UCI's CS1C@OC Computer Science Teacher Certificate Program
Backup Materials for CTC Application for
Computer Science Supplementary Authorization

Computer Science education is vital to the future of California and its students. CS is not just about being able to use computers, it is about innovation, collaboration and problem solving. CS education builds computational and critical thinking skills that allow students to create—not simply use—the next generation of applications, devices, and other computing technologies. We must ensure that all schools provide access to meaningful and sustainable teaching and learning opportunities in Computer Science, thereby giving every student the expectation to thrive in the 21st century. Every student should have this possibility, regardless of their ultimate field of study or work, but more importantly regardless of their gender, color, disability or economic status. However, there are not enough K-12 teachers in California prepared to teach quality computer science to the state's students.

To address this situation, the California Commission on Teacher Credentialing approved a Supplementary Authorization in Computer Science in 2016, replacing an outdated supplementary authorization on computer applications. Holders of single subject teaching credentials (other than Math, Business, Industrial and Technology Education, who are already authorized to teach Computer Science) are eligible to receive this supplementary authorization after receiving college units in computer science courses covering specified topics. For more detail, see CTC's Coded Correspondence 16-05 (tinyurl.com/CTC-CC16-05).

Through a special partnership with the Orange County Department of Education and funding from the National Science Foundation, the University of California—Irvine is offering a certificate program consisting of four courses intended to satisfy the requirements of the Supplementary Authorization in Computer Science. This certificate program is also helpful for teachers currently authorized to teach computer science but who would like to improve their classrooms, not only with respect to content, but also pedagogy addressing equitable and inclusive classrooms.

UCI's CS1C@OC Computer Science Teacher Certificate Program

Backup Materials for CTC Application for Computer Science Supplementary Authorization

Table of Contents

UCI's CS1C@OC Computer Science Teacher Certificate Program

EDUC X300.43, EDUC X300.44, EDUC X300.45, EDUC X300.46

This certificate program offers courses with evidence-based Computer Science instructional materials and assessment sets ready for implementation in the classroom. This two-year program (~eight months in the classroom over 16 months) consists of four hybrid (face-to-face and on-line) courses offered in Spring (April — early June) and early Summer (late June — early August) of consecutive years, on the following schedule:

Spring year 1	EDUC X300.43—Teaching Exploring Computer Science (4 units)
Summer year 1	EDUC X300.44—Teaching Computer Science Principles (5 units)
Spring year 2	EDUC X300.45—Methods for Teaching Computer Science (2 units)
Summer year 2	EDUC X300.46—Advanced Topics for CS Teachers (4 units)

A Professional Learning Community is offered for teacher participants during the Fall and Winter, when classes are not in session.

Further information is available at the program website: sites.uci.edu/cs1c.

CS1C@OC Program Director

Name: Debra J. Richardson
Email: djr@uci.edu
Website: www.ics.uci.edu/~djr

Dr. Debra J. Richardson is Professor of Informatics and founding dean of the UCIrvine's Donald Bren School of Information and Computer Sciences (ICS). Although her primary research contributions have been in Software Engineering, her more recent work has focused on Computer Science Education. In this context, Professor Richardson works on improving K-16 computer science education, specifically passionate about creating an environment in which *all* students—including female students, students of color, and students from under-resourced communities—have access to quality computer science education so that they might become the creators of the next computing technology and not just users.

Professor Richardson worked with California's Commission on Teacher Credentialing to establish the Supplementary Authorization in Computer Science, replacing an outdated authorization focused on computer applications. She is currently co-chairing the California Computer Science Strategic Implementation Plan Panel—a governor-appointed panel that is developing recommendations for a strategic plan to implement equitable K-12 Computer Science education throughout California. These recommendations will be provided to the Superintendent of Public Instruction, the State Board of Education, and the Legislature for development, adoption and implementation of the strategic plan.

A long-time advocate of increasing the participation of women and other underrepresented groups in computing, Professor Richardson has served on the leadership team of the National Center for Women and Information Technology (NCWIT) since its inception in 2004. She led the original NCWIT hub on undergraduate education, currently leads UCI's PaceSetter team, and recently led UCI's Extension Services team, which resulted in UCI receiving NCWIT's 2016 NEXT Award for Excellence in Promoting Women in Undergraduate Computing.

Richardson received her B.A. in Mathematics from the University of California—San Diego, and her M.S. and Ph.D. in Computer and Information Science at the University of Massachusetts—Amherst.

UCI's CS1C@OC Computer Science Teacher Certificate Program
Overview of Courses – Descriptions and Objectives

EDUC X300.43, EDUC X300.44, EDUC X300.45, EDUC X300.46
15 quarter units, ~8 months (30 weeks) spread over 16 months

The target audience for UCI's Computer Science Teacher Certificate Program is teachers looking to become better prepared to teach computer science, with a focus on two particular courses—Exploring Computer Science and Computer Science Principles. Moreover, the certificate program is designed to support credentialed teachers to receive California's Supplementary Authorization in Computer Science.

EDUC X300.43: Teaching Exploring Computer Science (4 units)
Year 1, Spring Quarter
Instructor: Nancy Se

Course Description:

Teaching Exploring Computer Science (ECS) is a professional development course, which is based on three major pillars— equity, inquiry, and computer science content/concepts— woven throughout the course and the Professional Learning Community. This Teaching ECS course introduces computer science content through various evidence-based instructional strategies: role playing, jigsaw-type collaborative activities, pair and small-group collaborative learning, structured tinkering, multiple solutions, and using manipulatives as well as simulations. While introducing the content and pedagogical knowledge, we will build a learning environment that encourages teachers to take autonomous roles and develop a growth mindset in the process. We also believe that ECS teachers and students are all part of a learning community where all members can contribute to the learning process in meaningful ways.

Course Objectives:

At the end of this course, teacher-participants (students) will be able to:

- Identify pedagogical content knowledge for teaching Exploring Computer Science
- Elaborate on strategies of integrating computational thinking in teaching and learning
- Create an equitable computer science classroom environment
- Design effective instruction for Exploring Computer Science
- Design student assessment for Exploring Computer Science
- Identify major components and functions of digital devices and computing systems
- Identify the impacts of computing on society
- Conduct information searching/validation using computers and communication platforms
- Develop web pages using HTML and CSS
- Develop functioning programs in Scratch to solve problems

EDUC X300.44: Teaching Computer Science Principles plus (5 units)

Instructors: Beth Simon and Art Lopez
Year 1, Summer Session I

Course Description:

Teaching Computer Science Principles (CSP) is a professional development course designed to support teachers in teaching the AP Computer Science Principles course or a similar computational thinking-based course. It is based in six Computational Practices: connecting computing, creating computational artifacts, abstracting, analyzing problems and artifacts, communicating, and collaborating. Additionally, the course focuses on seven Big Ideas: creativity, abstraction, data and information, algorithms, programming, the Internet, and global impact. This Teaching CSP course introduces computer science content through a variety of techniques including online videos (e.g. discussing student misconceptions, modeling classroom strategies, concept introductions, debugging advice examples, lesson plans overviews), guided engagement with lesson plans, scaffolded programming scrambles, online discussions, reflective writing, and guided lesson plan evaluation and peer review, in-person pair programming and peer instruction. While introducing the content and pedagogical knowledge, also focus on providing structure teachers can continue to use to build their computational thinking and programming skills as they teach the course. The course purports that CSP teachers and students are part of a learning community where all members contribute to the learning process in meaningful ways.

Course Objectives:

At the end of this course, students will be able to:

- Identify pedagogical content knowledge for teaching Computer Science Principles
- Elaborate on strategies of integrating computational thinking in teaching and learning
- Create an equitable computer science classroom environment
- Design effective instruction for Computer Science Principles
- Design student assessment and evaluation for the CS Principles Performance Tasks and written exam or similar activities
- Identify major components and functions of digital devices and computing systems???
- Identify the impacts of computing on society
- Develop computational artifacts using sequential execution, methods, parameters, events, mathematical expressions, functions, if statements, boolean expressions, loops and lists

EDUC X300.45: Methods for Teaching Computer Science (2 units)

Instructor: Nancy Se
Year 2, Spring Quarter

Course Overview:

Computer Science Teaching Methods and Student Assessment addresses pedagogical methods for teaching computer science at the high school level as well as methods to evaluate student learning. It is the capstone course in UCI's Computer Science Teacher Certificate Program, which targets providing secondary school teachers with the content and pedagogical background and experiences to be effective high school computer science teachers.

Course Objectives:

At the end of this course, teacher-participants (students) will be able to:

- Understand the K-12 Computer Science Framework and California K-12 Computer Science Standards enabling them to create and implement a curriculum to best meet expectations of this subject.
- Create lessons plans to engage diverse learners through culturally responsive pedagogy and analyze potential learning difficulties and adjust teaching for students with different needs.
- Create an equitable computer science classroom environment.
- Implement a variety of methods in the teaching process, including meaningful learning, collaborative learning, and inquiry-based learning.
- Create activities for students to solve specific problems and to document both their programming process and their debugging process.
- Develop detailed lesson plans for selective topics, consisting of goals and objectives, descriptions of activities and tasks, teaching methods, teaching aids, and evaluation.
- Develop various types of assessments (e.g., formative and summative, authentic, and performance assessments) and corresponding rubrics to evaluate student learning.
- Establish a bank of resources for teaching computer science in high school, including course materials, lab assignments, class activities, and websites.
- Keep up with research in computer science education and apply it to the teaching process and in the classroom.

