today:

- review:
  - event-driven programming
  - conditional execution
  - conditional repetition
- new:
  - programmer-defined functions

event-driven programming

- when web pages are event-driven, that means that they respond to input from a user
- when a robot’s behavior is event-driven, that means that it responds to input from its sensors
- in RoboLab, the following icons facilitate event-driven behavior in your robot:
  - wait-for-let-go  wait-for-push  wait-for-light  wait-for-dark
- the robot waits for an event to happen, then it executes the icons that follow the “wait for” icon in your code
- remember that this doesn’t mean that the robot sits still—if you give it motor commands, like forward before the “wait for” icon, then your robot will go forward until the “wait for” event happens

conditional execution

- there are times when you want your code to behave differently under different conditions
- for example, in the assignment for unit E:
  IF your robot sees something black, THEN it should stop for one second, then go backwards for two seconds, then go forward again.
  IF your robot sees something silver or gold, THEN it should stop for one second, then turn to the left and go forward again.
- the notion of conditional execution means that you define in your program multiple branches, and the code will follow a different branch depending on the conditions it encounters while running
- conditional execution is sometimes referred to as IF-THEN or IF-THEN-ELSE execution
- if the IF condition is true, then the THEN branch is executed; otherwise (if the IF condition is false), the ELSE branch is executed

- in RoboLab, the following icons facilitate conditional execution in your robot:
  - touch-sensor-fork  light-sensor-fork  fork-merge
- note that these are different from event-driven icons since the program will NOT wait for an event to happen but will simply evaluate the condition of the fork icon and execute a branch accordingly
- for example, when using the touch-sensor-fork icon:
  IF the touch sensor is not pressed when the program comes to the touch-sensor-fork icon in its execution, THEN the top branch of icons will be executed; ELSE the bottom branch of icons will be executed
- when using the light-sensor-fork icon:
  IF the light sensor reads a value greater than the one specified (you have to hang a numeric constant below the icon containing the threshold value for the IF-THEN-ELSE decision), THEN the top branch of icons will be executed; ELSE the bottom branch of icons will be executed
when writing a program that uses conditional execution, it is often easier to design your code first using a flowchart, before trying to write anything on the computer.

For example, here is a flowchart for the last challenge in the assignment for unit E:

conditional repetition

- there are times when you want your code to execute the same thing over and over again, repeatedly.
- this is called looping or iteration.
- we talked about three types of loops:
  - “forever” (or infinite) loops
  - counter-controlled loops
  - condition-controlled loops.
- in RoboLab, the following icons facilitate infinite loops:

  - yellow land
  - yellow jump

  These go BEFORE the code that you want to repeat and go AFTER the code that you want to repeat.

- in RoboLab, the following icons facilitate counter-controlled loops:

  - start-of-loop
  - end-of-loop

  These go BEFORE the code that you want to repeat and go AFTER the code that you want to repeat.

  - you have to hang a loop counter (numeric constant) from the start-of-loop icon indicating the number of times you want the loop to run.

- in RoboLab, the following icons facilitate condition-controlled loops:

  - loop-touch-sensor-pushed
  - loop-touch-sensor-released
  - loop-light-sensor-less-than
  - loop-light-sensor-greater-than

  For the light sensor loops, you have to hang a loop counter (numeric constant) from the “start of loop” icon indicating the number of times you want the loop to run.

  - for all the sensor-based loops, you have to hang the port number from the “start of loop” icon indicating which port the sensor is connected to.

  - for all the loops, the icons above show the “start of loop” icon; to end the loops, you use the end-of-loop at the end of the loop.
• compare the following programs:

- programmer-defined functions

  - in RoboLab, “programmer-defined functions” are called subroutines
  - the idea behind a subroutine is if you have some piece of code that is useful and you might want to use it many times—not just in a loop, but other times too—then you can group the icons together into something called a subroutine
  - example:

• subroutines work by having two parts:
  - first, you have to define the subroutine
  - second, you have to invoke or call the subroutine
  - the subroutine only runs when you call it
  - it does NOT run when you define it

  - the subroutine is defined with the create-subroutine icon
  - hanging from the create-subroutine icon is a numeric constant, assigning a number to the subroutine
  - this is in case you want to define more than one subroutine—you give each a number so that you can distinguish between them later
  - from the lower right corner of the create-subroutine icon, you string the icons that you want to belong to the subroutine
  - you end the subroutine with the end icon
  - from the top right corner of the create-subroutine icon, you continue with your program code

• when you want to call or invoke the subroutine, then you use the run-subroutine icon

  - here’s the example again: