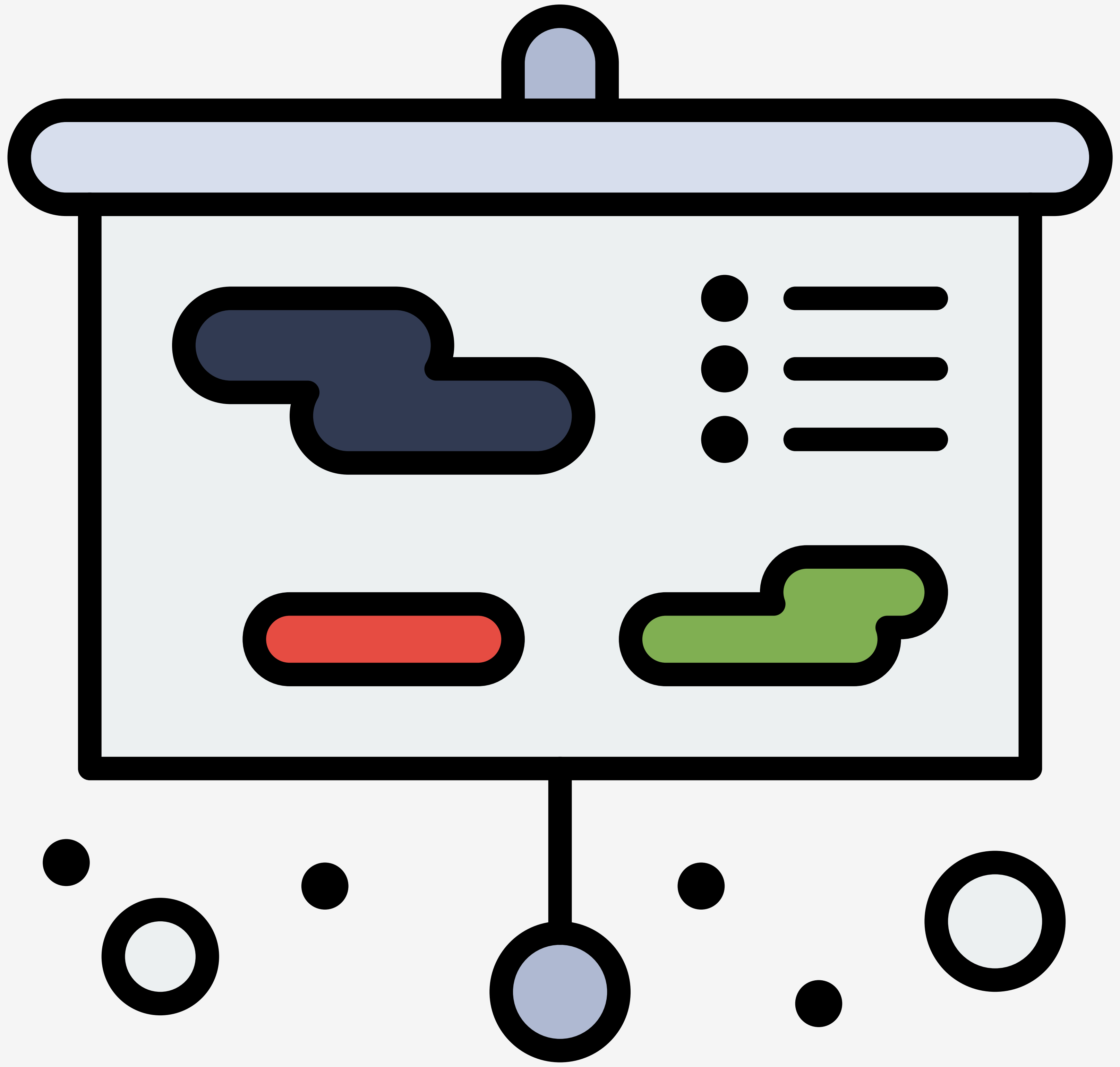
RESPONSIVE WEB DESIGN II

Lecture 19

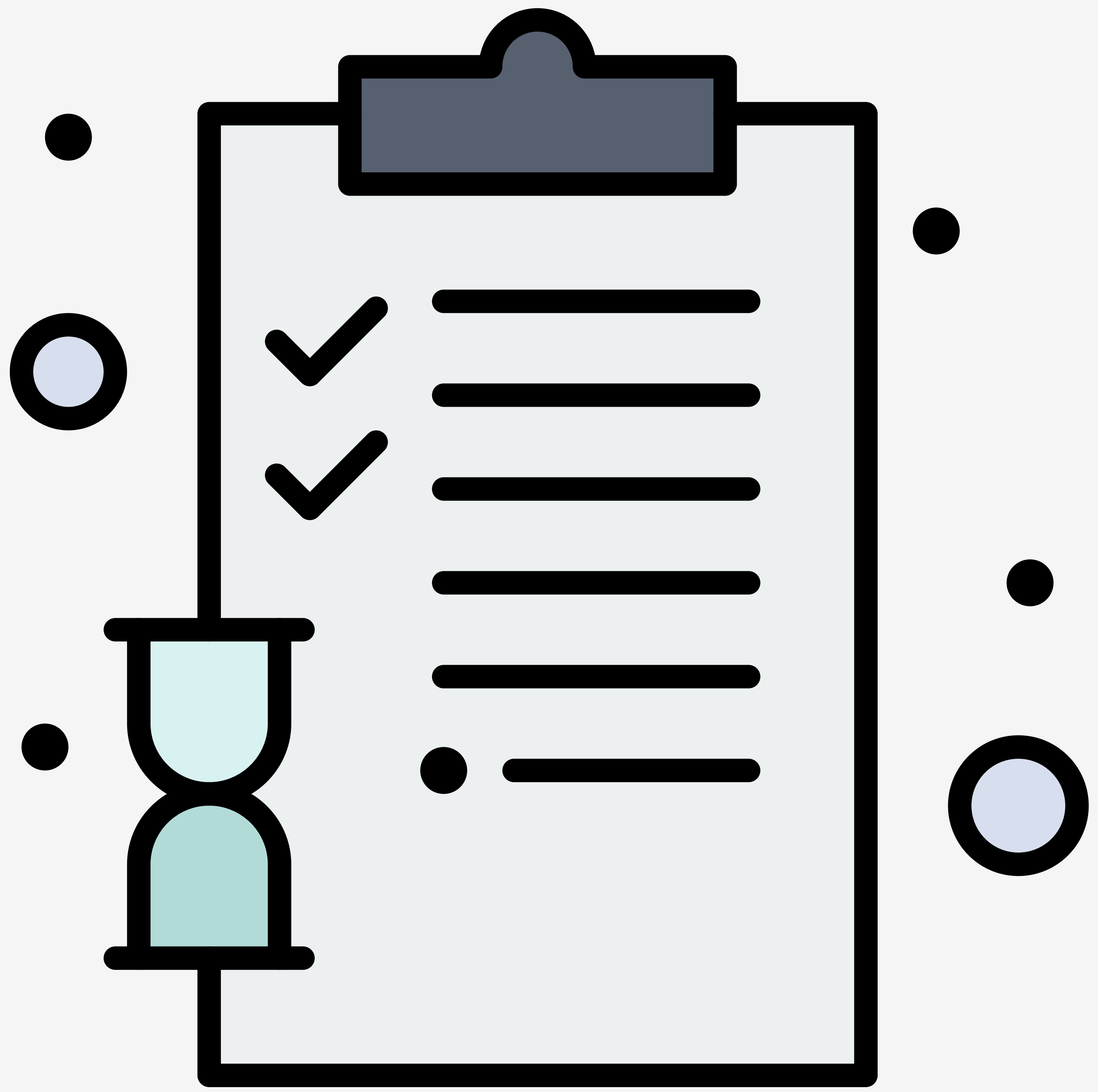
TODAY'S TOPICS



- Introduction to Programming
- SassScript
- Participation: Sassy Cats
- Exercise: Sassy Shapes