EDUC X300.46: Advanced Topics for Computer Science Teachers (4 units)

Instructor: Daniel Frost
Year 2, Summer Session I

Course Overview:

This course covers topics in computer science that are more advanced than those *required* to teach Exploring Computer Science and Computer Science Principles, but enable teachers of these courses to feel confident that they know more than most of their students and can answer student questions that might arise in the classroom. In addition to learning higher level-programming in Python, teachers in the course will learn basic data structures and algorithms (including tradeoffs between data representation and algorithm performance, searching and sorting in the context of problem solving using computational tools), and software design (including the processes of planning, engineering and implementing a software system to solve larger-scale problems).

Course Objectives:

At the end of this course, teacher-participants (students) will understand:

- Python programming
  - Write, run, and debug a Python program
  - Understand fundamental concepts of program design in Python
  - Explain several differences between Python, Scratch, and other programming languages, and decide which language is more appropriate for a given task
- Software Design and Development
  - Identify and describe the phases of the software design/development process
  - Explain the role of a requirements or specification document
  - Develop use cases for a software system design
  - Design and carry out systematic unit testing of a software module
- Algorithms and data structures
  - Implement several basic data structures and algorithms in Python
  - Describe the trade-offs between alternative data structures and algorithms for a specific problem specification
- Additional advanced topics in computer science
  - Artificial Intelligence
  - Privacy and security
  - Image file formats, data compression
  - Databases, networks, Internet, cloud computing
  - Big data, visualizing data
  - Careers in computing
- How computational thinking (as described in a previous course in the sequence) is manifested in the advanced topics of this course
- How to design effective instruction for advanced topics in Computer Science

UCI's CS1C@OC Computer Science Teacher Certificate Program
Coverage of CS Supplementary Authorization Topics

Introductory Computer Science Supplementary Authorization

Computational thinking
involves solving problems and designing systems, using fundamental computing concepts such as decomposition, data representation, generalization/abstraction, and algorithms

Computing practice and programming
should include expertise in at least one block-based, visual drag-and-drop programming language e.g., Alice, Blockly, Kodu, Logo, Scratch, Snap! or a modern, high-level programming language

Computer and communications devices
should cover the major components and functions of digital devices and the computing systems they compose

Impacts of computing
includes the social, ethical, and legal issues and impacts of computing, as well as the contributions of computer science to current and future innovations in the arts, business, humanities, medicine, and science

Computer science methods/pedagogy
not required but recommended

Specific Computer Science Supplementary Authorization

Programming
should include expertise in at least one modern, high-level programming language—e.g., Python, Java, C++

Data structures and algorithms
covers data representation, abstraction, searching and sorting in the context of solving problems using programming and computational tools

Digital devices, systems and networks
should cover computer and communication devices and the systems they compose, including the concepts and abstractions that enable stand-alone, networked, and mobile digital devices to operate and communicate

Software design
refers to the process of planning, engineering and implementing a software system to solve a problem, typically using both a design and a programming methodology

Impacts of computing
includes the social, ethical, and legal issues and impacts of computing, as well as the contributions of computer science to current and future innovations in the arts, business, humanities, medicine, and science

Computer science methods/pedagogy
not required but recommended

From California Commission on Teacher Credentialing Coded Correspondence 16-05 (https://www.ctc.ca.gov/docs/default-source/commission/coded/2016/1605.pdf)

Combined Computer Science Supplementary Authorization Topics

T1: Computational thinking *(this should be included in both, even though only listed in introductory)*
involves solving problems and designing systems, using fundamental computing concepts such as decomposition, data representation, generalization/abstraction, and algorithms

----

T2i: Computing practice and programming
should include expertise in at least one block-based, visual drag-and-drop programming language e.g., Alice, Blockly, Kodu, Logo, Scratch, Snap! or a modern, high-level programming language

T2s: Programming
should include expertise in at least one modern, high-level programming language—e.g., Python, Java, C++

T3i: Computer and communications devices
should cover the major components and functions of digital devices and the computing systems they compose

T3s: Digital devices, systems and networks
should cover computer and communication devices and the systems they compose, including the concepts and abstractions that enable stand-alone, networked, and mobile digital devices to operate and communicate

----

T4: Impacts of computing
includes the social, ethical, and legal issues and impacts of computing, as well as the contributions of computer science to current and future innovations in the arts, business, humanities, medicine, and science

----

T5: Computer science methods/pedagogy
not required but recommended

Specific CS Supplementary Authorization Only Topics

T6s: Data structures and algorithms
covers data representation, abstraction, searching and sorting in the context of solving problems using programming and computational tools

----

T7s: Software design
refers to the process of planning, engineering and implementing a software system to solve a problem, typically using both a design and a programming methodology

Shading indicates topic coverage: *Dark shading ⇨ primary course and deep coverage;* *Medium shading ⇨ significant coverage;* *Light shading ⇨ supportive/ introductory coverage*		**UCI's CS1C@OC CS Teacher Certificate Program**			
		Year 1 = 9 quarter units		**Year 2 = 6 quarter units**	
		Teaching Exploring Computer Science (4 qtr units)	**Teaching Computer Science Principles (5 qtr units)**	**Methods for Teaching Computer Science (2 qtr units)**	**Advanced Topics for Comp Sci Teachers (4 qtr units)**
Computer Science Supplementary Authorization Topics (see acronyms for topics above)	**T1: CT**	▓	■	▒	▒
	T2i: CPP	■	■	▒	▓
	T2s: HLP			▒	■
	T3i: CCD	▓	▒		▒
	T3s: DSN		▒	░	▓
	T4: IoC	■	▓	▒	░
	T5: CSM	■	▓	■	
	T6s: DSA	▓	▓		■
	T7s: SD	▒	▒		■

Teaching Exploring Computer Science

EDUC X300.43
4 Units
first offering – Spring 2017

Class Meeting Information

Modality: Hybrid over nine weeks

- Face-to-Face, three times – week 1, 5, 9
- Online, including weekly synchronous online classes twice per week
  - Tool: Zoom Video Conference

Website: sites.uci.edu/cs1c/teaching-exploring-computer-science

Instructor Information

Name: Nancy Se
Email: nse@uci.edu
Website: www.nancyse.com

Nancy Se is an LAUSD teacher and facilitator for Exploring Computer Science. Nancy graduated from the University of California—Los Angeles with a degree in Biology and received her Masters in Education from the University of Southern California. She currently teaches Exploring Computer Science and Chemistry at the Critical Design and Gaming School. Nancy is also one of the founders of Code Hawk Camp, a free, week-long computer science camp in South Central Los Angeles. In 2016, Nancy was invited to speak with other leaders in the CS Education community at “June Tea” with former Secretary of Education, John King. Nancy and her students were also featured in an article in the August 2016 issue of Scientific American called, “The Coding Revolution” where Nancy is quoted saying “In my classes, the students and I aren’t just exploring an academic subject. We’re reinventing their sense of themselves in the face of very powerful cultural messages.”

Communication

Please note that general questions about the course (content questions, due date clarifications, etc.) should be posted via the general Q&A forum on our **course site** so that all students can benefit from the answer. To reach the instructor regarding questions that are personal in nature (such as a request for an extension due to a family emergency or a request for an incomplete grade), please email the program email address cs1catoc@uci.edu

Course Description

Teaching Exploring Computer Science (ECS) is a professional development course, which is based on three major pillars— equity, inquiry, and computer science content/concepts—woven throughout the course and the Professional Learning Community. This Teaching ECS course introduces computer science content through various evidence-based instructional strategies: role playing, jigsaw-type collaborative activities, pair and small-group collaborative learning, structured tinkering, multiple solutions, and using manipulatives as well as simulations. While introducing the content and pedagogical knowledge, we will build a learning environment that encourages teachers to take autonomous roles and develop a growth mindset in the process. We also believe that ECS teachers and students are all part of a learning community where all members can contribute to the learning process in meaningful ways.

The Teaching ECS course and professional development consists of:

- A 9-week hybrid course (face-to-face and on-line) that develops teachers' pedagogical content knowledge of the first four units in the ECS curriculum (exploringcs.org/curriculum);
- A Professional Learning Community with workshops during the academic year to focus on the remaining two units and pedagogy and implementation of all six units.

Prerequisites and Course Sequencing (Classes or Knowledge Required Before Taking This Course)

Teaching Exploring Computer Science is being offered as part of UCI's Computer Science Teacher Certificate Program, which is currently funded by the National Science Foundation. This course is the first in a sequence of four courses in the program.
Thus, there are no class/knowledge prerequisites to this course. All teacher-participants (aka students) are admitted through the certificate program.

Course Objectives

At the end of this course, teacher-participants (students) will be able to:

- Identify pedagogical content knowledge for teaching Exploring Computer Science
- Elaborate on strategies of integrating computational thinking in teaching and learning
- Create an equitable computer science classroom environment
- Design effective instruction for Exploring Computer Science
- Design student assessment for Exploring Computer Science
- Identify major components and functions of digital devices and computing systems
- Identify the impacts of computing on society
- Conduct information searching/validation using computers and communication platforms
- Develop web pages using HTML and CSS
- Develop functioning programs in Scratch to solve problems

Course Material / *Required Reading* (to be provided):

1. Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the Shallow End: Education, Race, and Computing*. MIT Press.

