Interactive Coding Math Lesson: Number Detective Challenge

Platform: VS Code with Claude
Grade Level: Middle School (Grades 6-8)
Duration: 60-75 minutes (includes coding time)
Topic: Comparing and Ordering Fractions, Decimals, and Whole Numbers through Code

---

Generative Question

How can math help us see ourselves as creative thinkers and problem-solvers?

NEW: Why Add Coding?

Coding transforms abstract math into something students can:

- **Visualize** - See numbers on generated number lines
- **Experiment** - Test their comparison strategies instantly
- **Create** - Build their own number comparison tools
- **Debug** - Learn from mistakes in a low-stakes environment

---

Technical Setup (Before Class)

Requirements:

- VS Code installed on student computers
- Claude for VS Code extension (or access to Claude via browser)
- Python extension for VS Code (we'll use Python for its simplicity)
- Basic folder structure: Each student creates a `math_detectives` folder

Pre-Class Preparation:

1. Students create a new folder: `number_detective_project`
2. Open folder in VS Code
3. Create starter file: `number_tools.py`

Modified Lesson Flow

Opening Hook (5-7 minutes): "The Mystery Number Challenge"

Present the Same Scenario: *"Three friends are arguing about who lives closest to school. Maya says she lives 3/4 of a mile away, Jordan lives 0.6 miles away, and Alex lives 4/5 of a mile away. Who lives closest?"*

NEW Coding Connection:

- "Today, we're not just solving this problem—we're building TOOLS that can solve ANY comparison problem!"
- "We'll use Claude in VS Code to help us think through our code and test our strategies."

Activity 1 (15 minutes): "Building a Fraction-to-Decimal Converter"

Learning Goal: Understand equivalence by coding conversions

Student Task: Work with Claude to create a Python function that converts fractions to decimals.

Code Challenge:

python

```python
# Students work with Claude to complete this:

def fraction_to_decimal(numerator, denominator):
    """
    Convert a fraction to a decimal.

    Examples:
    fraction_to_decimal(3, 4) should return 0.75
    fraction_to_decimal(4, 5) should return 0.8
    """
    # TODO: Write your code here
    pass

# Test the function
print("Maya's distance: 3/4 =", fraction_to_decimal(3, 4))
print("Alex's distance: 4/5 =", fraction_to_decimal(4, 5))
print("Jordan's distance: 0.6")
```

Guiding Questions for Claude Interaction: Students ask Claude:

- "How do I convert a fraction to a decimal in Python?"
- "Can you explain why this code works?"
- "What happens if the denominator is zero? How should I handle that?"

Creative Extension:

- Handle mixed numbers (like 2 1/2)
- Round to specific decimal places
- Show repeating decimals (like 1/3)

Activity 2 (20 minutes): "Number Line Visualizer"

Learning Goal: Visualize number comparisons programmatically

Student Task: Create a text-based number line that shows where numbers fall.

Code Challenge:

python

```python
def create_number_line(numbers, start=0, end=1):
    """
    Create a visual number line showing where numbers fall.

    Args:
    numbers: list of numbers to plot (can be decimals)
    start: beginning of number line
    end: end of number line
    """
    # Students work with Claude to build this

    # Example output:
    # 0.0 |----*-------*---*-----| 1.0
    #         0.6    0.75  0.8

    pass

# Test with our friends' distances
distances = {
    "Jordan": 0.6,
    "Maya": 0.75,
    "Alex": 0.8
}

create_number_line(list(distances.values()))
```

Claude Collaboration Prompts:

- "Claude, how can I represent a number line using text characters?"
- "How do I calculate where to place each number on the line?"
- "Can you help me add labels showing who lives at each distance?"

Enhancement Ideas:

- Use ASCII art to make fancier number lines
- Color-code different number types (if terminal supports colors)
- Allow user input for custom numbers

Activity 3 (20 minutes): "Smart Number Comparator"

Learning Goal: Implement comparison strategies as algorithms

Student Task: Build a tool that compares ANY set of mixed numbers (fractions, decimals, whole numbers)

Code Challenge:

python

```python
def compare_numbers(numbers_list):
    """
    Compare and order a list of mixed numbers.

    Input can be:
    - Fractions as tuples: (3, 4) means 3/4
    - Decimals: 0.75
    - Whole numbers: 2

    Returns ordered list from smallest to largest
    """
    converted = []

    for num in numbers_list:
        if isinstance(num, tuple):  # It's a fraction
            decimal_value = num[0] / num[1]
            converted.append({
                'original': f"{num[0]}/{num[1]}",
                'value': decimal_value
            })
        else:  # It's already a decimal or whole number
            converted.append({
                'original': str(num),
                'value': num
            })

    # TODO: Sort the list and return it
    # Students work with Claude to complete

    pass

# Test with various numbers
test_numbers = [(3, 4), 0.6, (4, 5), 0.65, (1, 2)]
result = compare_numbers(test_numbers)
print("Ordered from smallest to largest:")
for item in result:
    print(f"{item['original']} = {item['value']}")
```

Claude Interaction:

- "Claude, how do I sort this list of dictionaries by the 'value' key?"
- "Can you explain what isinstance() does?"
- "How can I handle improper fractions like 5/4?"