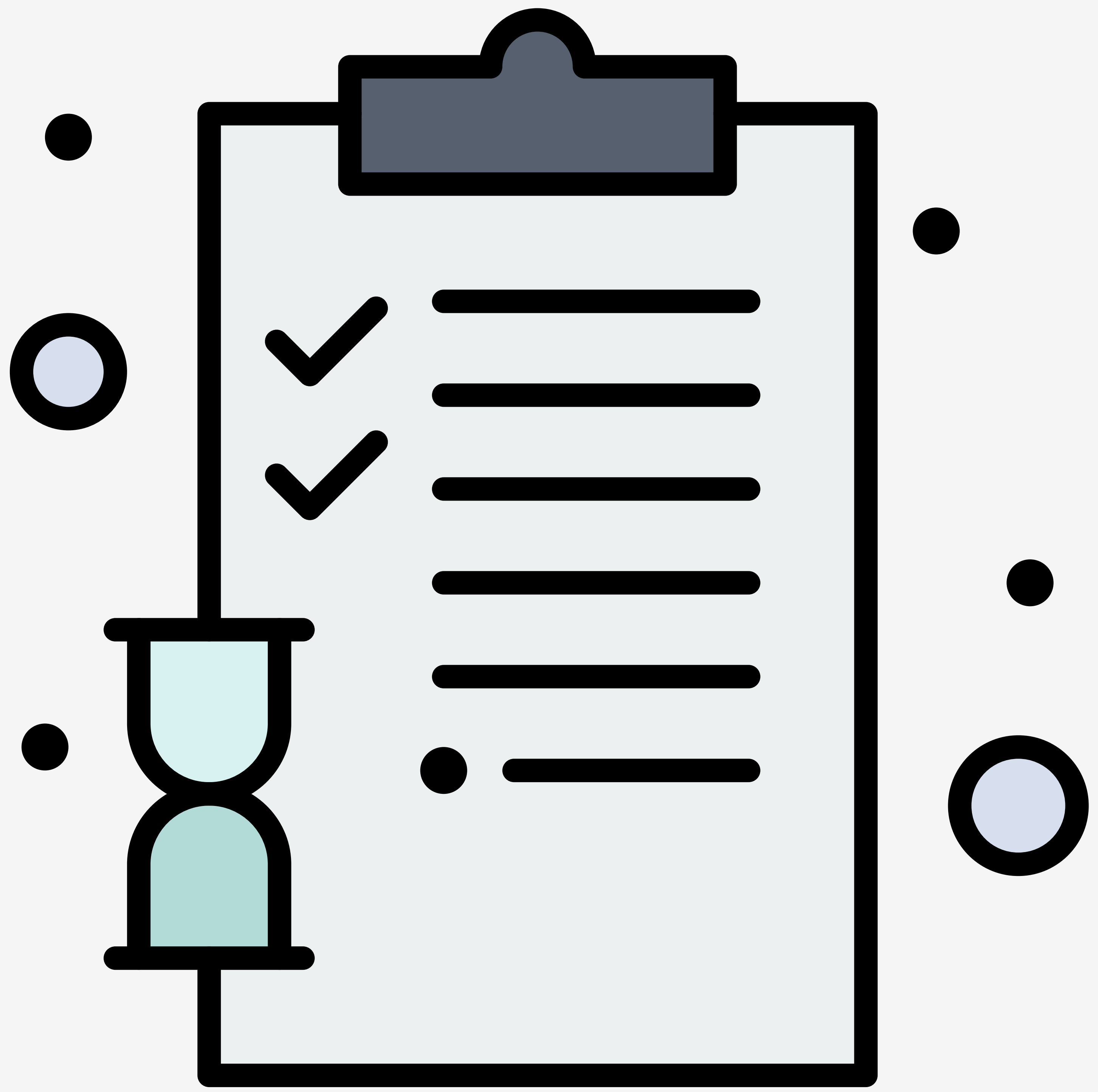
INTRODUCTION TO PROGRAMMING

PROGRAMMING



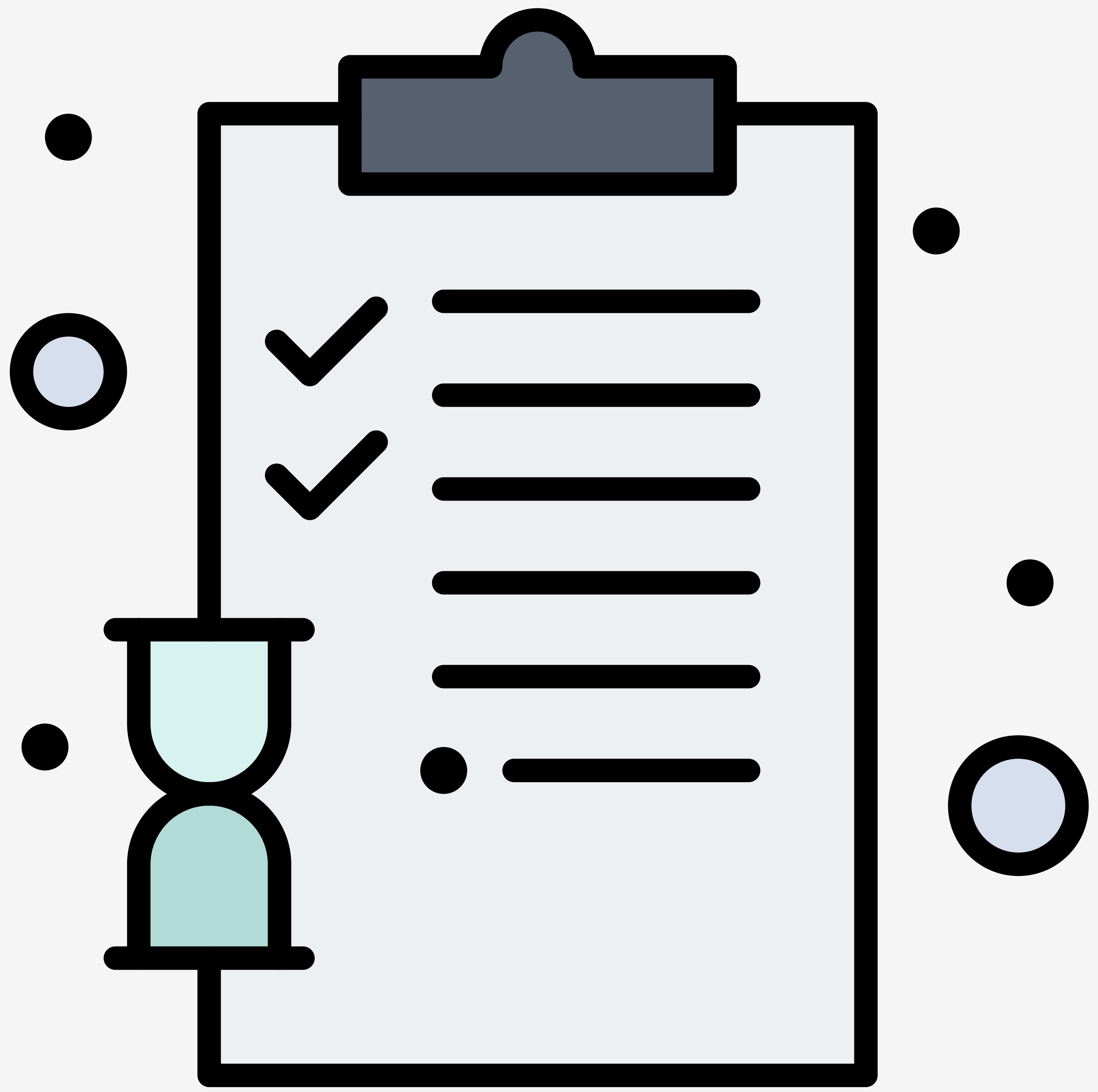
- A program is a set of the instructions for a computer
- The instructions must be in a language that computer understands
- The instructions must be in the proper syntax
- The instructions must be in the right order

