Sensor, Signal and Information Processing Center

Instructional Lesson Plan

Instructional Lesson Title	Battleships Python Game: AP Computer Science and App Development				
Subject Area *Highlight all subject areas that apply to this lesson.* *Subject area definitions*	Algebra	Biology	Chemistry	Computer Sci	Data Analysis
	Earth/Space	Geometry	Life Science	Measurement	Numbers
	Physical Sci	Physics	Problem Solving	Reasoning	Sci & Tech
Keywords (4-10 words)	AP Computer Science, Image Filtering, Image Recognition, AI, Problem-Based-Learning, Review Games, PBL				
Unit Duration (in min.)	580				
Focus Grade Level	11	**Grade Level Range**	10 to 12		

Instructional Unit Summary

This unit focuses on introducing AP computer science students to the fundamental concepts of code inputs, outputs, objects, and image processing through an engaging and collaborative group activity. The unit begins with an exploration of code inputs and outputs, emphasizing their importance in programming. Students learn about Python inputs and outputs and gain an understanding of how to dehaze premade images using Python libraries such as OpenCV. Through hands-on activities and experimentation, students develop their problem-solving skills while enhancing their understanding of image processing techniques.

In the second part of the unit, students delve into the concept of objects in Python programming. They learn how objects are used to represent real-world entities and gain insight into object-oriented programming. To apply their knowledge and foster teamwork, students participate in a group "Battleship" game. They collaborate to filter and analyze battleship board images, strategically attempting to "attack" other groups. This game allows students to apply their understanding of code inputs, outputs, and objects in a dynamic and interactive context. Overall, this unit combines theory, practical skills, and collaboration, providing students with a comprehensive introduction to Python programming, image processing, and problem-solving techniques.

Engineering Connection	Engineering Category
The concepts studied in this unit, such as code inputs, outputs, objects, and image processing, are highly relevant to real-world engineering problems and concerns. In the field of engineering, code inputs and outputs play a crucial role in developing software applications, controlling hardware systems, and analyzing data. Understanding how to effectively handle inputs and outputs is essential for designing efficient and reliable engineering solutions. *Additionally, image processing techniques, including dehazing, are widely used in various engineering domains. Engineers often encounter situations where images need to be enhanced,*	*Highlight ONE of the below to demonstrate this lesson's depth of engineering content.* 1. Engineering with some science/math

filtered, or analyzed to extract valuable information or improve visual clarity. By learning these techniques, students gain skills applicable to fields such as computer vision, robotics, remote sensing, medical imaging, and more. The unit's emphasis on object-oriented programming also aligns with industry practices, as it promotes modular and scalable software development, essential for engineering projects.	

Materials & Equipment

Based on the provided lesson summary, here is a list of materials and equipment that could be useful for teaching this unit on AP computer science:

1. Computers: Each student will need access to a computer with Python programming environment installed. This can be a desktop computer, laptop, or a computer lab setup.

2. Python Programming Environment: Ensure that the Python programming environment is installed on the computers. Popular options include Anaconda, IDLE, PyCharm, or Jupyter Notebook.

3. OpenCV Library: Install the OpenCV library for Python on the computers. This library is used for image processing and will be essential for the dehazing activity mentioned in the unit.

4. Premade Images: Prepare a collection of premade images for the dehazing activity. These images should have varying degrees of haze or fog that students can practice dehazing using Python and the OpenCV library.

5. Battleship Board Images: Create battleship board images that students will use during the "Battleship" game. These images should represent the boards with ships placed on them.

6. Projector or Interactive Whiteboard: Use a projector or interactive whiteboard to demonstrate code examples, share instructions, and showcase student work during the lessons.

7. Internet Access: Ensure that students have internet access to download and install Python libraries, access learning resources, and conduct research related to the unit.

8. Collaboration Tools: Provide collaboration tools such as shared documents, online platforms, or whiteboards where students can work together, plan their strategies for the "Battleship" game, and communicate effectively.

9. Whiteboard or Flipchart: Use a whiteboard or flipchart to illustrate concepts, write down code snippets, brainstorm ideas, or facilitate discussions during the lessons.

10. Printed Handouts or Digital Resources: Prepare handouts or digital resources that summarize key concepts, provide step-by-step instructions, and offer additional practice exercises for students to refer to during the unit.

