```
import numpy as np

temps = np.random.uniform(20, 40, (7, 24))  # 7 days x 24 hours
avg_per_day = temps.mean(axis=1)
hottest_day = np.argmax(avg_per_day)

print(f"Hottest day: Day {hottest_day + 1}, Avg Temp: {avg_per_day[hottest_day]:.2f}")
```

```
Hottest day: Day 4, Avg Temp: 31.71
```

```
sales = np.random.randint(100, 1000, (6, 5))  # 6 months x 5 products
total_sales = sales.sum(axis=0)
top_product = np.argmax(total_sales)

print(f"Top selling product: Product {top_product + 1} with {total_sales[top_product]} units.")
```

```
Top selling product: Product 1 with 3916 units.
```

```
visits = np.random.randint(50, 300, 30)  # 30 days
least_visits_day = np.argmin(visits)

print(f"Day {least_visits_day + 1} had the fewest visits: {visits[least_visits_day]}")
```

```
Day 8 had the fewest visits: 72
```

```
marks = np.random.randint(30, 100, 8)
highest = np.argmax(marks)
lowest = np.argmin(marks)

print(f"Highest: Subject {highest + 1}, Marks: {marks[highest]}")
print(f"Lowest: Subject {lowest + 1}, Marks: {marks[lowest]}")
```

```
Highest: Subject 5, Marks: 85
Lowest: Subject 7, Marks: 39
```

```
likes = np.random.randint(100, 1000, 28)  # 28 days
likes_by_week = likes.reshape(4, 7)
avg_likes = likes_by_week.mean(axis=1)

print("Average likes per week:", avg_likes)
```

```
Average likes per week: [590.71428571 554.42857143 646.42857143 531.14285714]
```

```
salaries = np.array([40000, 50000, 60000, 45000])
new_salaries = salaries * 1.10

print("Updated salaries:", new_salaries)
```

```
Updated salaries: [44000. 55000. 66000. 49500.]
```

```
rainfall = np.random.uniform(0, 10, (30, 24))  # 30 days x 24 hours
daily_rain = rainfall.sum(axis=1)
wettest_day = np.argmax(daily_rain)

print(f"Day {wettest_day + 1} had the most rain: {daily_rain[wettest_day]:.2f} mm")
```

```
Day 20 had the most rain: 155.50 mm
```

```
team1 = np.array([56, 78, 45, 89, 90])
team2 = np.array([67, 88, 70, 91, 85])
combined = np.stack((team1, team2), axis=1)

print("Combined scores:\n", combined)
```

```
Combined scores:
 [[56 67]
 [78 88]
 [45 70]
 [89 91]
 [90 85]]
```

```
scores = np.random.randint(50, 100, (5, 3))  # 5 students x 3 tests
avg_scores = scores.mean(axis=1)
top_student = np.argmax(avg_scores)

print(f"Top student: Student {top_student + 1} with average {avg_scores[top_student]:.2f}")
```

```
Top student: Student 1 with average 84.67
```

```
usage = np.random.rand(96 * 7)  # 96 readings/day * 7 days
daily_usage = usage.reshape(7, 96)

print("Daily usage shape:", daily_usage.shape)
```

```
Daily usage shape: (7, 96)
```

```
temps = np.random.uniform(30, 90, (7, 24))
daily_max = temps.max(axis=1)

print("Max temp per day:", daily_max)
```

```
Max temp per day: [89.77506948 89.13682788 87.80047635 88.88499868 87.29245409 89.90977912
 87.97065012]
```

```
ids = np.array([101, 102, 103, 102, 104, 101])
unique_ids = np.unique(ids)

print("Unique customer IDs:", unique_ids)
```

```
Unique customer IDs: [101 102 103 104]
```

```
weights = np.random.randint(50, 150, (7, 5))  # 7 days x 5 members
improvements = weights[-1] - weights[0]

print("Improvements over the week:", improvements)
```

```
Improvements over the week: [ 78 -22  49 -58 -68]
```

```
transactions = np.random.randint(1000, 20000, 100)
count = np.sum(transactions > 10000)

print("Transactions over $10,000:", count)
```

```
Transactions over $10,000: 49
```

```
ratings = np.random.randint(1, 6, 200)
five_star_count = np.sum(ratings == 5)

print("Number of 5-star ratings:", five_star_count)
```

```
Number of 5-star ratings: 46
```

```
durations = np.random.randint(30, 600, 15)  # seconds