VARIABLES

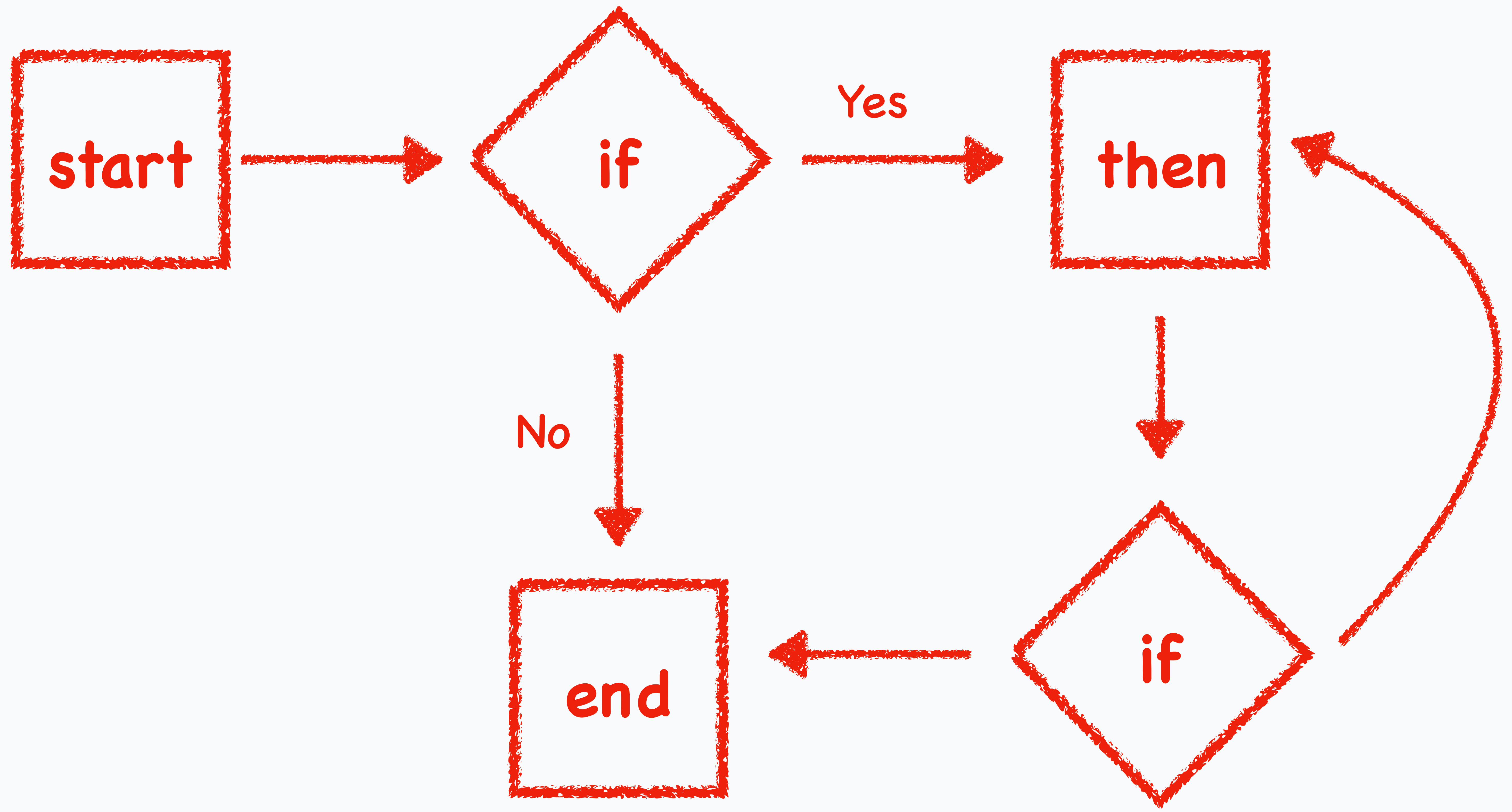


- A variable is a container with a label
- A variable can be of different data types
- Single value data types: strings, numbers, boolean
- Multiple value data types: arrays, lists, stacks, objects, maps