11. Evaluation Materials: Prepare evaluation materials such as rubrics, assessment criteria, or grading sheets to assess students' understanding, problem-solving skills, and collaboration during the activities.

Remember to adapt the list according to the specific needs and resources available in your teaching environment.

Attachments

1. *List any documents, files, images, etc. to attach to this lesson.* 2.	3. 4.

Prerequisite Student Knowledge

Skills and knowledge that students should already have in order to be successful in this lesson:

Basic understanding of programming concepts: Students should have a basic understanding of programming concepts such as variables, functions, conditionals, loops, and basic data types.

Familiarity with Python syntax: Students should have prior experience with Python programming or be familiar with the syntax and structure of Python code.

Educational Standards

List 3-4 educational STEM standards that students would learn as a result of completing this lesson. If students need a skill to complete an activity, but the activity assumes they already have that skill, then the activity does not teach that skill.
For each standard, please include the source, year, standard number(s)/letter(s), grade band and text (if available, the unique ID# for the standard is helpful). Example: North Carolina, science, 2004, 1.01 (grades 8-8): Identify and create questions and hypotheses that can be answered through scientific investigations. ID# S1028531

Provide at least ONE from each of the following:

1. List State Standards (Arizona K-12; Arizona MCC Course Competencies; other states via Teach Engineering)

Computing and Society (AP CS Principles)
Standard: CSA.CSP.1
Description: Analyze the impact of computing innovations on society, privacy, and personal data, and demonstrate responsible and ethical use of computing resources.

Learning Objective in relation to the lesson:

Students will explore the impact of image processing techniques on society, such as in the context of dehazing premade images, and discuss the ethical considerations and potential privacy implications of such techniques. They will also demonstrate responsible and ethical use of computing resources while working collaboratively on the "Battleship" game, respecting the privacy and intellectual property of other groups.

Note: The specific state standards may vary depending on the version and updates of the curriculum. The given standard is an example based on the general Computing and Society domain of AP Computer Science Principles in Arizona. It is important to refer to the most recent version of the standards provided by the Arizona Department of Education to ensure accurate alignment with the curriculum.

2. List Next Generation Science Standards (NGSS)

Among the Next Generation Science Standards (NGSS), the standard most applicable to this lesson would likely be:

HS-PS4-1: Use mathematical representations to support a claim regarding relationships among the frequency, wavelength, and speed of waves traveling in various media.

While this standard is not directly related to computer science or image processing, it aligns with the broader concept of using mathematical representations to support claims and understand relationships. In the context of the lesson, students may utilize mathematical representations and concepts related to image processing techniques, such as understanding the transformation of pixel values, color spaces, or the mathematical operations involved in dehazing images using Python and the OpenCV library.

It's important to note that NGSS standards are primarily designed for science education, and computer science topics may not have a direct one-to-one correspondence with these standards. However, teachers can often find connections between NGSS standards and cross-cutting concepts or practices when integrating computer science into their lessons.

3. List International Technology and Engineering Educators Association (ITEEA) Standards

Standard 2: Design

Students will apply the engineering design process to solve problems and create innovative solutions.

This standard aligns well with the lesson's focus on problem-solving skills, collaboration, and the application of programming concepts in real-world contexts. Throughout the lesson, students engage in hands-on activities, such as dehazing images and participating in the "Battleship" game, which require them to think critically, plan, iterate, and evaluate their designs or strategies. By following the engineering design process, students will develop a systematic approach to problem-solving and apply their knowledge of code inputs, outputs, and objects to create effective solutions.

4. List Teacher Leader Engineering Network (TaLENt) Criteria

Standard: Facilitate Collaborative Problem-Solving in Engineering Education

Description: Teacher leaders in engineering education facilitate collaborative problem-solving experiences that engage students in authentic, real-world engineering challenges.

In the context of the lesson, the teacher can incorporate the following aspects of this standard:

Facilitating collaborative problem-solving: The teacher guides students in working together to solve problems and complete activities, such as dehazing images or playing the "Battleship" game, fostering collaboration, communication, and teamwork among students.

