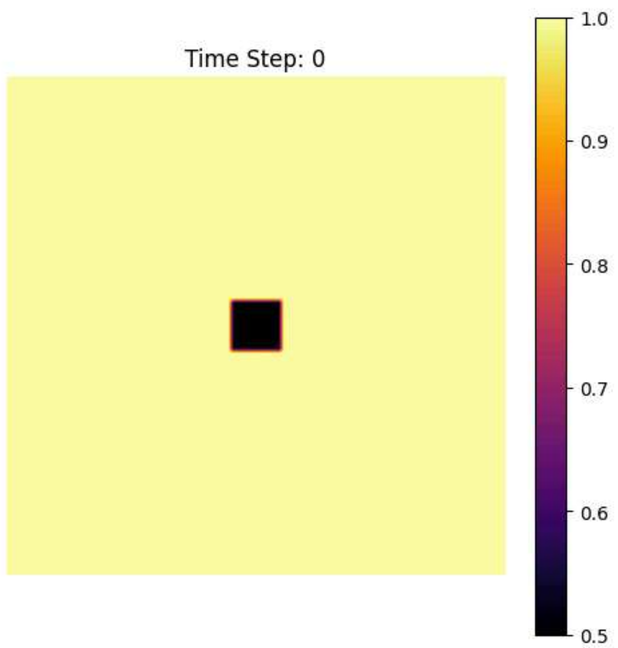
```
import numpy as np
import matplotlib.pyplot as plt
from ipywidgets import interact, IntSlider, Play, VBox, HBox
from google.colab import output
output.enable_custom_widget_manager()
from IPython.display import display

# Grid size
N = 100
r = 5  # radius of the square region

# Initialize arrays
u = np.ones((N, N))
v = np.zeros((N, N))

# Add a small square with different values in the center
cx, cy = N // 2, N // 2
u[cx - r:cx + r, cy - r:cy + r] = 0.50
v[cx - r:cx + r, cy - r:cy + r] = 0.25

# Constants (change as needed)
Du, Dv = 0.16, 0.08  # diffusion coefficients
F, k = 0.0545, 0.062  # feed & kill rates
dt = 1.0

def laplacian(Z):
    return (
        -4 * Z
        + np.roll(Z, (0, -1), (0, 1))
        + np.roll(Z, (0, 1), (0, 1))
        + np.roll(Z, (-1, 0), (0, 1))
        + np.roll(Z, (1, 0), (0, 1))
    )

# Simulate and store frames
steps = 10000
u_list = [u.copy()]
v_list = [v.copy()]
u_temp, v_temp = u.copy(), v.copy()

for i in range(steps):
    Lu = laplacian(u_temp)
    Lv = laplacian(v_temp)
    uvv = u_temp * v_temp**2
    u_temp += (Du * Lu - uvv + F * (1 - u_temp)) * dt
    v_temp += (Dv * Lv + uvv - (F + k) * v_temp) * dt
    u_temp = np.clip(u_temp, 0, 1)  # Keep values bounded
    v_temp = np.clip(v_temp, 0, 1)
    u_list.append(u_temp.copy())
    v_list.append(v_temp.copy())

# Plot function (activator or inhibitor)
def plot_at_time(t):
    plt.figure(figsize=(6, 6))
    img = plt.imshow(u_list[t], cmap='inferno', interpolation='bilinear')  # Change to v_list[t] if needed
    plt.colorbar(img, ax=plt.gca())
    plt.title(f"Time Step: {t}")
    plt.axis('off')
    plt.show()

# Slider and play widget for 10000 frames
play = Play(value=0, min=0, max=len(u_list) - 1, step=1, interval=100, description="Play")
slider = IntSlider(value=0, min=0, max=len(u_list) - 1, step=1, description='Time')
widgets = HBox([play, slider])
interact_ui = interact(plot_at_time, t=slider)

# Link play and slider
from ipywidgets import jslink
jslink((play, 'value'), (slider, 'value'))

# Display everything
display(widgets, interact_ui)
```

