```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Standard Library
from argparse import ArgumentParser, Namespace
from fractions import Fraction as Frac
from os.path import basename

# Relative
from utils import print_heading, print_matrix
from markov import exercise_1, exercise_2, exercise_3, exercise_4
from tests import test_exercise_1, test_exercise_2, test_exercise_3, test_exercise_4

parser: ArgumentParser = ArgumentParser(
    prog=basename(__file__).replace('.py', ''),
    description='solutions to stochastic systems assessment')

parser.add_argument(
    'exercise',
    help='the exercise number',
    choices=[1, 2, 3, 4],
    type=int)

parser.add_argument(
    '--test',
    help='run tests for the exercise instead of running it',
    action='store_true',
    default=False)

args: Namespace = parser.parse_args()

if args.exercise < 0 or args.exercise > 4:
    raise NotImplementedError(
        f'exercise number "{args.exercise}" is invalid, try 1-4')

if args.test:
    print_heading(f'tests for exercise {args.exercise}')
    if args.exercise == 1:
        test_exercise_1()
    elif args.exercise == 2:
        test_exercise_2()
    elif args.exercise == 3:
        test_exercise_3()
    elif args.exercise == 4:
        test_exercise_4()
else:
    print_heading(f'Solution to exercise {args.exercise}')

    if args.exercise == 1:
        SSP, trans_table = exercise_1()
        print_matrix(trans_table)

    elif args.exercise == 2:
        SSP, trans_table = exercise_2()
        print_matrix(trans_table)

    elif args.exercise == 3 or args.exercise == 4:
        std, p1, p3, p9 = exercise_3() if args.exercise == 3 else exercise_4()

        def show(n: int, p: Frac) -> None:
            print(f'probability for state {n}: {p}+-{std}')

        show(1, p1)
        show(3, p3)
        show(9, p9)
```