Engaging students in authentic, real-world challenges: The teacher ensures that the activities and projects in the lesson reflect real-world scenarios or applications of programming, image processing, and problem-solving techniques. This provides students with meaningful and relevant experiences, connecting their learning to the broader context of engineering and technology.

Learning Objectives

By the end of the lesson, students will be able to:

1. Describe the process of dehazing premade images using Python and OpenCV libraries.
   - Condition: Given a set of premade images with varying degrees of haze.
   - Criterion: Students will accurately describe the steps involved in dehazing images using Python and OpenCV, including the specific functions or techniques employed.

2. Identify the key concepts and characteristics of object-oriented programming (OOP) in Python.
   - Condition: Presented with examples and explanations of OOP in Python.
   - Criterion: Students will correctly identify and explain the key concepts of OOP, such as objects, classes, attributes, and methods, in Python.

3. Collaboratively strategize and apply code inputs, outputs, and objects in the "Battleship" game.
   - Condition: Engaged in a group activity where they participate in the "Battleship" game.

- Criterion: Working as a team, students will demonstrate effective collaboration, problem-solving, and application of code inputs, outputs, and objects to strategically analyze battleship board images and make informed decisions in the game.

Vocabulary	Definitions
vocab word/phrase (lower case)	Definition punctuated like a complete sentence even if it's only a phrase.

Code inputs: Values or data provided to a program or function as input.

Code outputs: Results or information produced by a program or function as output.

Image processing: The manipulation and analysis of digital images using computer algorithms to enhance or modify them.

Dehazing: The process of reducing or removing haze or fog from an image to improve visibility and clarity.

Python: A popular programming language known for its simplicity and readability.

OpenCV: Open Source Computer Vision Library, a Python library used for computer vision and image processing tasks.

Objects: In object-oriented programming, objects are instances of a class that represent real-world entities and possess attributes and behaviors.

Object-oriented programming (OOP): A programming paradigm that organizes code around objects, their properties (attributes), and actions (methods).

Battleship: A strategy-based board game where players attempt to sink each other's hidden ships by making guesses about their locations.

Collaboration: Working together with others to achieve a common goal, often involving communication, sharing ideas, and coordinating efforts.

Problem-solving: The process of finding solutions to challenges or puzzles by analyzing the problem, developing strategies, and evaluating potential solutions.

Algorithm: A step-by-step procedure or set of rules used to solve a specific problem or perform a task.

Image enhancement: Techniques used to improve the quality, visibility, or appearance of an image.

Lesson Procedure

Introduction/Motivation

How do you grab students' interest? Written toward the students. Include an engineering context.

Welcome to an exciting journey into the world of computer science and engineering! In this lesson, we are going to dive into the fascinating realm of code inputs, outputs, objects, and image processing. Get ready to explore the power of programming and how it can be applied in real-world engineering scenarios.

Imagine this: You are an engineer working on a cutting-edge project that requires you to analyze and enhance images using computer algorithms. You might be working on developing advanced surveillance systems or creating self-driving cars that can perceive and understand the world around them. These are just a few examples of how image processing and programming play a vital role in the field of engineering.

Have you ever wondered how computer programs take input data and produce meaningful outputs? Think about it as feeding instructions to a machine and receiving results in return. By understanding this process, we can unleash the potential of technology and achieve remarkable things.

In this lesson, we are going to learn about Python programming, a versatile language used by engineers and developers worldwide. We will uncover the magic behind code inputs and outputs, discovering their significance in creating innovative solutions. You will explore how to dehaze images using Python libraries like OpenCV, enhancing visibility and improving the quality of visuals.

But that's not all! We will also delve into the concept of objects in programming, which is like representing real-world entities in a virtual space. You will gain insights into object-oriented programming, a powerful approach used by engineers to design and build complex systems.

To make things even more exciting, we will engage in a group activity inspired by the classic game of "Battleship." You will collaborate with your teammates, analyzing and strategizing how to "attack" other groups by filtering and analyzing battleship board images. This interactive game will allow you to apply your newfound knowledge of code inputs, outputs, and objects in a dynamic and engaging context.

Throughout this lesson, we will not only enhance our programming skills but also develop problem-solving abilities, teamwork, and an understanding of how engineering concepts are applied in real-world scenarios. So get ready to unlock the world of code and image processing while immersing yourself in an exciting engineering context.

