isPrime[n]: returns True if n is prime, and False otherwise

Trial Division: see if n is divisible by any number $2, 3, 4, 5, \ldots, \lfloor \sqrt{n} \rfloor$

$\lfloor \sqrt{n} \rfloor$ = greatest integer less than or equal to $\sqrt{n}$

Implementation 1:

"Flag" — starts False, but will be set True if we find a divisor

```
isPrime[n_] := Module[{foundDivisor = False},
  Do[
   If[Divisible[n, k], foundDivisor = True],
   {k, 2, Sqrt[n]}
  ];
  ! foundDivisor (* return value *)
 ]
```

loop over k from 2 to $\sqrt{n}$

← if k divides n, then set foundDivisor to True.

NOT →

If no divisor is found, then n is prime.

If a divisor is found, then n is not prime.

Implementation 2:

divisor to test

max possible divisor

```
isPrime2[n_] := Module[ {foundDivisor = False, d = 2, m = Sqrt[n]},
  While[! foundDivisor && d ≤ m,
   If[Divisible[n, d], foundDivisor = True];
   d++
  ];
  ! foundDivisor (* return value *)
 ]
```

AND ← conditions

d++ increment d

loop until a divisor is found or $d > \sqrt{n}$

OR: ||

Print: prints some value to the screen.

Return: sends a value back to wherever the module was called. It may or may not be printed on screen.

Implementation 3:

```
isPrime3[n_Integer] := Module[{m = Sqrt[n]},
 Catch[
  Do[
   If[Divisible[n, d], Throw[False]],
   {d, 2, m}
  ];
  True
 ](* the Module returns the "caught" value *)
]
```

If d divides n, then "throw" the value False. This interrupts the loop and transfers control to the enclosing Catch block. The Catch statement has the value False.

If no divisor is found, then the Catch block has the value True.

The value of the Catch block is the module's return value.

LISTING PRIMES

listPrimes[n]: return a list of all primes up to n.

Plan:

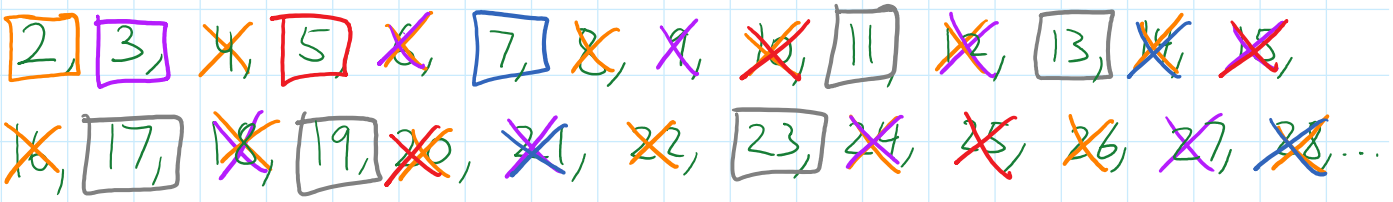
primes = { }

loop: k goes from 2 to n

if k is prime, put it in the list primes

return primes list

Sieve of Eratosthenes



How can we implement the sieve in Mathematica?

Make a plan on paper, then code in Mathematica.