2. Exploring Computer Science Curriculum v.7, http://www.exploringcs.org/curriculum

3. Various readings on pedagogy and content will be provided on-line as PDF files.

Course Outline

Week 1 **On-line** **Introduction**	**Topics/Objectives:** Pedagogy through the strands of Equity, Inquiry and CS Concepts	**Key Topics:** ● Creating an equitable and inclusive learning environment
	Learning Activities:	● *Stuck in the Shallow End* Discussion, Preface and Introduction ● On-line class
	Assignments Due:	1. Read and come prepared to discuss *Stuck in the Shallow End*, Preface and Intro during on-line class 2. Read and come prepared to discuss *Stuck in the Shallow End*, Chapter 2, 3, and 4 by Saturday

Kick-Off Face-to-Face Saturday 9am—3pm **Unit 1**[1]	**Topics/Objectives:** Human Computer Interaction and Pedagogy	**Key Topics:** ● Computers and the Internet ● Models of Intelligent Behaviors ● Societal impact of computing ● Communicating with students ● Using questioning and discussion techniques ● Structures to engage students in learning **Learning Objectives:** ● Evaluate the results of web searches and the reliability of information found on the Internet ● Conduct information searching and validation using computers and communication platforms ● Explain the differences between tasks that can and cannot be accomplished with a computer ● Analyze the effects of computing on society within economic, social, and cultural contexts ● Communicate legal and ethical concerns raised by computing innovation ● Explain the implications of communication as data exchange ● Design effective instruction for Exploring Computer Science
	Learning Activities	● Model lesson from Human Computer Interaction Unit with debrief ● *Stuck in the Shallow End* Discussion, Chapter 2-4 ● Lesson Plan: Unit 1 ● Presentation of lesson plans with debrief
	Assignments Due	1. *Stuck in the Shallow End* Poster Jigsaw Presentations during face-to-face session (group posters for each chapter) 2. Submit lesson plan with assessments by the end of the session 3. Presentation of lesson plan during face-to-face session

[1] Unit numbers refer to the units in the Exploring Computer Science curriculum.

Week 2 **On-line** **Units 1&2**	**Topics/Objectives:** Problem Solving and Classroom Culture: Equitable Practices and Teaching Strategies	**Key Topics:** ● Algorithms and abstraction ● Connections between Mathematics and Computer Science ● Creating a classroom environment of respect and rapport ● Establishing a culture for learning **Learning Objectives:** ● Name and explain the steps used in solving a problem ● Solve a problem by applying appropriate problem-solving techniques ● Express a solution using standard design tools ● Determine if a given algorithm successfully solves a stated problem ● Create algorithms that meet specified objectives ● Create an equitable computer science classroom environment
	Learning Activities	● Growth Mindset video lecture ● Data collection, process for solving problems, number systems (binary/decimal), and search algorithms (linear and binary) video lecture ● Lesson plan: Unit 2 ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Lesson plan due by Monday 11:59pm 2. Post comments for peer's posting by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects

Week 3 **On-line** **Unit 2**	**Topics/Objectives:** Problem Solving and Classroom Culture: Equitable Practices and Teaching Strategies	**Key Topics:** ● Algorithms and abstraction ● Connections between Mathematics and Computer Science ● Creating a classroom environment of respect and rapport ● Establishing a culture for learning **Learning Objectives:** ● Identify the definition of computational thinking ● Apply computational thinking in problem solving ● Explain the connections between binary numbers and computers ● Summarize the behavior of an algorithm ● Compare the tradeoffs between algorithms for solving the same problem ● Design student assessment for Exploring Computer Science
	Learning Activities	● Strategies for the Computer Science Classroom video lecture ● Sorting algorithms, graphing theory and minimum spanning tree video lecture ● Lesson plan: Unit 2 ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Lesson plan due by Monday 11:59pm 2. Post comments for peer’s posting by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects.

Week 4 **On-line** **Unit 3**	**Topics/Objectives:** Web Design and Computational Practices	**Key Topics:** ● Web page design and development **Learning Objectives:** ● Create web pages to address specific objectives ● Create web pages with a practical, personal, and/or societal purpose ● Select appropriate techniques when creating web pages ● Use abstraction to separate style from content in web page design and development ● Describe the use of a website with appropriate documentation
	Learning Activities	● Societal impacts of the web, image editing, responsible use of digital content, and HTML/CSS video lecture ● Website Project ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Website project by Monday 11:59pm 2. Post comments for peer's Website project by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects.

Mid-Point Face-to-Face Saturday 9am-3pm **Unit 4**	**Topics/Objectives:** Introduction to Programming; Scratch Intro	**Key Topics:** ● Understand the Scratch programming environment and animate sprites to move ● Communicate computation thought processes, procedures, and results to others **Learning Objectives:** ● Use appropriate algorithms to solve a problem ● Design, code, test, and execute a program that corresponds to a set of specification ● Select appropriate programming structures ● Locate and correct errors in a program ● Explain how a particular program functions ● Justify the correctness of a program ● Create programs with practical, personal, and/or societal intent
	Learning Activities	● Website Gallery Walk ● Model lessons from Scratch Unit with debrief ● Lesson Plan: Unit 4 ● Presentations of lesson and debrief
	Assignments Due	1. Presentation of website with code 2. Submit draft lesson plan for Scratch Unit by the end of the session 3. Presentation of lesson plan by the end of session

Week 5 **On-line** **Unit 4**	**Topics/Objectives:** Introduction to Programming; Scratch Animation	**Key Topics:** ● Apply broadcast and receive to programs to require sequences or take inputs. **Learning Objectives:** ● Use appropriate algorithms to solve a problem ● Design, code, test, and execute a program that corresponds to a set of specification ● Select appropriate programming structures ● Locate and correct errors in a program ● Explain how a particular program functions ● Justify the correctness of a program ● Create programs with practical, personal, and/or societal intent
	Learning Activities	● Event Driven Programming and Scratch Editor video lecture ● Scratch Project: Animation ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Scratch project by Monday 11:59pm 2. Post comments for peer’s Scratch project by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects

Week 6 **On-line** **Unit 4**	**Topics/Objectives:** Introduction to Programming; Scratch Variables and Expressions	**Key Topics:** ● Create and manipulate variables ● Construct logical expressions to evaluate given conditions **Learning Objectives:** ● Use appropriate algorithms to solve a problem ● Design, code, test, and execute a program that corresponds to a set of specification ● Select appropriate programming structures ● Locate and correct errors in a program ● Explain how a particular program functions ● Justify the correctness of a program ● Create programs with practical, personal, and/or societal intent
	Learning Activities	● Data, variables, conditionals, operators, and sensing blocks video lecture ● Scratch Project: Guess the Number ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Scratch project by Monday 11:59pm 2. Post comments for peer's Scratch project by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects

Week 7 **On-line** **Unit 4**	**Topics/Objectives:** Introduction to Programming; Scratch Conditionals Race and Class	**Key Topics:** ● Use if and if/else blocks to choose among alternative actions and execute an action. **Learning Objectives:** ● Use appropriate algorithms to solve a problem ● Design, code, test, and execute a program that corresponds to a set of specification ● Select appropriate programming structures ● Locate and correct errors in a program ● Explain how a particular program functions ● Justify the correctness of a program ● Create programs with practical, personal, and/or societal intent
	Learning Activities	● *(Race)ing to Class* reading and short reflection ● Facilitating Classroom Discussions video and reflection ● Group Lesson Plan for Final Presentation ● On-line class ● On-line Community Participation
	Assignments Due	1. Post *(Race)ing to Class* reflection due by Monday 11:59pm 2. Post comments for peer's posting by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos in on-line class and use video lectures to help create lesson plans and complete projects

Week 8 **On-line** **Unit 4**	**Topics/Objectives:** Introduction to Programming; Conditionals Race and Culture	**Key Topics:** ● Use if and if/else blocks to choose among alternative actions and execute an action **Learning Objectives:** ● Use appropriate algorithms to solve a problem ● Design, code, test, and execute a program that corresponds to a set of specification ● Select appropriate programming structures ● Locate and correct errors in a program ● Explain how a particular program functions ● Justify the correctness of a program ● Create programs with practical, personal, and/or societal intent
	Learning Activities	● *Why Race and Culture Matter in Schools* reading ● Name Story Project ● Scratch Project: Game ● Group Lesson Plan for Final Presentation ● On-line class ● On-line Community Participation
	Assignments Due	1. Post Scratch project by Monday 11:59pm 2. Post comments for peer's Scratch project by Tuesday 11:59pm 3. Be prepared to share thoughts on pedagogy reading/videos and Name Story during on-line class

Final **Face-to-Face** **Saturday** **9am—3pm**	**Topics/Objectives:** Lesson Presentations	**Key Topics:** ● Practice teaching ECS with peer feedback ● Introduction to Units 5 and 6 **Learning Objectives:** ● Demonstrate teaching effectiveness through micro-teaching
	Learning Activities	● Scratch Project Gallery Walk ● Teaching Presentations
	Assignments Due	1. Submit the lesson plan used in microteaching by the end of the session 2. Microteaching Demonstration 3. Submit the provided peer and self-evaluation document for presentations by the end of the session

