Python Movie Credits Animation

Learning Objective

By the end of the lesson, students will be able to create a beginner-level Python program that:

- Stores movie credits in a list
- Uses a `for` loop to process each item in the list
- Displays text using the `print()` function
- Creates pauses using `time.sleep()`
- Uses correct indentation, quotation marks, parentheses, and colons

Students will also practice reading error messages, correcting mistakes, and explaining what their code does.

> **Encouraging message:** Programming is like learning a new language. You do not need to understand everything immediately. Start small, practice, ask questions, and build from there. Mistakes are a normal and valuable part of learning to code.

Assessments

Formative Assessments

Use these quick checks throughout the lesson:

- Ask students to predict what a short code example will display.
- Have students identify which lines are lists, loops, and output statements.
- Ask students to explain why a colon and indentation are needed.
- Conduct a compilation check after each guided-practice task.
- Use a quick self-check:
    - **Green:** My program works.
    - **Yellow:** I have an error but know where to look.
    - **Red:** I need help identifying the problem.
- Ask students to explain one line of their program to a partner.

Summative Assessment

Students will submit a working Python program that:

- Includes at least five movie credits
- Stores the credits in a list
- Uses a `for` loop to visit each credit
- Prints each credit using `print()`
- Includes `import time`
- Uses `time.sleep()` between credits
- Runs without crashing
- Includes a comment describing the program

Student Reflection

Students respond to the following prompt:

> What was one part of programming that felt easier than you expected, and what was one part that was challenging?

Simple Rubric

Criteria	Beginning	Developing	Meets Expectations
List	List is missing or contains major errors	List is present but needs corrections	List contains at least five credits
Output	Output is missing or incorrect	Some credits print correctly	Credits print correctly
Loop	Loop is missing or does not work	Loop works with assistance	Correctly uses a `for` loop
Timing	Delay is missing or incorrect	Delay works with assistance	Correctly uses `time.sleep()`
Code quality	Several syntax errors remain	Minor errors remain	Program runs and includes a helpful comment

Key Points

Lists Store Multiple Items

A list stores several values in one place. In this lesson, the list will store movie credits.

```
credits = ["Director: Jordan Lee", "主演: Maya Chen", "Music: Alex Rivera"]
```

Python lists use:

- Square brackets
- Commas between items

- Quotation marks around text

A shorter example is:

```
names = ["Ava", "Mateo", "Sam"]
```

Python does not require students to declare the data type separately. This makes the syntax more concise and beginner-friendly.

The `print()` Function Displays Information

The `print()` function displays text on the screen.

```
print("Welcome to the movie!")
```

It can also display a variable:

```
title = "The Great Adventure"
print(title)
```

A `for` Loop Repeats an Action

A `for` loop performs an action for every item in a list.

```
for name in names:
    print(name)
```

Important details:

- `for` begins the loop.
- `name` is a temporary variable.
- `in` tells Python where to get the items.
- A colon is required at the end of the line.
- The indented line belongs to the loop.

Python uses indentation to show which instructions belong together. This is one of the features that makes Python readable.

Importing `time`

Python includes many built-in modules. A module is a collection of tools that programmers can use.

To use the time tools, write:

```
import time
```

This line should appear before the program uses `time.sleep()`.

Creating a Delay

The following statement pauses the program:

```
time.sleep(1)
```

The number represents seconds. For example:

```
time.sleep(2)
```

pauses for two seconds.

Basic Program Structure

A simple Python program may include:

1. A comment explaining the program
2. An import statement
3. A list of information
4. A loop
5. Instructions inside the loop

Python is often approachable for beginners because it uses fewer symbols and less required punctuation than many other programming languages.

Opening: 10–15 Minutes

Hook: Observe the Movie Credits Demo

Show the JavaScript movie credits demo that inspired the lesson.

Ask students:

- What do you notice about the way the names appear?
- Does everything appear at once?
- What seems to happen between each name?
- What instructions might the computer be following?
- How might a computer know which name should appear next?

Do not expect students to know the programming vocabulary yet. Accept observations such as:

- “The names appear one at a time.”
- “There is a pause.”

- “The names are in an order.”
- “The program repeats something.”

Introduce the Lesson

Explain:

> We are going to recreate this effect using Python, and you will write your first computer program today.

Tell students that the completed program will:

1. Store several credits
2. Look at each credit in order
3. Display the credit
4. Pause briefly
5. Continue to the next credit

Address Programming Anxiety

Share the following message:

> Programming is like learning a new language. We will start with a few words and rules, practice them together, and gradually combine them. You are not expected to get everything right on the first attempt.

