Install dependencies and create a virtual screen 🔽

The first step is to install the dependencies, we'll install multiple ones.

- `gymnasium[box2d]`: Contains the LunarLander-v2 environment 🌛
- `stable-baselines3[extra]`: The deep reinforcement learning library.
- `huggingface_sb3`: Additional code for Stable-baselines3 to load and upload models from the Hugging Face 🤗 Hub.

To make things easier, we created a script to install all these dependencies.

```
!apt install swig cmake
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cmake is already the newest version (3.22.1-1ubuntu1.22.04.2).
Suggested packages:
  swig-doc swig-examples swig4.0-examples swig4.0-doc
The following NEW packages will be installed:
  swig swig4.0
0 upgraded, 2 newly installed, 0 to remove and 20 not upgraded.
Need to get 1,116 kB of archives.
After this operation, 5,542 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig4.0 amd64 4.0.2-1ubuntu1 [1,110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig all 4.0.2-1ubuntu1 [5,632 B]
Fetched 1,116 kB in 1s (827 kB/s)
Selecting previously unselected package swig4.0.
(Reading database ... 124926 files and directories currently installed.)
Preparing to unpack .../swig4.0_4.0.2-1ubuntu1_amd64.deb ...
Unpacking swig4.0 (4.0.2-1ubuntu1) ...
Selecting previously unselected package swig.
Preparing to unpack .../swig_4.0.2-1ubuntu1_all.deb ...
Unpacking swig (4.0.2-1ubuntu1) ...
Setting up swig4.0 (4.0.2-1ubuntu1) ...
Setting up swig (4.0.2-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
```

```
!pip install -r https://raw.githubusercontent.com/huggingface/deep-rl-class/main/notebooks/unit1/requirements-unit1.txt
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 68.4 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 55.1 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 843.8 kB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.5 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 19.3 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.0 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.7 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 90.9 MB/s eta 0:00:00
Building wheels for collected packages: box2d-py
  Building wheel for box2d-py (setup.py) ... done
```

```
!sudo apt-get update
!sudo apt-get install -y python3-opengl
!apt install ffmpeg
!apt install xvfb
!pip3 install pyvirtualdisplay
```

```
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfontenc1 libxfont2 libxkbfile1 x11-xkb-utils xfonts-base xfonts-encodings xfonts-utils
  xserver-common
The following NEW packages will be installed:
  libfontenc1 libxfont2 libxkbfile1 x11-xkb-utils xfonts-base xfonts-encodings xfonts-utils
  xserver-common xvfb
0 upgraded, 9 newly installed, 0 to remove and 22 not upgraded.
Need to get 7,815 kB of archives.
After this operation, 11.9 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64 1:1.1.4-1build3 [14.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libxfont2 amd64 1:2.0.5-1build1 [94.5 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libxkbfile1 amd64 1:1.1.0-1build3 [71.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/main amd64 x11-xkb-utils amd64 7.7+5build4 [172 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all 1:1.0.5-0ubuntu2 [578 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64 1:7.7+6build2 [94.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-base all 1:1.0.5 [5,896 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 xserver-common all 2:21.1.4-2ubuntu1.7~22.04.12 [28.7 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 xvfb amd64 2:21.1.4-2ubuntu1.7~22.04.12 [864 kB]
Fetched 7,815 kB in 2s (4,401 kB/s)
Selecting previously unselected package libfontenc1:amd64.
(Reading database ... 128763 files and directories currently installed.)
Preparing to unpack .../0-libfontenc1_1%3a1.1.4-1build3_amd64.deb ...
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) ...
Selecting previously unselected package libxfont2:amd64.
Preparing to unpack .../1-libxfont2_1%3a2.0.5-1build1_amd64.deb ...
Unpacking libxfont2:amd64 (1:2.0.5-1build1) ...
Selecting previously unselected package libxkbfile1:amd64.
Preparing to unpack .../2-libxkbfile1_1%3a1.1.0-1build3_amd64.deb ...
Unpacking libxkbfile1:amd64 (1:1.1.0-1build3) ...
```

```
import os
os.kill(os.getpid(), 9)
```

```
# Virtual display
from pyvirtualdisplay import Display

virtual_display = Display(visible=0, size=(1400, 900))
virtual_display.start()
```

```
<pyvirtualdisplay.display.Display at 0x794986f94310>
```

```
import gymnasium

from huggingface_sb3 import load_from_hub, package_to_hub
from huggingface_hub import notebook_login # To log to our Hugging Face account to be able to upload models to the Hub.

from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.monitor import Monitor
```