FLOW CONTROL

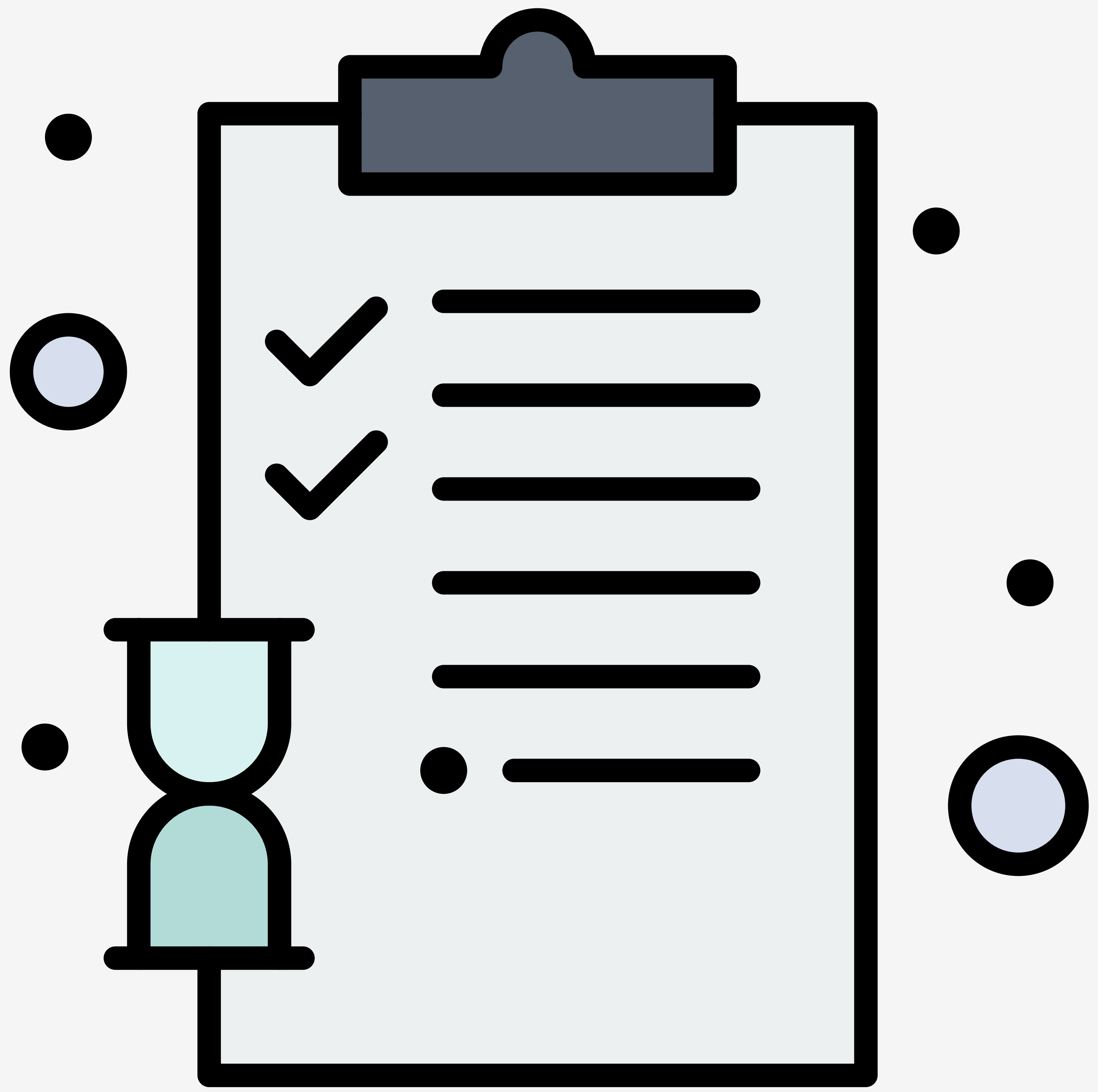


- Conditional statements fork the flow of a program or execute code only when a condition is met
- Loops will repeat code until a condition is met


start
if
Yes
then
No
end
if

SASS LISTS

SASS LISTS



Going Away

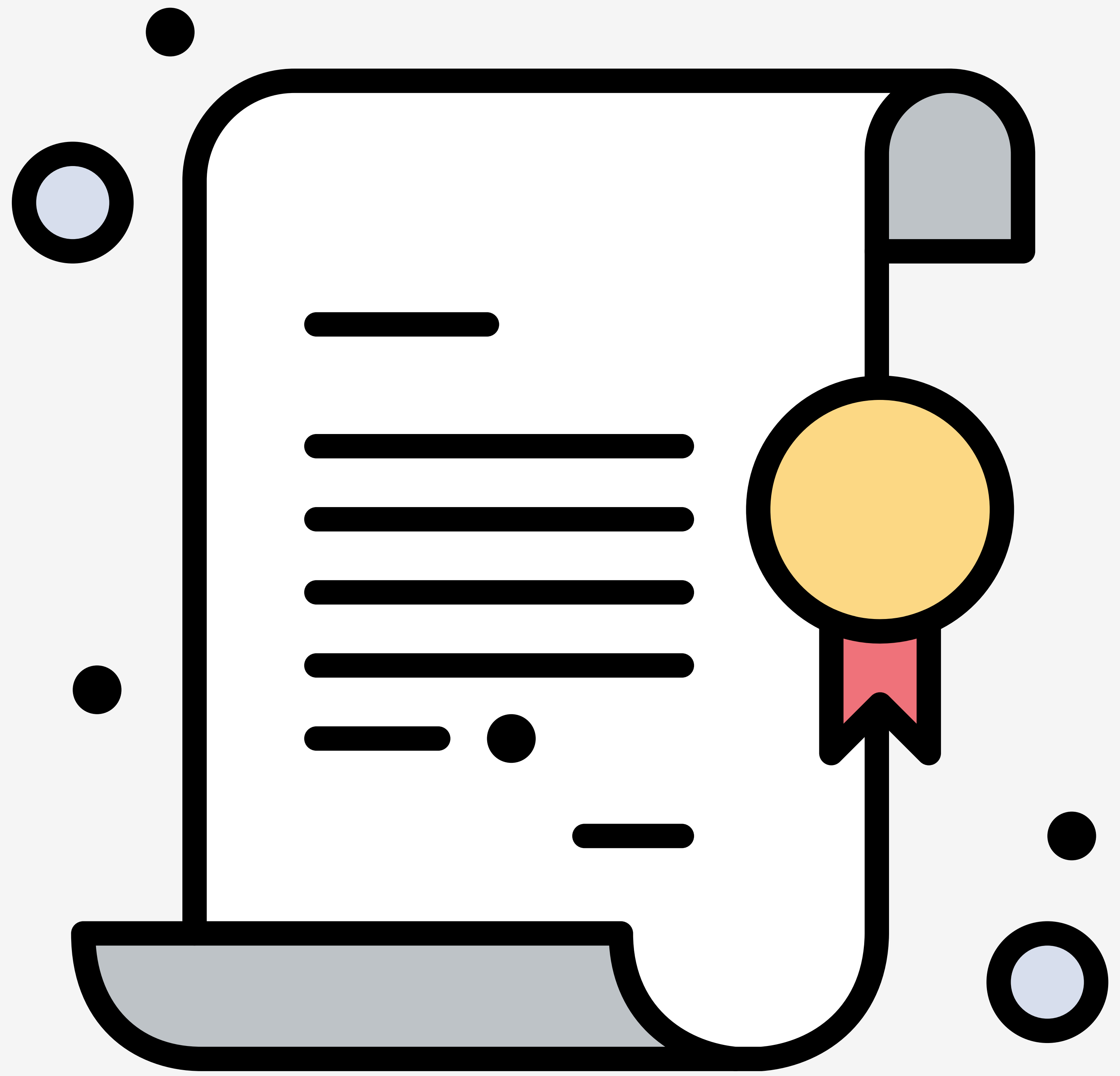
- Lists are a sequence of values
- Elements can be separated by commas or spaces
- Parentheses can be used, but not required
- The `nth()` function is used to access a single element in a list by their index
- The index `1` will return the first element, the index `-1` will return the last element

```scss
/* Sass */
$font-family: Helvetica, sans-serif;
$border-radius: 0px 20px 0px 20px;
$colors: (red, green, blue);

box {
  width: 100px;
  height: 100px;
  border: 1px solid #333;
  border-radius: $border-radius;
  background-color: nth($colors, 2);
  font-family: $font-family;
}
```

```css
/* CSS */
box {
  width: 100px;
  height: 100px;
  border: 1px solid #333;
  border-radius: 0px 20px 0px 20px;
  background-color: green;
  font-family: Helvetica, sans-serif;
}
```

