Weekly Planner: AP CSP week of 4.13.20

BIG IDEA for the week:

3- ALGORITHMS AND PROGRAMMING

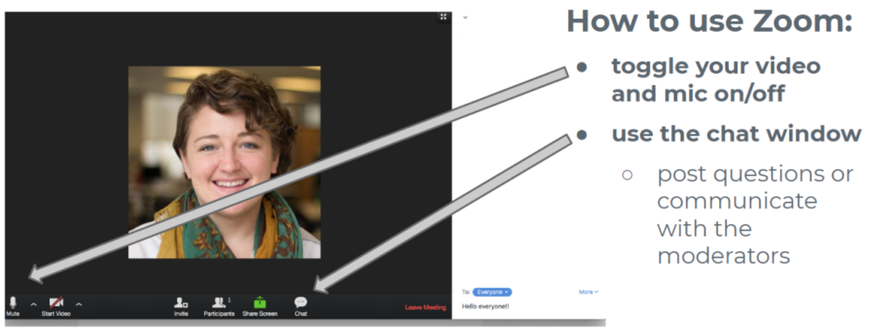
AP CSP

Flip Code: cccaapcsp

At this time, I have unlocked most of Unit 5 for you and you may work ahead of schedule if you like. **LET ME KNOW if you need me to unlock an assessment for you or if I can help you with any of the programs.**

Day	***SEE EMERGENCY MESSAGE ON SCHOOLOGY!*** **YOU ARE ELIGIBLE for a free laptop to KEEP!!**
Mon 4.13	**Announcement! This Monday morning, in your schoology inbox, I have prepared and sent to you a detailed break down of your Unit 5 progress as well as your current grade, missing lessons, and possible final grade upon completion.** **YOU CAN DO THIS!! Being exempt from the multiple choice exam is a gift that no one else will ever get..EVER. This means you need to step it up, just finish Unit 5, and be creative as well as technical for your AP TASKS!** Here's what you should shoot for: -Finish Unit 5 by April 26th -Do the create task -Check in with me every school day! -Always check our website and schoology daily!

	cb.org/tech = free device for AP kids Code break Wednesday code.org/break for one hour on Wed.
Tues 4.14	No APCSP multiple choice final. You will ONLY be graded on your CREATE and Explore tasks that you submit! For this reason, I am putting in my power points with teacher code into our schoology materials for UNIT 5. You MUST complete unit5 in order to know what you are doing for the create task. Please set aside time each day to LEARN the material and contact me when you are ready for me to unlock a unit5 code.org test for you. I will be in schoology and code.org daily to provide you with support. Also, we will have our daily ZOOM meetings (announced on schoology). YOU CAN DO THIS!! Being exempt from the multiple choice exam is a gift that no one else will ever get..EVER. This means you need to step it up and be creative as well as technical for your AP TASKS!
Wed 4.15	
Thurs 4.16	
Fri 4.17	



How to use Zoom:

- toggle your video and mic on/off
- use the chat window
  - post questions or communicate with the moderators

TUTORING HELP!! https://www.khanacademy.org/computing/ap-computer-science-principles

CODE break Wednesdays! https://code.org/break#signup

https://www.khanacademy.org/computing/ap-computer-science-principles/ap-csp-exam-preparation#learn-ap-csp-exam-pseudocode

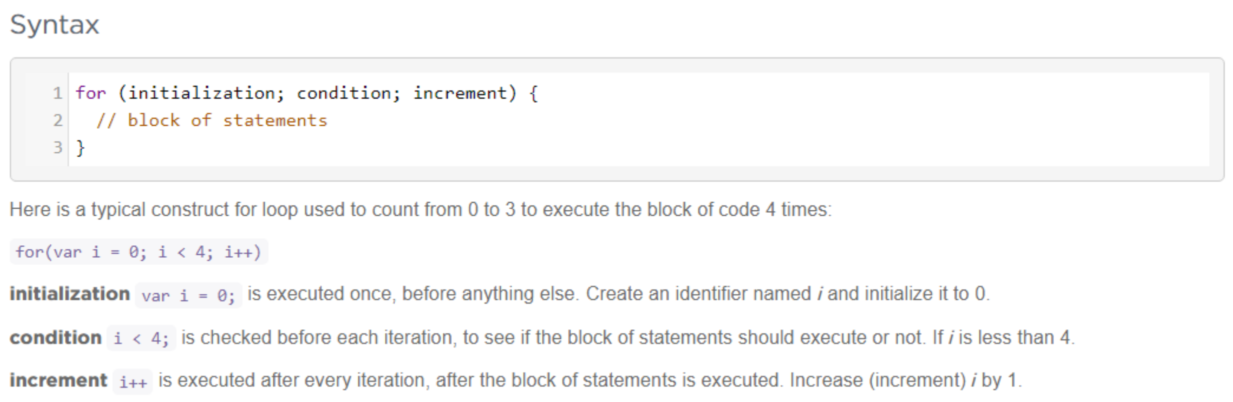
https://online-learning.harvard.edu/course/cs50s-introduction-game-development?delta=0