Get set to embark on an extraordinary adventure that will challenge your mind, expand your skills, and inspire you to think like an engineer. Let's dive in and discover the wonders of computer science and engineering together!

Lesson Background/Teacher Concepts

Summarize pertinent information a teacher would need to teach this lesson.

To effectively teach this lesson, the teacher should be familiar with the following information:

1. Lesson Summary: Understand the overall objectives and flow of the lesson, which introduces AP Computer Science students to code inputs, outputs, objects, and image processing through collaborative activities and an engineering context.

2. Learning Objectives: Review the three learning objectives of the lesson, which focus on students being able to describe dehazing images using Python and OpenCV, identify key concepts of object-oriented programming in Python, and collaboratively apply code inputs, outputs, and objects in the "Battleship" game.

3. Materials and Equipment: Take note of the required materials and equipment for the lesson, such as computers with Python and OpenCV installed, premade images with varying degrees of haze, access to the "Battleship" game materials, and collaboration tools if necessary.

4. Vocabulary: Familiarize yourself with the important vocabulary terms related to the lesson, as these will be introduced and defined throughout the instruction.

5. Pre-requisite Knowledge: Understand the necessary skills and knowledge students should already have to be successful in the lesson, such as a basic understanding of programming concepts and Python.

6. Standards Alignment: Determine how the lesson aligns with relevant educational standards, such as the AP Computer Science Principles framework, state-specific computer science standards, and any applicable science or engineering standards.

7. Introduction and Motivation: Be prepared to engage students by providing an exciting introduction that grabs their interest, emphasizing an engineering context, and highlighting the relevance of the lesson to real-world applications.

8. Instructional Strategies: Plan for effective instructional strategies that engage students in hands-on activities, promote collaboration, encourage problem-solving, and provide opportunities for experimentation and exploration.

9. Demonstration and Examples: Prepare demonstrations and examples that illustrate the concepts and techniques covered in the lesson, ensuring clarity and understanding.

10. Assessment and Evaluation: Determine how student learning and progress will be assessed, such as through formative assessments, group presentations, or individual assignments. Consider rubrics or criteria to evaluate students' understanding and application of the key concepts.

11. Safety and Ethical Considerations: Discuss any safety considerations related to using computers and software, and address ethical considerations related to image processing, privacy, and responsible use of technology.

By having a thorough understanding of these key aspects, the teacher will be well-equipped to effectively deliver the lesson, engage students in meaningful learning experiences, and help them develop their programming, problem-solving, and collaborative skills in an engineering context.

Learning Activities/Strategies

In this lesson, several learning activities and strategies can be employed to enhance student engagement, understanding, and application of the concepts. Here are some suggested activities and strategies:

1. Guided Discussion: Engage students in a guided discussion to explore the importance of code inputs and outputs in programming. Prompt them with questions to encourage critical thinking and participation. Provide real-life examples to help students understand the relevance of inputs and outputs in various applications. You can use the Socratic method or open-ended questions to stimulate discussion and encourage students to share their ideas.

2. Hands-On Image Dehazing: Organize a hands-on activity where students work individually or in pairs to dehaze premade images using Python and OpenCV libraries. Provide step-by-step instructions or a tutorial to guide students through the process. Encourage students to experiment with different techniques and parameters to observe the effects on the images. This activity allows students to gain practical experience in image processing and develop problem-solving skills.

3. Object-Oriented Programming Exploration: Facilitate an exploration of object-oriented programming (OOP) concepts using Python. Provide examples and explanations of classes, objects, attributes, and methods. Students can work individually or in small groups to analyze and discuss code snippets that showcase OOP principles. Encourage them to identify objects and their associated attributes and behaviors. This activity promotes critical thinking, analysis, and comprehension of OOP concepts.

4. Collaborative "Battleship" Game: Conduct a collaborative group activity based on the "Battleship" game, where students collaborate and strategize to analyze battleship board images. Each group can have a set of board images to analyze, applying image processing techniques to identify and attack targets strategically. Students can take turns analyzing the images, discussing possible moves, and making informed decisions. The game promotes teamwork, collaboration, and the application of code inputs, outputs, and objects in a dynamic and interactive context.

