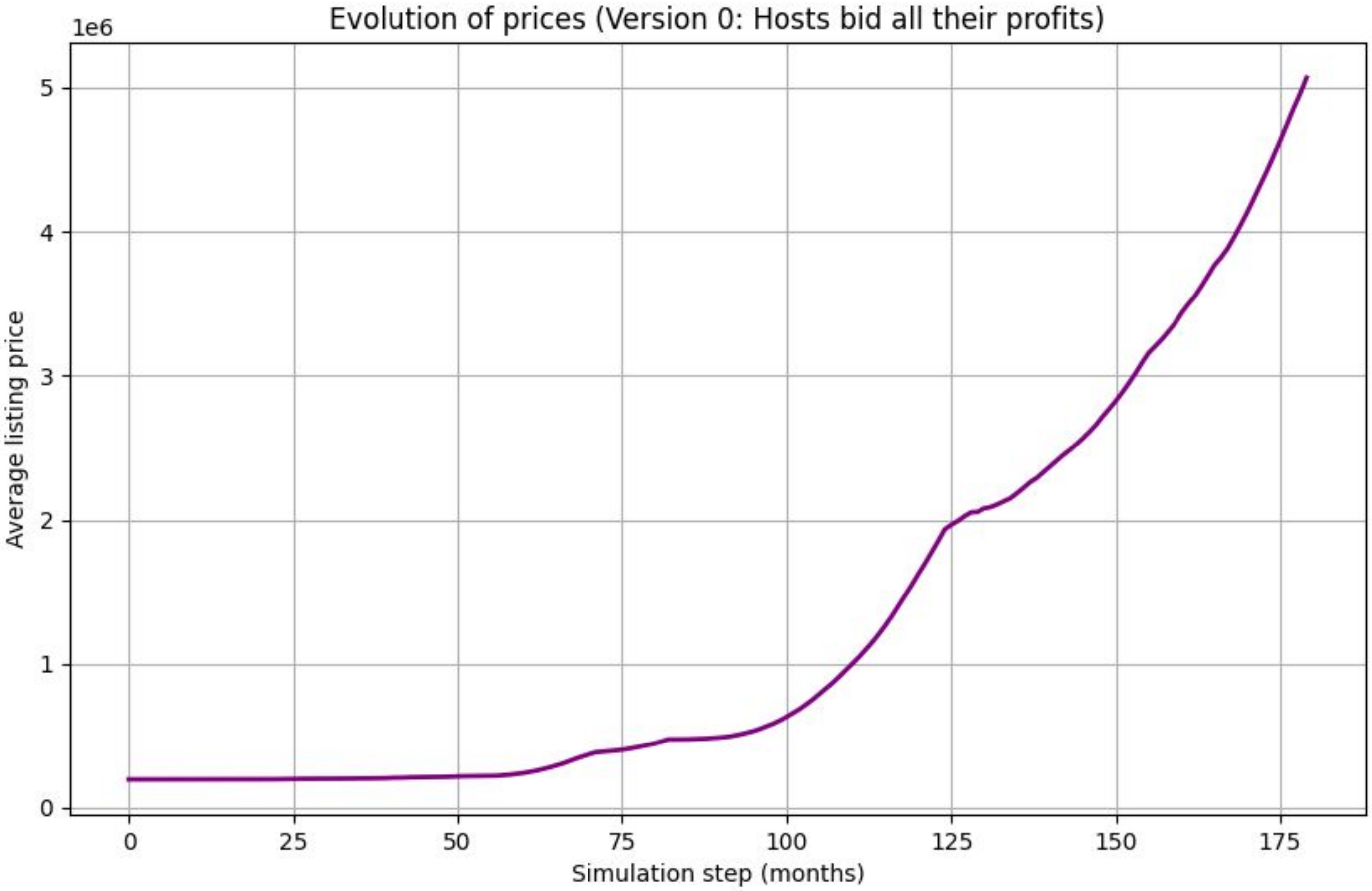
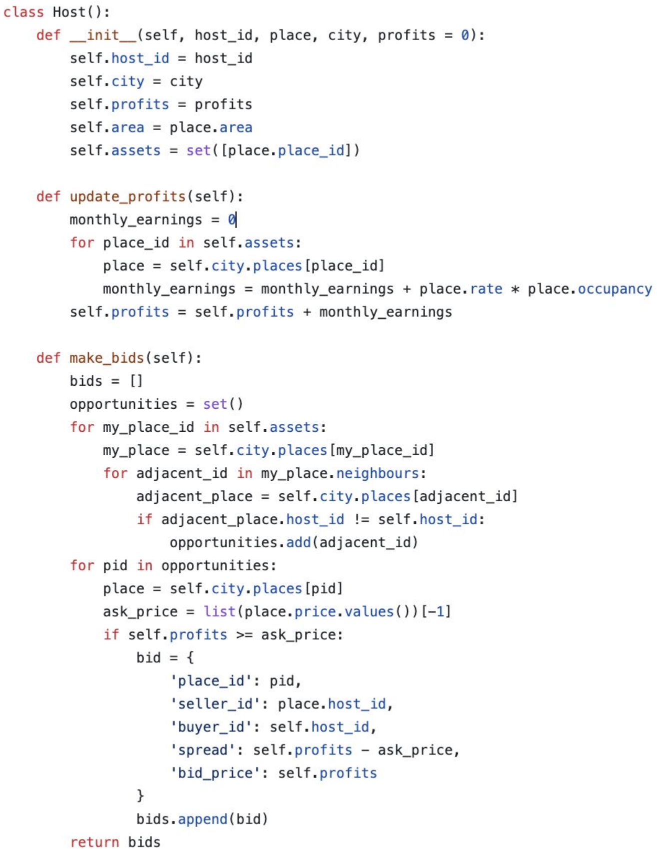
Part 1:
Algorithmic Simulation



