Hour of AI - Creative Coding with Python and AI

Overview and Purpose

In this one-hour lesson, students will explore how AI can support creative coding with Python. They'll learn how to write simple code to create pixel art and animations while using an AI-powered Debugging Buddy to identify and fix errors. Students will practice computational thinking, creativity, and responsible use of AI.

Suggested Learning Targets

1. Understand what AI is and how it supports coding.
2. Create simple Python programs using variables, RGB colors, functions, loops, and animations.
3. Use the **Debugging Buddy** responsibly to identify and fix coding errors.
4. Demonstrate perseverance and growth mindset. .

Glossary

- **AI (Artificial Intelligence)**: A type of computer technology that can learn from information and make decisions like humans.
- **Matrix**: A grid of pixels used for creating digital images or animations.
- **Debugging**: The process of finding and fixing errors (bugs) in code.
- **Variable**: A name that stores a piece of information in a program.
- **Function**: A reusable block of code that performs a specific task.
- **Animation**: A sequence of frames that creates the illusion of movement.

Documentation

```
m[2][3] = on  # Turns on a pixel
m[2][3] = (255, 0, 0)  # Colors the pixel red using RGB
eye = (25, 200, 207)  # Creates a variable for eye color

def smiley(eye, mouth):
    m[2][2] = eye
    m[2][5] = eye
    m[5][1] = mouth
smiley(A, R)

a = Animation()  # Creates an animation
a.add_frame(m)   # Adds a frame to the animation
```

Teacher Tips

- Encourage students to debug independently before using the Debugging Buddy.
- Highlight how AI provides guidance, not answers — students must evaluate its responses.
- Remind them to check indentation and syntax carefully when using loops and functions.

Standards

CSTA Standards

CSTA Level 1B (Grades 3-5):

Computing Systems (CS)

- 1B-CS-01: Describe how internal and external parts of computing devices function to form a system.
- 1B-CS-02: Model how computer hardware and software work together as a system to accomplish tasks.
- 1B-CS-03: Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.

Data & Analysis (DA)

- 1B-DA-06: Organize and present collected data visually to highlight relationships and support a claim.
- 1B-DA-07: Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.

Algorithms & Programming (AP)

- 1B-AP-08: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
- 1B-AP-09: Create programs that use variables to store and modify data.
- 1B-AP-10: Create programs that include sequences, events, loops, and conditionals.
- 1B-AP-11: Decompose (break down) problems into smaller, manageable sub-problems to facilitate the program development process.
  1B-AP-12: Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
- 1B-AP-13: Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.
- 1B-AP-14: Observe intellectual property rights and give appropriate attribution when creating or remixing programs.
- 1B-AP-15: Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
- 1B-AP-16: Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
- 1B-AP-17: Describe choices made during program development using code comments, presentations, and demonstrations.

Impacts of Computing (IC)

- 1B-IC-18: Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.
- 1B-IC-19: Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.
- 1B-IC-20: Seek diverse perspectives for the purpose of improving computational artifacts.

CSTA Level 2 (Grades 6-8):

Computing Systems (CS)

- 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Data & Analysis (DA)

- 2-DA-07: Represent data using multiple encoding schemes.

Algorithms & Programming (AP)

- 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
- 2-AP-13: Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
- 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
- 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
- 2-AP-17: Systematically test and refine programs using a range of test cases.
- 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
- 2-AP-19: Document programs in order to make them easier to follow, test, and debug.
- Impacts of Computing (IC)
- 2-IC-20: Compare trade-offs associated with computing technologies that affect people's everyday activities and career options.
- 2-IC-21: Discuss issues of bias and accessibility in the design of existing technologies.

ISTE:

1.1 Empowered Learner

- 1.1.a – Students connect their learning needs, strengths and interests to their goals and use technology to help achieve them and reflect on their progress.
- 1.1.b – Students build networks and customize their learning environments in ways that support the learning process.
- 1.1.c – Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.

- 1.1.d – Students understand fundamental concepts of how technology works, demonstrate the ability to choose and use current technologies effectively, and are adept at thoughtfully exploring emerging technologies.

1.2 Digital Citizen
- 1.2.b – Students demonstrate empathetic, inclusive interactions online and use technology to responsibly contribute to their communities.
- 1.2.c – Students safeguard their well-being by being intentional about what they do online and how much time they spend online.

1.3 Knowledge Constructor
- 1.3.a – Students use effective research strategies to find resources that support their learning needs, personal interests and creative pursuits.
- 1.3.c – Students curate information from digital resources using a variety of tools and methods to create collections of artifacts that demonstrate meaningful connections or conclusions.
- 1.3.d – Students build knowledge by exploring real-world issues and gain experience in applying their learning in authentic settings.