Understand Gymnasium and how it works 🤖

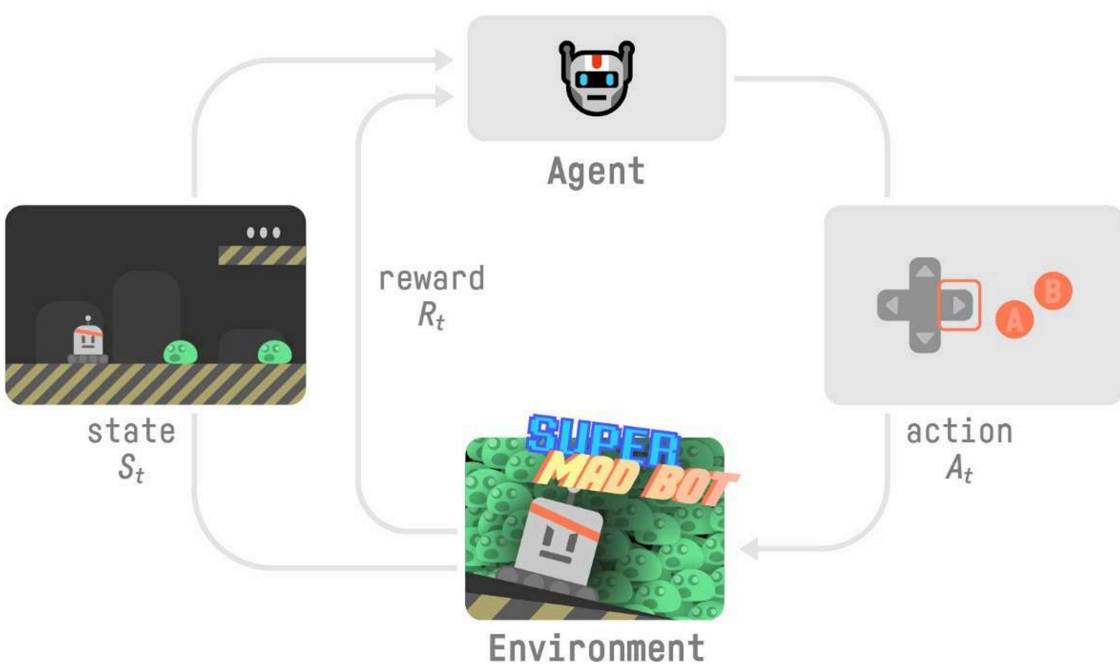
🏋 The library containing our environment is called Gymnasium. **You'll use Gymnasium a lot in Deep Reinforcement Learning.**

Gymnasium is the **new version of Gym library** [maintained by the Farama Foundation](https://github.com/Farama-Foundation/Gymnasium).

The Gymnasium library provides two things:

- An interface that allows you to **create RL environments**.
- A **collection of environments** (gym-control, atari, box2D...).

Let's look at an example, but first let's recall the RL loop.



At each step:

- Our Agent receives a **state (S0)** from the **Environment** — we receive the first frame of our game (Environment).
- Based on that **state (S0),** the Agent takes an **action (A0)** — our Agent will move to the right.
- The environment transitions to a **new state (S1)** — new frame.
- The environment gives some **reward (R1)** to the Agent — we're not dead *(Positive Reward +1)*.

With Gymnasium:

1️⃣ We create our environment using `gymnasium.make()`

2️⃣ We reset the environment to its initial state with `observation = env.reset()`

At each step:

3️⃣ Get an action using our model (in our example we take a random action)

4️⃣ Using `env.step(action)`, we perform this action in the environment and get

- `observation`: The new state (st+1)
- `reward`: The reward we get after executing the action
- `terminated`: Indicates if the episode terminated (agent reach the terminal state)
- `truncated`: Introduced with this new version, it indicates a timelimit or if an agent go out of bounds of the environment for instance.
- `info`: A dictionary that provides additional information (depends on the environment).