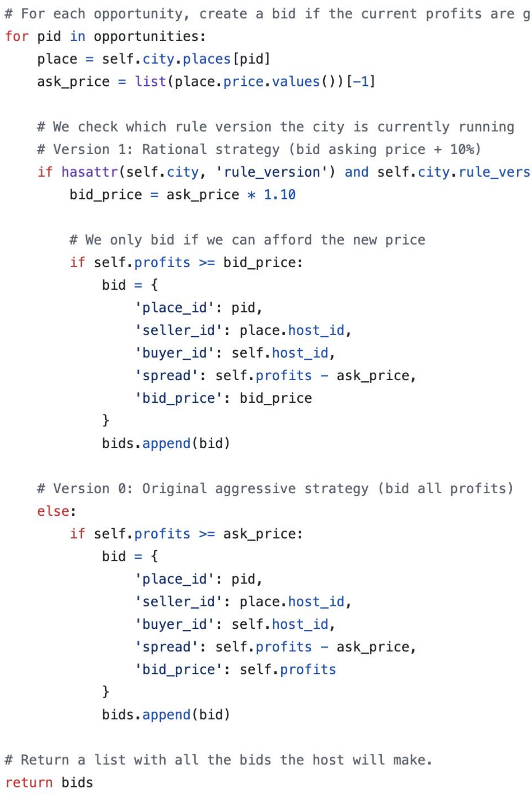
Rule Change Snippet

Version 0 (original)

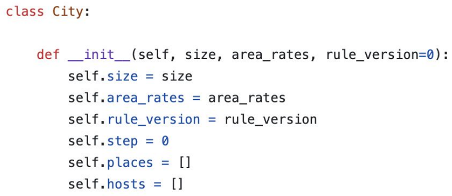


```
class Host():
    def __init__(self, host_id, place, city, profits = 0):
        self.host_id = host_id
        self.city = city
        self.profits = profits
        self.area = place.area
        self.assets = set([place.place_id])

    def update_profits(self):
        monthly_earnings = 0
        for place_id in self.assets:
            place = self.city.places[place_id]
            monthly_earnings = monthly_earnings + place.rate * place.occupancy
        self.profits = self.profits + monthly_earnings

    def make_bids(self):
        bids = []
        opportunities = set()
        for my_place_id in self.assets:
            my_place = self.city.places[my_place_id]
            for adjacent_id in my_place.neighbours:
                adjacent_place = self.city.places[adjacent_id]
                if adjacent_place.host_id != self.host_id:
                    opportunities.add(adjacent_id)
        for pid in opportunities:
            place = self.city.places[pid]
            ask_price = list(place.price.values())[-1]
            if self.profits >= ask_price:
                bid = {
                    'place_id': pid,
                    'seller_id': place.host_id,
                    'buyer_id': self.host_id,
                    'spread': self.profits - ask_price,
                    'bid_price': self.profits
                }
                bids.append(bid)
        return bids
```

