Intro:
Hello viewer, I'm starting a new series of arduino tutorials, I'll try to make them quick and as informative as possible. Today we'll discuss arduino input and output for both analog and digital signals.

Analog vs Digital:
Before starting it's important to know the difference between digital and analog signals.

Analog signals are continuous signals that vary over a range of values, such as the voltages produced by a potentiometer, or most senors.

Digital signals, on the other hand, are discrete signals that can only take on a limited number of values, such as the binary values of 0 or 1.

Digital Input and Output:
The arduino is capable of digital input and output through all of it's digital pins, let's connect an LED to pin 10 on the arduino and write this simple code where we'll define a global variable LED that points to value 10, and in the setup we'll use the pinMode function to tell the arduino that pin 10 is an output pin, and finally in the loop we'll use the digitalWrite write function to set the output on pin 10 to High, let's upload the code and we see the LED turns on, on the contrary changing HIGH to LOW turns the LED off.
As for digital input, let's add a button to our circuit, we'll connect one side of the switch to 5v and the other to digital pin 9, additionally we'll add a 10k ohm resistor to pull the input to GND, more on that later. On the code side we'll define a global variable called "Button", and in the setup function we'll set the button pin to INPUT additionally we'll begin the serial monitor to print our input signal, finally in the loop we'll use the digitalRead function to read the input values and store them in a variable called "state", we can print our state variable to the Serial monitor and use it to control our LED. After uploading the code and opening our serial monitor we see that as the button gets pressed the value true is printed to the serial monitor and when released false gets printed, additionally the LED is triggered on and off.
If your wondering why we have a 10k Ohm resistor, it's used to pull the input to LOW or GND when the switch is open without it the signal goes crazy, fortunately you don't need it, as arduino has a builtin 20k Ohm pull up resistor which you can use by changing Input ot Input_pullup in the pinMode function.

Analog Input and Output:
Arduino is also capable of reading analog signals through its 6 analog pins A0- A5, and outputting an analog voltage through only it's PWM pins, you can tell it's a PWM pin by looking for the squiggly line before the digital pin number, in the case of the arduino UNO we can output an analog signal only through digital pins 3, 5, 6, 9, 10, and 11. So going back to our LED circuit where our LED is connected to PWM pin 10, we can output an analog signal to not only turn the LED on and OFF but to adjust its brightness. As in our original LED code, Let's keep everything the same but we'll change the digitalWrite function to analogWrite and we'll set a PWM value to a value between 0 and 255 to control our LED's brightness. Setting it 50, we see the LED is dimmer than before, and when set to 255 it's bright again. Let's add a for loop to adjust the brightness value, the LED goes from bright to dim in a smooth way.
Analog output is cool and all, but let's try to read an input analog signal. For this we can use a potentiometer, I'm using one provided by KeyeStudio. They have an awesome collection of sensors, which I'll link to in the description below. But you can use any potentiometer just follow this simple circuit diagram. I'll connect the negative side to GND, the positive to 5v, and the signal to analog pin 0. As for the code, lets define a global variable called "potentiometer" that points to value A0, and another global integer variable to store the input value, we'll call that "potentiometerValue", in the setup function, we'll only start the Serial monitor, and finally in the loop we'll store the input value in the "potentiometerValue" variable and print it to the serial monitor. After uploading the code and adjusting the potentiometer we see the value on the serial monitor range from **0 to 1023**, we can divide this value by 4 and use it to adjust our LED's brightness.

Conclusion:
That's all for today's video but before I go, I'll leave you with this cool project that utilizes all the knowledge you've acquired from today's video, and I encourage you to build the circuit and play around with the code. I've included a link to the github repository with all the code and wiring diagrams in the description, as well as links to arduino kits and components I recommend.