Establish these classroom expectations:

- Errors are information, not evidence of failure.
- Read the error message before asking for help.
- Change one thing at a time.
- Ask, “What did I expect to happen, and what actually happened?”
- Test code frequently instead of waiting until the end.

Introduction to New Material: 20–25 Minutes

Mini-Lesson 1: Creating a List

Begin with a familiar example:

> A movie credit list is like a list on paper. It has several items in a particular order.

Write:

```
credits = ["Director: Jordan Lee", "Actor: Maya Chen", "Music: Alex Rivera"]
```

Explain each part:

- `credits` is the list's name.
- `=` assigns information to that name.
- Square brackets contain the list.
- Each item is text.
- Text is surrounded by quotation marks.
- Commas separate the items.

Ask students to predict:

> How many credits are stored in this list?

Mini-Lesson 2: Printing Text

Demonstrate:

```
print("Hello, Python!")
```

Then demonstrate:

```
credits = ["Director: Jordan Lee", "Actor: Maya Chen", "Music: Alex Rivera"]

print(credits)
```

Explain that this displays the entire list. To display one individual item, Python can use a position number:

```
print(credits[0])
```

Explain that Python begins counting positions at zero. Avoid requiring students to memorize indexing during the first activity; the main goal is to introduce the idea that a list contains multiple items.

Mini-Lesson 3: Printing Each Item with a `for` Loop

Show the loop:

```
credits = ["Director: Jordan Lee", "Actor: Maya Chen", "Music: Alex Rivera"]

for credit in credits:
    print(credit)
```

Read the code in everyday language:

> For each credit in the credits list, print the credit.

Emphasize the structure:

```
for credit in credits:
    print(credit)
```

- The first line begins with `for`.
- `credit` is the temporary name for the current item.
- `in credits` means the loop looks through the list.
- The colon ends the loop instruction.
- The indented `print()` line is repeated.

Demonstrate what happens when indentation is removed:

```
for credit in credits:
print(credit)
```

Explain that Python uses indentation to identify the instructions inside the loop.

Mini-Lesson 4: Adding a Delay

First import the time module:

```
import time
```

Then add a delay:

```
time.sleep(1)
```

Combine the loop and delay:

```
import time

credits = ["Director: Jordan Lee", "Actor: Maya Chen", "Music: Alex Rivera"]

for credit in credits:
    print(credit)
    time.sleep(1)
```

Explain the program in plain language:

1. Python loads the time tools.
2. Python creates the credits list.
3. Python takes the first credit.
4. Python prints the credit.
5. Python pauses for one second.

6. Python repeats the process for the remaining credits.

Mini-Lesson 5: Live Coding Demonstration

Type the following program slowly. Narrate each decision.

```
# A simple movie credits program

import time

credits = [
    "A Classroom Production",
    "Director: Jordan Lee",
    "Actors: Maya Chen and Alex Rivera",
    "Music: Sam Patel",
    "Thank you for watching!"
]

for credit in credits:
    print(credit)
    time.sleep(1)
```

Run the program and let students observe the output.

Ask:

- What happened first?
- What happened repeatedly?
- Which line created the pause?
- Which line caused the credit to appear?
- What would happen if another credit were added to the list?

Common Beginner Misconceptions

Forgetting the Colon

Incorrect:

```
for credit in credits
    print(credit)
```

Correct:

```
for credit in credits:
    print(credit)
```

Explain that the colon tells Python that the instructions for the loop are about to begin.

Incorrect Indentation

Incorrect:

```
for credit in credits:
print(credit)
```

Correct:

```
for credit in credits:
    print(credit)
```

Use four spaces for the indented line. Most Python editors insert the indentation automatically after a colon.

Forgetting Quotation Marks

Incorrect:

```
credits = [Director, Actor, Music]
```

Correct:

```
credits = ["Director", "Actor", "Music"]
```

Explain that quotation marks tell Python that these are words or text.

Forgetting to Import `time`

Incorrect:

```
time.sleep(1)
```

Correct:

```
import time

time.sleep(1)
```

Explain that Python needs to load the time module before using its tools.

Mixing Up the List Name

If the list is named `credits`, the loop must use that same name:

```
credits = ["Director", "Actor"]

for credit in credits:
    print(credit)
```

Python treats names such as `credits`, `Credits`, and `credit` as different names.