Week 9 **On-line** **Final**	**Topics/Objectives:** CS Education and Equity	**Key Topics:** ● Identify pedagogical content knowledge for teaching Exploring Computer Science ● Elaborate on strategies of integrating computational thinking in teaching and learning ● Create an equitable computer science classroom environment ● Design effective instruction for Exploring Computer Science ● Design student assessment for Exploring Computer Science
	Learning Activities	N/A
	Assignments Due	1. Submit final paper on Computer Science Education and Equity by Friday at 11:59PM 2. Submit Exploring Computer Science Portfolio, including all reflections and lesson plans developed throughout the course updated with any revisions based on feedback provided by instructors and colleagues by Friday at 11:59PM

Evaluation and Grading

Evaluation of Student Performance Weighted as Percentages of the Total Grade

15%	Reading/video reflections
15%	Lesson plans
20%	Computing artifacts—i.e., Website and Scratch projects
20%	Final lesson plan and teaching demonstration
10%	Final Paper on CS Education and Equity
20%	Participation in on-line professional learning community (including peer to peer comments on posts, on-line discussion, and face-to-face meetings)

To receive full points on your reading/video reflections, lesson plans, and computing artifacts you will need to post substantive responses for the assignment by 11:59pm PST on the Monday of each week and provide critical feedback for one of your peer's submissions by 11:59pm PST on the Tuesday of each week. Although you are not expected to meet a specific word count, your submissions should be well thought out and make interesting contributions to the conversation. Simply posting a link to a resource or letting your classmate know that he/she did a "Great job!" will not result in full points. When crafting an original and meaningful response please either expand the thought, add additional insights, or respectfully disagree and explain why.

Earning participation points: Each week, there will be opportunity for students to engage with their classmates in synchronous on-line classes. Likewise, students are expected to attend and participate in discussions in face-to-face classes. Participation in these discussions are required and count as 20 % of your final grade, spread out over the performance items above. You will have an opportunity to earn up to 2 points for your participation in weekly discussion forums. The rubric below demonstrates how these points are assigned:

- 0 points No activity; did not respond to the forum and/or did not respond by the listed deadlines
- 1 point Provided feedback to classmates but did not craft an original response to the prompt
- 2 points Crafted an original and meaningful response

Grading Scale:
Students can choose to receive a letter grade or Pass/Not Pass.

If a letter grade is selected:

A	=	90% – 100%
B	=	80% – 89%
C	=	70% – 79%
D	=	60% – 69%
F	=	59% or less

If P/NP is selected:

P >=	70%
NP <	70%

Attendance

All participants in the course are expected to attend all face-to-face classes and participate in synchronous on-line classes. Attendance is important for active participation in a professional learning community. It is also important for receiving instruction, information, and feedback for the computing artifacts, lesson plans, etc. A limited number of excused attendances may be tolerated with permission.

Code of Conduct

All participants in the course are bound by the University of California Code of Conduct, found at http://www.ucop.edu/ethics-compliance-audit-services/_files/stmt-stds-ethics.pdf

Netiquette

In an on-line course, the majority of our communication takes place in the course forums. However, when we have a need for communication that is private, whether personal, interpersonal, or professional, we will use individual email or telephone. Our primary means of communication is written. The written language has many advantages: more opportunity for reasoned thought, more ability to go in-depth, and more time to think through an issue before posting a comment. However, written communication also has certain disadvantages, such a lack of the face-to-face signaling that occurs through body language, intonation, pausing, facial expressions, and gestures. As a result, please be aware of the possibility of miscommunication and compose your comments in a positive, supportive, and constructive manner.

Academic Honesty Policy

The University is an institution of learning, research, and scholarship predicated on the existence of an environment of honesty and integrity. As members of the academic community, faculty, students, and administrative officials share responsibility for maintaining this environment. It is essential that all members of the academic community subscribe to the ideal of academic honesty and integrity and accept individual responsibility for their work. Academic dishonesty is unacceptable and will not be tolerated at the University of California, Irvine. Cheating, forgery, dishonest conduct, plagiarism, and collusion in dishonest activities erode the University's educational, research, and social roles.

Students who knowingly or intentionally conduct or help another student engage in dishonest conduct, acts of cheating, or plagiarism will be subject to disciplinary action at the discretion of UCI Division of Continuing Education.

Disability Services

If you need support or assistance because of a disability, you may be eligible for accommodations or services through the Disability Service Center at UC Irvine. Please contact the DSC directly at (949) 824-7494 or TDD (949) 824-6272. You can also visit the DSC's website: http://www.disability.uci.edu/. The DSC will work with your instructor to make any necessary accommodations. Please note that it is your responsibility to initiate this process with the DSC.

Teaching Computer Science Principles
EDUC X300.44

5 Units

first offering – Summer 2017

Class Meeting Information

Modality: Hybrid over five weeks

- Face-to-Face once per week
- Online, including weekly synchronous online classes twice per week
  - Tool: Zoom Video Conference

Website: sites.uci.edu/cs1c/teaching-computer-science-principles

Instructor Information

Name:	Beth Simon
Email:	bsimon@ucsd.edu
Website:	https://sites.google.com/ucsd.edu/bsimon

Note: general questions about the course (content questions, due date clarifications, etc.) should be posted via the general Q&A forum on our **course site** via the Learning Management System (LMS) so that all students can benefit from answers. To reach the instructor regarding personal questions (such as a request for an extension due to a family emergency or a request for an incomplete grade), please email cs1catoc@uci.edu

Dr. Beth Simon is an Associate Teaching Professor in the Department of Education Studies at UC San Diego. Her research interests lie in the areas of computing education and online and technology-enhanced teaching. Beth is currently involved in development of high school computing curriculum and the training and community needs of K-12 teachers wanting to bring computing education opportunities to their students. Previously, Beth has studied the impacts of evidence-based active learning practices (e.g., Peer Instruction) on student outcomes in higher education, student conceptions of computing concepts, and novice computing students' self-efficacy. Beth was one of the initial university pilot instructors for the AP CS Principles course. Beth currently serves as Faculty Advisor for Technology-Enhanced Teaching for UCSD's Teaching and Learning Commons, where she supports faculty and instructional staff in the use of technology to support their educational efforts both on-campus and through MOOCs (Massive Open Online Courses).

Beth formerly served as a Teaching Professor in the Computer Science and Engineering Department and as Director of UCSD's Center for Teaching Development (now part of the Teaching and Learning Commons). From 2014-2015, Beth served as the Principal Teaching and Learning Specialist at Coursera, supporting faculty in development of MOOCs and advising on pedagogical platform development. During 2007-2008, Beth served as a Science Teaching and Learning Fellow in the Carl Wieman Science Education Initiative at the University of British Columbia.

Course Description

Teaching Computer Science Principles (CSP) is a professional development course designed to support teachers in teaching the AP Computer Science Principles course or a similar computational thinking-based course. It is based on six *Computational Practices*: connecting computing, creating computational artifacts, abstracting, analyzing problems and artifacts, communicating, and collaborating. Additionally, the course focuses on seven *Big Ideas*: creativity, abstraction, data and information, algorithms, programming, the Internet, and global impact. This Teaching CSP course introduces computer science content through a variety of techniques including online videos (e.g. discussing student misconceptions, modeling classroom strategies, concept introductions, debugging advice examples, lesson plans overviews), guided engagement with lesson plans, scaffolded programming scrambles, online discussions, reflective writing, and guided lesson plan evaluation and peer review, in-person pair programming and peer instruction. While introducing the content and pedagogical knowledge, also focus on providing structure teachers can continue to use to build their computational thinking and programming skills as they teach the course. The course purports that CSP teachers and students are part of a learning community where all members contribute to the learning process in meaningful ways.

The Teaching CSP course and professional development consists of:

- A 5-week hybrid course (face-to-face and online) that develops teachers' pedagogical content knowledge and content knowledge for the AP CSP curriculum.
- A Professional Learning Community with workshops during the academic year to support teachers as they implement the curriculum for the first time.

Prerequisites — Classes or Knowledge Required Before Taking This Course

Teaching CSP is the second course in UCI's Computer Science Teacher Certificate Program, which is currently funded by the National Science Foundation. As such, it is expected that all participants will have completed the first course in the program, Teaching ECS. All students must be in-service teachers and admitted through that program.

Course Objectives

At the end of this course, students will be able to:

- Identify pedagogical content knowledge for teaching Computer Science Principles
- Elaborate on strategies of integrating computational thinking in teaching and learning
- Create an equitable computer science classroom environment
- Design effective instruction for Computer Science Principles
- Design student assessment and evaluation for the CS Principles Performance Tasks and written exam or similar activities
- Identify major components and functions of digital devices and computing systems???
- Identify the impacts of computing on society
- Develop computational artifacts using sequential execution, methods, parameters, events, mathematical expressions, functions, if statements, boolean expressions, loops and lists

Course Material

All course materials will be provided in the online LMS.

