CommAI-env

Marco Baroni, **Allan Jabri**, Armand Joulin, Germán Kruszewski,
Angeliki Lazaridou, Tomas Mikolov, Klemen Simonic

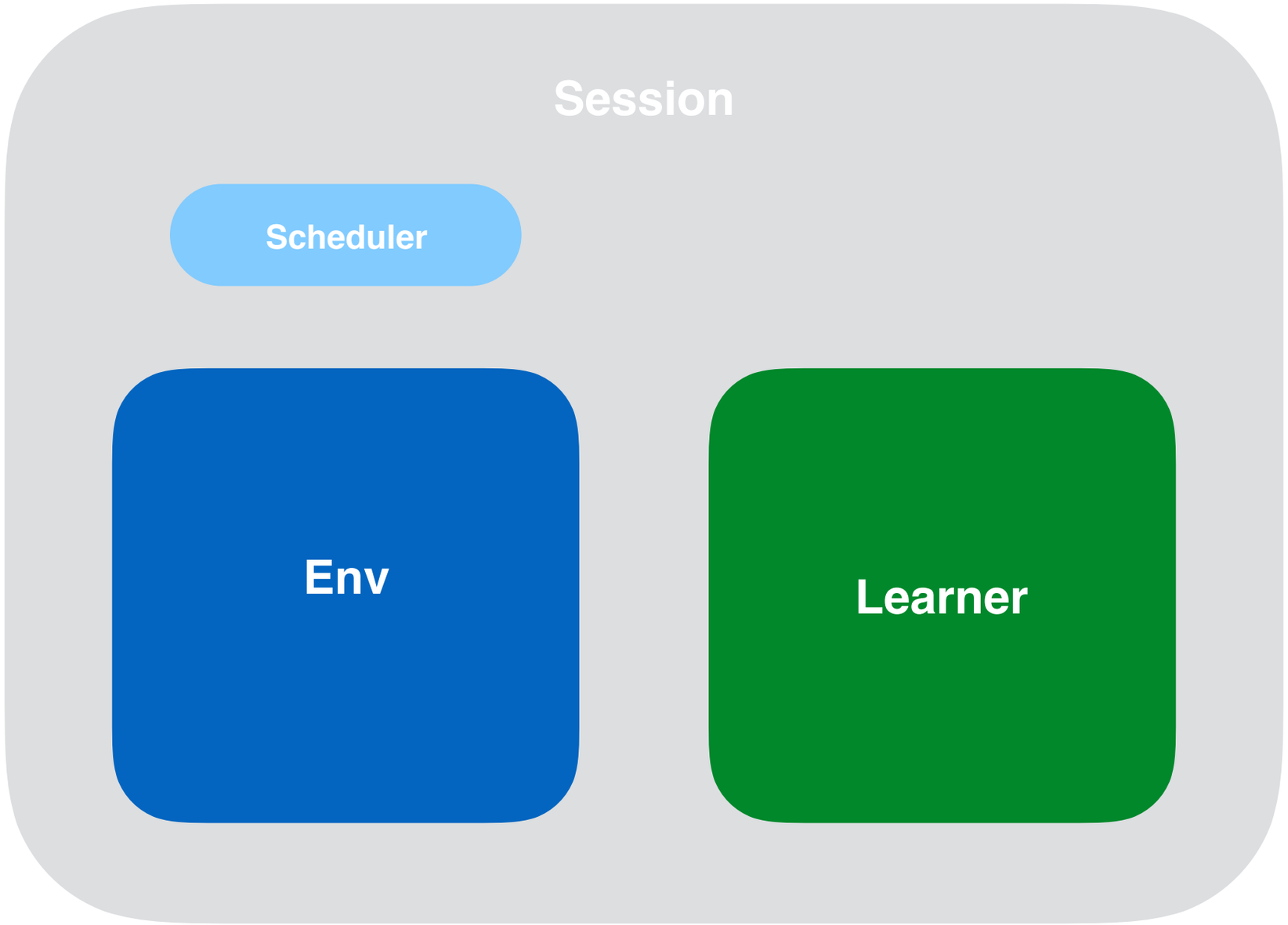
Facebook Artificial Intelligence Research

MAchine INtelligence Workshop NIPS 2016

Communication-based AI Environment

- environment in python, learner in any language
- lightweight text environment (command line)
- github.com/facebookresearch/CommAI-env

```
git clone https://github.com/facebookresearch/CommAI-env.git
cd src
python run.py my_tasks.json
```

