In [1]:
```
import sqlalchemy
from sqlalchemy import create_engine
import pymysql
import os
import pandas as pd
import matplotlib.pyplot as plt
from finta import TA

from datetime import datetime

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import ray
import ray.rllib.agents.ppo as ppo
import tensortrade.env.default as default
from gym.spaces import Discrete
from ray import tune
from ray.tune.registry import register_env
from symfit import parameters, variables, sin, cos, Fit
from tensortrade.env.default.actions import TensorTradeActionScheme
from tensortrade.env.default.rewards import TensorTradeRewardScheme
from tensortrade.env.generic import Renderer
from tensortrade.feed.core import DataFeed, Stream
from tensortrade.oms.exchanges import Exchange
from tensortrade.oms.exchanges import ExchangeOptions
from tensortrade.oms.instruments import Instrument
from tensortrade.oms.orders import proportion_order
from tensortrade.oms.services.execution.simulated import execute_order
from tensortrade.oms.wallets import Wallet, Portfolio

from tensortrade.env.default.actions import BSH
from tensortrade.env.default.rewards import PBR

import tensortrade.stochastic as sp

import tensortrade.env.default.rewards as rewards
from tensortrade.env.default.actions import ManagedRiskOrders
from tensortrade.env.default.rewards import SimpleProfit
```

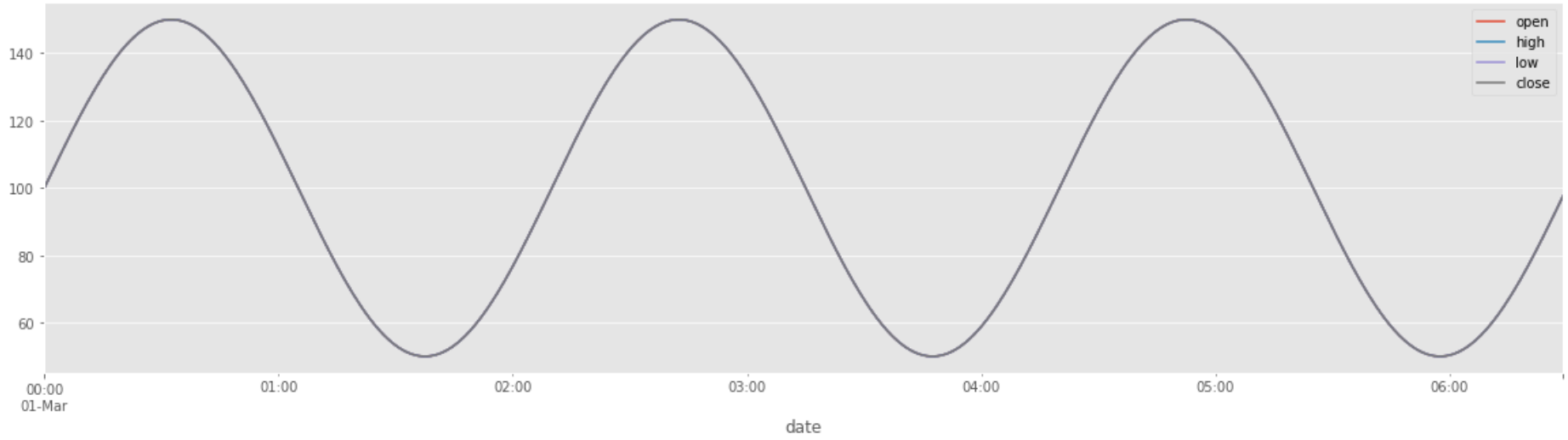
```
WARNING:tensorflow:From /root/anaconda3/lib/python3.8/site-packages/tensorflow/python/compat/v2_compat.py:9
```

```
6: disable_resource_variables (from tensorflow.python.ops.variable_scope) is deprecated and will be removed
in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [2]:
```python
def get_data():
    x = np.arange(0, 2*np.pi, 2*np.pi / 390)
    y = 50*np.sin(3*x) + 100
    data = pd.DataFrame(data=y, columns=["open"])
    data["date"] = pd.date_range(start='03/01/2021', freq='MIN', periods=390)
    data["high"] = data["open"]
    data["low"] = data["open"]
    data["close"] = data["open"]
    data["volume"] = 10
    data = data[["date", "open", "high", "low", "close", "volume"]]
    return data
```

In [3]:
```python
data = get_data()
data.plot(x="date", y=["open", "high", "low", "close"], figsize=(20,5))
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa6c01621c0>
```



In [4]:

```
USD = Instrument("USD", 2, "U.S. Dollar")
TSLA = Instrument("TSLA", 8, "Tesla, Inc.")

def create_env(config):

