Part 1:  
1)First Try:

"Equal to the number" was printed because the age variable was strictly equal to the number 19 (using ===).

"Equal to the string" was printed because the age variable (19) was loosely equal to the string "19" (using ==), allowing type coercion.

"Truthy!" was printed because age was set to a non-zero value, and non-zero numbers are truthy in JavaScript.

Second Try:

No strings were printed because the age variable was set to 0, and no conditions were met:

age === 19 was false because age was 0.

age == "19" was false because 0 is not equal to "19".

age was falsy because 0 is a falsy value.

2)I used === for strict equality (checking both value and type) and == for loose equality (allowing type coercion). In condition #2, == allowed type coercion, so the number 19 was loosely equal to the string "19", and the message "Equal to the string" was printed.

3)In the first try, age was set to 19, which is a truthy value. In the second try, age was set to 0, which is a falsy value. Falsy values in JavaScript include 0, null, undefined, NaN, false, and an empty string "". Therefore, the "Truthy!" message did not print in the second try.

4)We used let because it allows us to reassign the value of the age variable in the second try. If we had used const, which is for constants, we wouldn't be able to reassign the value of age, and the program would throw an error.

Part 2:

Array Output :

---- Part 2: Arrays and Iteration ----

COSC 2328

COSC 2329

MATH 1010

Sunday (0)

Monday (1)

Tuesday (2)

Wednesday (3)

Thursday (4)

Friday (5)

Saturday (6)

Sunday

COSC 2328

COSC 2329

MATH 1010

Monday

COSC 2328

COSC 2329

MATH 1010

Tuesday

COSC 2328

COSC 2329

MATH 1010

Wednesday

COSC 2328

COSC 2329

MATH 1010

Thursday

COSC 2328

COSC 2329

MATH 1010

Friday

COSC 2328

COSC 2329

MATH 1010

Saturday

COSC 2328

COSC 2329

MATH 1010

1) used the forEach method to iterate through the arrays. forEach is a built-in JavaScript method used to loop through each element in an array. For example:

classes.forEach((course) => { console.log(course); });

days.forEach((day, index) => { console.log(${day} (${index})); });

2)We made classes and days constants by using the const keyword because their references should not be reassigned. Although the contents of the arrays can be modified (i.e., elements can be added or removed), the variables themselves cannot be reassigned to new arrays.

Example: const classes = ["COSC 2328", "COSC 2329", "MATH 1010"];

Part 3:  
1) I used dot notation to access the object properties. For example:

myself.firstName for accessing the firstName property.

myself.schoolAddress.streetAddress for accessing the nested schoolAddress object’s streetAddress property.

2) I used template literals for printing the sentence about my graduation and address. Template literals allow us to embed expressions inside strings using backticks () and ${}`:

Example : console.log(`${myself.firstName} ${myself.lastName} graduates in ${myself.graduationYear} from ${myself.schoolAddress.schoolName} at ${myself.schoolAddress.streetAddress}, ${myself.schoolAddress.city}, ${myself.schoolAddress.state}`);

Part 4:  
1) The output correctly displays my class schedule for each day. On Monday, Wednesday, and Friday, I have COSC 2328 and ENGR 1010, and on Tuesday and Thursday, I have MATH 1010 and PHYS 2020

2) Creating the arrays first improves code organization, readability, and reusability. It allows us to define the class schedules independently of the schedule object, which is helpful when the same set of classes applies to multiple days. If we had hard-coded the classes directly inside the schedule array, the code would be less maintainable and more redundant.