Course Outline

Week 1 **Online**	**Topics/Objectives:** Peer Instruction, Creating a Supportive Classroom, Global Impacts, Programming Proficiency, Comparing Curriculum	**Key Topics:** ● Structures to engage students in deep learning ● Unconscious Bias ● Social, ethical and legal issues and impacts of computing ● Programming Concepts: sequential/parallel execution, methods, parameters **Learning Objectives:** ● Analyze the value of computational thinking for all students ● Review research on Peer Instruction and review materials for high school classrooms ● Discuss modifications of classroom practices for reducing unconscious bias ● Implement programs using methods, parameters, sequential execution, repetition ● Experience a scaffolded programming introduction ● Identify, summarize, and critique a TED talk on technology and society
	Learning Activities:	● Peer review assignment: Brainstorming access issues in your school (30 min) ● Programming assignments (10 hours) ● Peer Instruction videos and readings (1 hour) ● Classroom Resource Development (2 hours) ● Unconscious Bias videos and discussion (1 hour) ● Classroom Design Checklist (30 min) ● Exploration, summary and reflection, and discussion on technology and society (1.5 hours) ● Synchronous online class (1 hour)
	Assignments Due:	1. Brainstorming access list submission and review of three peers' submissions 2. Two online discussion forum posts and response to two peers' posts 3. Classroom Design Checklist 4. Five Alice programs

Week 1 **Face-to-Face**	**Topics/Objectives:** AP CS Principles Curriculum Framework and Course Variants Peer Instruction, Creating a Supportive Classroom, Programming Proficiency	**Key Topics:** ● AP CS Principles Curriculum Framework and Course Variants ● Peer Instruction ● Unconscious Bias ● Programming Concepts: methods, parameters **Learning Objectives:** ● Experience Peer Instruction as a student ● Observe a teacher doing Peer Instruction ● Use Peer Instruction sentence starters ● Create a revised unconscious bias checklist ● Contrast programming instruction designs
	Learning Activities:	● Overview of AP CS Principles Framework and exploration of APCSP course variants (45 min) ● Peer Instruction student experience (30 min) ● Peer Instruction debrief (30 min) ● Unconscious Bias checklist Think/Pair/Share + update (45 min) ● Programming Concepts Q&A (30 min) ● Think/Pair/Share contrasting programming instruction designs (60 min)
	Assignments Due:	1. Revised Bias Checklist 2. Comparison/Contrast worksheet on Programming Instruction Design

Week 2 **Online**	**Topics/Objectives:** Programming Proficiency, Explore Performance Task, Internet, Data	**Key Topics:** ● Internet, Data and Information ● The APCSP Performance Task Framework: Explore and Create ● Introduction to APCSP Explore Task ● Programming Concepts: events, mathematical expressions, functions, variables **Learning Objectives:** ● Explain how information is routed via the Internet ● Experience various types of lessons for teaching cybersecurity, encoding, and encryption ● Perform the APCSP Explore Task ● Apply the Explore Task Rubric to sample student work ● Create programs and functions which use basic Booleans expressions and mathematical expressions
	Learning Activities:	● Internet Online activities & reading (2 hours) ● Encryption activity review, critique and modification (2 hours) ● Read, perform, and submit APCSP Explore Performance Task (8 hours) ● Programming assignments (8 hours) ● Synchronous online class (1 hour)
	Assignments Due:	1. Two submissions to Internet activity lessons 2. Lesson modification and review of two peers' lessons 3. Submission of the APCSP Explore Task (written responses + computational artifact) 4. Three Alice programs

Week 2 **Face-to-Face**	**Topics/Objectives:** Explore Performance Task, Developing Programming Proficiency	**Key Topics:** ● APCSP Performance Task: Explore ● Programming Concepts: events, mathematical expressions, functions, and variable **Learning Objectives:** ● Evaluate APCSP Explore Performance Task ● Deepen understanding of common student misconceptions around expressions, functions, and variables ● Critique a Peer Instruction session
	Learning Activities:	● Explore programming misconceptions through Peer Instruction (1 hour) ● Identify and evaluate resources for supporting Explore Performance Task (1 hour) ● Create personalized lesson Plan for Explore Performance Task (2 hours)
	Assignments Due:	1. Collaboratively produce list of resources for Explore Performance Task 2. Lesson plan for Explore Performance Task preparation

Week 3 **Online**	**Topics/Objectives:** Pair Programming, Scaffolded Problem Solving, Developing Programming Proficiency, Curriculum Planning	**Key Topics:** ● Pair Programming and running a programming lab ● Scaffolded learning of problem solving ● Programming Concepts: if statements, compound Boolean expressions, nested if statements ● Curriculum planning ● Computer Science pencil & paper exams **Learning Objectives:** ● Plan and critique approaches for implementing Pair Programming and lab time ● Explore Analyze-Level options for developing programming knowledge and skills ● Create programs using if statements, nested if statements, and compound Boolean expressions ● Evaluate Course Planning & Pacing Guide ● Demonstrate to and guide students in using best practices for pencil & paper computing exams
	Learning Activities:	● Peer Review: APCSP Course Planning and Pacing Guide (3 hours) ● Videos, online discussion, and draft of plans for implementing Pair Programming (3 hours) ● Explore & reflect on programming scrambles to scaffold development of programming skills (2 hours) ● Create programs using if statements, nested if statements, and compound Boolean expressions (8 hours) ● Videos & peer review showing application of test taking recommended practices (1 hour) ● Synchronous online class (1 hour)
	Assignments Due:	1. Evaluation of the Course Planning and Pacing Guide and review of two peers' evaluations 2. Online discussion post and response to two peers' posts 3. Analysis of the Programming Scrambles experience and review of two peers' analyses 4. Two Alice programs 5. Submission of three exam questions using best test taking practices

Week 3 **Face-to-Face**	**Topics/Objectives:** Pair Programming, Scaffolded Problem Solving, Developing Programming Proficiency	**Key Topics:** ● Pair Programming and running a programming lab ● Scaffolded learning of Problem Solving ● Programming Concepts: of Statements, compound Boolean expressions, nested if statements **Learning Objectives:** ● Experience Pair Programming ● Reflect on challenges of implementing Pair Programming ● Discuss differences in learning to program by creating vs. analyzing ● Deepen understanding of common student misconceptions around if statements, nested if statements and compound Boolean expressions
	Learning Activities:	● Experience Pair Programming and reflect (1 hour) ● Think/Pair/Share differences in programming by scramble and programming from scratch (40 min) ● Discuss assessment options using Programming Scrambles (20 min) ● Peer Instruction implementation practice and feedback (2 hours)
	Assignments Due:	1. Lesson plan for using Programming Scrambles 2. Teaching Demonstration: Peer Instruction

Week 4 **Online**	**Topics/Objectives:** Computing Education Leadership, Create Performance Task, Developing Programming Proficiency	**Key Topics:** ● Developing a computing education culture in your schools and districts ● Introduction to APCSP Performance Task: Create ● Programming Concepts: loops, lists **Learning Objectives:** ● Explore approaches for leading computing education in your school ● Understand the APCSP Create Performance Task requirements ● Create programs using loops and lists ● Implement Curriculum Design for your class (e.g. Canvas or other)
	Learning Activities	● Reading and discussion of APCSP Create Performance Task Requirements (2 hours) ● Quiz of applying the APCSP Rubric (1 hour) ● Videos, readings, and web exploration for peer review of computing education leadership goals and concerns at your school (2 hours) ● Create programs using loops and lists (8 hours) ● Fill in first two months of curriculum platform for your class (4 hours) ● Synchronous online class (1 hour)
	Assignments Due	1. Online discussion post and response to two peers' posts 2. Online quiz applying the APCSP Create Rubric 3. Outline of leadership goals and concerns and review of two peers' outlines 4. Three Alice programs 5. PT Create Submission (1 video, 4 written responses)

Week 4 **Face-to-Face**	**Topics/Objectives:** Computing Education Leadership, Create Performance Task, Developing Programming Proficiency, Curriculum Planning	**Key Topics:** ● Developing a computing education culture in your schools and districts ● APCSP Performance Task: Create ● Programming Concepts: loops, lists ● Curriculum planning **Learning Objectives:** ● Understand the various components of developing a culture of computing education in your environment ● Understand the requirements of the APCSP Create Performance Task and Evaluation Rubric ● Deepen understanding of common student misconceptions around loops and lists ● Reflect on challenges of curriculum planning
	Learning Activities:	● Structured sharing and discussion on computing education leadership (guest speaker) (1 hour) ● Question and answer period on APCSP Create Performance Task (30 min) ● Collaborative craft of Create Performance Task preparation activities (30 min) ● Peer Instruction implementation practice and feedback (2 hours)
	Assignments Due:	1. Collaboratively created leadership activity planner 2. Lesson plan draft for Create Performance Task preparation 3. Teaching Demonstration: Peer Instruction

Week 5 **Online**	**Topics/Objectives:** Create Performance Task, Curriculum Planning, Exam Preparation Approaches	**Key Topics:** ● APCSP Create Performance Task ● Curriculum planning & pacing review ● Exam Techniques for Students **Learning Objectives:** ● Experience the APCSP Create Performance Task and submission process ● Implement curriculum design for your class (e.g. Canvas or other) ● Demonstrate to and guide students in using best practices for pencil & paper computing exams
	Learning Activities:	● Performing a guided practice for the APCSP Create Performance Task (Lyft) (6 hours) ● Videos & peer review showing application of test taking recommended practices (1 hour) ● Fill in scaffold materials for remaining months of curriculum platform for your class (6 hours) ● Synchronous online class (1 hour)
	Assignments Due:	1. Video demonstration (screen capture) of guiding students through a pencil and paper exam question 2. Identify and describe algorithms and abstraction from a program