For more explanations check this 👉 https://gymnasium.farama.org/api/env/#gymnasium.Env.step

If the episode is terminated:

- We reset the environment to its initial state with `observation = env.reset()`

Let's look at an example! Make sure to read the code

```
import gymnasium as gym

# First, we create our environment called LunarLander-v2
env = gym.make("LunarLander-v2")

# Then we reset this environment
observation, info = env.reset()

for _ in range(20):
  # Take a random action
  action = env.action_space.sample()
  print("Action taken:", action)

  # Do this action in the environment and get
  # next_state, reward, terminated, truncated and info
  observation, reward, terminated, truncated, info = env.step(action)
  print(reward)
  print(observation)
  print(terminated)
  print(truncated)

  # If the game is terminated (in our case we land, crashed) or truncated (timeout)
  if terminated or truncated:
      # Reset the environment
      print("Environment is reset")
      observation, info = env.reset()

env.close()
```

```
Action taken: 1
-1.1223890014733218
[-1.1947632e-03  1.2671293e+00 -3.0168828e-02 -6.7983365e-01
 -2.3416940e-02  8.2156947e-03  0.0000000e+00  0.0000000e+00]
False
False
Action taken: 2
1.7565513588630324
[-0.00150394  1.2519485  -0.03121868 -0.674698   -0.02311482  0.00604295
  0.          0.        ]
False
False
Action taken: 1
-0.9695994976902387
[-0.00187359  1.2361677  -0.03881315 -0.7013475  -0.02129037  0.03649235
  0.          0.        ]
False
False
Action taken: 1
-0.6912739463114679
[-0.00230932  1.219801   -0.04708974 -0.7273626  -0.01780432  0.06972723
  0.          0.        ]
False
False
Action taken: 3
-0.7682113321771726
[-0.00268545  1.2028313  -0.03962933 -0.75418293 -0.01581722  0.03974576
  0.          0.        ]
False
False
Action taken: 2
```

Let's see what the Environment looks like:

```
# We create our environment with gym.make("<name_of_the_environment>")
env = gym.make("LunarLander-v2")
env.reset()
print("_____OBSERVATION SPACE_____ \n")
print("Observation Space Shape", env.observation_space.shape)
print("Sample observation", env.observation_space.sample()) # Get a random observation
```

```
_____OBSERVATION SPACE_____

Observation Space Shape (8,)
Sample observation [-19.766953   -38.19351      4.840538     4.8456907   -2.1288986
  -0.65268654   0.23250899   0.23430432]
```

We see with `Observation Space Shape (8,)` that the observation is a vector of size 8, where each value contains different information about the lander:

- Horizontal pad coordinate (x)
- Vertical pad coordinate (y)
- Horizontal speed (x)
- Vertical speed (y)
- Angle
- Angular speed
- If the left leg contact point has touched the land (boolean)
- If the right leg contact point has touched the land (boolean)

```
print("\n _____ACTION SPACE_____ \n")
print("Action Space Shape", env.action_space.n)
print("Action Space Sample", env.action_space.sample()) # Take a random action
```

Vectorized Environment

- We create a vectorized environment (a method for stacking multiple independent environments into a single environment) of 16 environments, this way, **we'll have more diverse experiences during the training.**

```
# Create the environment
env = make_vec_env('LunarLander-v2', n_envs=32)
```

Stable-Baselines3 is easy to set up:

1️⃣ You **create your environment** (in our case it was done above)

2️⃣ You define the **model you want to use and instantiate this model** `model = PPO("MlpPolicy")`

3️⃣ You **train the agent** with `model.learn` and define the number of training timesteps

```
# Create environment
env = gym.make('LunarLander-v2')

# Instantiate the agent
model = PPO('MlpPolicy', env, verbose=1)
# Train the agent
model.learn(total_timesteps=int(2e5))
```

```
# TODO: Define a PPO MlpPolicy architecture
# We use MultiLayerPerceptron (MLPPolicy) because the input is a vector,
# if we had frames as input we would use CnnPolicy
model = PPO("MlpPolicy"
            , env = env
            , n_steps=2048
            , batch_size=64
            , n_epochs=5
            , gamma = 0.999
            , gae_lambda=0.98
            , ent_coef=0.01
            , verbose=1)
```

```
Using cuda device
```