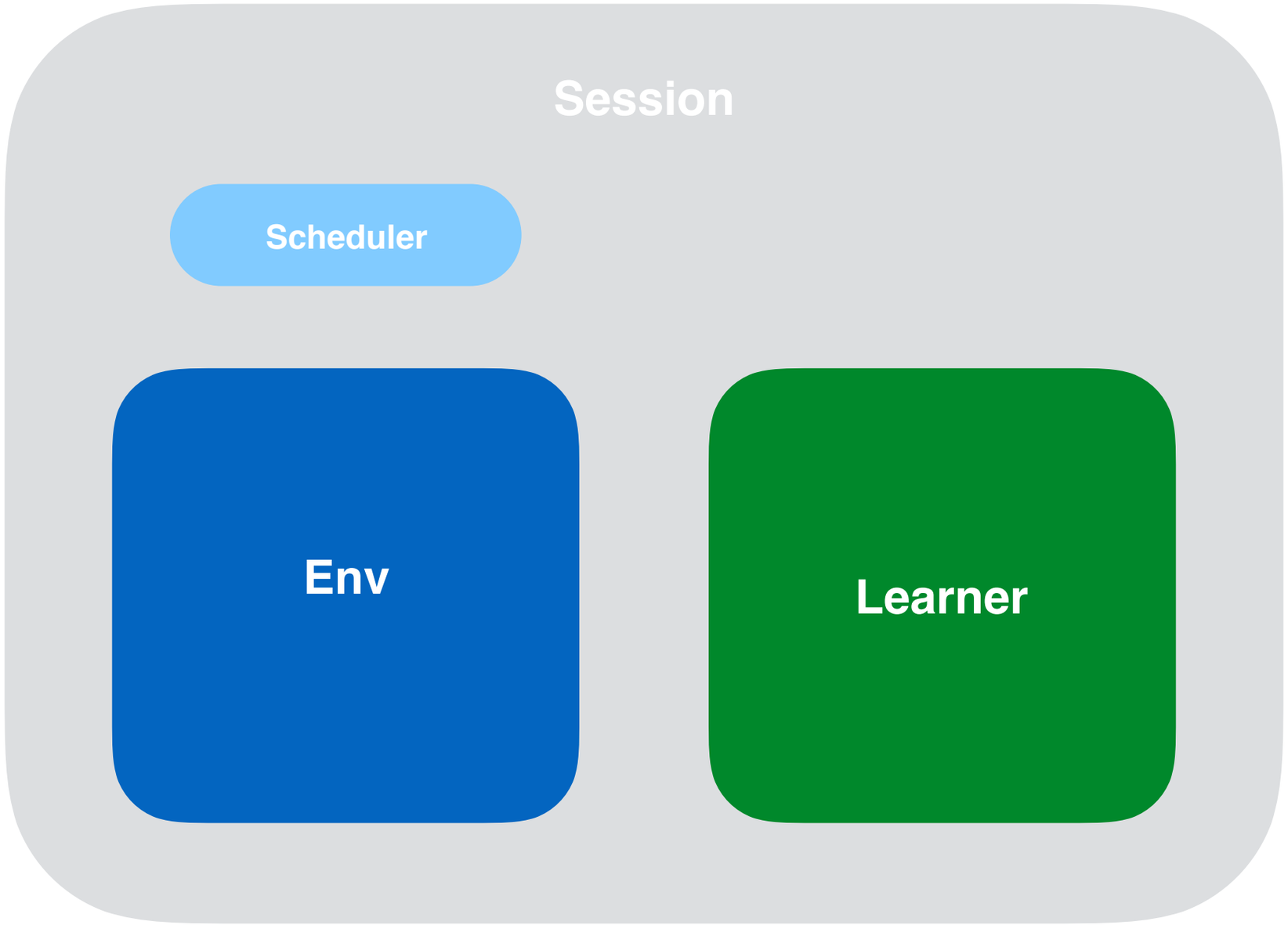

Session
Scheduler
Env
Learner

python run.py tasks_config.sample.json

```
{
  "worlds": {
    "gw": {
      "type": "worlds.grid_world.GridWorld"
    }
  },
  "tasks":
  {
    "k0": {
      "type": "tasks.competition.repetition.BeSilentTask",
      "world": "gw"
    },
    "g15": {
        "type": "tasks.competition.repetition.RepeatCharacterTask",
        "world": "gw"
    },
    "k2": {
        "type": "tasks.competition.repetition.RepeatWhatISayTask",
        "world": "gw"
    },

                              . . .

    "aj2": {
        "type": "tasks.competition.navigation.LookAroundTask",
        "world": "gw"
    },
    "aj3": {
        "type": "tasks.competition.navigation.FindObjectAroundTask",
        "world": "gw"
    }
  },
  "scheduler":
    {
      "type": 'core.scheduler.RandomTaskScheduler",
      "args": {
          'tasks": ["k0", "g15", "k2", "k5", "k6", "k7", "k8", "k9",
            "a1", "a2",
            "t1", "t2", "t3",
            "m1", "m2", "m3", "m4", "m5", "m6", "m7", "m8", "m9", "m10", "m11", "m12",
            "g0", "g1", "g2", "g3", "g4", "g6", "g7", "g8", "g9", "g10", "g13", "g14",
            "aj1", "aj2", "aj3"]
        }
    }
}
```

python run.py tasks_config.sample.json


```
"aj2": {
    "type": "tasks.competition.navigation.LookAroundTask",
    "world": "gw"
},
"aj3": {
    "type": "tasks.competition.navigation.FindObjectAroundTask",
    "world": "gw"
}
  },
  "scheduler":
    {
      "type": "core.scheduler.RandomTaskScheduler",
      "args": {
          "tasks": ["k0", "g15", "k2", "k5", "k6", "k7", "k8", "k9",
            "a1", "a2",
            "t1", "t2", "t3",
            "m1", "m2", "m3", "m4", "m5", "m6", "m7", "m8", "m9", "m10", "m11", "m12",
            "g0", "g1", "g2", "g3", "g4", "g6", "g7", "g8", "g9", "g10", "g13", "g14",
            "aj1", "aj2", "aj3"]
        }
    }
}
```


Session
Scheduler
Env
Learner

Loop

Pick next task

While not task.finished:

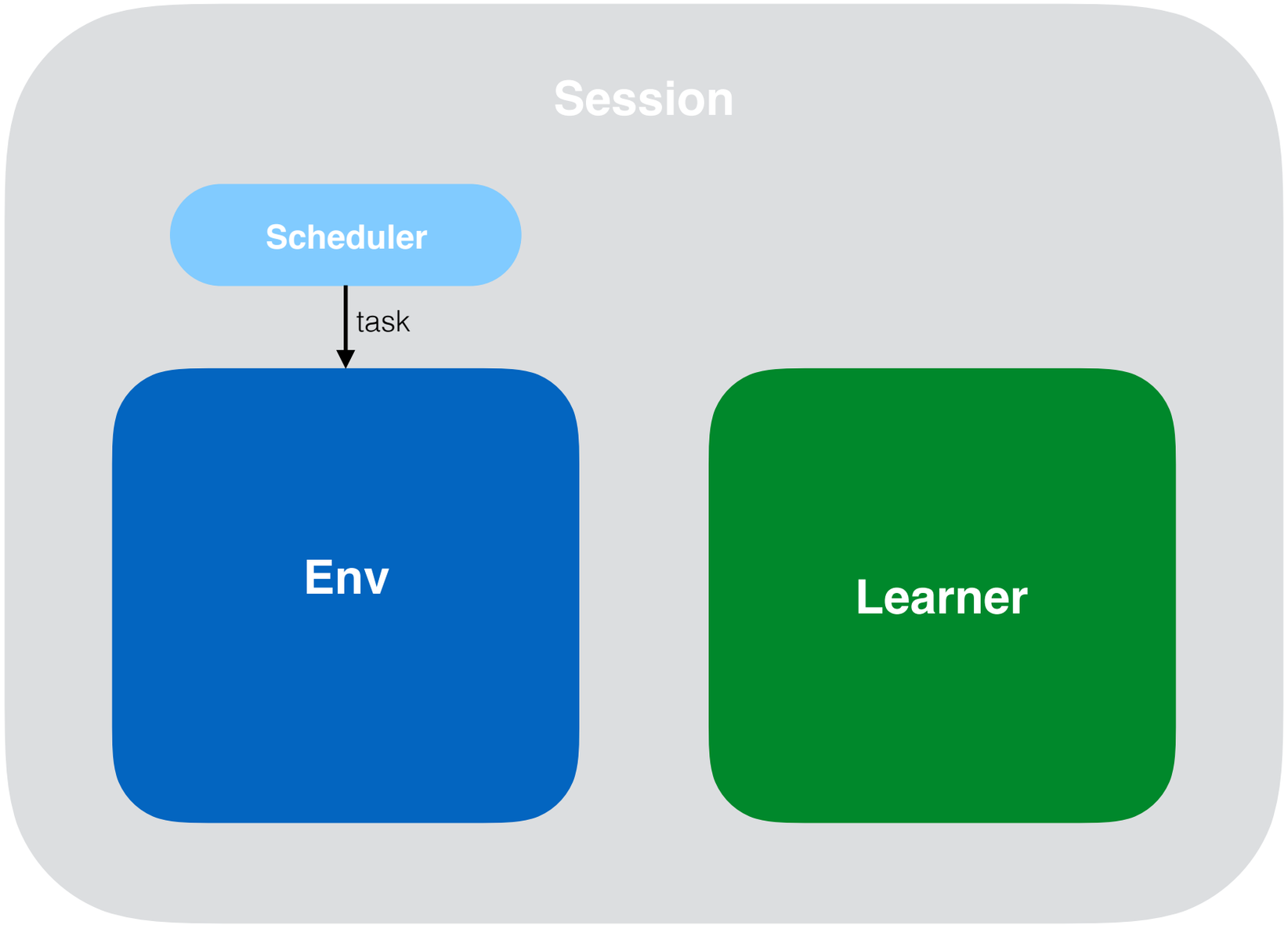
env sends a bit

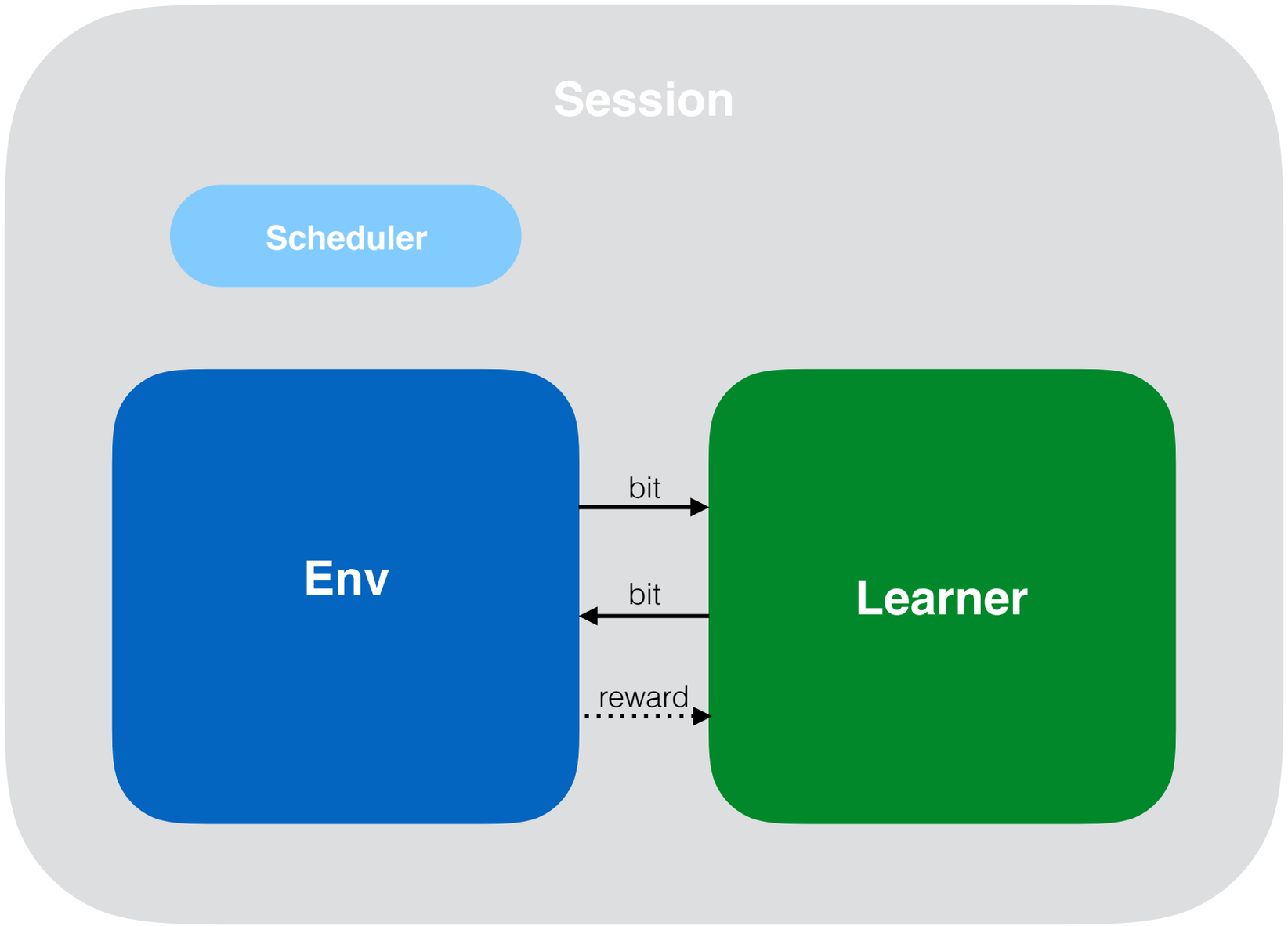
