SCOPE & SEQUENCE

PYHON 101

Lesson	Purpose	CSTA Standards	CS Concepts	Commands / sample code
1	**All that syntax: Python command structure & syntax**	1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. 2-AP-17 Systematically test and refine programs using a range of test cases.	Algorithm; Python command syntax structure; Differences in syntax when declaring strings or numbers; Correcting syntax errors	`player.say(45+2505)` `player.say("Hi")`

2	**Location, Location, Location: Parameters & Coordinates**	1A-AP-11
		Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
		1B-AP-08
		Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
		2-AP-17
		Systematically test and refine programs using a range of test cases.
			Relative coordinates
			Positional programming in Minecraft
			Command structure
				`blocks.place(PLANKS_OAK, pos(-1, -1, 0))`
				`blocks.place(PLANKS_OAK, pos(-2, -1, 0))`
				`blocks.place(PLANKS_OAK, pos(-3, -1, 0))`
				`blocks.place(PLANKS_OAK, pos(-4, -1, 0))`
				`blocks.place(PLANKS_OAK, pos(-5, -1, 0))`

3	**A Varied Outcome: Variables**	1A-AP-08
		Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
		1A-AP-11
		Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
		2-AP-17
		Systematically test and refine programs using a range of test cases.
		1B-AP-09
		Create programs that use variables to store and modify data.
			Creating a variable and assign a string/ numerical value to it
			The concept of world coordinates
			Using predefined locations.
				`location1 = world(-24, 40, -18)`
				`location2 = world(-31, 40, -11)`
				`location3 = world(-28, 40, -16)`
				`location4 = world(-25, 40, -13)`
				`location5 = world(-31, 40, -17)`
				
				`blocks.place(MELON_BLOCK, location 1)`
				`blocks.place(PUMPKIN, location2)`
				`blocks.place(MELON_BLOCK, location3)`
				`blocks.place(PUMPKIN, location4)`
				`blocks.place(MELON_BLOCK, location5)`

4	**Animals Are Friends: Lists & Methods**	3A-AP-14
		Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
		2-AP-13
		Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
		2-AP-11
		Create clearly named variables that represent different data types and perform operations on their values.
			Creating lists
			Using methods with lists
			Modifying and sorting lists
			Zero-based numbering
				`Bone = world(-21, 45, -31)`
				`Beef = world(-21, 45, -29)`
				`Chicken = world(-21, 45, -27)`
				`Biscuit = world(-21, 45, -25)`
				`Vitamins = world(-21, 45, -23)`
				`# Replace the lines below with your code #`
				
				`Dog_Food=[Bone, Beef, Chicken, Biscuit]`
				`Dog_Food.append(Vitamins)`
				`Dog_Food.pop(1)`
				
				`blocks.place(REDSTONE_BLOCK, Dog_Food[0])`
				`blocks.place(REDSTONE_BLOCK, Dog_Food[1])`
				`blocks.place(REDSTONE_BLOCK, Dog_Food[2])`
				`blocks.place(REDSTONE_BLOCK, Dog_Food[3])`
				`blocks.place(REDSTONE_BLOCK, Dog_Food[4])`

5	**A Helper For The Home: Agent & Loops**	1A-AP-10
		Develop programs with sequences and simple loops, to express ideas or address a problem.
		1B-AP-11
		Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
		1B-AP-15
		Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended
			For Loops
			Syntax
			Gaining familiarity with Nested loops
				`for i3 in range(3):`
				`  for i1 in range(7):`
				`  agent.collect_all()`
				`  agent.move(FORWARD, 1)`
				`  agent.move(RIGHT, 1)`
				`  for i2 in range(7):`
				`  agent.collect_all()`
				`  agent.move(BACK, 1)`
				`  agent.move(RIGHT, 1)`
				`agent.drop_all(FORWARD)`

6	**Driving Around: Conditionals & Booleans**	2-AP-13
		Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
		1B-AP-10
		Create programs that include sequences, events, loops, and conditionals.
		2-AP-17
		Systematically test and refine programs using a range of test cases
			Concept of conditionals
			Using if, if-else & elif conditionals
			Using Boolean logic with conditionals
				`for index in range(23):`
				`  if agent.detect(AgentDetection.BLOCK, FORWARD) and not (agent.detect(AgentDetection.BLOCK, LEFT)):`
				`  agent.move(LEFT, 1)`
				`  elif agent.detect(AgentDetection.BLOCK, FORWARD)`
				`and agent.detect(AgentDetection.BLOCK, LEFT):`
				`  agent.move(RIGHT, 2)`
				`  else:`
				`  agent.move(FORWARD, 1)`