1.4 Innovative Designer
- 1.4.a – Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- 1.4.b – Students select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
- 1.4.c – Students develop, test and refine prototypes as part of a cyclical design process.
- 1.4.d – Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

1.5 Computational Thinker
- 1.5.a – Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- 1.5.b – Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- 1.5.c – Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- 1.5.d – Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6 Creative Communicator
- 1.6.a – Students choose the appropriate platforms and digital tools for meeting the desired objectives of their creation or communication.
- 1.6.b – Students create original works or responsibly repurpose or remix digital resources into new creations.
- 1.6.c – Students use digital tools to visually communicate complex ideas to others.

- 1.6.d – Students publish or present content that customizes the message and medium for their intended audiences.

1.7 Global Collaborator

- 1.7.a – Students use digital tools to connect with learners from a variety of backgrounds, recognizing diverse viewpoints and broadening mutual understanding.
- 1.7.b – Students use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.
- 1.7.c – Students contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.
- 1.7.d – Students explore local and global issues and use collaborative technologies to work with others to investigate solutions.

Materials and Resources

Getting Started Guide - imagi Edu
Teacher slides - CREATIVE CODING WITH PYTHON AND AI

Suggested Lesson Structure

Part 1: Coding Basics and Debugging Buddy

1. Intro to Python and the Matrix [Direct Instruction]

Introduce students to the imagi Edu platform and explain how code interacts with the pixel matrix. Show the interface: code editor, error messages, and output grid. Explain how a pixel can be turned on and off using simple Python commands such as `m[row][col] = on`. Provide quick examples and have students predict what will happen before running code.

2. Run Your First Code! [Guided Practice]

Students create a new project and run the sample code. Ask: “What does this line do?” Discuss coordinates and matrix structure (rows and columns). Guide students through common errors (e.g., syntax, index out of range) and introduce the Debugging Buddy as a tool to help identify and fix errors.

```python
m[2][3] = on
```

3. Debugging Buddy [Unplugged + Interactive Discussion]

Define “bugs” and “debugging.”Explain why the Debugging Buddy doesn’t fix code but gives hints. Have students identify an intentional syntax error on the board and work

together to debug it. Reinforce responsible AI use — ask: “When should we ask the Debugging Buddy for help?”

4. Add Some Color! [Guided Practice]

Students modify their code to replace `on` with color variables (e.g., `R`, `G`, `B`).

Demonstrate how to use RGB color values such as `(255, 0, 0)` for red. Show how to use the Color Picker to find RGB combinations. Discuss computational thinking over memorization: patterns > memorizing values.

5. Variables for Efficiency [Direct Instruction]

Introduce variables as a way to store color values. Show how changing one variable updates multiple parts of the code. Ask students to describe why this makes code easier to manage.

```python
eye = (25, 200, 207)
mouth = (255, 165, 150)
m[2][2] = eye
m[2][5] = eye
m[5][1] = mouth
m[6][2] = mouth
m[6][3] = mouth
m[6][4] = mouth
m[6][5] = mouth
m[5][6] = mouth
```

6. My Colorful Smiley [Guided Practice]

Students recreate a colorful smiley face using variables for different elements (eyes, mouth). Encourage experimentation with colors and patterns. Walk around and assist students using Debugging Buddy tips.

Part 2: Functions, Loops, and Animation

1. Functions for Reusable Code [Direct Instruction]

Explain what functions are and why they’re useful. Walk through defining and calling a function. Call the function using `smiley(A, R)` and discuss parameters.

```python
def smiley(eye, mouth):
```

```
    m[2][2] = eye
    m[2][5] = eye
    m[5][1] = mouth
    m[6][2] = mouth
    m[6][3] = mouth
    m[6][4] = mouth
    m[6][5] = mouth
    m[5][6] = mouth
smiley(A, R)
```

2. Animate It! [Guided Practice]

Introduce the concept of animation using frames. Students observe how frames create motion. Discuss: “What happens if we swap the colors or add more frames?”

Python

```
a = Animation()
def smiley(eye, mouth):
    m[2][2] = eye
    m[2][5] = eye
    m[5][1] = mouth
    m[6][2] = mouth
    m[6][3] = mouth
    m[6][4] = mouth
    m[6][5] = mouth
    m[5][6] = mouth
smiley(R, A)
a.add_frame(m)
smiley(A, R)
a.add_frame(m)
```

3. For Loops and Patterns [Direct Instruction + Guided Practice]

Explain how loops reduce repetition. Show how to fill a full row of pixels with a loop. Discuss indentation and the concept of iteration. Prompt students to predict output before running.

Python

```
for col in range(0, 8):
    m[0][col] = R
```

4. Rainbow Challenge [Guided Practice]

Students expand their loop to create multiple colored rows. Encourage exploration: What happens if we change the range or colors? Introduce adding animation frames inside the loop.

```python
for col in range(0, 8):
    m[0][col] = R
    m[1][col] = O
    m[2][col] = Y
    m[3][col] = G
    m[4][col] = A
    m[5][col] = B
    m[6][col] = P
    m[7][col] = M
```

5. Animate Your Rainbow! [Guided Practice]

Students combine loops, animation, and color to create a moving rainbow. Encourage creativity—students can make wave effects or flashing lights. Have them debug independently or with the Debugging Buddy.

```python
a = Animation()
for col in range(0, 8):
    m[0][col] = R
    m[1][col] = O
    m[2][col] = Y
    m[3][col] = G
    m[4][col] = A
    m[5][col] = B
    m[6][col] = P
    m[7][col] = M
    a.add_frame(m)
```

6. Responsible AI Use [Additional Activity]

Discuss how to use AI responsibly:

- Try problem-solving before asking the AI.
- Ask clear, specific questions.
- Always evaluate the AI's responses critically.

