TOKEN BUCKET :-

```
import time

class TokenBucket:
    def __init__(self, rate, capacity):
        self.rate = rate            # tokens added per second
        self.capacity = capacity    # maximum number of tokens
        self.tokens = capacity      # start full
        self.last_checked = time.monotonic()

        self.allowed = 0            # total allowed requests
        self.denied = 0             # total denied requests

    def _add_tokens(self):
        now = time.monotonic()
        elapsed = now - self.last_checked
        added = elapsed * self.rate
        self.tokens = min(self.capacity, self.tokens + added)
        self.last_checked = now

    def consume(self, tokens=1):
        self._add_tokens()
        if self.tokens >= tokens:
            self.tokens -= tokens
            self.allowed += 1
            return True
        else:
            self.denied += 1
```

```
            return False


if __name__ == "__main__":
    rate = float(input("Enter token generation rate (tokens per second): "))
    capacity = float(input("Enter bucket capacity (max tokens): "))
    num_requests = int(input("Enter number of requests to simulate: "))
    interval = float(input("Enter interval between requests (seconds): "))

    bucket = TokenBucket(rate, capacity)

    for i in range(num_requests):
        if bucket.consume():
            print(f"Request {i+1} allowed (tokens left: {bucket.tokens:.2f})")
        else:
            print(f"Request {i+1} denied (tokens left: {bucket.tokens:.2f})")
        time.sleep(interval)

    print("\nSimulation Summary:")
    print(f"Total allowed requests: {bucket.allowed}")
    print(f"Total denied requests: {bucket.denied}")
    print(f"Tokens remaining in bucket: {bucket.tokens:.2f}")
```