7	**Emergency Response: while loops & sequences**	2-AP-13
		Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
		1B-AP-10
		Create programs that include sequences, events, loops, and conditionals.
		2-AP-17
		Systematically test and refine programs using a range of test cases.
			While loops
			Using loops with different conditions
			Sequencing tasks
				`while agent.detect(AgentDetection.REDSTONE, FORWARD):`
				`  agent.place(LEFT)`
				`  agent.move(FORWARD, 1)`
				`  while agent.detect(AgentDetection.REDSTONE, LEFT):`
				`  agent.turn(LEFT_TURN)`
				`  agent.move(FORWARD, 2)`
				`  while agent.detect(AgentDetection.REDSTONE, RIGHT):`
				`  agent.place(LEFT)`
				`  agent.move(FORWARD, 1)`
				`  agent.place(LEFT)`
				`  agent.move(BACK, 1)`
				`  agent.turn(RIGHT_TURN)`
				`  agent.place(LEFT)`
				`  agent.move(FORWARD, 1)`

8	**Planting a seed: functions**	1B-AP-10
		Create programs that include sequences, events, loops, and conditionals.
		2-AP-14
		Create procedures with parameters to organize code and make it easier to reuse.
		1B-AP-11
		Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
			Concept of functions & using functions;
			Creating their own functions;
			Using code comments.
				`# Place your functions below #`
				`# This function makes the Agent remove and pickup rocks`
				`def remove_rock():`
				`  agent.destroy(FORWARD)`
				`  agent.collect_all()`
				`  agent.move(FORWARD, 1)`
				`# This function makes the Agent plant saplings behind itself`
				`def plant_tree():`
				`  agent.move(FORWARD, 1)`
				`  agent.till(BACK)`
				`  agent.place(BACK)`
				`# Replace the lines below with your code #`
				`for index in range(12):`
				`  if agent.inspect(AgentInspection.BLOCK, FORWARD) == STONE:`
				`  remove_rock()`
				`  elif agent.inspect(AgentInspection.BLOCK, DOWN) == GRASS:`
				`  plant_tree()`
				`  else:`
				`  agent.move(FORWARD, 1)`

9	**All Fun and Games**	1B-AP-10
		Create programs that include sequences, events, loops, and conditionals.
		2-AP-14
			tackling
			writing larger codes.
			the concept of decomposition.
				`def fire():`
				`  global score`
				`  if blocks.test_for_block(GOLD_BLOCK,`
				`  positions.add(agent.get_position(), pos(0, 2, 0))):`

		Create procedures with parameters to organize code and make it easier to reuse.
		1B-AP-11
		Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
			using all the coding concepts they have learned, in one larger code.
			adding to a relative position.
				`mobs.spawn(FIREWORKS_ROCKET, agent.get_position())`
				`loops.pause(100)`
				`blocks.place(AIR, positions.add(agent.get_position(), pos(0, 2, 0)))`
				`score += 1`
				`elif blocks.test_for_block(GOLD_BLOCK, positions.add(agent.get_position(), pos(0, 3, 0))):`
				`mobs.spawn(FIREWORKS_ROCKET, agent.get_position())`
				`loops.pause(100)`
				`blocks.place(AIR, positions.add(agent.get_position(), pos(0, 3, 0)))`
				`score += 1`
				`def move():`
				`if blocks.test_for_block(LIGHT_BLUE_CONCRETE, pos(0, -1, 0)):`
				`agent.move(RIGHT, 1)`
				`elif blocks.test_for_block(RED_CONCRETE, pos(0, -1, 0)):`
				`agent.move(LEFT, 1)`
				`# Replace the lines below with your code #`
				`score = 0`
				`gameplay.title(mobs.target(NEAREST_PLAYER), "Agent Invaders", "Start")`
				`while score <= 15:`
				`move()`
				`fire()`
				`gameplay.title(mobs.target(NEAREST_PLAYER), "Congratulations!", "You win!")`
				`mobs.spawn(LIGHTNING_BOLT, agent.get_position())`
				`if score > 15:`
				`player.execute("scoreboard players set @p score 15")`

10	**Coding Battle**	1B-AP-10 Create programs that include sequences, events, loops, and conditionals. 1B-AP-08 Compare and refine multiple algorithms for the same Challenge and determine which is the most appropriate. 1B-AP-15	Use all the accumulated coding knowledge to complete 10 challenges within a given time; Have the possibility to compete in multiplayer	

		Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.		

By the end of this course:
The students will

- Have a good grasp of Python syntax
- Gain a good grasp of algorithms, for loops, conditional loops, sequencing, variables, lists, functions. The concept of decomposition

CSTA Standards covered:

1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
1A-AP-10 Develop programs with sequences and simple loops, to express ideas or address a problem.

1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
1B-AP-09 Create programs that use variables to store and modify data.
1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.

2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.
2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-17 Systematically test and refine programs using a range of test cases.
2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.

3A-AP-14 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.