Time 0



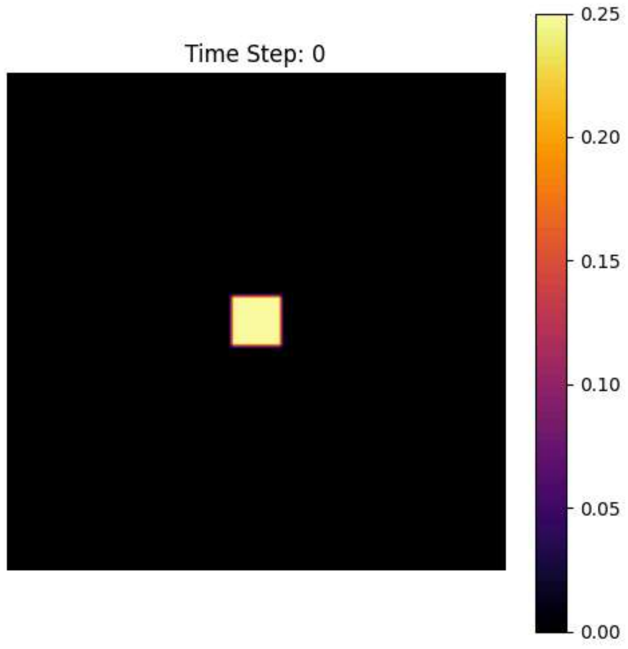
Time 0

```
plot_at_time
def plot_at_time(t)
```

```
import numpy as np
import matplotlib.pyplot as plt
from ipywidgets import interact, IntSlider, Play, VBox, HBox
from google.colab import output
output.enable_custom_widget_manager()
from IPython.display import display

# Grid size
N = 100
r = 5  # radius of the square region

# Initialize arrays
u = np.ones((N, N))
v = np.zeros((N, N))

# Add a small square with different values in the center
cx, cy = N // 2, N // 2
u[cx - r:cx + r, cy - r:cy + r] = 0.50
v[cx - r:cx + r, cy - r:cy + r] = 0.25

# Constants (change as needed)
Du, Dv = 0.16, 0.08  # diffusion coefficients
F, k = 0.0545, 0.062  # feed & kill rates
dt = 1.0

def laplacian(Z):
    return (
        -4 * Z
        + np.roll(Z, (0, -1), (0, 1))
        + np.roll(Z, (0, 1), (0, 1))
        + np.roll(Z, (-1, 0), (0, 1))
        + np.roll(Z, (1, 0), (0, 1))
    )

# Simulate and store frames
steps = 10000
u_list = [u.copy()]
v_list = [v.copy()]
u_temp, v_temp = u.copy(), v.copy()

for i in range(steps):
    Lu = laplacian(u_temp)
    Lv = laplacian(v_temp)
    uvv = u_temp * v_temp**2
```

```
        u_temp += (Du * Lu - uvv + F * (1 - u_temp)) * dt
        v_temp += (Dv * Lv + uvv - (F + k) * v_temp) * dt
        u_temp = np.clip(u_temp, 0, 1)  # Keep values bounded
        v_temp = np.clip(v_temp, 0, 1)
        u_list.append(u_temp.copy())
        v_list.append(v_temp.copy())

    # Plot function (activator or inhibitor)
    def plot_at_time(t):
        plt.figure(figsize=(6, 6))
        img = plt.imshow(v_list[t], cmap='inferno', interpolation='bilinear')  # Change to v_list[t] if needed
        plt.colorbar(img, ax=plt.gca())
        plt.title(f"Time Step: {t}")
        plt.axis('off')
        plt.show()

    # Slider and play widget for 10000 frames
    play = Play(value=0, min=0, max=len(u_list) - 1, step=1, interval=100, description="Play")
    slider = IntSlider(value=0, min=0, max=len(u_list) - 1, step=1, description='Time')
    widgets = HBox([play, slider])
    interact_ui = interact(plot_at_time, t=slider)

    # Link play and slider
    from ipywidgets import jslink
    jslink((play, 'value'), (slider, 'value'))

    # Display everything
    display(widgets, interact_ui)
```

Time 0



Time 0

```
plot_at_time
def plot_at_time(t)

<no docstring>
```

```
    import numpy as np
    import matplotlib.pyplot as plt
    from ipywidgets import interact, IntSlider, Play, HBox, VBox, Dropdown, ToggleButtons, jslink
    from google.colab import output
    from IPython.display import display

    output.enable_custom_widget_manager()

    # Grid size
    N = 100
    r = 5  # radius of the square region
    steps = 10000  # reduce steps for faster simulation in Colab
```