SASS CONTENT BLOCKS

SASS CONTENT BLOCK



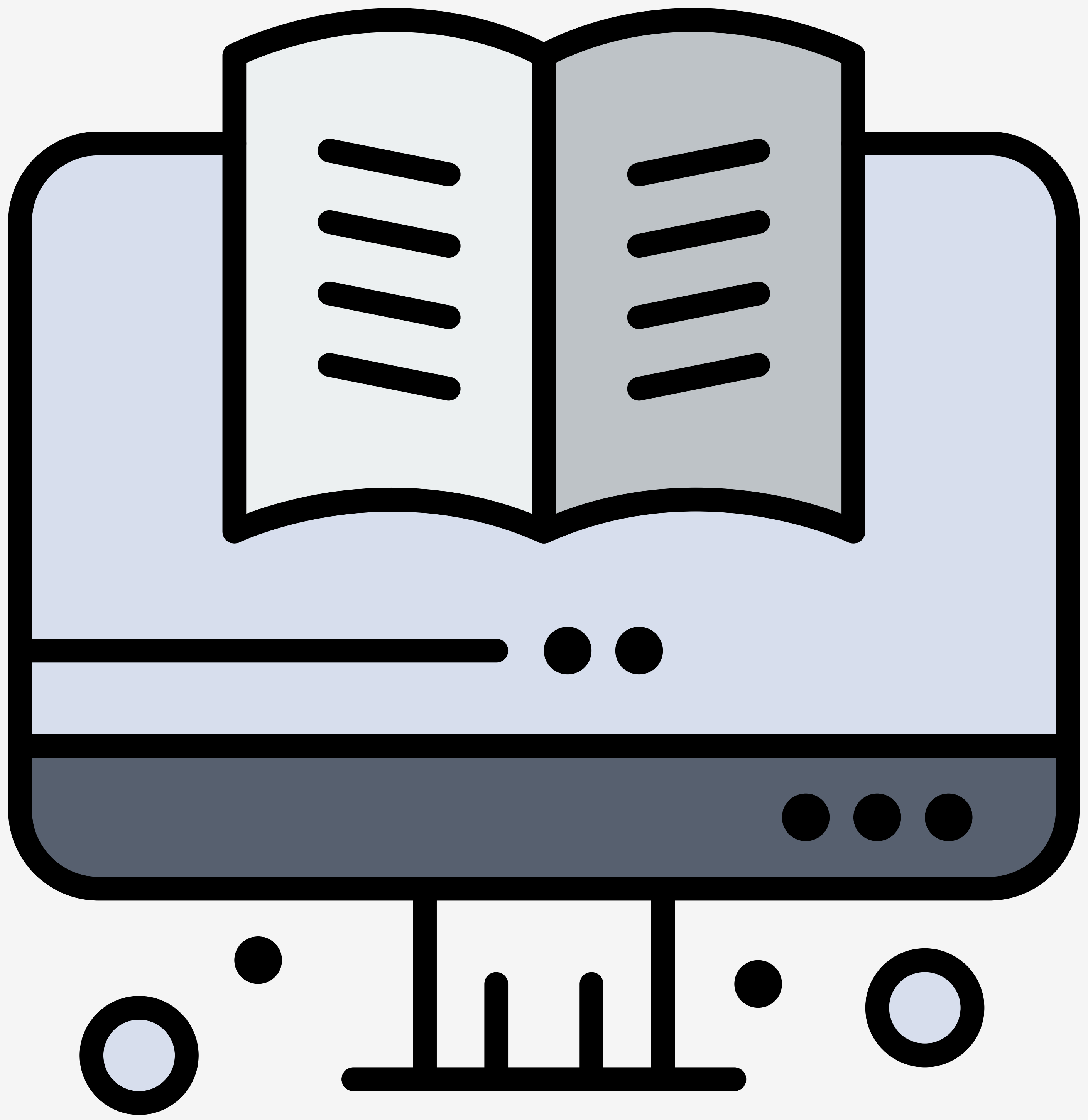
- A content block is a block of style that is passed to a mixin
- A content block can be included in a mixin using `@content` rule

```scss
/* Sass */
@mixin hover {
  &:not([disabled]):hover {
    @content;
  }
}

.button {
  background-color: #007bff;

  @include hover {
    background-color: #0069d9;
    cursor: pointer;
  }
}
```

```css
/* CSS */
.button {
  background-color: #007bff;
}

.button:not([disabled]):hover {
  background-color: #0069d9;
  cursor: pointer;
}
```

SASS @IF

SASS @IF



- The `@if` rule can be used to conditional evaluate blocks
- The `@if` rule expression will return `true` or `false`
- If the expression is `true` the block will be evaluated
- The `@else` rule can be added and will be evaluated if the `@if` expression is false
- The `@else if` rule can be used when more than one condition needs to be tested

