AIM: The aim of this case study is to identify the most effective strategy for winning a simple card-based game by simulating different player strategies using Python.

ALGORITHM: 1. Start the simulation and initialize scores for each strategy. 2. Create a deck of card values (1–13). 3. Shuffle the deck. 4. Draw a random card for each strategy. 5. Decide whether to keep or replace based on strategy rules. 6. Compare final card values. 7. Award point to winning strategy. 8. Repeat for N rounds. 9. Display total wins and identify the best strategy.

PYTHON CODE:

```
import random

def random_strategy(card):
    return card if random.random() > 0.5 else random.randint(1, 13)

def greedy_strategy(card):
    return card if card >= 8 else random.randint(1, 13)

def smart_strategy(card):
    if card >= 10:
        return card
    elif card <= 4:
        return random.randint(5, 13)
    else:
        return random.randint(1, 13)

rounds = 1000
scores = {"Random": 0, "Greedy": 0, "Smart": 0}

for _ in range(rounds):
    random_card = random_strategy(random.randint(1, 13))
    greedy_card = greedy_strategy(random.randint(1, 13))
    smart_card = smart_strategy(random.randint(1, 13))

    max_card = max(random_card, greedy_card, smart_card)

    if random_card == max_card:
        scores["Random"] += 1
    if greedy_card == max_card:
        scores["Greedy"] += 1
    if smart_card == max_card:
        scores["Smart"] += 1

print(scores)
```