Creative Additions: Students can ask Claude to help them:

- Add benchmark identification (is it closest to 0, 0.5, or 1?)
- Create comparison statements ("0.75 is greater than 0.6 by 0.15")
- Handle negative numbers
- Build a menu system for user input

Activity 4 (10 minutes): "Create Your Own Challenge Generator"

Learning Goal: Apply understanding by creating problems for others

Student Task: Build a random problem generator that creates comparison challenges.

Code Challenge:

python

```python
import random

def generate_challenge():
    """
    Generate a random number comparison challenge.
    Creates a mix of fractions, decimals, and whole numbers.
    """
    # Generate random fractions
    fraction1 = (random.randint(1, 9), random.randint(2, 10))
    fraction2 = (random.randint(1, 9), random.randint(2, 10))

    # Generate random decimals
    decimal1 = round(random.uniform(0, 2), 2)

    # Mix them together
    numbers = [fraction1, fraction2, decimal1]

    # Create a story problem
    stories = [
        f"Who ran farther: Alex who ran {fraction1[0]}/{fraction1[1]} miles, "
        f"or Sam who ran {decimal1} miles?",

        f"Which recipe uses more sugar: Recipe A with {fraction1[0]}/{fraction1[1]} cups "
        f"or Recipe B with {decimal1} cups?",
    ]

    # TODO: Students expand with more stories and complete the function

    pass

# Generate 3 random challenges
for i in range(3):
    print(f"\nChallenge {i+1}:")
    generate_challenge()
```

Pair Programming: Students work in pairs:

- One "drives" (types)
- One "navigates" (guides and asks Claude questions)
- Switch roles after 5 minutes

Closing Discussion (10 minutes): "Code as Creative Problem-Solving"

Reflection Questions:

Discuss as a class:

1. **"How did coding change the way you thought about comparing numbers?"**
   - Expected: "I had to be more precise," "I could test many examples quickly"
2. **"What was the most creative part of coding your solutions?"**
   - Expected: Creating visualizations, naming variables, choosing approaches
3. **"How did Claude help you as a problem-solving partner?"**
   - Expected: Explained concepts, caught errors, suggested improvements
4. **"What strategies did you use when your code didn't work?"**
   - Expected: Asked Claude, tested with simpler numbers, printed intermediate results
5. **"How is being a mathematical problem-solver similar to being a coder?"**
   - Expected: Both require logical thinking, creativity, debugging/checking work

Key Takeaway: "Today you used code to explore math concepts. You discovered that programming and mathematics both reward creative thinking, strategic planning, and persistence. YOU are not just math students—you're computational thinkers and creators!"

Assessment Opportunities

Formative (During Lesson):

- Review student code in VS Code (teacher circulates virtually or in-person)
- Check Claude conversation history for problem-solving approaches
- Observe how students debug and iterate on their code

Exit Ticket (Submitted as Code):

Students create a file `exit_ticket.py` that:

1. Compares these numbers: 5/8, 0.7, 2/3, 0.55
2. Includes comments explaining their strategy
3. Produces output showing the numbers in order

Example:


python

```python
# Exit Ticket - [Student Name]
# Strategy: I converted all numbers to decimals to compare them

def exit_ticket():
    """
    My strategy for comparing numbers is to convert everything
    to decimals because that makes it easier to see which is bigger.
    This shows creative problem-solving because I chose the method
    that makes the most sense to me.
    """
    numbers = {
        '5/8': 5/8,
        '0.7': 0.7,
        '2/3': 2/3,
        '0.55': 0.55
    }

    # Sort and display
    sorted_nums = sorted(numbers.items(), key=lambda x: x[1])

    print("Numbers from smallest to largest:")
    for name, value in sorted_nums:
        print(f"{name} = {value:.4f}")

exit_ticket()
```

Differentiation Strategies

For Students Who Need Support:

- Provide more complete starter code with TODO comments
- Pair with a coding buddy
- Give Claude specific prompts to ask (template questions)
- Focus on completing Activity 1 & 2 thoroughly rather than rushing through all

For Students Ready for More Challenge:

- Build a graphical number line using matplotlib or turtle graphics
- Create a Flask web app for their number comparator
- Add features like percentage calculations or ratio comparisons
- Explore floating-point precision issues (why 0.1 + 0.2 ≠ 0.3)

For Visual Learners:

- Emphasize the number line visualizer activity
- Explore adding ASCII art or colors
- Use print statements frequently to see intermediate results

For Students New to Coding:

- Start with very simple modifications to existing code
- Use Claude heavily for explanations
- Pair with experienced coder
- Focus on understanding logic before syntax

Technical Troubleshooting Guide

Common Issues & Solutions:

Issue: "My code has a syntax error"

- **Solution:** Ask Claude: "I'm getting a syntax error on line X. Can you help me understand what's wrong?"