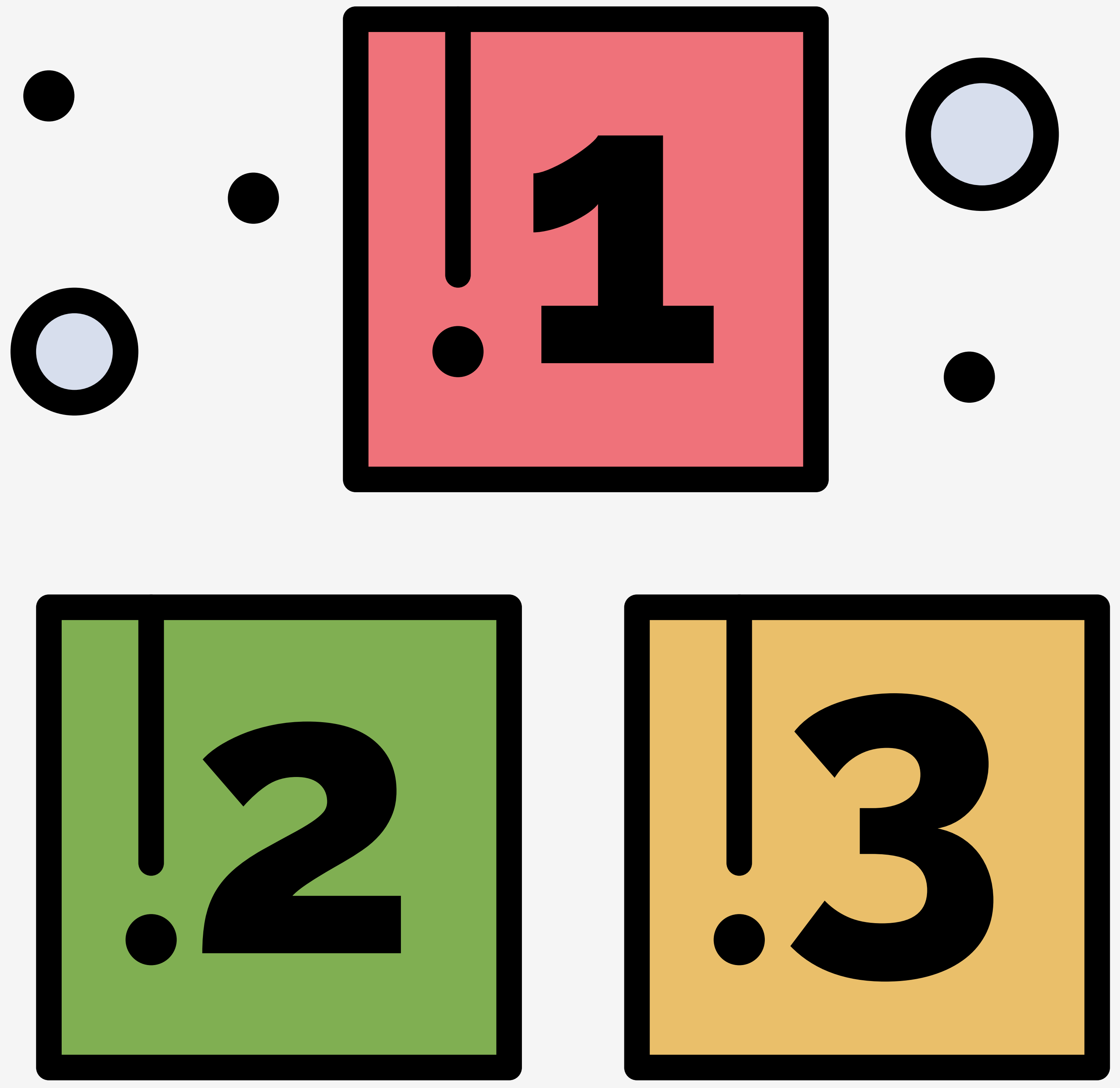
```
/* Sass */
@mixin breakpoint ($size) {
  @if $size=='medium' {
    @media screen and (min-width: 640px) {
      @content;
    }
  }

  @else if $size=='large' {
    @media screen and (min-width: 1024px) {
      @content;
    }
  }

  @else {
    @error "Unknown size #{$size}."
  }
}
```

```scss
/* Sass */
main {
  display: grid;
  grid-gap: 5px;
  grid-template-columns: 1fr;

  @include breakpoint('medium') {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
  }

  @include breakpoint('large') {
    display: grid;
    grid-template-columns: 200px 1fr 1fr 200px;
  }
}
```

```css
/* CSS */
main {
  display: grid;
  grid-gap: 5px;
  grid-template-columns: 1fr;
}

@media screen and (min-width: 640px) {
  main {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
  }
}

@media screen and (min-width: 1024px) {
  main {
    display: grid;
    grid-template-columns: 200px 1fr 1fr 200px;
  }
}
```

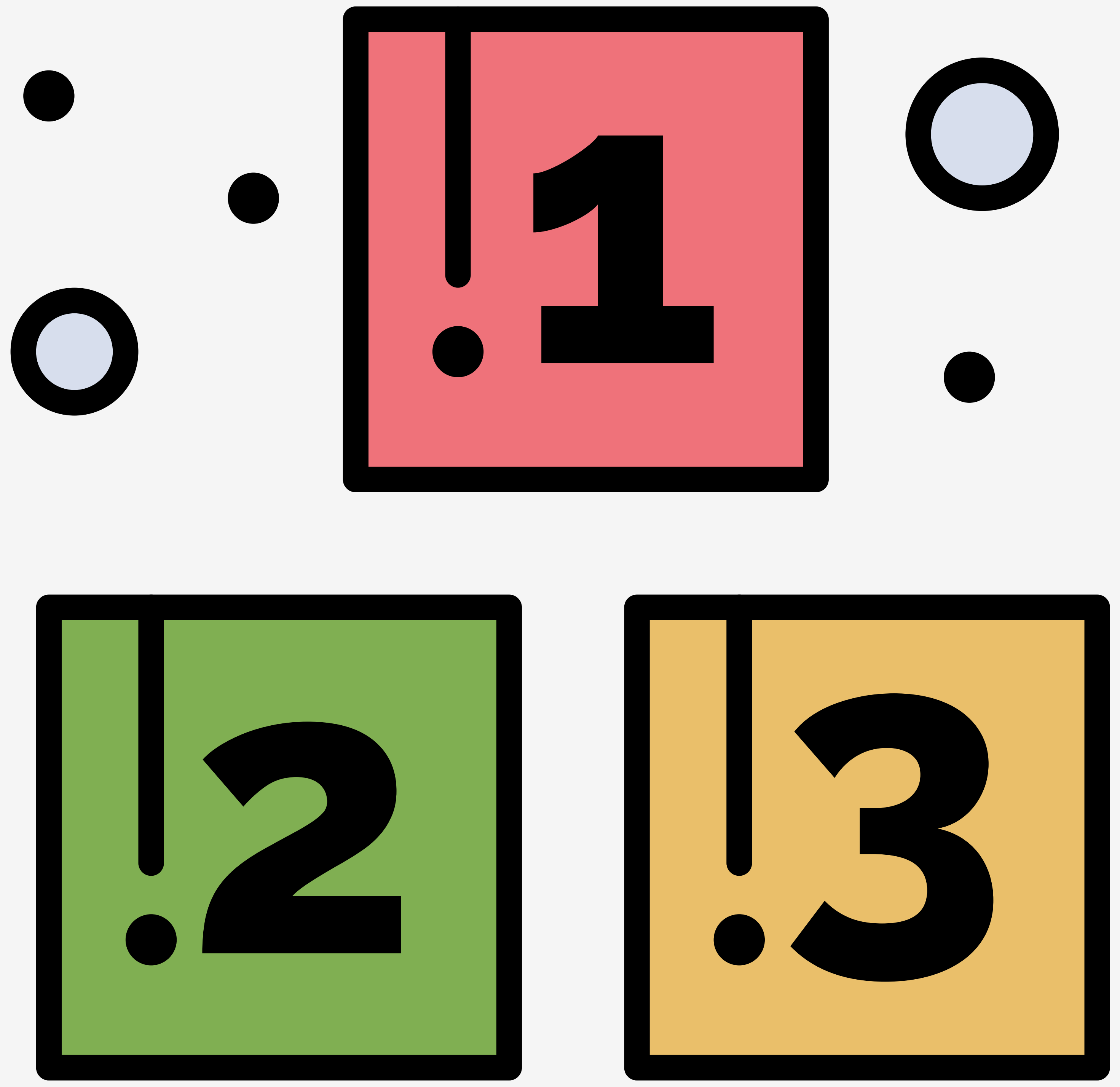
SASS @FOR

SASS @FOR



- The `@for` rule is used to count through a range of numbers
- The `@for` rule expression includes an iterator and a range of numbers
- The range of numbers can be connected using `to`, which excludes the final number, or `through`, which includes the final number
- For each iteration, the iterator is set to the current number in the range and the block is evaluated