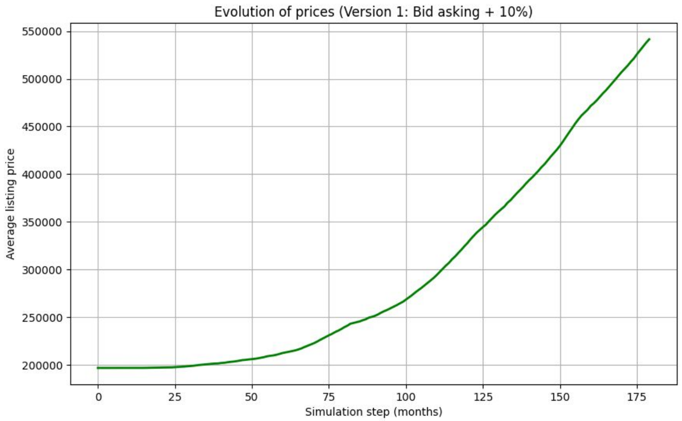
Version 1 (modified)



```
# For each opportunity, create a bid if the current profits are greater than the ask price.
for pid in opportunities:
    place = self.city.places[pid]
    ask_price = list(place.price.values())[-1]

    # We check which rule version the city is currently running
    # Version 1: Rational strategy (bid asking price + 10%)
    if hasattr(self.city, 'rule_version') and self.city.rule_version == 1:
        bid_price = ask_price * 1.10

        # We only bid if we can afford the new price
        if self.profits >= bid_price:
            bid = {
                'place_id': pid,
                'seller_id': place.host_id,
                'buyer_id': self.host_id,
                'spread': self.profits - ask_price,
                'bid_price': bid_price
            }
            bids.append(bid)

    # Version 0: Original aggressive strategy (bid all profits)
    else:
        if self.profits >= ask_price:
            bid = {
                'place_id': pid,
                'seller_id': place.host_id,
                'buyer_id': self.host_id,
                'spread': self.profits - ask_price,
                'bid_price': self.profits
            }
            bids.append(bid)

# Return a list with all the bids the host will make.
return bids
```

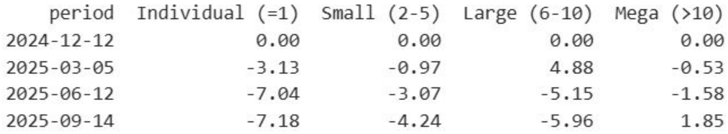


```
class City:

    def __init__(self, size, area_rates, rule_version=0):
        self.size = size
        self.area_rates = area_rates
        self.rule_version = rule_version
        self.step = 0
        self.places = []
        self.hosts = []
```

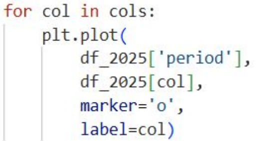


Part 2:
Data Driven Analysis

Graph creation:



```
      period  Individual (=1)  Small (2-5)  Large (6-10)  Mega (>10)
2024-12-12             0.00         0.00          0.00        0.00
2025-03-05            -3.13        -0.97          4.88       -0.53
2025-06-12            -7.04        -3.07         -5.15       -1.58
2025-09-14            -7.18        -4.24         -5.96        1.85
```

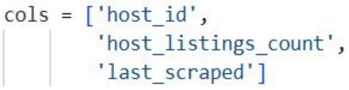


```
for col in cols:
    plt.plot(
        df_2025['period'],
        df_2025[col],
        marker='o',
        label=col)
```

Motivation:

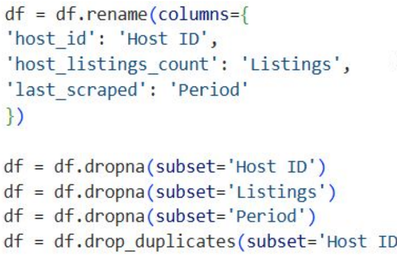
Who does regulation benefit?