shortest = durations.min()
longest = durations.max()

print(f"Shortest: {shortest}s, Longest: {longest}s")
```

```
Shortest: 71s, Longest: 521s
```

```
attendance = np.random.randint(0, 2, 180)  # 180 days
percentage = np.mean(attendance) * 100

print(f"Attendance Percentage: {percentage:.2f}%")
```

```
Attendance Percentage: 52.78%
```

```
visits = np.random.randint(50, 300, 60)
weekly = visits.reshape(12, 5)
busiest_week = np.argmax(weekly.sum(axis=1))

print(f"Busiest week: Week {busiest_week + 1}")
```

```
Busiest week: Week 4
```

```
times = np.random.randint(5, 90, 200)
fastest = times.min()
slowest = times.max()
average = times.mean()

print(f"Fastest: {fastest} min, Slowest: {slowest} min, Avg: {average:.2f} min")
```

```
Fastest: 5 min, Slowest: 89 min, Avg: 46.62 min
```

```
logins = np.random.randint(10, 300, 50)
top5 = np.argsort(logins)[-5:][::-1]

print("Top 5 active students (indices):", top5)
```

```
Top 5 active students (indices): [42 40 23 48 21]
```

```
calls = np.random.randint(60, 1800, 100)  # in seconds
avg_duration = calls.mean()

print(f"Average call duration: {avg_duration:.2f} seconds")
```

```
Average call duration: 959.85 seconds
```

```
sales = np.random.randint(10, 500, 10)
best = np.argmax(sales)
worst = np.argmin(sales)

print(f"Best: Book {best + 1}, Worst: Book {worst + 1}")
```

```
Best: Book 2, Worst: Book 7
```

```
consumption = np.random.uniform(5, 20, 10)  # liters per 100km
most_efficient = np.argmin(consumption)

print(f"Most fuel efficient: Vehicle {most_efficient + 1} ({consumption[most_efficient]:.2f} L/100km)")
```

```
Most fuel efficient: Vehicle 9 (5.69 L/100km)
```

```
stocks = np.random.randint(100, 1000, 15)
most = np.argmax(stocks)
least = np.argmin(stocks)

print(f"Most stock: Section {most + 1}, Least stock: Section {least + 1}")
```

```
Most stock: Section 13, Least stock: Section 15
```

```
grades = np.random.uniform(2.0, 4.0, (100, 4))
avg_per_sem = grades.mean(axis=0)

print("Average GPA per semester:", avg_per_sem)
```

```
Average GPA per semester: [2.95483388 2.97747459 3.06734931 3.02701489]
```

```
foot_traffic = np.random.randint(0, 100, (30, 24))
daily_peak = foot_traffic.max(axis=1)

print("Peak visits per day:", daily_peak)
```

```
Peak visits per day: [97 98 98 98 99 97 95 97 92 99 99 93 99 99 89 98 99 93 99 98 98 99 97 99
 85 91 98 94 94 99]
```

```
ph_levels = np.random.uniform(4.0, 9.0, 20)
most_acidic = np.argmin(ph_levels)
most_alkaline = np.argmax(ph_levels)

print(f"Most acidic region: {most_acidic + 1}, pH: {ph_levels[most_acidic]:.2f}")
print(f"Most alkaline region: {most_alkaline + 1}, pH: {ph_levels[most_alkaline]:.2f}")
```

```
Most acidic region: 13, pH: 4.15
Most alkaline region: 20, pH: 8.86
```

```
heart_rates = np.random.randint(60, 180, 600)
summary = {
    "min": heart_rates.min(),
    "max": heart_rates.max(),
    "avg": heart_rates.mean()
}
print("Heart rate summary over 10 minutes:", summary)
```

```
Heart rate summary over 10 minutes: {'min': np.int64(60), 'max': np.int64(179), 'avg': np.float64(119.51833333333333)}
```

```
delays = np.random.randint(0, 60, 100)
delayed = delays[delays > 0]
avg_delay = delayed.mean()

print(f"Average delay (excluding on-time): {avg_delay:.2f} mins")
```

```
Average delay (excluding on-time): 32.45 mins
```

```
trips = np.random.randint(10, 100, (7, 20))  # 7 days x 20 taxis
total_trips = trips.sum(axis=0)
most_active = np.argmax(total_trips)

print(f"Most active taxi: Taxi {most_active + 1} with {total_trips[most_active]} trips")
```

```
Most active taxi: Taxi 18 with 453 trips
```