SASS INTERPOLATION



- Interpolation is used to embed the results of an expression into CSS
- Interpolation syntax is `#{ }`
- Interpolation is often used to create selectors, property names and custom property values

```scss
/* Sass */
// loops 10 times
@for $i from 1 through 10 {
  .box:nth-child(#{$i}) {
    background-color: lighten(#000, $i * 10%);
  }
}
```

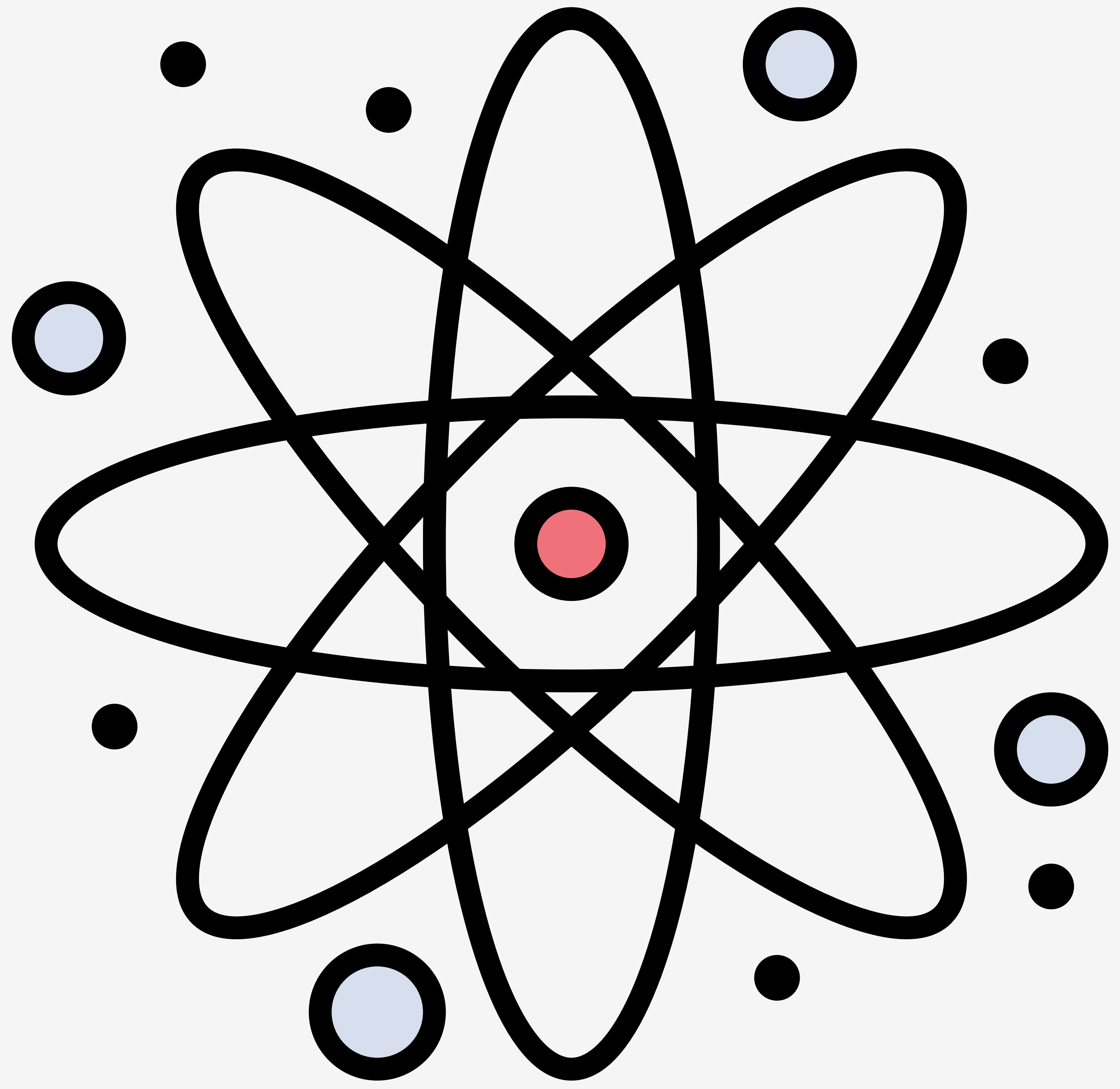
```css
/* CSS */
.box:nth-child(1) {
  background-color: #1a1a1a;
}

.box:nth-child(2) {
  background-color: #333333;
}

...

.box:nth-child(10) {
  background-color: white;
}
```

```scss
/* Sass */
// loops 9 times
@for $i from 1 to 10 {
  .box:nth-child(#{$i}) {
    background-color: lighten(#000, $i * 10%);
  }
}
```

```css
/* CSS */
.box:nth-child(1) {
  background-color: #1a1a1a;
}

.box:nth-child(2) {
  background-color: #333333;
}

...

.box:nth-child(9) {
  background-color: #e6e6e6;
}
```

