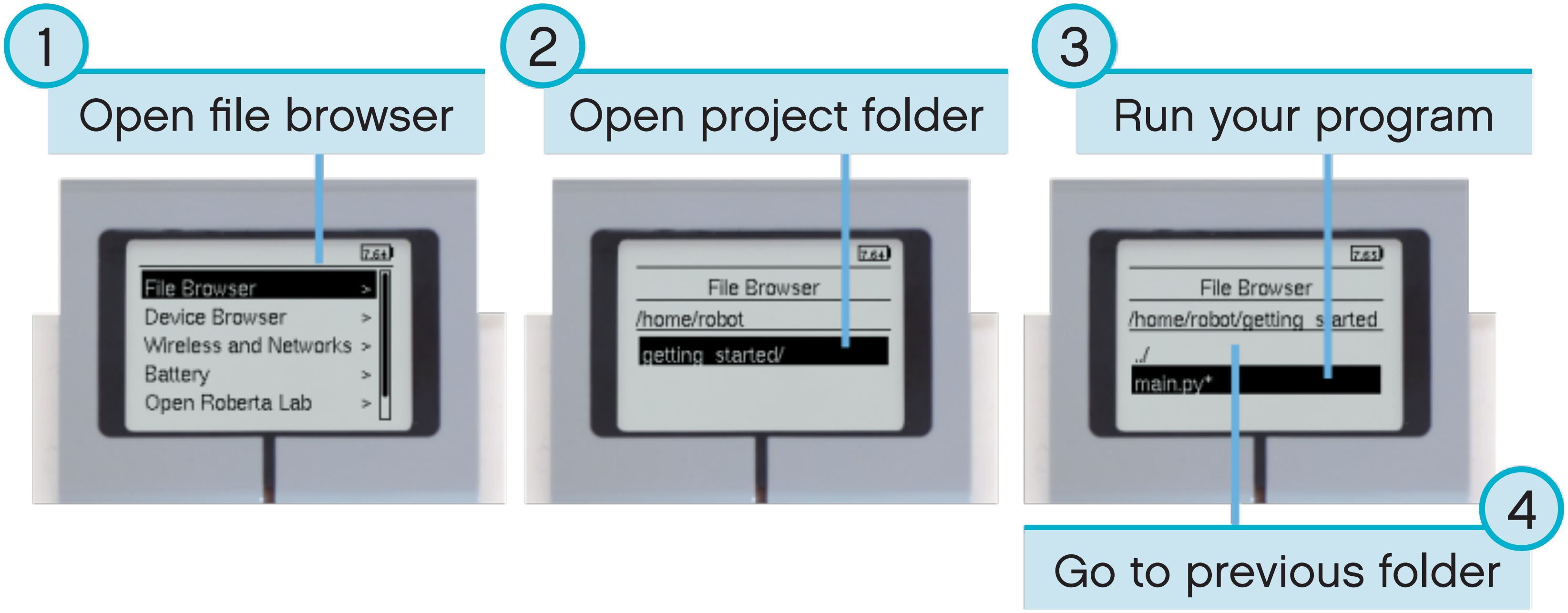
Using our variables, we can tell the robot to perform the desired actions, like moving straight, turning, displaying eyes, beeping, and speaking.

```
# Set eyes
ev3.screen.load_image(Image(ImageFile.NEUTRAL))

# Go forward and backwards for one meter.
robot.straight(500)
ev3.speaker.beep()

robot.straight(-500)
ev3.speaker.beep()

# Turn clockwise by 360 degrees and back again.
robot.turn(360)
ev3.speaker.beep()

robot.turn(-360)
ev3.speaker.say(“hello campers”)
```

Common Programming Mistakes

- Spelling matters!
- Punctuation matters!
- Indentation matters!
- Little mistakes can cause crazy behavior on your robots.
- Test your code often!