5. Formative Assessments: Incorporate formative assessments throughout the lesson to gauge student understanding and progress. These can include short quizzes, exit tickets, or class discussions where students can share their insights and ask questions. Formative assessments provide valuable feedback to both students and teachers, helping identify areas for clarification or reinforcement.

6. Reflection and Summarization: Allocate time for students to reflect on their learning and summarize the key concepts covered in the lesson. This can be done through individual reflection exercises, group discussions, or written reflections. Summarization activities help consolidate learning and encourage students to articulate their understanding in their own words.

For a more detailed description and implementation of these activities and strategies, you can refer to the attached document [link to the document] that provides step-by-step instructions, prompts, and additional resources for each activity.

TeachEngineering Associated Activities

Go to the TeachEngineering Activities website and find an activity that could be used in association with this proposed lesson. Write the name of the activity and a short 1-2 sentence description.
Example: What a Drag! Students investigate the forces of flight using paper helicopters they design and construct.

Building a Piezoelectric Generator — Activity
Grade Level: 9 (9-12)

Engineering Category: Partial design

Total Time: 45 minutes

Students learn how to build simple piezoelectric generators to power LEDs.

Closure

Congratulations, students, on completing this exciting journey into the world of code inputs, outputs, objects, and image processing! You've gained valuable knowledge and skills that will serve you well in the realm of computer science and engineering.

Now, let's bring it all together and reflect on what we've learned. Throughout this lesson, we explored the significance of code inputs and outputs in programming. We discovered how inputs provide instructions to a program, and outputs are the valuable results we obtain. By understanding this process, we can harness the power of programming to create innovative solutions and make a real impact in the engineering world.

We also delved into the fascinating realm of image processing. We learned about dehazing images using Python and OpenCV libraries, and how these techniques can enhance visibility and improve the quality of visuals. By mastering image processing techniques, you have acquired a valuable skillset that can be applied in various fields, from developing advanced surveillance systems to creating self-driving cars that can "see" and understand their surroundings.

Additionally, we explored the concept of objects in programming, specifically in the context of object-oriented programming (OOP). Objects allow us to represent real-world entities in a virtual space, enabling us to design and build complex systems efficiently. By grasping the fundamentals of OOP, you have unlocked a powerful approach that engineers and developers use to create robust and scalable software solutions.

Let's not forget the exciting "Battleship" game activity, where you collaborated, strategized, and applied your knowledge of code inputs, outputs, and objects. This interactive experience allowed you to put theory into practice and reinforced your problem-solving, teamwork, and critical thinking skills. Remember, collaboration and effective communication are essential components of successful engineering endeavors.

As you move forward in your learning journey, continue exploring the fascinating world of computer science, programming, and engineering. Stay curious, keep experimenting, and always seek opportunities to apply your skills in real-world contexts. Remember, the knowledge and skills you've acquired in this lesson are just the beginning of an exciting and rewarding adventure.

Now, take a moment to reflect on your learning, discuss with your peers, and ask any remaining questions you may have. Embrace the knowledge you've gained, and carry it with you as you embark on future projects and challenges. You are now equipped with a comprehensive introduction to Python programming, image processing, and problem-solving techniques.

Well done, students! Keep coding, exploring, and making a difference with your newfound skills. The possibilities are endless, and I can't wait to see the amazing things you will accomplish as future engineers and innovators.

Assessment

Pre-Assessment

To determine students' prior knowledge before starting the lesson, I would employ a pre-assessment to gauge their understanding of relevant concepts and skills. The pre-assessment can take various forms, such as a written quiz, a short online survey, or a class discussion. The assessment should focus on key areas that will be covered in the lesson, including programming concepts, image processing, and problem-solving techniques. Here are some sample items that could be included:

1. Programming Concepts:
   - What is the purpose of code inputs in a program?
   - Provide an example of a code output and explain its significance.
   - Describe the role of variables in programming.

2. Image Processing:
   - Define image processing in your own words.
   - Name a Python library commonly used for image processing.
   - What is the purpose of dehazing an image? Provide a brief explanation.

3. Problem-Solving Techniques:
   - Explain the steps involved in the problem-solving process.
   - Provide an example of a problem you have solved using a systematic approach.
   - How would you collaborate with a team to solve a complex problem?