https://www.edx.org/course/using-python-for-research

Scoring Guidelines and Notes for the 2019 Exam Administration

Explore – Sample Responses	Create – Sample Responses	Scoring Guidelines	Commentary
Explore A: Artifact Explore A: Written response	Create A: Video Create A: Written response Create B: Video	Scoring Guidelines Chief	Explore: Commentary Create: Commentary Scoring Distribution

Explore – Sample Responses	Create – Sample Responses	Scoring Guidelines	Commentary
Explore B: Artifact Explore B: Written response Explore C: Artifact Explore C: Written response Explore D: Artifact Explore D: Written response Explore E: Artifact Explore E: Written response Explore F: Artifact Explore F: Written response Explore G: Artifact Explore G: Written response Explore H: Artifact Explore H: Written response Explore I: Artifact Explore I: Written response Explore J: Artifact Explore J: Written response	Create B: Written response Create C: Video Create C: Written response Create D: Video Create D: Written response Create E: Video Create E: Written response Create F: Video Create F: Written response Create G: Video Create G: Written response Create H: Video Create H: Written response Create I: Video Create I: Written response Create J: Video Create J: Written response	Reader Report	



Syntax

```
for (initialization; condition; increment) {
  // block of statements
}
```

Here is a typical construct for loop used to count from 0 to 3 to execute the block of code 4 times:

`for(var i = 0; i < 4; i++)`

initialization `var i = 0;` is executed once, before anything else. Create an identifier named *i* and initialize it to 0.

condition `i < 4;` is checked before each iteration, to see if the block of statements should execute or not. If *i* is less than 4.

increment `i++` is executed after every iteration, after the block of statements is executed. Increase (increment) *i* by 1.

AP CSP pseudocode

Assignment, display, and input

`a ← expression`

- Evaluates `expression` and assigns the result to the variable `a`.
- □Practice: Storing data in variables

`DISPLAY (expression)`

- Displays the value of `expression`, followed by a space.
- □Practice: Programming basics

`INPUT ()`

- Accepts a value from the user and returns it.

Arithmetic operators and numeric procedures

`a + b a - b a * b a / b`

- The arithmetic operators, +, -, *, and /, are used to perform arithmetic on `a` and `b`.
- □Practice: Mathematical expressions

`a MOD b`

- Evaluates to the remainder when a is divided by b. Assumes that `a` and `b` are positive integers.
- □Practice: Mathematical expressions

`RANDOM(a, b)`

- Evaluates to a random integer from `a` to `b`, including `a` and `b`.
- □Practice: Random numbers

Relational and Boolean operators

`a = b a ≠ b a > b a < b a ≥ b a ≤ b`

- The relational operators, =, ≠, >, <, ≥, and ≤ are used to test the relationship between two expressions, variables, or values.
- □Practice: Conditionals with if, else, and Booleans

`NOT condition`

- Evaluates to `true` if `condition` is `false`; otherwise evaluates to `false`.
- □Practice: Compound Booleans with logical operators

`condition1 AND condition2`

- Evaluates to `true` if both `condition1` and `condition2` are true; otherwise evaluates to `false`.
- □Practice: Compound Booleans with logical operators

`condition1 OR condition2`

- Evaluates to `true` if `condition1` is `true` or if `condition2` is `true` or if both `condition1` and `condition2` are true; otherwise evaluates to false.
- □Practice: Compound Booleans with logical operators

