* Testēšana - 2 veiksmes scenāriji
* Zinot, ievadot izveli 1, spele turpinas ar “Piedāvāt noziedzniekam drošu izeju un advokātu", 80 procentu rezultats.

Zinot, ievadot izveli 1, spele turpinas ar "Pavēlēt nekavējoties padoties", 30 procentu rezultats.

* Testēšana - 4 lietošanas scenāriji

Zinot, ievadot izveli 5, spele turpinas ar atkal piedavajot izveli no 1 lidz 4

Zinot, ievadot izveli 8, spele turpinas ar atkal piedavajot izveli no 1 lidz 4

Zinot, ievadot izveli 9, spele turpinas ar atkal piedavajot izveli no 1 lidz 4

Zinot, ievadot izveli 0, spele turpinas ar atkal piedavajot izveli no 1 lidz 4

* Testēšana - 2 robež-scenāriji

Zinot, ievadot izveli 1, spele turpinas ar "Pavēlēt nekavējoties padoties", 30 procentu rezultats.

Zinot, ievadot izveli 2, spele turpinas ar “Piedāvāt noziedzniekam drošu izeju un advokātu", 80 procentu rezultats.

* Testēšana - 4 automatizēti testi (pytest bilbiotēka)

class TestNegotiator(unittest.TestCase):

def test\_choice\_1(self):

result = play\_situation(situation1, user\_choice=1)

self.assertEqual(result["choice"], 1)

self.assertEqual(result["selected\_text"], "Piedāvāt noziedzniekam drošu izeju un advokātu")

self.assertEqual(result["success\_rate"], 80)

self.assertEqual(result["result\_msg"], "Lieliski! Jūs pārliecinoši virzāties uz mērķi.")

def test\_choice\_2(self):

result = play\_situation(situation1, user\_choice=2)

self.assertEqual(result["choice"], 2)

self.assertEqual(result["success\_rate"], 30)

self.assertEqual(result["result\_msg"], "Sarunas neveicas jūsu labā...")

def test\_choice\_3(self):

result = play\_situation(situation1, user\_choice=3)

self.assertEqual(result["choice"], 3)

self.assertEqual(result["success\_rate"], 60)

self.assertEqual(result["result\_msg"], "Pieņemams rezultāts, bet ir kur tiekties.")

def test\_invalid\_choice(self):

with self.assertRaises(ValueError):

play\_situation(situation1, user\_choice=5)

# Запуск тестов

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()