Visual Studio Projects on Brick



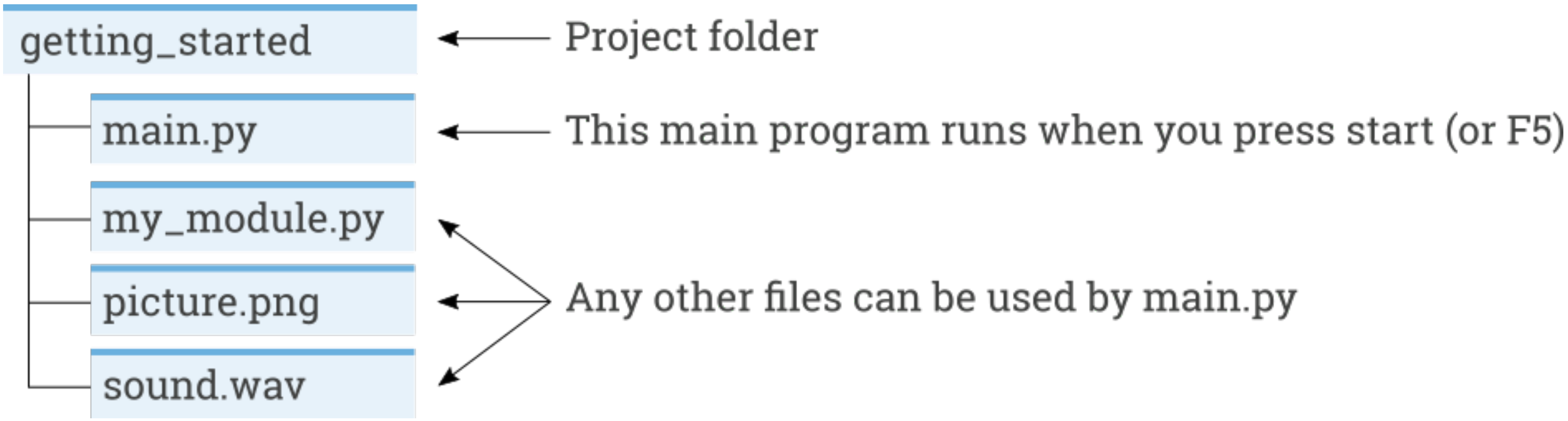
1 Open file browser

2 Open project folder

3 Run your program

4 Go to previous folder

Visual Studio Projects on Brick



getting_started

main.py

my_module.py

picture.png

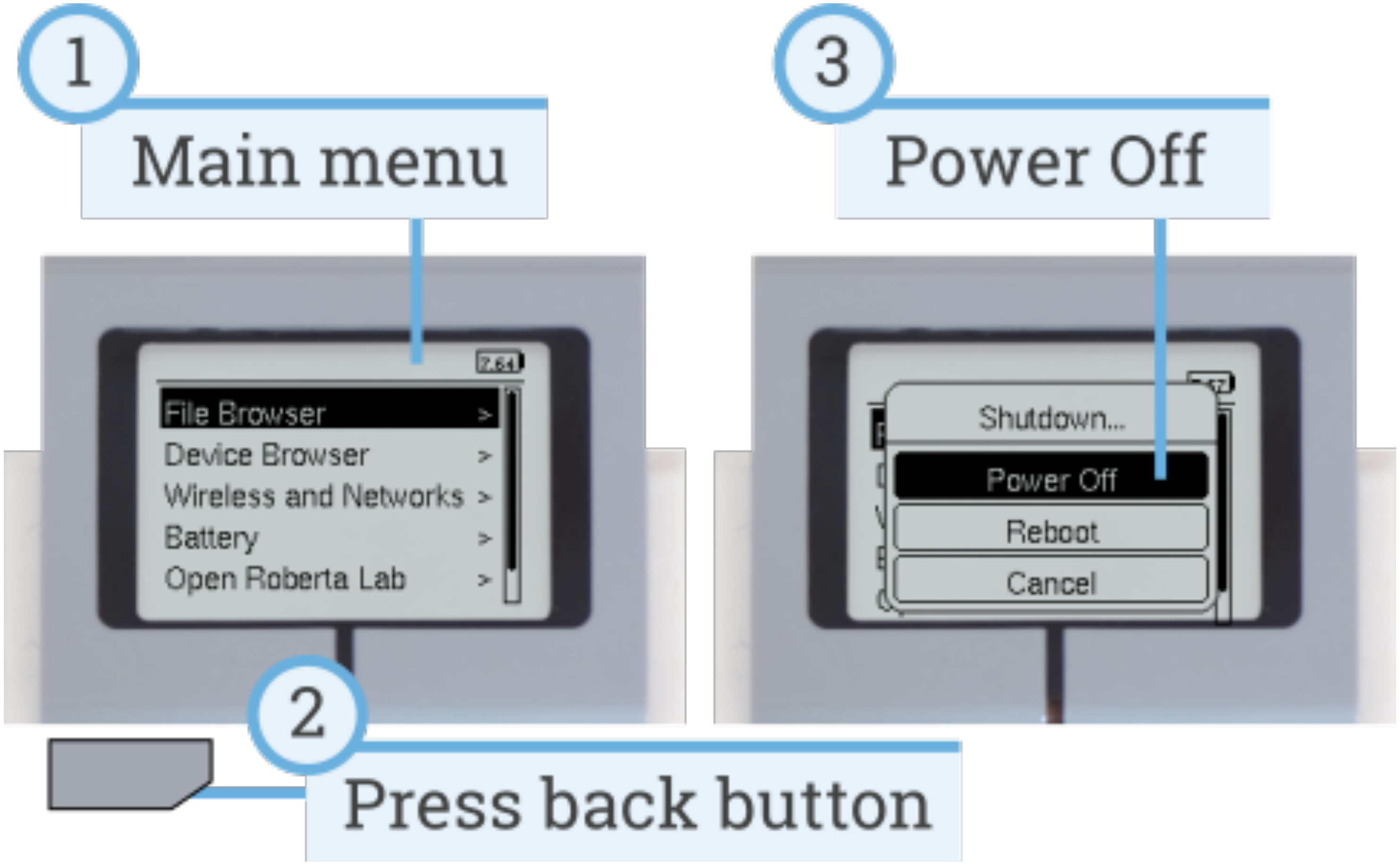
sound.wav

Project folder

This main program runs when you press start (or F5)

Any other files can be used by main.py

Power Off Brick



1 Main menu

2 Press back button

3 Power Off

Lab

- Step 1
  - Build your robots. Follow the instructions carefully. **Raise your hand if you need help!**
- Step 2
  - Connect your robot to your computer using the USB cable. Open Visual Studio. Try running the basic movement program on your robot. **Raise your hand if you need help!**
- Step 3
  - Make your robots dance and sing by adding more commands! We'll demo your creations to your parents at dismissal. (Demo fun_actions)
- Step 4
  - Don't forget to clean up your workspace before you go. Please plug in your brick!

Fun Robot Actions

- **Movement**
- `robot.straight(x)` – drive forward for x millimeters
- `robot.drive(x, y)` – drive forward at speed x and turn rate y
- `robot.turn(x)` – turn in place x degrees
- **Light**
- `ev3.light.on(color)` – turns light on to specific color (try Color.RED, Color.GREEN, Color.YELLOW)
- `ev3.light.off()` – turns off the light
- **Sound**
- `ev3.speaker.beep()` – beep speaker once
- `ev3.speaker.say("text")` – speak the text specified
- `ev3.speaker.play_notes(notes)` – plays a sequence of musical notes. For example, try ['C4/4', 'D4/4', 'E4/4', 'F4/4', 'G4/4']. This plays C, D, E, F, G as quarter notes (/4)
- **Screen**
- `ev3.screen.load_image(Image(ImageFile.NEUTRAL))` – sets "neutral" eyes. Also try ANGRY, DIZZY, SLEEPING, EVIL.

Lunch break!

- After lunch, we'll build your robots!