SASS @EACH

SASS @EACH



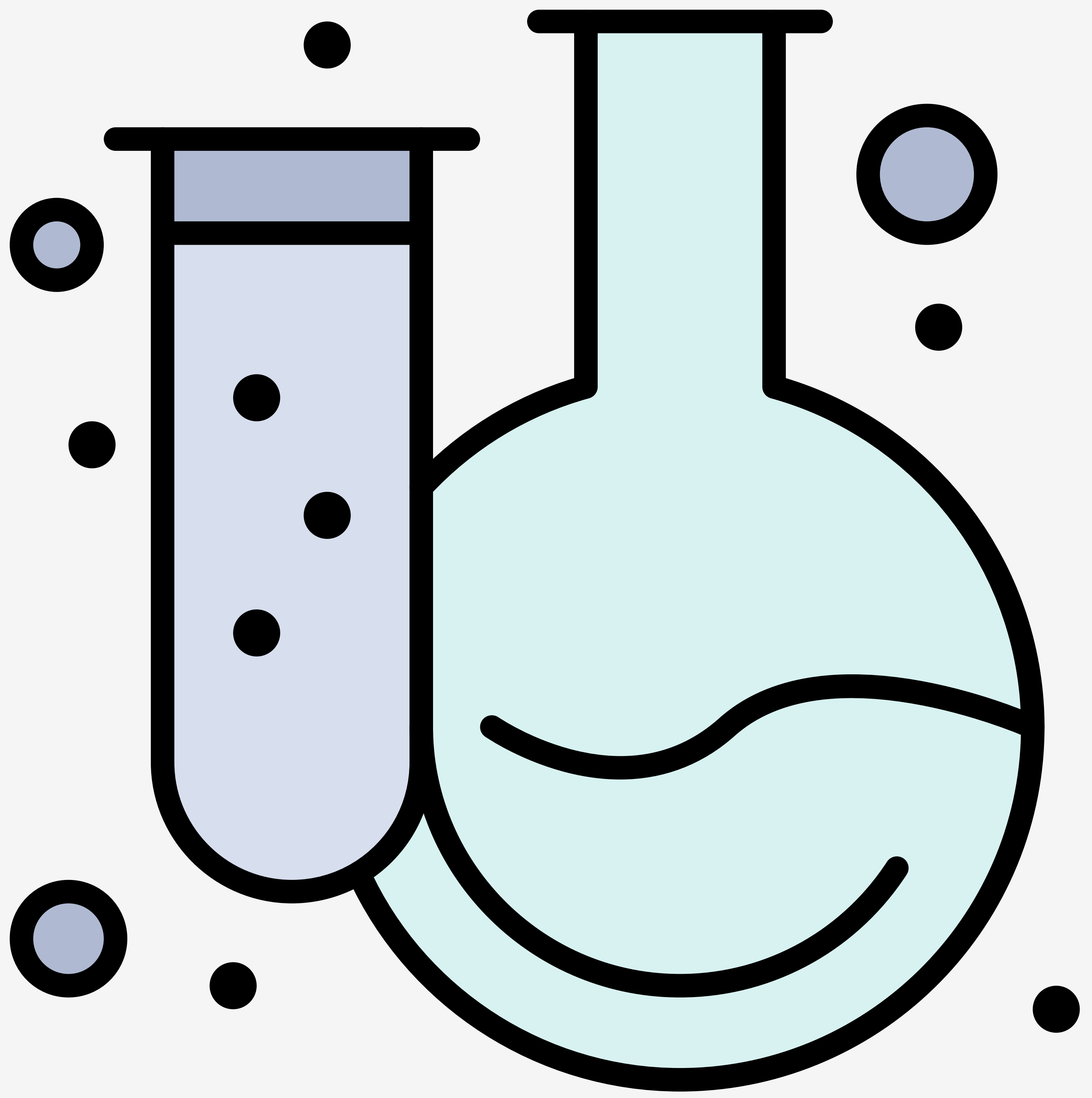
- The `@each` rule will iterate over each element in a list or map
- For each element, the block will be evaluated
- The block will have access to the key and value of each element

```scss
/* Sass */
$colors: red, blue, green;

%box {
  display: inline-block;
  width: 100px;
  height: 100px;
}

@each $color in $colors {
  .box-#{$color} {
    @extend %box;
    background-color: $color;
  }
}
```

```css
/* CSS */
.box-red, .box-blue, .box-green, .box-orange {
  display: inline-block;
  width: 100px;
  height: 100px;
}

.box-red {
  background-color: red;
}

.box-blue {
  background-color: blue;
}

.box-green {
  background-color: green;
}
```

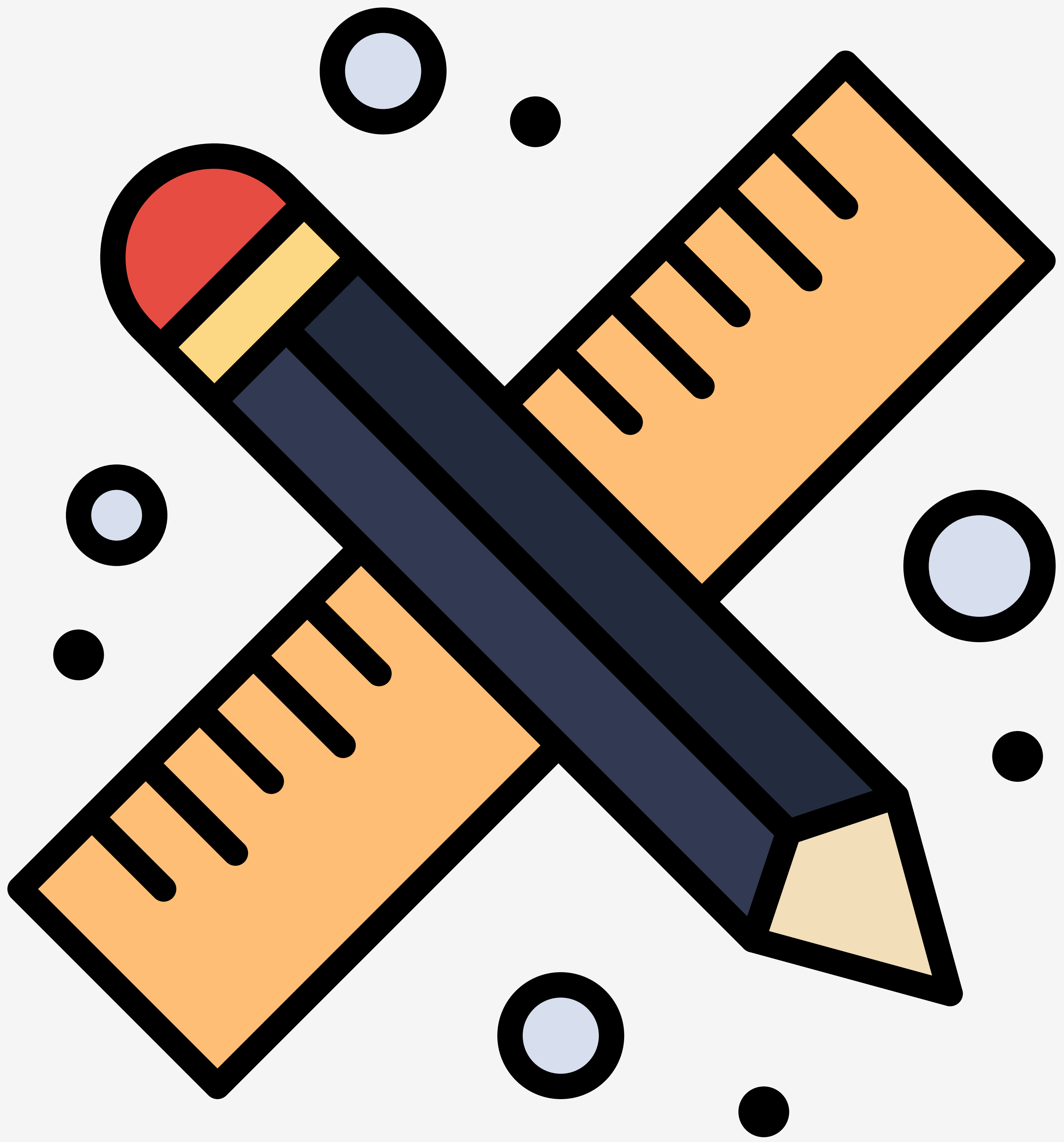
HANDS-ON

SASSY CATS



- *FORK THE PEN!*
- Use Sass to create the CSS styles to make cats different sizes and colors
- Use a `@for` rule to create the sizes classes.
- Create a list and use the `@each` rule to create the colors
- Submit the URL to your pen
- *DUE:* **Thu. Apr 9 @ 11:59 PM**

SASSY SHAPES



- Create 3 different shapes in 5 different colors using Sass
- Dynamically create the CSS using Mixins, `@extend`, and loops
- *TEST YOUR CODE*
- Submit the `.scss` file to BrightSpace
- *DUE:* **Thu. Apr. 9 @ 11:59 PM**

NEXT TIME...

- Work Periods