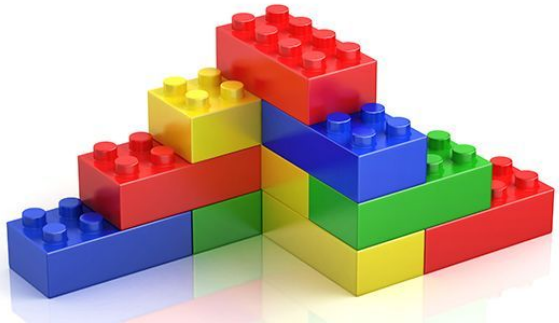
OBJECTS

FUNCTIONS -> CLASSES -> OBJECTS

WHAT ARE THE BASIC BUILDING BLOCKS OF COMPUTER SOFTWARE?

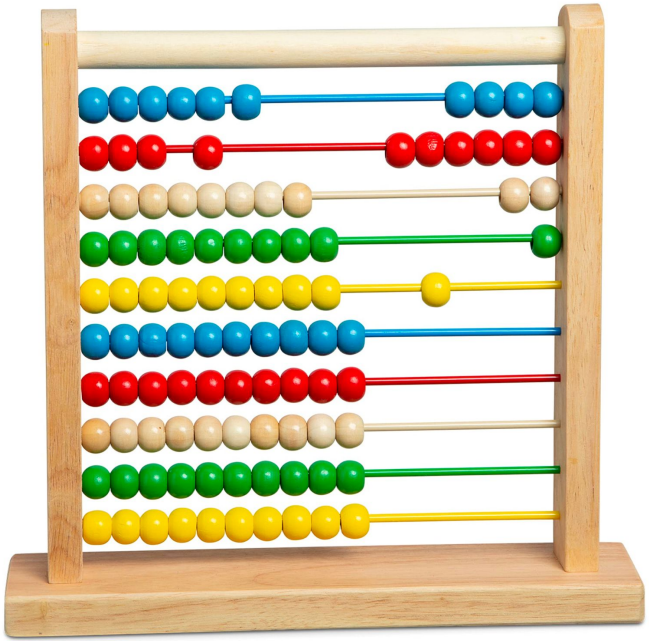


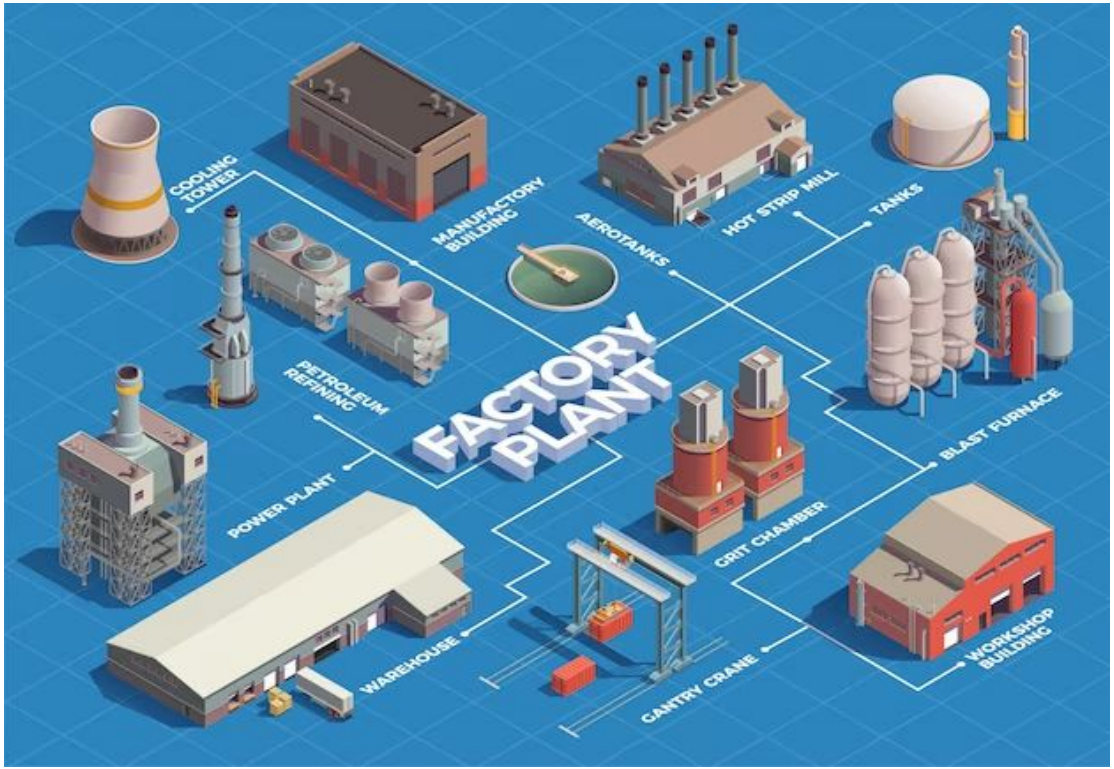

Stable
2310/2310
0/10
You need to build more houses.
AGE OF EMPIRES
HD EDITION

$x \in$
N

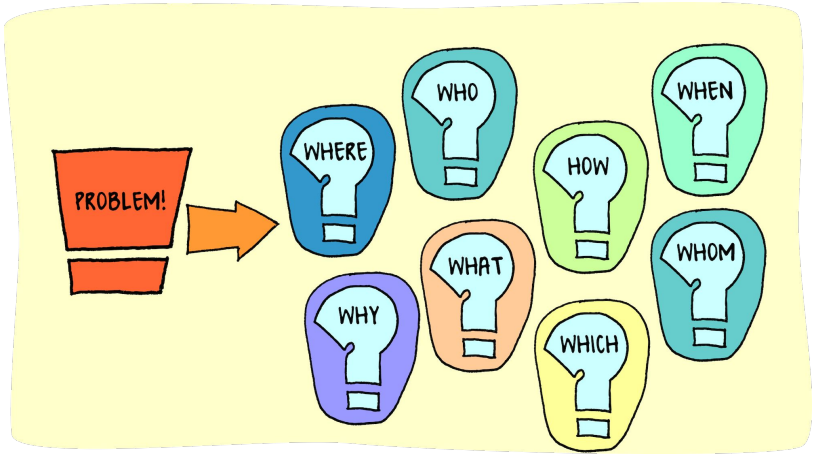
$$S = \{ x \in \mathbf{N} \mid x > 1 \}$$


1 one
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine
10 ten




FACTORY PLANT
COOLING TOWER
MANUFACTORY BUILDING
AEROTANKS
HOT STRIP MILL
TANKS
PETROLEUM REFINING
POWER PLANT
BLAST FURNACE
GRIT CHAMBER
WAREHOUSE
GANTRY CRANE
WORKSHOP BUILDING


TASK LIST!
1.
2.
3.
4.


PROBLEM!
WHERE
WHO
WHEN
HOW
WHAT
WHOM
WHY
WHICH

VARIABLES
CONDITIONAL LOGIC
LOOPING STATEMENTS
ARRAYS

VARIABLES
CONDITIONAL LOGIC
LOOPING STATEMENTS
ARRAYS

=>

ALGORITHMS

ALGORITHMS ARE TASK-SPECIFIC PROBLEM SOLVING SEQUENCES. USING OBJECT-ORIENTED PROGRAMMING, WE CAN KNIT TOGETHER A PIECE OF SOFTWARE COMPOSED OF MULTIPLE ALGORITHMS THAT CAN SOLVE AN AGGREGATE OF POSSIBLE PROBLEMS.

FUNCTIONS

```
void setup(){

  size(400, 400);

}

void draw(){

  rect(width/2, height/2, 10, 10);
}
```

```
_________________________________
Console:
```

FUNCTIONS

```
int a = 1;
int b = 2;
boolean y = true;



void setup(){

  print(y);

}

void draw(){

  print(1 + 2);
}

—-----------------------
Console: true333333333333
333333333333333333333333
```

FUNCTIONS

```
void setup(){

  size(400, 400);
  background(200);
  stroke(255, 0, 0);
}


void draw(){

  rect(mouseX, mouseY, 10,
10);

}

—----------------------
Console:
```

FUNCTIONS

```
void setup(){
  size(400, 400);
}

void draw(){

  squareBot(100, 100, 0);
  squareBot(200, 200, 255);

}


void squareBot(float x, float y, float s){

  stroke(s);
  rect(x, y, 10, 10);
}
```

FUNCTIONS

```
void setup(){
  size(400, 400);
}

void draw(){

  squareBot(100, 100, 0);
  squareBot(200, 200, 255);

}


void squareBot(float x, float y, float s){

  stroke(s);
  rect(x + mouseX, y + mouseY, 10, 10);
}
```

CLASSES

```
SquareBot square = new SquareBot(100, 200, 100);


void setup(){
  size(400, 400);
}

void draw(){

  square.run();
}
```

```
class SquareBot{

  float xpos, ypos, scolor;

  SquareBot(float x, float y, float s){
    xpos = x;
    ypos = y;
    scolor = s;
  }

  void run(){

    render();
  }

  void render(){

    stroke(scolor);
    rect(xpos, ypos, 10, 10);
  }

}
```