learner sends a bit

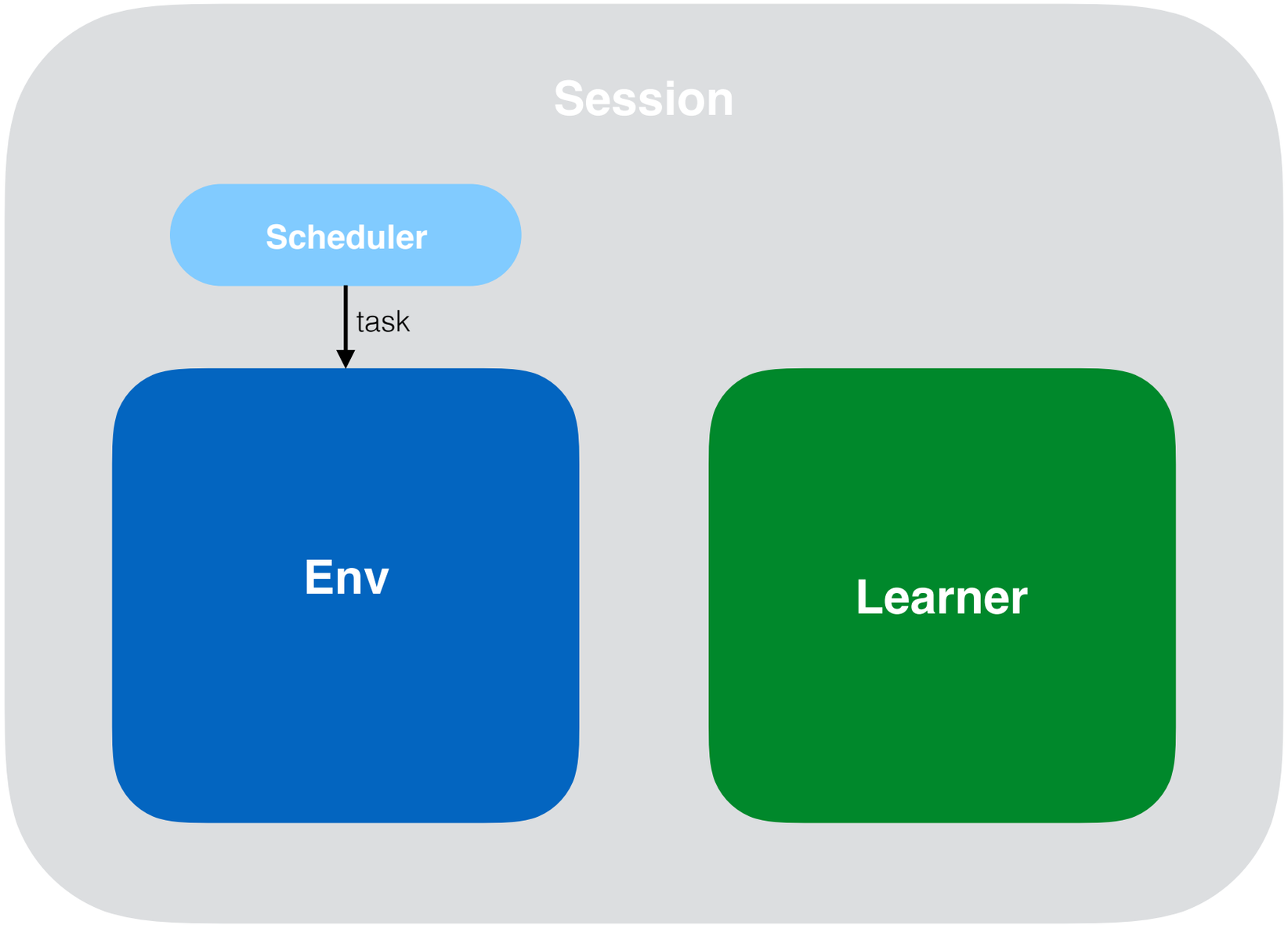
if learner succeeds, fails, or time out:

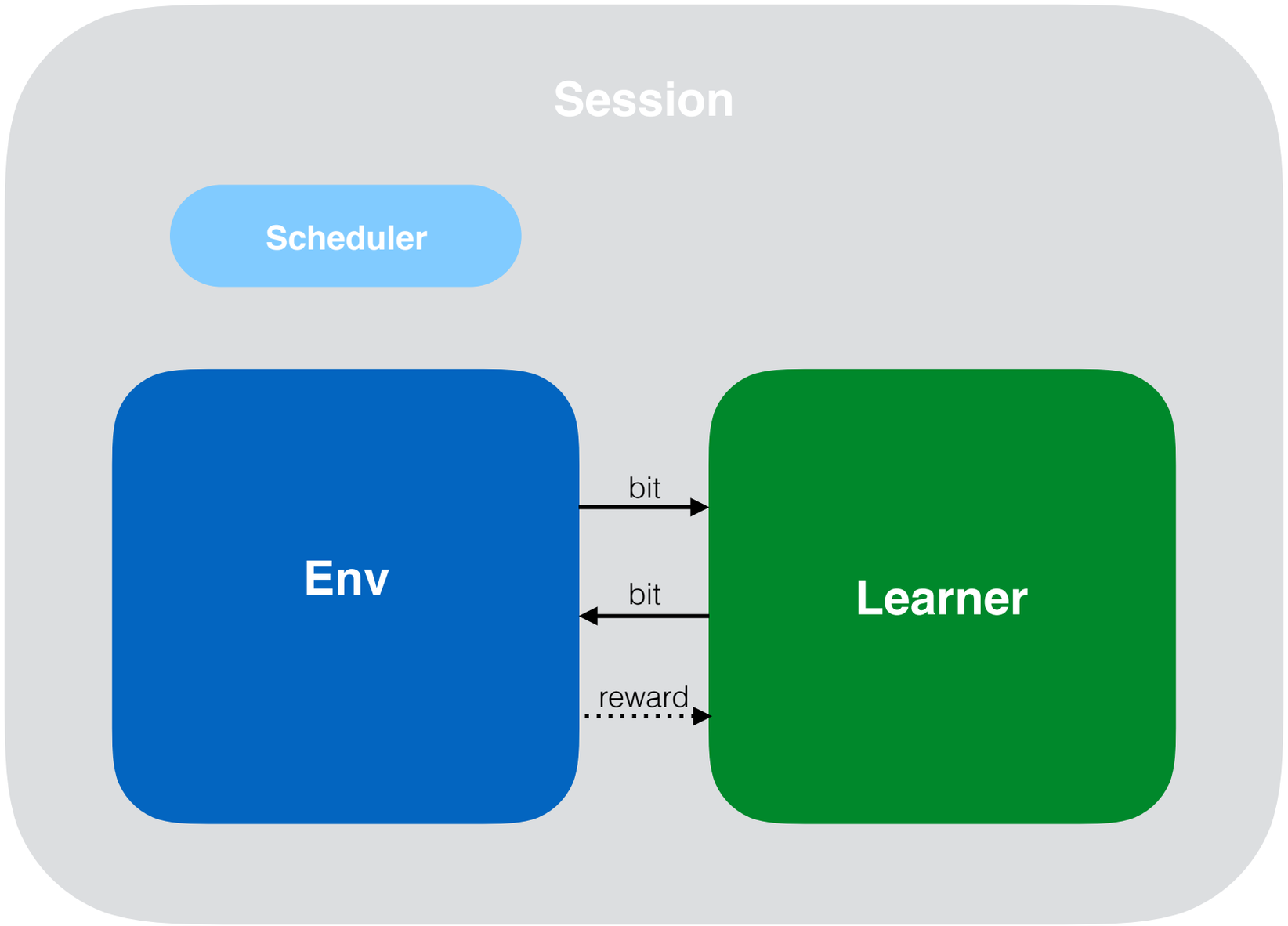
env sends reward

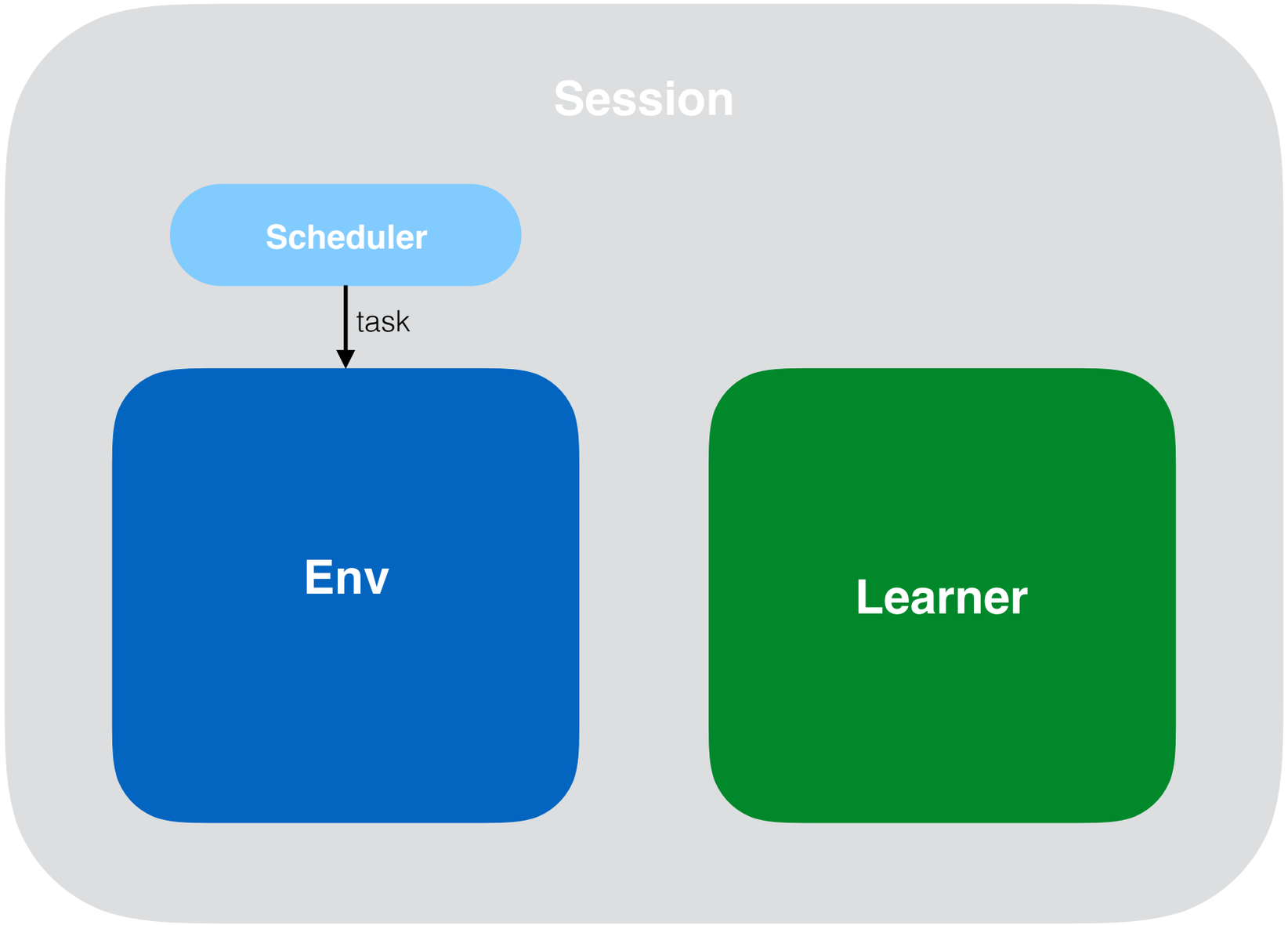
task.finished = true

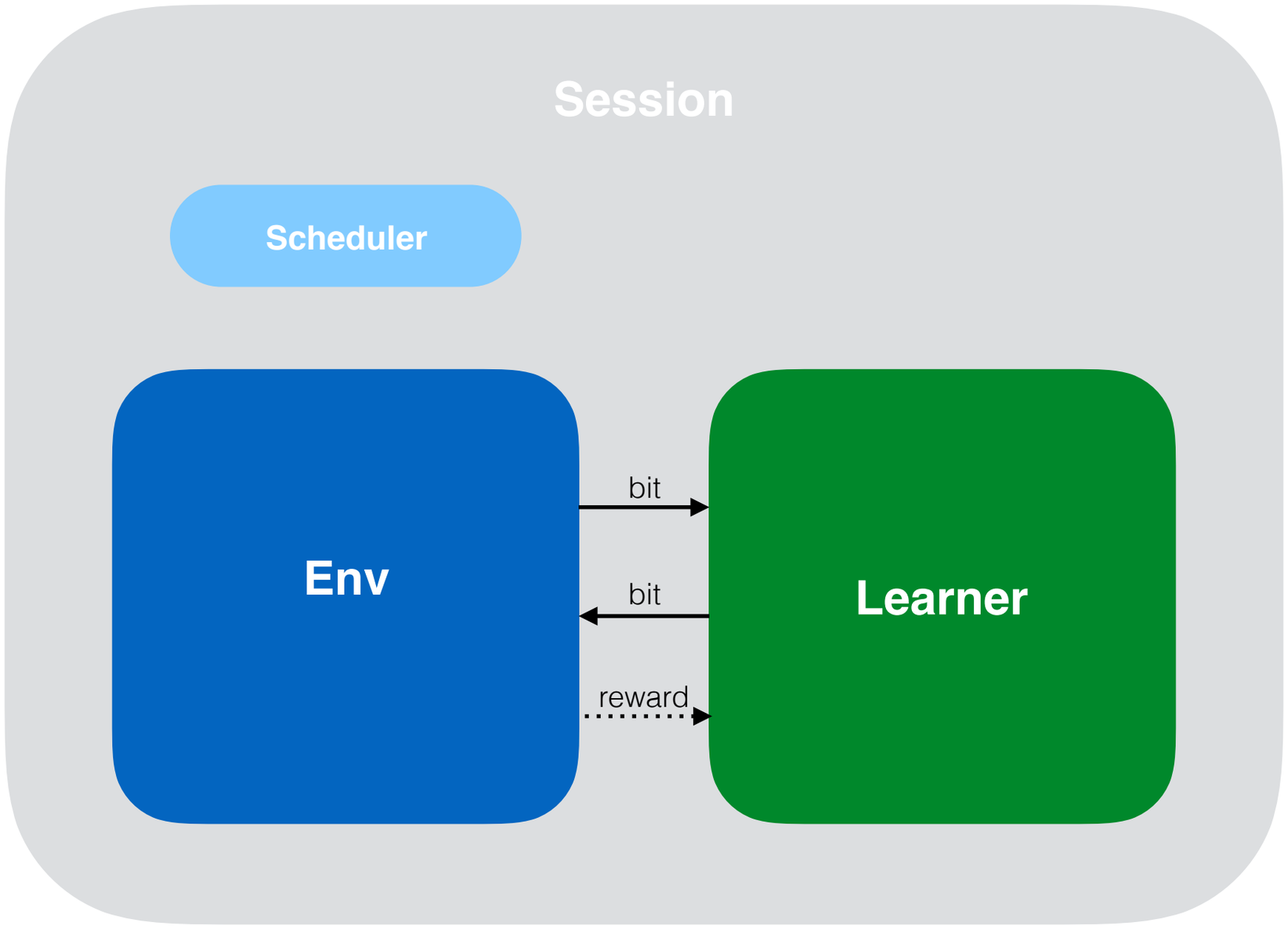

Session
Scheduler
task
Env
Learner


Session
Scheduler
Env
bit
bit
reward
Learner


Session
Scheduler
task
Env
Learner


Session
Scheduler
Env
bit
bit
reward
Learner


Session
Scheduler
task
Env
Learner


Session
Scheduler
Env
bit
bit
reward
Learner

Demo

- Human mode:
  - python run.py my_tasks.json
- Scrambled:
  - python run.py my_tasks.json --scramble
- With a learner:
  - python run.py my_tasks.json \
    -l learners.sample_learners.SampleRepeatingLearner
- Remote learner:
  - python run.py my_tasks.json -l learners.base.RemoteLearner \
    --learner-cmd "/my/learner/binary"

Events

on_init

on_start

on_message

on_sequence

on_output_message

on_output_sequence

on_state_changed

on_timeout

on_world_init

on_world_start

BeSilentTask

```
class BeSilentTask(Task):
    def __init__(self, world=None):
        super(BeSilentTask, self).__init__(world=world,
                                           max_time=random.randint(100, 1000))

    # give instructions at the beginning of the task
    @on_start()
    def on_start(self, event):
        self.set_message(random.choice(["be silent now.",
                                        "do not say anything."]))

    # catch any non-space character
    @on_message("[^ ]")
    def on_message(self, event):
        self.set_reward(0, random.choice(msg.failed))

    # when the maximum amount of time set for the task has elapsed
    @on_timeout()
    def on_timeout(self, event):
        self.set_reward(1, random.choice(msg.congratulations))
```

