

Lab: Market-Driven Business Simulation

Objective:

Simulate a simple market where companies compete using price and advertising strategies. The market determines demand, and companies calculate revenue, cost, and profit. Learn to apply object-oriented programming, randomness, and market modeling.

Part 1: Company

Each company has:

- **Intrinsic strategy** (price, advertising) defined at creation.
- Production cost per unit.
- Efficiency in advertising (how effective advertising is in increasing demand).
- Ability to calculate **units sold, revenue, cost, and profit** after market allocation.

```
class Company:
    def __init__(self, name, production_cost, efficiency, price, advertising):
        self.name = name
        self.production_cost = production_cost
        self.efficiency = efficiency
        self.price = price
        self.advertising = advertising

        self.units_sold = 0
        self.revenue = 0
        self.cost = 0
        self.profit = 0

    def receive_demand(self, units_demanded):
        """Calculate revenue, cost, and profit based on market demand."""
        self.units_sold += units_demanded
        self.cost += units_demanded * self.production_cost
        self.revenue += units_demanded * self.price
        self.profit += (units_demanded * self.price - units_demanded * self.production_cost)

    def __str__(self):
        return (f"{self.name}: Units Sold={self.units_sold}, "
                f"Revenue={self.revenue:.1f}, Cost={self.cost:.1f}, Profit={self.profit:.1f}")
```

Part 2: Market

The market:

- Allocates units to companies based on their strategy (advertising and price).
- Uses randomness to simulate market fluctuations.

```
import random

class Market:
    def __init__(self, total_units):
        self.total_units = total_units

    def calculate_demand(self, companies):
```

```

"""Allocate market units based on company strategies."""
scores = []
for c in companies:
    score = c.advertising * c. efficiency - 0.5 * c.price
    scores.append(max(score, 0))

total_score = sum(scores)
allocation = {}
remaining_units = self.total_units

for i, score in enumerate(scores):
    if total_score > 0:
        units = int(self.total_units * (score / total_score) * random.uniform(0.9, 1.1))
    else:
        units = 0
    units = min(units, remaining_units)
    allocation[i] = units
    remaining_units -= units
return allocation

```

Part 3: Market Share

```

class MarketShare:
    def __init__(self, companies):
        self.companies = companies

    def show(self):
        total_units = sum(c.units_sold for c in self.companies)
        print("\n--- Market Share ---")
        for c in self.companies:
            share = (c.units_sold / total_units * 100) if total_units > 0 else 0
            print(f"{c.name}: {share:.1f}%")

```

Part 4: Single-Round Simulation

```

class MarketSimulation:
    def __init__(self, market, companies):
        self.market = market
        self.companies = companies

    def run(self):
        print("\n--- Market Simulation (Single Round) ---")
        allocation = self.market.calculate_demand(self.companies)

        # Companies receive their allocated units and calculate profit
        for i, company in enumerate(self.companies):
            company.receive_demand(allocation[i])
            print(company)

        # Determine best-performing company
        winner = max(self.companies, key=lambda c: c.profit)
        print(f"\n Best-performing company: {winner.name} with profit {winner.profit:.1f}")

```

Part 5: Example Setup

```

# Create companies with intrinsic strategies
c1 = Company("SodaKing", production_cost=2, efficiency=8, price=5, advertising=5)
c2 = Company("FizzCo", production_cost=1.8, efficiency=6, price=6, advertising=4)
c3 = Company("BubblyInc", production_cost=2.2, efficiency=7, price=5.5, advertising=6)

```

```
# Create market
market = Market(total_units=100)

# Run single-round simulation
simulation = MarketSimulation(market, [c1, c2, c3])
simulation.run()

# Show market share
market_share = MarketShare([c1, c2, c3])
market_share.show()
```

Part 6: Lab Exercises

1. Record **units sold, revenue, cost, and profit** for each company.
2. Discuss how **price and advertising strategy** affected demand.
3. Experiment with different **strategies for each company** and compare results.
4. Analyze **market share** and identify the most effective strategy.
5. Run multiple simulations to see how **market randomness** affects outcomes.