By analyzing the responses from the pre-assessment, I can identify students' prior knowledge, misconceptions, and any gaps in understanding. This information will guide my instructional approach, allowing me to tailor the lesson to meet students' needs and build upon their existing knowledge. It will also help me identify areas that require additional clarification or reinforcement.

Please note that as an AI language model, I don't have the capability to attach specific assessments or documents. However, you can create a pre-assessment using the sample items provided or design your own assessment based on the learning objectives and content of the lesson.

Formative Assessment
During the learning process, I would employ various formative assessments to check for students' understanding and progress. These assessments can be conducted throughout the lesson and can take different formats, such as quizzes, discussions, hands-on activities, or individual/group assignments. Here are some examples of how I would check for understanding:

1. Concept Check Questions: Throughout the lesson, I would pause at key points and ask concept check questions to assess students' understanding. These questions can be posed verbally or through written prompts. Sample items include:

- What is the purpose of an input in programming? Provide an example.
- How would you dehaze an image using Python and OpenCV? Explain the steps.
- Describe the concept of object-oriented programming and give an example.

2. Hands-On Activity Reflection: After the hands-on activity where students dehaze images or work on the "Battleship" game, I would ask students to reflect on their experience and understanding. Sample prompts include:

- What challenges did you encounter during the image dehazing activity? How did you overcome them?
- In the "Battleship" game, how did you collaborate with your team to strategize and analyze the battleship board images?

3. Group Discussions: Organize group discussions where students can share their ideas, ask questions, and engage in peer-to-peer learning. Sample discussion prompts include:

- Explain the concept of code inputs and outputs to your group members.
- Discuss the advantages of using objects in programming compared to procedural programming.

4. Exit Tickets: Use brief exit tickets at the end of the lesson to assess students' comprehension of the main concepts covered. Sample questions include:

- Name two image processing techniques you learned in today's lesson.
- Write one benefit of using object-oriented programming in software development.

As an AI language model, I cannot attach specific assessments or documents. However, you can create your own formative assessments based on the examples provided or design assessments that align with the specific learning objectives and content of the lesson. The goal is to obtain ongoing feedback on students' understanding, identify areas that require clarification, and make necessary instructional adjustments to support their learning progress.

Summative Assessment
After the learning is complete, I would conduct a final check for understanding to assess students' overall comprehension of the key concepts and skills covered in the lesson. This assessment can take the form of a summative evaluation, such as a written quiz or project-based assignment. Here are some examples of how I would assess students' understanding:

1. Written Quiz: Design a written quiz that covers the main topics and learning objectives of the lesson. Sample items could include:

- Define code inputs and code outputs in programming.
- Explain the purpose of image dehazing and how it is achieved using Python and OpenCV.
- Describe the key principles of object-oriented programming and provide an example.

2. Project-Based Assignment: Assign a project where students are required to apply their knowledge and skills in a practical context. For example:

- Design a Python program that takes user inputs and performs a specific task, demonstrating your understanding of code inputs and outputs.
- Develop an image processing application using Python and OpenCV to enhance the quality of images.

3. Performance Assessment: Evaluate students' ability to apply their learning through a performance-based assessment. This can involve solving a programming problem or analyzing and manipulating images using Python and OpenCV.

Please note that as an AI language model, I cannot attach specific assessments or documents. However, you can create your own final check for understanding assessment based on the examples provided. The assessment should align with the learning objectives of the lesson and evaluate students' overall mastery of the concepts and skills covered.

The purpose of the final check for understanding is to consolidate students' learning and provide a comprehensive evaluation of their progress. It helps assess the effectiveness of the instruction, identifies any remaining misconceptions or areas for further reinforcement, and provides students with feedback on their overall performance.

Homework

Students will research one other open source python code in an area of their interest. They will turn in a summary of:

1) *What it is/does*
2) *Why they were interested*
3) *What they would need to do to outfit it for their computer.*

Contributors

Individuals

Alejandro Hinojos

Supporting Program

RET Site: Sensor, Signal and Information Processing Algorithms and Software

Sensor, Signal and Information Processing Center (SenSIP), in partnership with Arizona State University and the National Science Foundation.

Funding Acknowledgements

This project is funded by the National Science Foundation (NSF) Award 1953745. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the NSF.