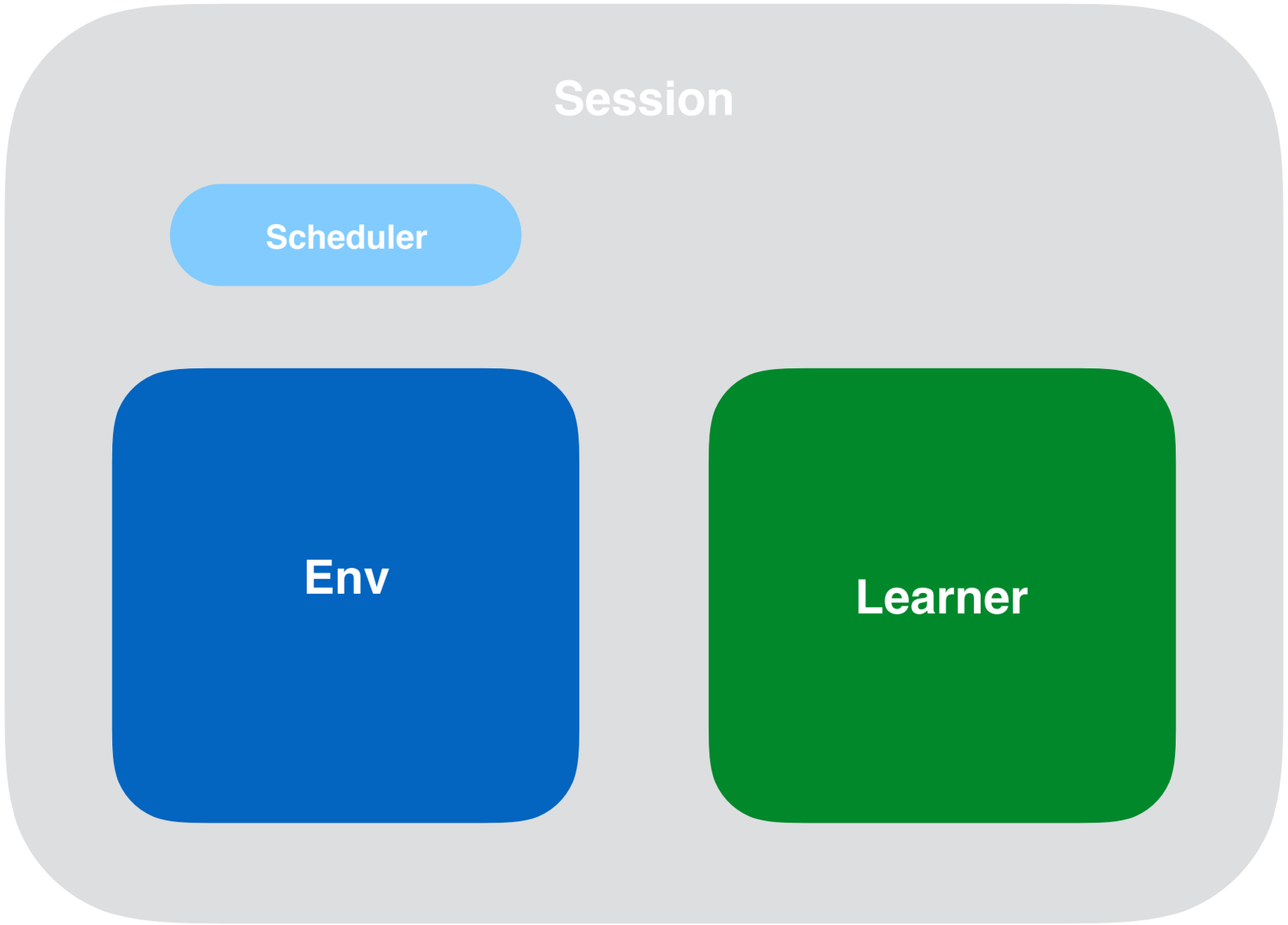
SampleRepeatingLearner

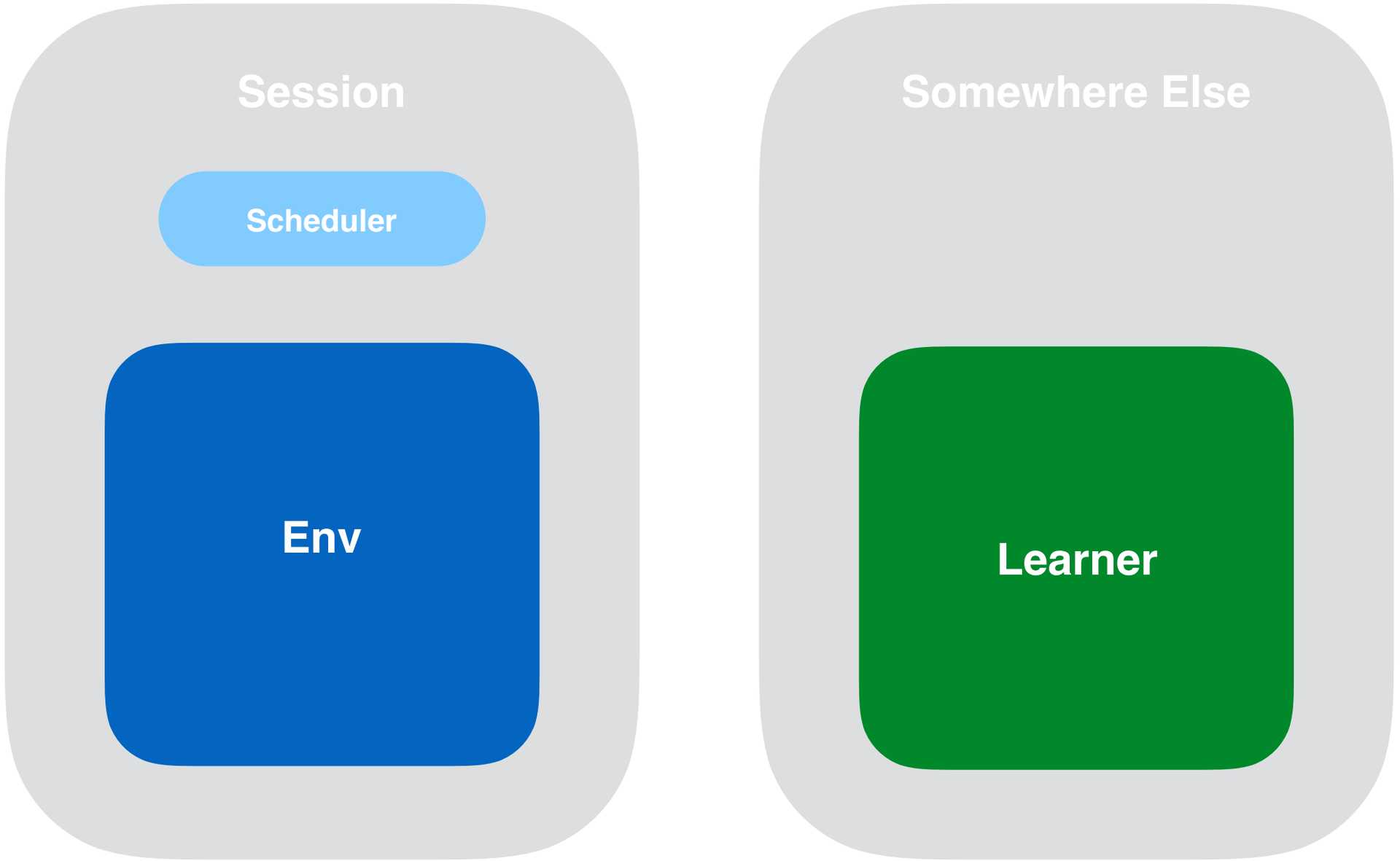
```
class SampleRepeatingLearner(BaseLearner):
    def reward(self, reward):
        # YEAH! Reward!!! Whatever...
        pass

    def next(self, input):
        # do super fancy computations
        # return our guess
        return input
```