Week 5 **Face-to-Face**	**Topics/Objectives:** Create Performance Task, Curriculum Planning, Professional Learning Community	**Key Topics:** ● APCSP Create Performance Task ● Curriculum planning ● Professional Learning Community **Learning Objectives:** ● Prepare for supporting APCSP Create Performance Task ● Share planned practices from Curriculum Planning ● Prepare for PLC engagement throughout year
	Learning Activities:	● Discuss and co-edit APCSP Create Performance Task review, rubric revision, and student checklist/guide (1 hour) ● Peer instruction implementation practice and feedback (2 hours) ● Curriculum Planning share out (1 hour) ● PLC Walkthrough (1 hour)
	Assignments Due:	1. Teaching Demonstration: Peer Instruction

Evaluation and Grading

Evaluation of Student Performance Weighted as Percentages of the Total Grade

15%	F2F meetings attendance and participation
10%	Weekly synchronous online attendance and participation
30%	Computing artifacts and programming related quizzes
20%	Online material usage/completion
20%	Online engagement activities (discussion prompts, peer review assignments)
10%	Peer Instruction Sample Teaching Session

Categories of Points:
You will find that points for online activities fall into two general types: points for correctness and points for effortful participation. Points for correctness are given for activities where there are specific correct/incorrect answers (e.g. quizzes, prescribed programming assignments). However a lot of online activities (participation in discussion prompts, submitting peer review assignments and reviewing the work of others, contributing to googledoc-based resource creation, etc.) are graded for effortful participation. Effortful participation means you engaged with the activity in a meaningful way (e.g. not just responding "Great job!" in replying to a discussion response, completing peer reviews on time, and having roughly equal participation in contributing to googledoc resource creation (graphically viewable via the Chrome Add One Docuviz). Points for F2F meetings are given for effortful participation.

Grading Scale
Students have can choose to receive a letter grade or Pass/Not Pass.

If a letter grade is selected:

A	=	90% – 100%
B	=	80% – 89%
C	=	70% – 79%
D	=	60% – 69%
F	=	59% or less

If P/NP is selected:

P >= 70%
NP < 70%

Code of Conduct

All participants in the course are bound by the University of California Code of Conduct, found at http://www.ucop.edu/ethics-compliance-audit-services/_files/stmt-stds-ethics.pdf

Netiquette

In an online course, the majority of our communication takes place in the course forums. However, when we have a need for communication that is private, whether personal, interpersonal, or professional, we will use individual email or telephone. Our primary means of communication is written. The written language has many advantages: more opportunity for reasoned thought, more ability to go in-depth, and more time to think through an issue before posting a comment. However, written communication also has certain disadvantages, such a lack of the face-to-face signaling that occurs through body language, intonation, pausing, facial expressions, and gestures. As a result, please be aware of the possibility of miscommunication and compose your comments in a positive, supportive, and constructive manner.

Academic Honesty Policy

The University is an institution of learning, research, and scholarship predicated on the existence of an environment of honesty and integrity. As members of the academic community, faculty, students, and administrative officials share responsibility for maintaining this environment. It is essential that all members of the academic community subscribe to the ideal of academic honesty and integrity and accept individual responsibility for their work. Academic dishonesty is unacceptable and will not be tolerated at the University of California, Irvine. Cheating, forgery, dishonest conduct, plagiarism, and collusion in dishonest activities erode the University's educational, research, and social roles.

Students who knowingly or intentionally conduct or help another student engage in dishonest conduct, acts of cheating, or plagiarism will be subject to disciplinary action at the discretion of UCI Division of Continuing Education.

Disability Services

If you need support or assistance because of a disability, you may be eligible for accommodations or services through the Disability Service Center at UC Irvine. Please contact the DSC directly at (949) 824-7494 or TDD (949) 824-6272. You can also visit the DSC’s website: http://www.disability.uci.edu/. The DSC will work with your instructor to make any necessary accommodations. Please note that it is your responsibility to initiate this process with the DSC.

Methods for Teaching Computer Science

EDUC X300.45

2 Units

first offering—Spring 2018

Class Meeting Information

Modality: Hybrid over nine weeks

- Face-to-Face, three times – week 1, 5, 9
- Online, including weekly synchronous online classes once per week
  - Tool: Zoom Video Conference

Website: sites.uci.edu/cs1c/methods-for-teaching-computer-science/

Instructor Information

Name: Nancy Se
Email: nancyse@gmail.com, nse@uci.edu
Website: www.nancyse.com

Nancy Se is an LAUSD teacher and facilitator for Exploring Computer Science. Nancy graduated from the University of California—Los Angeles with a degree in Biology and received her Masters in Education from the University of Southern California. She currently teaches Exploring Computer Science and Chemistry at the Critical Design and Gaming School. Nancy is also one of the founders of Code Hawk Camp, a free, week-long computer science camp in South Central Los Angeles. In 2016, Nancy was invited to speak with other leaders in the CS Education community at “June Tea” with former Secretary of Education, John King. Nancy and her students were also featured in an article in the August 2016 issue of Scientific American called, “The Coding Revolution” where Nancy is quoted saying “In my classes, the students and I aren’t just exploring an academic subject. We’re reinventing their sense of themselves in the face of very powerful cultural messages.”

Communication

Please note that general questions about the course (content questions, due date clarifications, etc.) should be posted via the general Q&A forum on our **course site** so that all students can benefit from the answer. To reach the instructor regarding questions that are personal in nature (such as a request for an extension due to a family emergency or a request for an incomplete grade), please email the program email address cs1catoc@uci.edu

Course Description

This course is designed to support computer science teachers in understanding computer science concepts at a level where they are comfortable developing ways to represent and formulate the same concepts for their own students. The course will explore effective strategies for teaching and retaining students who are traditionally marginalized in computer science. The course begins with understanding the CS Framework and California Computer Science Standards so teachers are informed by what leaders in this community hope students will know and be able to do by the end of their K-12 education. Additionally, by understanding computational thinking, teachers will be equipped to develop engaging and rigorous learning experiences that will prepare all of their students to face the problems in our technology rich world. Lastly, teachers will have created both formative and summative assessments to measure student progress and use that information to inform next steps with individual students.

Prerequisites — Classes or Knowledge Required Before Taking This Course

Methods for Teaching Computer Science is being offered as part of UCI's Computer Science Teacher Certificate Program, which is currently funded by the National Science Foundation. This course is the third in a sequence of four courses in the program. Teachers are expected to have successfully completed the first two courses, offered in the previous year (Spring and Summer):

1. Teaching Exploring Computer Science (4 units)
2. Teaching Computer Science Principles (5 units)

All teacher-participants (aka students) are admitted through the certificate program.

Course Objectives

At the end of this course, teacher-participants (students) will be able to:

- Understand the K-12 Computer Science Framework and California K-12 Computer Science Standards enabling them to create and implement a curriculum to best meet expectations of this subject.
- Create lessons plans to engage diverse learners through culturally responsive pedagogy and analyze potential learning difficulties and adjust teaching for students with different needs.
- Create an equitable computer science classroom environment.
- Implement a variety of methods in the teaching process, including meaningful learning, collaborative learning, and inquiry-based learning.
- Create activities for students to solve specific problems and to document both their programming process and their debugging process.
- Develop detailed lesson plans for selective topics, consisting of goals and objectives, descriptions of activities and tasks, teaching methods, teaching aids, and evaluation.
- Develop various types of assessments (e.g., formative and summative, authentic, and performance assessments) and corresponding rubrics to evaluate student learning.
- Establish a bank of resources for teaching computer science in high school, including course materials, lab assignments, class activities, and websites.
- Keep up with research in computer science education and apply it to the teaching process and in the classroom.

Course Materials / *Readings* *(to be provided):*

1. Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting*, Vancouver, BC, Canada, 1–25.

2. Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43.

3. McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, 18(2), 67-92.

4. Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students?. *Journal of Research on Technology in Education*, 46(3), 277-296.

5. Margolis, J., Ryoo, J. J., Moreno Sandoval, C. D., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *Inroads*, 3(4), 72–78.

6. Brown, J. C., & Crippen, K. J. (2017). The Knowledge and Practices of High School Science Teachers in Pursuit of Cultural Responsiveness. Science Education, 101(1), 99–133.

7. Hazzan, O., Lapidot, T., & Ragonis, N. (2011). *Guide to Teaching Computer Science*: An Activity-Based Approach. Chapter 10 Assessment, 187-205. Spring Verlag.

8. Google resources on Computational Thinking:
   https://edu.google.com/resources/programs/exploring-computational-thinking/
   https://computationalthinkingcourse.withgoogle.com/

Course Outline

Kick-Off Face-to-Face Saturday 9am—12pm **The K-12 CS Education Stage**	**Topics:** Understanding the Stage: CS Framework and California Computer Science Standard	**Learning Objectives:** Understand the CS Framework and California Computer Science Standards in order to create and implement a curriculum to best meet expectations of this subject.
	Learning Activities:	In Class: • Website Scavenger Hunt • Lesson Plan and Presentation

Week 1 Online **Diversity vs. Equity**	**Topic(s):** Diversity vs Equity in Computer Science: Where are we and how did we get here? Explore what's happening in CS Education, what continues to keep students out and what we can do about it.	**Learning Objectives:** Create an equitable computer science classroom environment. Establish a bank of resources for teaching computer science in high school, including possible materials, lab assignments, class activities, and websites.
	Learning Activities:	Before Class: • Read: Beyond access: Broadening participation in high school computer science • Forum Post In Class: • Group Presentation of lesson plan based on provided scenario After Class: • Equity Paper Part 2: Rethink the Equity Paper you wrote in Teaching ECS, specifically including reflections you have to guide others on teaching computer science with more equity • Rewrite/resubmit at the end of the course

Week 2 **Online** **Computational Thinking**	**Topic(s):** Defining Computational Thinking	**Learning Objectives:** Keep up with research in the area of computer science education, and apply it to the teaching process.
	Learning Activities:	Before Class: • Read: Computational Thinking in K-12: A Review of the State of the Field • Forum Post • Optional: Explore Google Computational Thinking websites In Class: • Elevator Pitch: What is Computational Thinking and why is it important?