Solution

Train the PPO agent 🏃

- Let's train our agent for 1,000,000 timesteps, don't forget to use GPU on Colab. It will take approximately ~20min, but you can use fewer timesteps if you just want to try it out.
- During the training, take a ☕ break you deserved it 🤗

```
# TODO: Train it for 1,000,000 timesteps

# TODO: Specify file name for model and save the model to file

model.learn(total_timesteps=1200000)
model_name = "ppo-LunarLander-v2"
model.save(model_name)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 90.5     |
|    ep_rew_mean     | -176     |
| time/              |          |
|    fps             | 5158     |
|    iterations      | 1        |
|    time_elapsed    | 12       |
|    total_timesteps | 65536    |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 92.7        |
|    ep_rew_mean          | -140        |
| time/                   |             |
|    fps                  | 2689        |
|    iterations           | 2           |
|    time_elapsed         | 48          |
|    total_timesteps      | 131072      |
| train/                  |             |
|    approx_kl            | 0.008538063 |
|    clip_fraction        | 0.0836      |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.38       |
```

```
|    explained_variance   | -0.000501    |
|    learning_rate        | 0.0003       |
|    loss                 | 824          |
|    n_updates            | 5            |
|    policy_gradient_loss | -0.00722     |
|    value_loss           | 2.63e+03     |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 87           |
|    ep_rew_mean          | -103         |
| time/                   |              |
|    fps                  | 2307         |
|    iterations           | 3            |
|    time_elapsed         | 85           |
|    total_timesteps      | 196608       |
| train/                  |              |
|    approx_kl            | 0.0075173494 |
|    clip_fraction        | 0.0695       |
|    clip_range           | 0.2          |
|    entropy_loss         | -1.37        |
|    explained_variance   | -3.39e-05    |
|    learning_rate        | 0.0003       |
|    loss                 | 370          |
|    n_updates            | 10           |
|    policy_gradient_loss | -0.00604     |
|    value_loss           | 1.14e+03     |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 93          |
|    ep_rew_mean          | -86.8       |
| time/                   |             |
|    fps                  | 2134        |
```

Solution

Evaluate the agent 📈

- Remember to wrap the environment in a [Monitor](https://stable-baselines3.readthedocs.io/en/master/common/monitor.html).
- Now that our Lunar Lander agent is trained 🚀, we need to **check its performance**.
- Stable-Baselines3 provides a method to do that: `evaluate_policy`.
- To fill that part you need to [check the documentation](https://stable-baselines3.readthedocs.io/)
- In the next step, we'll see **how to automatically evaluate and share your agent to compete in a leaderboard, but for now let's do it ourselves**

💡 When you evaluate your agent, you should not use your training environment but create an evaluation environment.

```
# TODO: Evaluate the agent
# Create a new environment for evaluation
eval_env = Monitor(gym.make('LunarLander-v2'))

# Evaluate the model with 10 evaluation episodes and deterministic=True
mean_reward, std_reward = evaluate_policy(model, model.get_env(), n_eval_episodes=10, deterministic=True)

# Print the results
print(f"mean_reward={mean_reward:.2f} +/- {std_reward}")
```

```
mean_reward=233.32 +/- 41.61997686557959
```

About the Unit 1 Exercise

- I had 3 iterations and ended up uploading at -> https://huggingface.co/Vanheart/ppo-LunarLander-v2 a model with 254.49 +/- 18.85 reward.
- Overall it was a nice exercise to understand the ideas of setting up enverinoments
- Also very cool to see the LunarLander working in detail.
- For bonus I created a huggy model (a puppy that catches a twig basically, but the model uses ML Agents from unity to learn how to control their legs which are 3D, functioning like a motor. You can check it here -> https://huggingface.co/Vanheart/ppo-Huggy