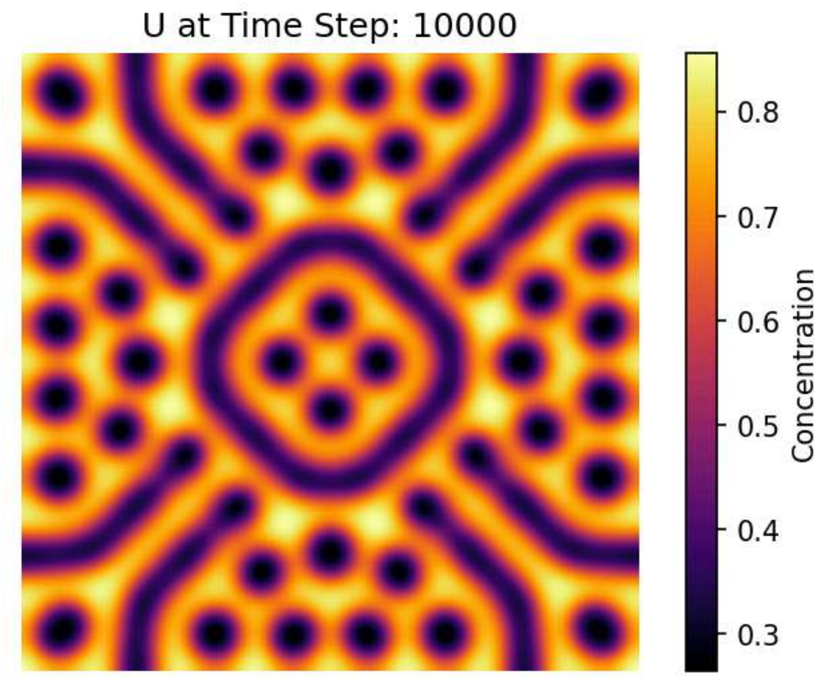
```
# Initialize arrays
def init_grid():
    u = np.ones((N, N))
    v = np.zeros((N, N))
    cx, cy = N // 2, N // 2
    u[cx - r:cx + r, cy - r:cy + r] = 0.50
    v[cx - r:cx + r, cy - r:cy + r] = 0.25
    return u, v

u_temp, v_temp = init_grid()
u_list = [u_temp.copy()]
v_list = [v_temp.copy()]

# Constants
Du, Dv = 0.16, 0.08
F, k = 0.0545, 0.062
dt = 1.0

def laplacian(Z):
    return (
        -4 * Z
        + np.roll(Z, (0, -1), (0, 1))
        + np.roll(Z, (0, 1), (0, 1))
        + np.roll(Z, (-1, 0), (0, 1))
        + np.roll(Z, (1, 0), (0, 1))
    )

# Simulation function
def simulate():
    global u_temp, v_temp, u_list, v_list
    u_temp, v_temp = init_grid()
    u_list = [u_temp.copy()]
    v_list = [v_temp.copy()]
    for i in range(steps):
        Lu = laplacian(u_temp)
        Lv = laplacian(v_temp)
        uvv = u_temp * v_temp**2
        u_temp += (Du * Lu - uvv + F * (1 - u_temp)) * dt
        v_temp += (Dv * Lv + uvv - (F + k) * v_temp) * dt
        u_temp = np.clip(u_temp, 0, 1)
        v_temp = np.clip(v_temp, 0, 1)
        u_list.append(u_temp.copy())
        v_list.append(v_temp.copy())

# Initial simulation
simulate()

# Plot function with species toggle
def plot_at_time_species(t, species):
    plt.figure(figsize=(6, 4), dpi=150)
    if species == 'u':
        img = plt.imshow(u_list[t], cmap='inferno', interpolation='bilinear')
    else:
        img = plt.imshow(v_list[t], cmap='inferno', interpolation='bilinear')
    plt.colorbar(img, ax=plt.gca(), label="Concentration")
    plt.title(f"{species.upper()} at Time Step: {t}")
    plt.axis('off')
    plt.show()

# Widgets
play = Play(value=0, min=0, max=len(u_list) - 1, step=1, interval=100)
slider = IntSlider(value=0, min=0, max=len(u_list) - 1, step=1, description='Time')
jslink((play, 'value'), (slider, 'value'))

species_toggle = ToggleButtons(
    options=[('Activator (u)', 'u'), ('Inhibitor (v)', 'v')],
    value='u',
    description='Species:',
    style={'description_width': 'initial'}
)

# Dropdown for famous patterns
patterns = {
    "Labyrinths (F=0.0545, k=0.062)": (0.0545, 0.062),
    "Maze (F=0.038, k=0.059)": (0.038, 0.059),
    "Spots (F=0.03, k=0.06)": (0.03, 0.06),
    "Chaos (F=0.02, k=0.05)": (0.02, 0.05),
    "Stable (F=0.06, k=0.062)": (0.06, 0.062)
}

pattern_selector = Dropdown(
    options=patterns,
    value=(0.0545, 0.062),
    description='Patterns:',
    style={'description_width': 'initial'}
)

def update_params(selection):
    global F, k
    F, k = selection
    simulate()  # rerun simulation with new parameters

# Connect dropdown to simulation update
interact(update_params, selection=pattern_selector)

# Connect slider and toggle to plot
interact(plot_at_time_species, t=slider, species=species_toggle)

# Display widgets
display(HBox([play, slider]), species_toggle, pattern_selector)
```

Patterns: Spots (F=0.03, k=0.06)

Time 10000

Species:

Activator (u) Inhibitor (v)



Time 10000

Species:

Activator (u) Inhibitor (v)

Patterns: Spots (F=0.03, k=0.06)