Week 3 **Online** **CS Unplugged**	**Topic(s):** Computational Thinking: Unplugged Activities	**Learning Objectives:** Implement a variety of methods in the teaching process, including meaningful learning, collaborative learning, and inquiry-based learning.
	Learning Activities:	Before Class: • Read: New frameworks for studying and assessing the development of computational thinking • Forum Post In Class: • Unplugged Computational Thinking Problems

Week 4 **Online** **Debugging**	**Topic(s):** Computing Practices: Debugging Strategies	**Learning Objectives:** Create activities for students to solve specific problems and to document both their programming process and their debugging process.
	Learning Activities:	Before Class: • Read: Debugging: a review of the literature from an educational perspective • Forum Post In Class: • Create Lab Activity about Debugging

Mid-Point **Face-to-Face** **Saturday** **9am—12pm** **Micro-Teaching**	**Topic(s):** Micro-Teaching	**Learning Objectives:** Develop detailed lesson plans for selective topics, consisting of goals and objectives, descriptions of activities and tasks, teaching methods, teaching aids, and evaluation.
	Learning Activities:	In Class: • Group Presentation • Micro-Teaching Feedback Forms

Week 5 **Online** **Collaboration**	**Topic(s):** Computing Practices: Collaboration and Team Projects	**Learning Objectives:** Implement a variety of methods in the teaching process, including meaningful learning, collaborative learning, and inquiry-based learning.
	Learning Activities	Before Class: • Read: Pair programming: Under what conditions is it advantageous for middle school students • Forum Post In Class: • Modify a Team Project

Week 6 **Online** **Engaging Diverse Learners**	**Topic(s):** Computer Science Pedagogy: Engagement and Differentiation	**Learning Objectives:** Create lessons plans to engage diverse learners through culturally responsive pedagogy.
	Learning Activities:	Before Class: • Read: The Knowledge and Practices of High School Science Teachers in Pursuit of Cultural Responsiveness • Forum Post In Class: • Modify a lesson plan to be culturally responsive

Week 7 **Online** **Student Assessment**	**Topic(s):** Writing Student Assessments	**Learning Objectives:** Develop various types of assessments (e.g., formative and summative, authentic, and performance assessments) and corresponding rubrics to evaluate student learning. Conduct information searching and validation using computers and communication platforms.
	Learning Activities:	Before Class: • Read: Chapter 10 Assessment in Guide to Teaching Computer Science • Forum Post In Class • Create a rubric (using a rubric) and create an assessment that would evaluate students in meeting a standard

Week 8 **Online** **Assessment Tools**	**Topic(s):** Looking at Data and Using Student Assessment Tools	**Learning Objectives:** Develop various types of assessments (e.g., formative and summative, authentic, and performance assessments) and corresponding rubrics to evaluate student learning. Conduct information searching and validation using computers and communication platforms.
	Learning Activities:	Before Class: • Read: Chapter 10 Assessment in Guide to Teaching Computer Science • Forum Post In Class: • Create Student Portfolio Assignment

Final **Face-to-Face** **Saturday** **9am—12pm** **Micro-Teaching**	**Topic(s):** Micro-Teaching	**Learning Objectives:** Develop detailed lesson plans for selective topics, consisting of goals and objectives, descriptions of activities and tasks, teaching methods, teaching aids, and evaluation.
	Learning Activities:	In Class: • Group Presentation • Micro-Teaching Feedback Forms

Evaluation and Grading

Evaluation of Student Performance Weighted as Percentages of Total Grade

10%	Weekly synchronous online class attendance and discussion participation
10%	Face-to-face class attendance and discussion participation
10%	Participation in discussion forums on weekly readings
10%	Equity Essay Part 2
30%	Micro-Teaching presentations
30%	Weekly in-class assignments

Earning participation points: Generally, each week includes an opportunity for you to engage with your classmates in written and oral discussion forums. You are expected to contribute written forum posts regarding the weekly readings as well as to attend and participate in discussions in the face-to-face and synchronous on-line classes. Participation in these discussions contributes to your final grade as described above. You will have an opportunity to earn up to 2 points for your participation in each discussion. The rubric below demonstrates how these points are assigned:

- 0 points—No activity; did not participate in the discussion
- 1 point—Provided meaningful feedback but did not craft an original response to the prompt
- 2 points—Crafted an original and meaningful response

To receive full points for a written Forum Post to the weekly reading, you will need to post a substantive response for the assignment *before* that week's synchronous, on-line class. Although you are not expected to meet a specific word count on these posts, your submissions should be well thought out and interesting contributions to the conversation. A substantive post is generally >100 words and introduces a new idea or is a meaningful response to another person's post. When crafting an original and meaningful response please either expand the thought, add additional insights, or respectfully disagree and explain why.

Grading Scale:
Students can choose to receive a letter grade or Pass/Not Pass.

If letter grade is selected:

A	=	90% – 100%
B	=	80% – 89%
C	=	70% – 79%
D	=	60% – 69%
F	=	59% or less

If P/NP is selected:

P	>=	70%
NP	<	70%

Attendance

All participants in the course are expected to attend all face-to-face classes and participate in synchronous on-line classes. Attendance is important for active participation in a professional learning community. It is also important for receiving instruction, information, and feedback for the computing artifacts, lesson plans, etc. A limited number of excused attendances may be tolerated with permission.

Code of Conduct

All participants in the course are bound by the University of California Code of Conduct, found at http://www.ucop.edu/ethics-compliance-audit-services/_files/stmt-stds-ethics.pdf

Netiquette

In an online course, the majority of our communication takes place in the course forums. However, when we have a need for communication that is private, whether personal, interpersonal, or professional, we will use individual email or telephone. Our primary means of communication is written. The written language has many advantages: more opportunity for reasoned thought, more ability to go in-depth, and more time to think through an issue before posting a comment. However, written communication also has certain disadvantages, such a lack of the face-to-face signaling that occurs through body language, intonation, pausing, facial expressions, and gestures. As a result, please be aware of the possibility of miscommunication and compose your comments in a positive, supportive, and constructive manner.

Academic Honesty Policy

The University is an institution of learning, research, and scholarship predicated on the existence of an environment of honesty and integrity. As members of the academic community, faculty, students, and administrative officials share responsibility for maintaining this environment. It is essential that all members of the academic community subscribe to the ideal of academic honesty and integrity and accept individual responsibility for their work. Academic dishonesty is unacceptable and will not be tolerated at the University of California, Irvine. Cheating, forgery, dishonest conduct, plagiarism, and collusion in dishonest activities erode the University's educational, research, and social roles.

Students who knowingly or intentionally conduct or help another student engage in dishonest conduct, acts of cheating, or plagiarism will be subject to disciplinary action at the discretion of UCI Division of Continuing Education.

Disability Services

If you need support or assistance because of a disability, you may be eligible for accommodations or services through the Disability Service Center at UC Irvine. Please contact the DSC directly at (949) 824-7494 or TDD (949) 824-6272. You can also visit the DSC's website: http://www.disability.uci.edu/. The DSC will work with your instructor to make any necessary accommodations. Please note that it is your responsibility to initiate this process with the DSC.

Advanced Topics for Computer Science Teachers

EDUC 300.46

4 Units

first offering—Summer 2018

Class Meeting Information

Modality: Hybrid over five weeks

- Face-to-Face once per week
- Online, including weekly synchronous online classes offered twice per week
  - Tool: Zoom Video Conference

Website:	sites.uci.edu/cs1c/advanced-topics-for-computer-science-teachers
Canvas website:	canvas.instructure.com/courses/1356390

Instructor Information

Name:	Dan Frost
Email:	frost@uci.edu
Website:	frost.ics.uci.edu

Dr. Dan Frost is Professor Emeritus of Teaching in the departments of Computer Science and Informatics at UC Irvine, where he was a faculty member for 19 years. He has developed and taught courses in a variety of topics, including Introductory Programming, Artificial Intelligence, Software Engineering, Social Impact of Computing, Computer Graphics, and Computer Games. He co-founded UC Irvine's major in Computer Game Science, taught several courses in the major and served as the chair of the CGS Steering Committee.

Dr. Frost was Principal Investigator on a National Science Foundation grant that brought computer science, game design, and cultural education to American Indian high school students, who created games that retold traditional stories and cultural practices. He chaired the Computer Science Teachers Association committee that wrote A Model Curriculum for K-12 Computer Science: Level 1 Objectives and Outlines (the K-8 CS curriculum) sponsored by the Association for Computing Machinery (ACM). He co-wrote the K-12 Computer Science Framework, under the auspices of ACM, Code.org, and many states and school districts.