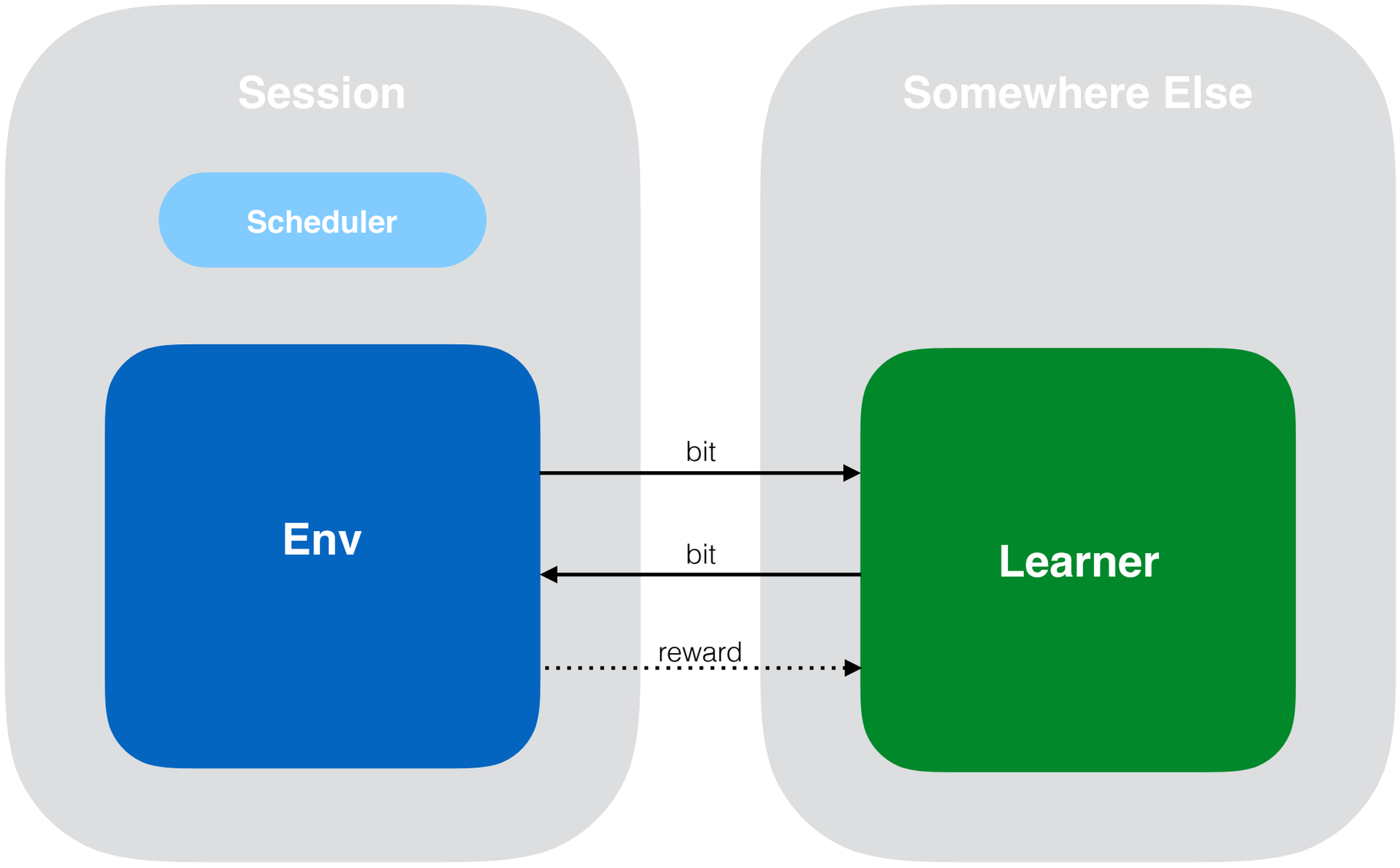
A learner in C

```c
#include <string.h>
#include <zmq.h>

// This is an example of a silly learner that always replies with '.'

int main()
{
   char reply[1];
   int n = 0;
   const char* response = "00101110";  // '.' utf-8 code

   // connect
   void *context = zmq_ctx_new();
   void *to_env = zmq_socket(context, ZMQ_PAIR);
   int rc = zmq_connect(to_env, "tcp://localhost:5556");

   zmq_send(to_env, "hello", 5, 0);    // handshake

   while (true) {
      zmq_recv(to_env, reply, 1, 0);   // receive reward
      zmq_recv(to_env, reply, 1, 0);   // receive teacher/env bit

      reply[0] = response[n % strlen(response)];
      zmq_send(to_env, reply, 1, 0);
      n += 1;
   }
   return 0;
}
```

```
while (true) {
    zmq_recv(to_env, reply, 1, 0);    // receive reward
    zmq_recv(to_env, reply, 1, 0);    // receive teacher/env bit

    reply[0] = response[n % strlen(response)];
    zmq_send(to_env, reply, 1, 0);
    n += 1;
}
```


Session
Scheduler
Env
Learner


Session
Scheduler
Env
Somewhere Else
Learner


Session
Scheduler
Env
bit
bit
reward
Somewhere Else
Learner

Examples

Counting
Object existence 1
Object existence 2
Associate object with property
Verify that object has property
List properties of an object
Name a property of an object
List objects with a certain property
Name an object with a property
Who has a certain object with a certain property
List the properties that an object has in a basket only
List the properties that an object has in all baskets
How many objects have a certain property
How many properties does an object have
Italian how many properties does an object have
Guess The Number Asking Questions (Explicit Model)
Guess The Number Asking Questions
Guess The Number Asking For Digits (Explicit Model)
Guess The Number Asking For Digits
Look
Look Around
Look For
Turning
Moving Forward
Moving relative
Moving absolute
Pick up Task
Pick up in front Task
Pick up around Task
Giving
Pick up around and give
Pick up around many and give
Counting in inventory
Counting in inventory with giving

T: say anything you want.
L: blablabla.
T: correct. [R+1]

T: say apple.
L: apple.
T: correct. [R+1]

T: repeat hello world.
L: blabla.
T: wrong, you should have said hello world.

T: i have an apple. do i have a banana?
L: yes.
T: wrong, i do not have a banana.

T: i have an apple and a banana and no pear. do i have a pear?
L: no
T: correct [R+1]

T: i have an apple, an apple and a banana. how many bananas do I have?
L: one
T: bravo! [R+1]

T: apple in john’s basket is green. how is apple?
L: green.

T: is apple green in john’s basket?
L: yes.

T: which properties does apple have in mary’s basket?
L: red, sweet and hard.

T: who has a yellow pineapple in the basket?
L: john and mary.

T: how many objects are yellow in john’s basket?
L: two.

T: move forward.
L: i move forward.
T: you moved [R+1].

T: i have placed an apple where you are. pick up the apple.
L: i pick up the apple.
T: you picked up the apple [R+1].

T: there is an apple in front of you. pick up the apple.
L: i pick up the apple.
T: there is no apple here.
L: i move forward.
T: you moved.
L: i pick up the apple.
T: you picked up the apple [R+1].

Task Documentation

https://github.com/facebookresearch/CommAI-env/blob/master/TASKS.md

T: Thank the audience.

L: Thank the audience.

T: Sigh... Just say "Thank you".

L: Thank you!

T: Good job. [+1]