Variables selection:

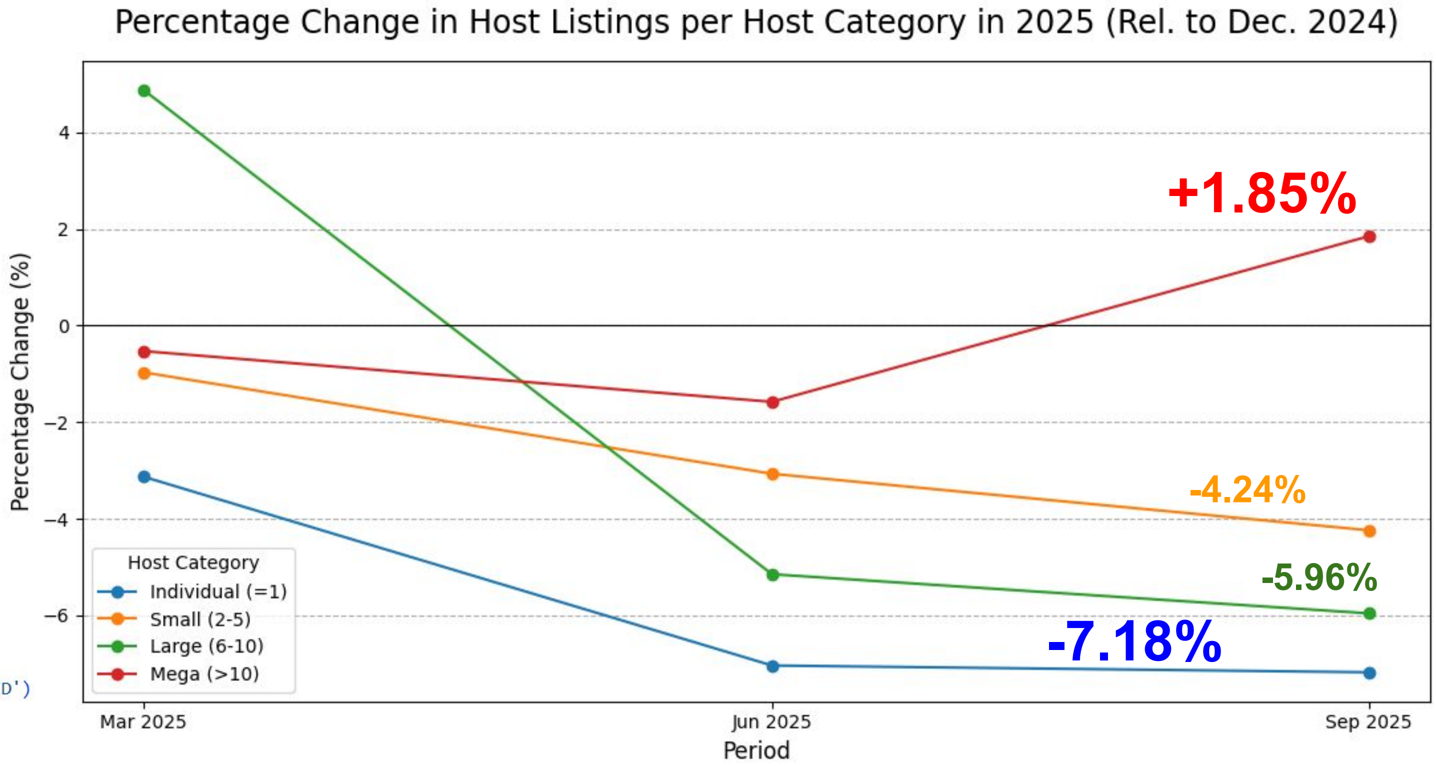


```
cols = ['host_id',
        'host_listings_count',
        'last_scraped']
```

Data cleaning:



```
df = df.rename(columns={
'host_id': 'Host ID',
'host_listings_count': 'Listings',
'last_scraped': 'Period'
})

df = df.dropna(subset='Host ID')
df = df.dropna(subset='Listings')
df = df.dropna(subset='Period')
df = df.drop_duplicates(subset='Host ID')
```



Percentage Change in Host Listings per Host Category in 2025 (Rel. to Dec. 2024)