Selection

`IF (<condition>) { <block of statements> }`

- The code in `block of statements` is executed if the Boolean expression `condition` evaluates to `true`; no action is taken if `condition` evaluates to false.
- □Practice: Conditionals with if, else, and Booleans

`IF (<condition>) { <first block of statements> } ELSE { <second block of statements> }`

- The code in `first block of statements` is executed if the Boolean expression `condition` evaluates to `true`; otherwise the code in `second block of statements` is executed.
- □Practice: Conditionals with if, else, and Booleans, Nested conditionals

Iteration

`REPEAT n TIMES { <block of statements> }`

- The code in `block of statements` is executed `n` times.
- □Practice: Numbered repetition of instructions

`REPEAT UNTIL (condition) { <block of statements> }`

- The code in `block` of statements is repeated until the Boolean expression `condition` evaluates to `true`.
- □Practice: Conditional repetition of instructions

List operations

For all list operations, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program terminates.

`list[i]`

- Refers to the element of `list` at index `i`. The first element of `list` is at index 1.
- □Practice: Storing and updating lists

`list[i] ← list[j]`

- Assigns the value of `list[j]` to `list[i]`.
- Practice: Storing and updating lists

`list ← [value1, value2, value3]`

- Assigns `value1`, `value2`, and `value3` to `list[1]`, `list[2]`, `list[3]`, respectively.
- Practice: Storing and updating lists

`FOR EACH item IN list { <block of statements> }`

- The variable `item` is assigned the value of each element of `list` sequentially, in order from the first element to the last element. The code in `block of statements` is executed once for each assignment of item.
- Practice: Iterating over lists with loops

`INSERT (list, i, value)`

- Any values in `list` at indices greater than or equal to `i` are shifted to the right. The length of `list` is increased by 1, and `value` is placed at index `i` in `list`.
- Practice: Storing and updating lists

`APPEND(list, value)`

- The length of `list` is increased by 1, and `value` is placed at the end of `list`.
- Practice: Storing and updating lists

`REMOVE(list, i)`

- Removes the item at index `i` in `list` and shifts to the left any values at indices greater than `i`. The length of `list` is decreased by 1.
- Practice: Storing and updating lists

`LENGTH(list)`

- Evaluates to the number of elements in list.

Procedures

`PROCEDURE name (parameter1, parameter2, ...) { <instructions> }`

- A procedure, `name`, takes zero or more parameters. The procedure contains programming instructions.
- Practice: Defining a procedure, Procedures with parameters

`PROCEDURE name (parameter1, parameter2, ...) { <instructions> RETURN (expression) }`

- A procedure, `name`, takes zero or more parameters. The procedure contains programming instructions and returns the value of `expression`. The `RETURN` statement may appear at any point inside the procedure and causes an immediate return from the procedure back to the calling program.
- Practice: Procedures with return values

Robot

If the robot attempts to move to a square that is not open or is beyond the edge of the grid, the robot will stay in its current location and the program will terminate.

`MOVE_FORWARD ()`

- The robot moves one square forward in the direction it is facing.
  `ROTATE_LEFT ()`
- The robot rotates in place 90 degrees counterclockwise (i.e. makes an in-place left turn).
  `ROTATE_RIGHT ()`
- The robot rotates in place 90 degrees clockwise (i.e. makes an in-place right turn).
  `CAN_MOVE (direction)`
- Evaluates to `true` if there is an open square one square in the `direction` relative to where the robot is facing; otherwise evaluates to `false`. The value of `direction` can be `left`, `right`, `forward`, or `backward`.
  □Practice of robot-like questions are throughout the Repetition lesson.

  There are many differences between the AP CSP pseudocode and the JavaScript language syntax.

  This table highlights the biggest differences:

Concept	**Pseudocode**	**JavaScript**
Assignment	`a ← expression`	`var a = expression;`
Equality	`a = b`	`a == b`
Inequality	`a ≠ b`	`a != b`
Repetition	`REPEAT n TIMES`	`for (var i = 0; i < n; i++)`
Repetition	`REPEAT UNTIL (condition)`	`while (condition)`
List index	`list[1]` is first item	`list[0]` is first item
List iteration	`FOR EACH item IN list`	`for (var i = 0; i < list.length; i++)`