Accidentally Using the Wrong Indentation

Explain that code inside the loop should line up:

```
for credit in credits:
    print(credit)
    time.sleep(1)
```

Both instructions are part of the loop because both are indented equally.

Guided Practice: 15–20 Minutes

Students should work individually or with a partner while the teacher circulates. Encourage students to run their code after every small change.

Task 1: Create a List and Print It

Students create a list of three names and print the whole list.

```
names = ["Ava", "Mateo", "Sam"]

print(names)
```

Students should confirm that all three names appear.

Quick Check

Ask:

- What is the name of the list?
- How many items are in it?
- Which symbols surround the list?
- Which function displays the list?

Task 2: Use a `for` Loop

Students change their program so that each name appears separately.

```
names = ["Ava", "Mateo", "Sam"]

for name in names:
    print(name)
```

Ask students to predict the output before running the program.

Quick Check

Students should be able to point to:

- The list

- The loop
- The temporary variable
- The colon
- The indented instruction

Task 3: Add a Delay

Students import `time` and add a one-second pause.

```
import time

names = ["Ava", "Mateo", "Sam"]

for name in names:
    print(name)
    time.sleep(1)
```

Invite students to change the delay to:

```
time.sleep(2)
```

Ask how the output changes.

Guiding Questions for Circulation

Use questions that help students reason instead of immediately giving the answer:

- What did you expect the program to do?
- What did the program actually do?
- Which line did Python identify as the problem?
- Does the list name match the name used in the loop?
- Does the `for` line end with a colon?
- Is the instruction inside the loop indented?
- Did you import `time` before using `time.sleep()`?
- Are the names inside quotation marks?
- What happens when you run the smallest working version?

Formative Check: Code Detective

Display the following code:

```
names = ["Ava", "Mateo", "Sam"]

for name in names
    print(name)
```

Ask students to identify the error before running it.

Expected correction:

```
names = ["Ava", "Mateo", "Sam"]

for name in names:
    print(name)
```

Independent Practice: 15–20 Minutes

Assignment: Create a Movie Credits Program

Students create their own movie credits animation program.

They may invent a movie or create credits for a fictional class production.

Starter Template

```
# My movie credits program

import time

credits = [
    "Title: _____",
    "Director: _____",
    "Actor: _____",
    "Music: _____",
    "Thanks for watching!"
]

for credit in credits:
    print(credit)
    time.sleep(1)
```

Student Directions

1. Add a title to the program comment.
2. Create a list named `credits`.
3. Include at least five credits.
4. Put quotation marks around every credit.
5. Use a `for` loop to visit each credit.
6. Print each credit.

7. Add a one-second delay between credits.
8. Run the program and correct any errors.
9. Personalize the credits with names, roles, or a fictional movie title.

Success Criteria

A successful program:

- Runs without crashing
- Includes `import time`
- Includes a list with at least five credits
- Uses a `for` loop
- Uses `print(credit)`
- Uses `time.sleep(1)`
- Has correct indentation
- Includes quotation marks around each text item
- Includes a helpful comment

Optional Formatting

Students may add blank lines to make the output easier to read:

```
for credit in credits:
    print()
    print(credit)
    time.sleep(1)
```

They may also add an opening message:

```
print("Now showing...")
time.sleep(2)
```

Differentiation

Below Grade Level or Students Needing Additional Support

Provide a fill-in-the-blank template:

```
# My movie credits program

import time
```

```
credits = [
    "____________________",
    "____________________",
    "____________________"
]

for __________ in __________:
    print(__________)
    time.sleep(__________)
```

Additional supports:

- Reduce the requirement to three credits.
- Provide a syntax reference card.
- Use pair programming with rotating roles:
    - **Driver:** types the code
    - **Navigator:** reads the instructions and checks punctuation
- Provide a completed example for students to copy and modify.
- Highlight matching indentation.
- Allow students to choose credits from a prepared list.
- Encourage students to test one line or section at a time.
- Provide a teacher-created list of common errors and corrections.

Syntax Reference Card

```
# Comment

import time

items = ["item 1", "item 2", "item 3"]

for item in items:
    print(item)
    time.sleep(1)
```

Remember:

- Lists use square brackets.
- Text uses quotation marks.
- `for` statements end with a colon.
- Instructions inside a loop are indented.

- `time` must be imported before `time.sleep()` is used.