CLASSES

```
SquareBot square1 = new SquareBot(100, 200, 10);
SquareBot square2 = new SquareBot(300, 200, 20);
SquareBot square3 = new SquareBot(400, 200, 80);
SquareBot square4 = new SquareBot(500, 200, 100);
SquareBot square5 = new SquareBot(600, 200, 200);



void setup(){
  size(800, 800);
}

void draw(){

  square1.run();
  square2.run();
  square3.run();
  square4.run();
  square5.run();
}
```

```
class SquareBot{

  float xpos, ypos, scolor;

  SquareBot(float x, float y, float s){
    xpos = x;
    ypos = y;
    scolor = s;
  }

  void run(){

    render();
  }

  void render(){

    stroke(scolor);
    fill(scolor);
    rect(xpos, ypos, 10, 10);
  }

}
```


1 one
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine
10 ten

$x \in$
N

$$\sum_{i=3}^{14} i$$

CLASSES

array of SquareBot squares

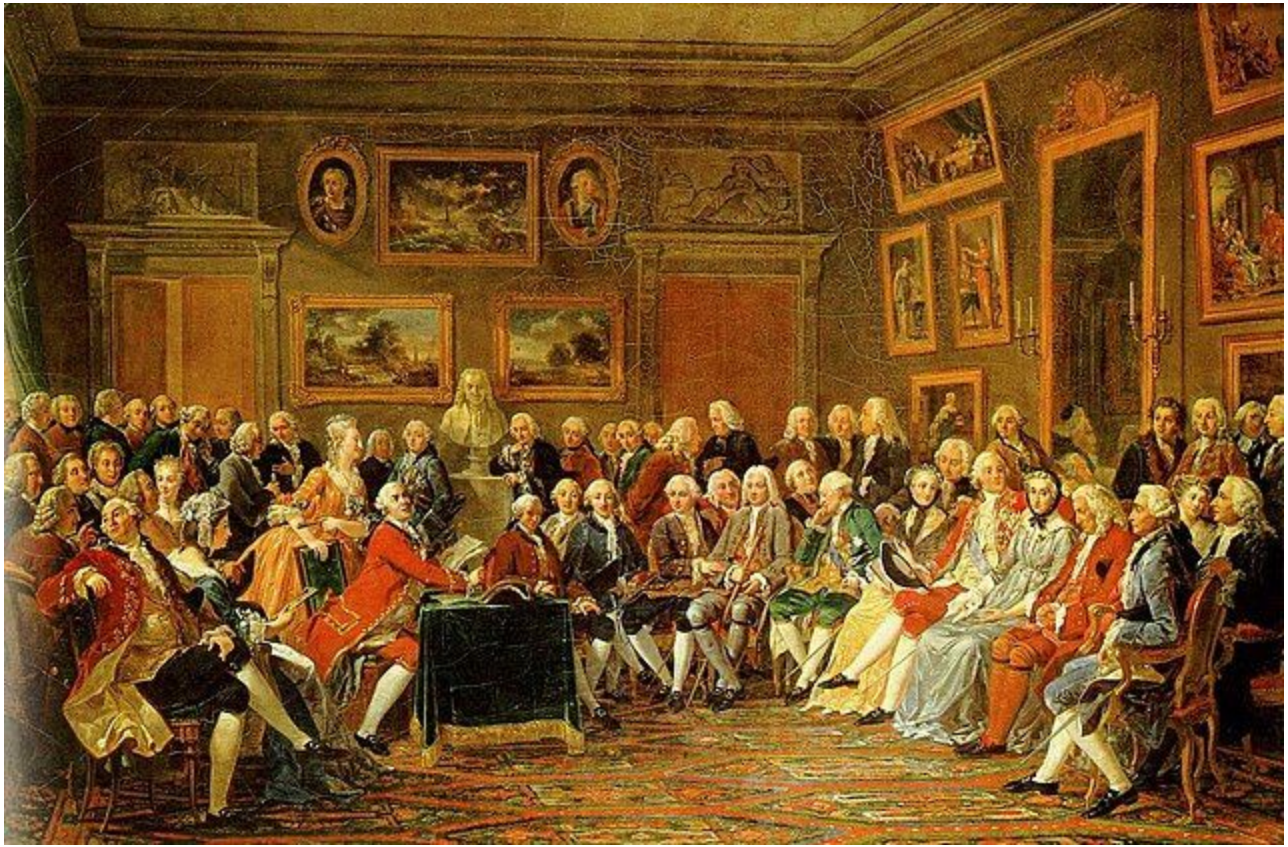
a *for loop* to count through that array of squares

class SquareBot

CLASSES

```
SquareBot[] squares = new SquareBot[100];

void setup(){
  size(800, 800);
  background(70, 150, 255);
}

void draw(){


  for(int i = 0; i < 10; i++){
   for(int j = 0; j < 10; j++){

     squares[j] = new SquareBot(i * 12, j * 12,
255/10 * i);
     squares[j].run();
    }
  }

}
```

```
class SquareBot{

  float xpos, ypos, fcolor;

  SquareBot(float x, float y, float f){
    xpos = x;
    ypos = y;
    fcolor = f;
  }

  void run(){

    render();
  }

  void render(){

    noStroke();
    fill(fcolor);
    rect(xpos, ypos, 10, 10);
  }

}
```