Students reflect: "Which AI skill do I want to improve—problem-solving, communication, or perseverance?"

Common Misconceptions
Address misconceptions that might have come up. Examples:

Misconception	Explanation
AI will fix my code for me.	The Debugging Buddy helps students understand their mistakes - it doesn't fix them automatically. Students must still apply problem-solving and reasoning skills to correct errors.
I should ask the Debugging Buddy right away when I see an error.	Students should first try to solve the problem themselves, review their code, and think critically before asking for help. This develops resilience and strengthens problem-solving skills.
Debugging is only about finding syntax errors.	Debugging also includes fixing logic errors, improving sequencing, and testing for unexpected outcomes. Teaching this helps students develop computational thinking.

Checking for Understanding

Administer a quick exit ticket at the end of class or assign any of the following as homework:

Quiz

1. What does the Debugging Buddy do?
   a) Writes your code for you
   b) Helps you fix and understand errors
   c) Adds colors automatically
   d) Makes your project faster

2. What is a variable used for?
   a) To repeat code
   b) To store information
   c) To color the matrix
   d) To debug programs

3. What should you do before asking AI for help?
   a) Give up
   b) Ask a friend
   c) Try to fix the issue yourself
   d) Restart the project

Answer Key:

1. b
2. b
3. c

Exercises

1. Write code that colors 3 rows of pixels in different RGB colors.
2. Modify the code in Exercise 1 to animate the colors using frames.
3. Identify and fix the bug in the following code:

```python
for col in range(0, 8):
m[0][col] = (255, 0, 0)
```

Solutions:

1.

```python
# Three distinct RGB colors (you can change these!)
row0 = (224, 182, 34)   # gold
row1 = (143, 88, 152)   # purple
row2 = (25, 200, 207)   # aqua

# Color the top three rows
for col in range(8):
    m[0][col] = row0
    m[1][col] = row1
    m[2][col] = row2
```

2.

```python
row0 = (224, 182, 34)
row1 = (143, 88, 152)
row2 = (25, 200, 207)

a = Animation()

for col in range(8):
```

```
m[0][col] = row0
m[1][col] = row1
m[2][col] = row2
a.add_frame(m)
```

3. **Error message:** expected an indented block after 'for' statement on line 1 (<string>, line 2).

   Debugging Buddy: The error occurs because the line inside your for loop is not indented. In Python, code blocks that follow statements like `for` need to be indented to show that they belong to that block. To fix this, you need to add an indentation (usually 4 spaces) before the line that sets the color of the pixel.

```
Python

for col in range(0, 8):
    m[0][col] = (255, 0, 0)
```

Student Reflection

1. What went well today?
2. What problem(s) came up?
3. What did you find most interesting in the lesson?
4. What did you find most challenging in the lesson?
5. What do you still have questions about?

Rubric

Suggested rubric for evaluating how well the student has met each learning target.

Area	Learning Target	Beginning	Developing	Proficient	Exceptional
Knowledge	Understand what AI is and how it supports coding	Cannot explain the role of AI in coding	Vaguely understands the role of AI in coding	Explains the AI's purpose clearly	Connects AI's role to real-world applications

Skill	Create code using variables, RGB colors, and animation	Requires significant help	Completes steps with guidance	Works independently	Adds creative features
Thinking	Debug using DB responsibly	Relies on DB for all help	Occasionally consults with DB	Evaluates and applies DB suggestions critically	Demonstrates independent reasoning when applying DB
Intrapersonal	*Demonstrate perseverance and growth mindset*	*Easily gives up*	*Needs encouragement*	*Shows persistence*	*Reflects on challenges and growth*

Follow-on activities

For educators and students eager to continue exploring creative coding and AI, imagi Edu offers full-length courses and cross-curricular lessons that build upon the concepts introduced in this lesson.

Grades 3-5	Creative Computer Science with Python ◆ Beginner 1	15 weeks (15-30 hours)
Grades 3-5	Creative Computer Science with Python ◆ Beginner 2	15 weeks (15-30 hours)
Grades 6-8	Creative Computer Science with Python ◆ Beginner	15 weeks (30-45 hours)
Grades 6-8	Creative Computer Science with Python ◆ Intermediate	15 weeks (30-45 hours)
Grades 3-5,	Cross-curricular Lessons	

6-8		

Explore educator resources here: https://imagilabs.com/pages/educator-resources