    # Sine wave
    data = get_data()

    p = Stream.source(list(data["close"]), dtype="float").rename("USD-TSLA")

    simulated_exchange = Exchange(
        "simulated_exchange",
        service=execute_order,
        options=ExchangeOptions(commission=0.03)
    )( p )

    cash = Wallet(simulated_exchange, 10000 * USD)
    asset = Wallet(simulated_exchange, 0 * TSLA)

    portfolio = Portfolio(USD, [
        cash,
        asset
    ])

    feed = DataFeed([
        Stream.source(list(data["open"]), dtype="float").rename("open"),
        Stream.source(list(data["high"]), dtype="float").rename("high"),
        Stream.source(list(data["low"]), dtype="float").rename("low"),
        Stream.source(list(data["close"]), dtype="float").rename("close"),
        Stream.source(list(data["volume"]), dtype="float").rename("volume"),
        p.ewm(span=10).mean().rename("fast"),
        p.ewm(span=50).mean().rename("medium"),
        p.ewm(span=100).mean().rename("slow"),
        p.log().diff().fillna(0).rename("lr")
    ])

    # A simple reward scheme that rewards the agent for incremental increases in net worth
    reward_scheme = SimpleProfit()

    # A discrete action scheme that determines actions based on managing risk
    action_scheme = ManagedRiskOrders()
```

```
    renderer_feed = DataFeed([
        Stream.source(list(data["date"]), dtype="datetime64").rename("date"),
        Stream.source(list(data["open"]), dtype="float").rename("open"),
        Stream.source(list(data["high"]), dtype="float").rename("high"),
        Stream.source(list(data["low"]), dtype="float").rename("low"),
        Stream.source(list(data["close"]), dtype="float").rename("close"),
        Stream.source(list(data["volume"]), dtype="float").rename("volume"),
    ])


    environment = default.create(
        feed=feed,
        portfolio=portfolio,
        action_scheme=action_scheme,
        reward_scheme=reward_scheme,
        renderer_feed=renderer_feed,
        renderer=default.renderers.PlotlyTradingChart(display=True, auto_open_html=False, save_format="png"),
        window_size=25,
        max_allowed_loss=0.6
    )
    return environment


register_env("TradingEnv", create_env)
```

In [5]:
```
analysis = tune.run(
    "PPO",
    stop={
      "info/num_steps_trained": 200000,
    },
    config={
        "env": "TradingEnv",
        "env_config": {
            "window_size": 25
        },
        "framework": "torch",
        "ignore_worker_failures": True,
        "num_workers": os.cpu_count() - 1,
        "num_gpus": 1,
        "clip_rewards": True,
        "lr": 8e-6,
        "gamma": 0,
        "observation_filter": "MeanStdFilter",
        "lambda": 0.72,
        "vf_loss_coeff": 0.5,
        "entropy_coeff": 0.01
    },
    checkpoint_at_end=True
)
```

```
        vf_loss: 0.14206236239635583
    num_steps_sampled: 12600
    num_steps_trained: 12600
  iterations_since_restore: 3
  node_ip: 192.168.0.18
  num_healthy_workers: 3
  off_policy_estimator: {}
  perf:
    cpu_util_percent: 68.15
    gpu_util_percent0: 0.11428571428571428
    ram_util_percent: 22.9
    vram_util_percent0: 0.22789896670493678
  pid: 1053045
  policy_reward_max: {}
  policy_reward_mean: {}
  policy_reward_min: {}
  sampler_perf:
```

```
_
mean action processing ms: 0.04718771616787459
```

In [6]:
```
checkpoints = analysis.get_trial_checkpoints_paths(
    trial=analysis.get_best_trial(metric="episode_reward_mean", mode="max"),
    metric="episode_reward_mean"
)
checkpoint_path = checkpoints[0][0]

# Restore agent
agent = ppo.PPOTrainer(
    env="TradingEnv",
    config={
        "env_config": {
            "window_size": 25
        },
        "framework": "torch",
        "ignore_worker_failures": True,
        "num_workers": 1,
        "num_gpus": 0,
        "clip_rewards": True,
        "lr": 8e-6,
        "gamma": 0,
        "observation_filter": "MeanStdFilter",
        "lambda": 0.72,
        "vf_loss_coeff": 0.5,
        "entropy_coeff": 0.01
    }
)
agent.restore(checkpoint_path)
```