Dr. Frost received his Ph.D. in Information and Computer Science at the University of California, Irvine, and his Bachelor's degree from Harvard University.

Communication

Please note that general questions about the course (content questions, due date clarifications, etc.) should be posted via the general Q&A forum on our **course site** so that all students can benefit from the answer. To reach the instructor regarding questions that are personal in nature (such as a request for an extension due to a family emergency or a request for an incomplete grade), please email the program email address cs1catoc@uci.edu.

Course Description

Advanced Topics for Computer Science Teachers covers topics in computer science that are more advanced than those required to teach Exploring Computer Science and Computer Science Principles, but that enable teachers of these courses to feel confident that they know more than most of their students and can answer student questions that might arise in the classroom. In addition to learning basic programming in Python, teachers in the course will learn basic data structures and algorithms (including tradeoffs between data representation and algorithm performance), and software design (including the process of planning, engineering and implementing a software system to solve larger-scale problems). The course emphasizes the breadth, coherence, and approachability of Computer Science as a field of intellectual inquiry. In addition to several homework assignments, each student will complete two projects, one centered on computer programming and software design, and one on developing pedagogical materials for high school computer science.

Prerequisites — Classes or Knowledge Required Before Taking This Course

Advanced Topics for Computer Science Teachers is being offered as part of UCI's Computer Science Teacher Certificate Program, which is currently funded by the National Science Foundation. This course is the fourth in a sequence of four courses in the program. Teachers are expected to have successfully completed the first three courses, offered in the previous year (Spring and Summer) and earlier this year (Spring):

1. Teaching Exploring Computer Science (4 units)
2. Teaching Computer Science Principles (5 units)
3. Methods for Teaching Computer Science (2 units)

All teacher-participants (aka students) are admitted through the certificate program.

Course Objectives

At the end of this course, teacher-participants (students) will understand:

- Python programming
  - Write, run, and debug a Python program
  - Understand fundamental concepts of program design in Python
  - Explain several differences between Python, Scratch, and other programming languages, and decide which language is more appropriate for a given task
- Software Design and Development
  - Identify and describe the phases of the software design/development process
  - Explain the role of a requirements or specification document
  - Develop use cases for a software system design
  - Design and carry out systematic unit testing of a software module
- Algorithms and data structures
  - Implement several basic data structures and algorithms in Python
  - Describe the trade-offs between alternative data structures and algorithms for a specific problem specification
- Additional advanced topics in computer science
  - Artificial Intelligence
  - Privacy and security
  - Image file formats, data compression
  - Databases, networks, Internet, cloud computing
  - Big data, visualizing data
  - Careers in computing
- How computational thinking (as described in a previous course in the sequence) is manifested in the advanced topics of this course
- How to design effective instruction for advanced topics in Computer Science

Course Material/Required Reading

1. Brad Miller and David Ranum. *How To Think Like A Computer Scientist.* Available online at runestone.academy.

2. Alan Turing, “Computing Machinery and Intelligence”, *Mind*, 1950.

Week 1 **Face to Face**	**Topics/Objectives/Activities** Introduction and Overview • Underlying themes and concerns of Computer Science • How is CS like mathematics? Other sciences? Art? Engineering?	**Assignments** • Set up an account at Runestone, read the first three sections of *How To Think*, and write some comments **(pre-assignment, due at start of class)** • Prepare a question about CS to discuss in class (due at start of next F2F class)
	Python Programming • Using *How to Think* online • Basic language features – names, expressions, procedures, functions • In-class exercises: example program – guessing game	• Selected exercises from *How to Think* (due at start of next on-line class)
	Data structures, algorithms, algorithm analysis • Examples of familiar algorithms • Examples of alternate data structures • Analyzing algorithm performance	• Selected sections and exercises from *How to Think* (due at start of next F2F class)
	Introduction to Software Engineering • The challenge and its scope • Process models • Requirements Engineering, use cases	• Find a current post or article relating to software engineering (due at start of next F2F class)
	Preparation for next face to face meeting.	• Read “Computing Machinery and Intelligence”, respond to four questions (due at start of next F2F class)
Week 1 **Online**	Discussion of Python programming Group work on assigned exercises	

Week 2 **Face to Face**	**Topics/Objectives/Activities** Artificial Intelligence • Reactions to Turing paper • Turing biosketch • Four perspectives on AI: Human/Optimal, Thinking/Performance • AI Today: hype or reality?	**Assignments** • Find a current post or article relating to artificial intelligence (due at start of next F2F class)
	Python Programming • Lists, ranges • Loops • if statements • In-class exercises	• Selected exercises from *How to Think* (some due at start of next on-line class, some due at start of next F2F class)
	Introduction to Software Engineering • Customer interview – a guest business person will describe a desired system	• Write a specification document, based on the interview, working in small teams (due at start of next F2F class)
	Preparation for class projects	• Write a proposal for Pedagogy project (due at start of next on-line class) • Write a proposal for Python Programming project (due at start of next F2F class)
	Discussion of Python programming Small groups discuss project proposals	
Week 2 **Online**	Python discussion Break out into groups to work on specification assignment	

Week 3 **Face to Face**	**Topics/Objectives/Activities** Review and discussion of project proposals • Scoping • Planning	**Assignments** • Detailed plan for each project, with one component identified and completed (due at start of next F2F class)
	Python Programming • Working with collections of data • Accessing data from the web • Using existing classes and objects • In-class exercises	• Selected exercises from *How to Think* (some due at start of next on-line class; some due at start of next F2F class)
	Images on computers • Pixels and color	• Complete provided photo-manipulation Python program (due at start of next F2F class)
	Introduction to Software Engineering • Testing – its role in systems large and small • Black box and white box test case development	• Write and run black box test cases on a provided system (due at start of next F2F class)
Week 3 **Online**	Discussion of Python programming Small groups discuss project proposals Small groups work on testing assignment	

Week 4 **Face to Face**	**Topics/Objectives/Activities** Review and discussion of project proposals	**Assignments** • Both projects near-completed (due at start of next F2F class)
	Python Programming • Designing classes and objects	• Selected exercises from *How to Think* (due at start of next on-line class)
	Interesting Computer Science Topics – such as: • Privacy and Security • Computer Architecture • Computer Graphics • CS and mathematics • Operating Systems • Internet and other networks • Databases • Programming Languages • Cloud computing • Big data • Visualizing data	
	Careers in computing	
Week 4 **Online**	Small groups discuss project progress	

Week 5 **Face to Face**	**Topics/Objectives/Activities** Pedagogy project presentations	**Assignments** • Both projects completed (due end of next week)
	Continued Interesting Topics Great Python Libraries	
Week 5 **Online**	Wrap up and review Office hours	Small groups discuss project proposals (as needed) Review projects Q&A

Evaluation and Grading

Evaluation of Student Performance Weighted as Percentages of the Total Grade

10%	Homework assignments
35%	Python programming project
35%	Computer Science pedagogy project
10%	Course Participation, primarily attendance at and participation in face-to-face and on-line classes

Grading Scale:
Students can choose to receive a letter grade or Pass/Not Pass.

If letter grade is selected:

A	=	90% – 100%
B	=	80% – 89%
C	=	70% – 79%
D	=	60% – 69%
F	=	59% or less

If P/NP is selected:

P	>=	70%
NP	<	70%

Attendance

All participants in the course are expected to attend all Face-to-Face classes and participate in synchronous on-line classes (for which there will be two options per week). Attendance is important for active participation in a professional learning community. It is also important for receiving instruction, information, and feedback for the computing artifacts, lesson plans, etc. A limited number of excused attendances may be tolerated with permission.

Code of Conduct

All participants in the course are bound by the University of California Code of Conduct, found at http://www.ucop.edu/ethics-compliance-audit-services/_files/stmt-stds-ethics.pdf

Netiquette

In an on-line course, the majority of our communication takes place in the course forums. However, when we have a need for communication that is private, whether personal, interpersonal, or professional, we will use individual email or telephone. Our primary means of communication is written. The written language has many advantages: more opportunity for reasoned thought, more ability to go in-depth, and more time to think through an issue before posting a comment. However, written communication also has certain disadvantages, such a lack of the face-to-face signaling that occurs through body language, intonation, pausing, facial expressions, and gestures. As a result, please be aware of the possibility of miscommunication and compose your comments in a positive, supportive, and constructive manner.

Academic Honesty Policy

The University is an institution of learning, research, and scholarship predicated on the existence of an environment of honesty and integrity. As members of the academic community, faculty, students, and administrative officials share responsibility for maintaining this environment. It is essential that all members of the academic community subscribe to the ideal of academic honesty and integrity and accept individual responsibility for their work. Academic dishonesty is unacceptable and will not be tolerated at the University of California, Irvine. Cheating, forgery, dishonest conduct, plagiarism, and collusion in dishonest activities erode the University's educational, research, and social roles.

Students who knowingly or intentionally conduct or help another student engage in dishonest conduct, acts of cheating, or plagiarism will be subject to disciplinary action at the discretion of UCI Division of Continuing Education.

Disability Services

If you need support or assistance because of a disability, you may be eligible for accommodations or services through the Disability Service Center at UC Irvine. Please contact the DSC directly at (949) 824-7494 or TDD (949) 824-6272. You can also visit the DSC's website: http://www.disability.uci.edu/. The DSC will work with your instructor to make any necessary accommodations. Please note that it is your responsibility to initiate this process with the DSC.