# Beamer Feature Showcase

Reference for LLM-Generated Video Slides

System Reference

January 11, 2026

This presentation demonstrates features for automated video generation.

## Incremental Reveals

To keep the video engaging, reveal content step-by-step using <+->.

- **Step 1**: Introduce the concept.

## Incremental Reveals

To keep the video engaging, reveal content step-by-step using <+->.

- **Step 1**: Introduce the concept.
- **Step 2**: Expand on details.

## Incremental Reveals

To keep the video engaging, reveal content step-by-step using <+->.

- **Step 1**: Introduce the concept.
- **Step 2**: Expand on details.
- **Step 3**: Conclude the point.

## Incremental Reveals

To keep the video engaging, reveal content step-by-step using <+->.

- **Step 1**: Introduce the concept.
- **Step 2**: Expand on details.
- **Step 3**: Conclude the point.

**Key Takeaway**

Incremental reveals prevent cognitive overload and sync with the narration.

## Code Evolution

We can simulate "Magic Move" by showing code changes across overlays.

```python
def calculate_area(radius):
    pi = 3.14
    return pi * radius * radius

```

Initial State

## Code Evolution

We can simulate "Magic Move" by showing code changes across overlays.

```python
import math

def calculate_area(radius):
    return math.pi * radius * radius

```

Refactored (Import Math)

## Code Evolution

We can simulate "Magic Move" by showing code changes across overlays.

```python
import math

def calculate_area(radius: float) -> float:
    return math.pi * radius ** 2

```

Type Hinting Added

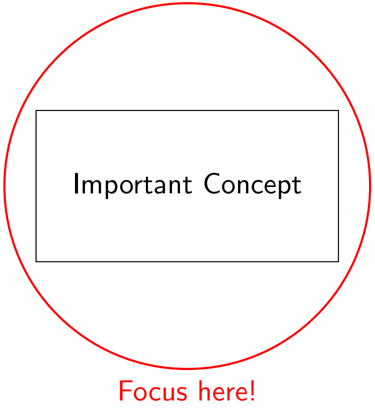
## Visual Annotations

Use TikZ to draw attention to specific elements.

Important Concept

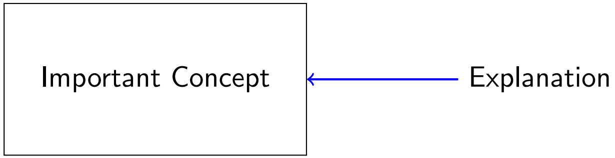
## Visual Annotations

Use TikZ to draw attention to specific elements.



## Visual Annotations

Use TikZ to draw attention to specific elements.



## Comparing Approaches

Use columns to compare side-by-side.

**Approach A (Recursive)**

```python
def fib(n):
    if n <= 1: return n
    return fib(n-1) + fib(n-2)

```

**Approach B (Iterative)**

```python
def fib(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a

```

## Comparing Approaches

Use columns to compare side-by-side.

**Approach A (Recursive)**

```python
def fib(n):
    if n <= 1: return n
    return fib(n-1) + fib(n-2)

```

**Approach B (Iterative)**

```python
def fib(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a

```

**Verdict:** Approach B is much faster.