```
2021-03-26 13:44:12,643 INFO trainer.py:641 -- Current log_level is WARN. For more information, set 'log_leve
l': 'INFO' / 'DEBUG' or use the -v and -vv flags.
(pid=1056829) WARNING:tensorflow:From /root/anaconda3/lib/python3.8/site-packages/tensorflow/python/compat/v2
_compat.py:96: disable_resource_variables (from tensorflow.python.ops.variable_scope) is deprecated and will
 be removed in a future version.
(pid=1056829) Instructions for updating:
(pid=1056829) non-resource variables are not supported in the long term
2021-03-26 13:44:18,279 INFO trainable.py:371 -- Restored on 192.168.0.18 from checkpoint: /root/ray_results/
PPO/PPO_TradingEnv_0f284_00000_0_2021-03-26_13-34-58/checkpoint_48/checkpoint-48
2021-03-26 13:44:18,283 INFO trainable.py:379 -- Current state after restoring: {'_iteration': 48, '_timestep
s_total': None, '_time_total': 543.5990972518921, '_episodes_total': 1114}
```

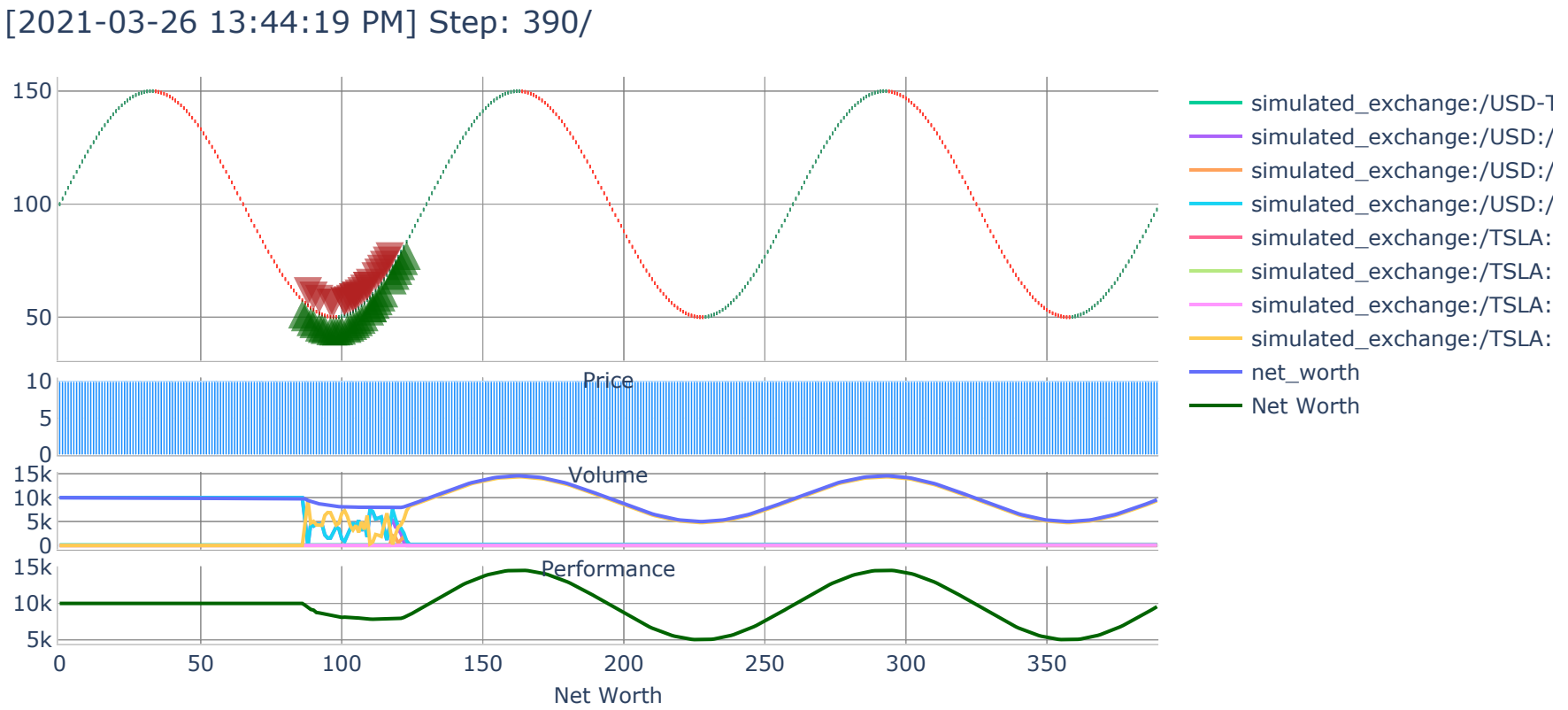
In [7]:

```
env = create_env({
    "window_size": 25
})


episode_reward = 0
done = False
obs = env.reset()

while not done:
    action = agent.compute_action(obs)
    obs, reward, done, info = env.step(action)
    episode_reward += reward

env.render()
```



[2021-03-26 13:44:19 PM] Step: 390/