On Grade Level

Students independently write a complete program with:

- Five or more credits
- A list
- A `for` loop
- `print()`
- `time.sleep()`
- Correct indentation and syntax

Above Grade Level

Students may choose one or more extensions:

Add User Input for the Delay

```
import time

credits = [
    "Director: Jordan Lee",
    "Actor: Maya Chen",
    "Music: Alex Rivera"
]

seconds = float(input("How many seconds should appear between credits? "))

for credit in credits:
    print(credit)
    time.sleep(seconds)
```

Explain that `input()` collects information typed by the user.

Create Two Lists

```
import time

roles = ["Director", "Actor", "Music"]
names = ["Jordan Lee", "Maya Chen", "Alex Rivera"]

for index in range(len(roles)):
    print(roles[index] + ": " + names[index])
    time.sleep(1)
```

This extension introduces list positions and `range()`. It may be completed with teacher guidance.

Experiment with Blank Lines

```
for credit in credits:
    print()
    print(credit)
    time.sleep(1)
```

Students can compare the output with and without the extra `print()` statement.

Closing: 5–10 Minutes

Share-Out

Invite several students to run their programs for the class or show them to a partner.

Students should identify:

- The list
- The `for` loop
- The output line
- The delay line

Reflection

Students respond to:

> What was easier than you expected about programming?
> What was more difficult than you expected?
> What is one Python rule you learned today?

Class Recap

Reinforce these ideas:

- A list stores multiple items.
- `print()` displays information.
- A `for` loop repeats instructions for each item.
- `time.sleep()` pauses a program.
- Indentation matters in Python.
- Errors help programmers locate problems.

Preview of the Next Lesson

Explain that the next lesson may build on this program by introducing:

- Variables
- User input
- Making decisions with `if`
- Personalizing the credits program
- Improving the visual design of the output

Extension / Above-Grade Challenge

Students who finish early may enhance the program in creative ways.

Add ASCII Art

```
print("************************")
print("*      THE END         *")
print("************************")
```

Add Color

Color support depends on the programming environment. In environments that support ANSI color codes, students may try:

```
print("\033[95m" + credit + "\033[0m")
```

Explain that not every editor displays color codes in the same way.

Add a Changing Delay

```
import time

credits = [
    "Opening Credit",
    "Director: Jordan Lee",
    "Actor: Maya Chen",
    "Music: Alex Rivera"
]

delay = 1

for credit in credits:
    print(credit)
    time.sleep(delay)
    delay = delay + 1
```

Students can observe how the delay changes during the program.

Create a Credits Banner

```
print("==============================")
print("        MY MOVIE               ")
print("==============================")
```

Explore Screen Clearing

Some environments support screen-clearing commands, but the exact command depends on the computer system and editor. Students should test this only with teacher guidance.

A platform-specific example may look like:

```
import os
import time

credits = ["Director: Jordan Lee", "Actor: Maya Chen", "Music: Alex Rivera"]

for credit in credits:
    os.system("cls" if os.name == "nt" else "clear")
    print(credit)
    time.sleep(1)
```

Explain that this is an optional challenge because it introduces operating-system commands that are not necessary for the main lesson.

Homework

Adapt the Movie Credits Program

Students modify their program by:

- Adding at least three more credits
- Changing the delay to a different value
- Adding a movie title
- Adding a final message such as `"Thank you for watching!"`
- Running the program to confirm that it works

Example:

```
import time

credits = [
    "MY FIRST PYTHON MOVIE",
    "Director: Jordan Lee",
    "Actor: Maya Chen",
    "Music: Alex Rivera",
```

```
    "Editor: Sam Patel",
    "Special Thanks: My Class",
    "Thank you for watching!"
]

for credit in credits:
    print(credit)
    time.sleep(1.5)
```

Written Reflection

Students write a short response:

> Describe your first experience writing a computer program. What did your program do? What mistake or challenge did you encounter, and how did you solve it?

Students should understand that debugging is a normal part of programming, not a sign that they are doing poorly.

Standards Aligned

The standards-retrieval search did not return matching standard codes or descriptions. Therefore, no specific standards codes are listed here.

This lesson supports introductory computer science practices commonly included in programming standards:

- Developing and testing a simple program
- Using a sequence of instructions
- Recognizing repetition in an algorithm
- Representing information in a list
- Using iteration to process multiple items
- Reading and responding to error messages
- Debugging through testing and revision
- Explaining how code produces an observable result
- Practicing persistence and collaboration during problem solving