Issue: "I don't understand what this code does"

- **Solution:** Ask Claude: "Can you explain this code line by line in simple terms?"

Issue: "My function returns None"

- **Solution:** Check if you're using `return` instead of just `print`

Issue: "Division by zero error"

- **Solution:** Add error checking for denominator = 0

Extensions & Home Projects

Coding Extensions:

1. **Data Analysis Project:**
    - Compare your class's quiz scores (as fractions of total points)
    - Visualize with bar charts or histograms
2. **Game Creation:**
    - "Number Line Jump" - guess where a number falls
    - "Fraction Wars" - compare randomly generated numbers
3. **Real-World Data:**
    - Scrape sports statistics (batting averages)
    - Compare stock prices or cryptocurrency values
    - Analyze recipe ingredient ratios

Home Connection:

"Code a Math Helper" Create a Python tool that helps with your math homework:

- Convert between fractions, decimals, and percentages
- Simplify fractions
- Find equivalent fractions
- Check if your comparison answers are correct

Standards Alignment

This lesson addresses:

Math Standards:

- Understanding rational numbers and their relationships
- Using visual models to represent and compare numbers
- Applying multiple problem-solving strategies

Computer Science Standards:

- Creating computational artifacts
- Using abstraction to manage complexity
- Testing and debugging programs
- Collaborating with AI tools

21st Century Skills:

- Computational thinking
- Problem decomposition
- Algorithmic thinking
- Digital literacy
- AI literacy (working with Claude)

Teacher Preparation Checklist

Before Class:

- [ ] Test all code examples in VS Code
- [ ] Ensure Claude for VS Code is working (or browser access ready)
- [ ] Prepare student accounts/access
- [ ] Create a shared folder with starter code
- [ ] Test on student machines if possible
- [ ] Prepare backup plan if tech fails (paper-based version of lesson)

During Class:

- [ ] Have working example code ready to share
- [ ] Monitor student-Claude conversations for misconceptions
- [ ] Save interesting student solutions to share with class
- [ ] Take screenshots of creative solutions

After Class:

- [ ] Collect student code files
- [ ] Review Claude conversation logs (if available)
- [ ] Note which activities worked best
- [ ] Identify students who need extra support or challenge

Tips for Using Claude with Students

Effective Claude Prompts to Teach Students:

Good Prompts: ✅ "Can you explain how to convert a fraction to a decimal in Python?" ✅ "My code on line 15 isn't working. Can you help me debug it?" ✅ "How can I make my number line visualization look better?" ✅ "Can you suggest a way to test if my function works correctly?"

Less Effective Prompts: ❌ "Do my homework" ❌ "Write all the code for me" ❌ "What's the answer?"

Teaching Students to Be Good Claude Partners:

1. **Ask specific questions** - Not "help me" but "how do I do X?"
2. **Show your thinking** - "I tried X because Y, but got Z error"
3. **Request explanations** - "Can you explain why this works?"
4. **Iterate** - "That works, but can we improve it by...?"

Why This Coding Approach Works

Benefits of the VS Code + Claude Format:

1. **Immediate Feedback:** Students see if their logic is correct instantly
2. **Low Stakes Experimentation:** Easy to try different approaches
3. **Scaffolded Support:** Claude provides just-in-time help
4. **Personalized Learning:** Students work at their own pace
5. **Real-World Skills:** Learning to code and work with AI
6. **Tangible Products:** Students create tools they can keep and share
7. **Growth Mindset:** Bugs become learning opportunities, not failures

Addresses the Generative Question:

"How can math help us see ourselves as creative thinkers and problem-solvers?"

- **Creative:** Students design their own functions and visualizations
- **Thinkers:** Must understand math concepts to code them
- **Problem-Solvers:** Debug code and choose strategies
- **Empowered:** Build tools that solve real problems

Sample Parent Letter

Dear Families,

This week in math, students explored comparing fractions, decimals, and whole numbers in a new way—through coding! Using VS Code and Claude AI, students:

- Built tools to convert fractions to decimals
- Created visual number lines
- Designed their own math problem generators

This approach helps students see math as creative problem-solving while developing valuable computational thinking skills. Students learned to work with AI as a learning partner, asking good questions and iterating on solutions.

At Home: Your student can continue exploring by creating Python programs that help with homework or solve everyday math problems. All code from class is saved in their `number_detective_project` folder.

Questions? Please reach out! We're excited about this new way of learning math.

Final Teacher Notes

Remember: The goal isn't perfect code—it's mathematical understanding and creative problem-solving. Celebrate:

- Students who try multiple approaches
- Creative variable names and comments
- Questions asked to Claude that show deep thinking

- Code that "fails" but shows good mathematical reasoning
- Collaboration and peer teaching

The code is a tool for exploring math, not an end in itself.

Students are building confidence as both mathematicians and coders—two identities that will serve them well in an increasingly computational world.