OBJECTS

```
SquareBot[] squares = new SquareBot[100];

void setup(){
  size(800, 800);
  background(70, 150, 255);
}

void draw(){

  for(int i = 0; i < 10; i++){
   for(int j = 0; j < 10; j++){

     if(random(1) > 0.5){

     squares[j] = new SquareBot(i * 12, j * 12,
255/10 * i);
     } else {
     squares[j] = new SquareBot(i * 12, j * 12,
0);
     }
     squares[j].run();
   }
  }
}
```

```
class SquareBot{

  float xpos, ypos, fcolor;

  SquareBot(float x, float y, float f){
    xpos = x;
    ypos = y;
    fcolor = f;
  }

  void run(){

    render();
  }

  void render(){

    noStroke();
    fill(fcolor);
    rect(xpos, ypos, 10, 10);
  }

}
```

OBJECTS

```
SquareBot[] squares = new SquareBot[100];

void setup(){
  size(800, 800);
  background(70, 150, 255);

  for(int i = 0; i < 10; i++){
   for(int j = 0; j < 10; j++){

     if(random(1) > 0.5){
     squares[j] = new SquareBot(i * 12, j * 12,
255/10 * i);
     } else {
     squares[j] = new SquareBot(i * 12, j * 12,
0);
     }
     squares[j].run();
   }
  }
}

void draw(){

}
```

```
class SquareBot{

  float xpos, ypos, fcolor;

  SquareBot(float x, float y, float f){
    xpos = x;
    ypos = y;
    fcolor = f;
  }

  void run(){

    render();
  }

  void render(){

    noStroke();
    fill(fcolor);
    rect(xpos, ypos, 10, 10);
  }

}
```

OBJECTS

```
SquareBot[] squares = new SquareBot[100];

void setup(){
  size(800, 800);
  background(70, 150, 255);

  for(int i = 0; i < 10; i++){
   for(int j = 0; j < 10; j++){

     if(random(1) > 0.5){
     squares[j] = new SquareBot(i * 30, j * 30,
255/10 * i);
     } else {
     squares[j] = new SquareBot(i * 30, j * 30,
0);
     }
     squares[j].run();
   }
  }
}

void draw(){

}
```

```
class SquareBot{

  float xpos, ypos, fcolor;

  SquareBot(float x, float y, float f){
    xpos = x;
    ypos = y;
    fcolor = f;
  }

  void run(){

    render();
  }

  void render(){

    noStroke();
    fill(fcolor);

    for(int i = 0; i < 3; i++){
      for(int j = 0; j < 3; j++){
          rect(i * 11 + xpos, j * 11 + ypos, 10,
      }
    }
  }
}
```

WRITING A CLASS

```
class SquareBot{

    //VARIABLES

    //PARAMETERS

    //FUNCTIONS

  }
```

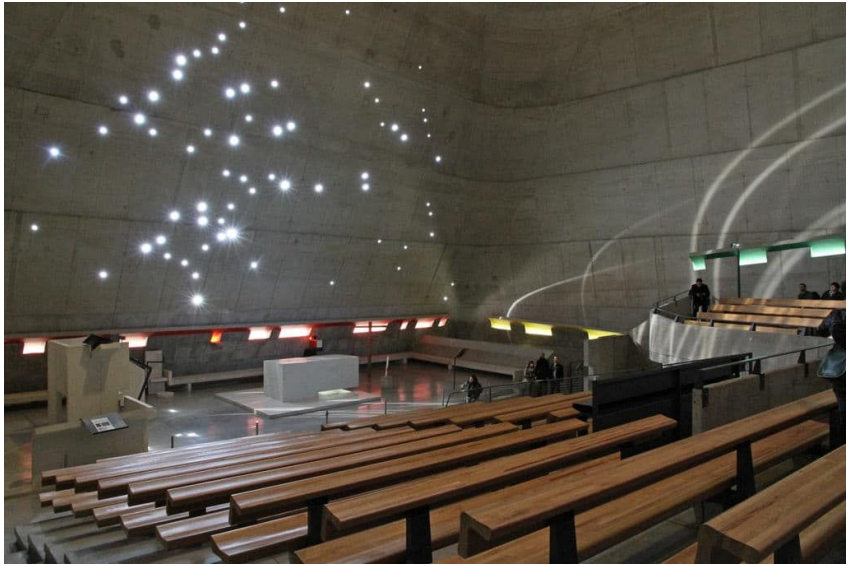
WRITING A CLASS

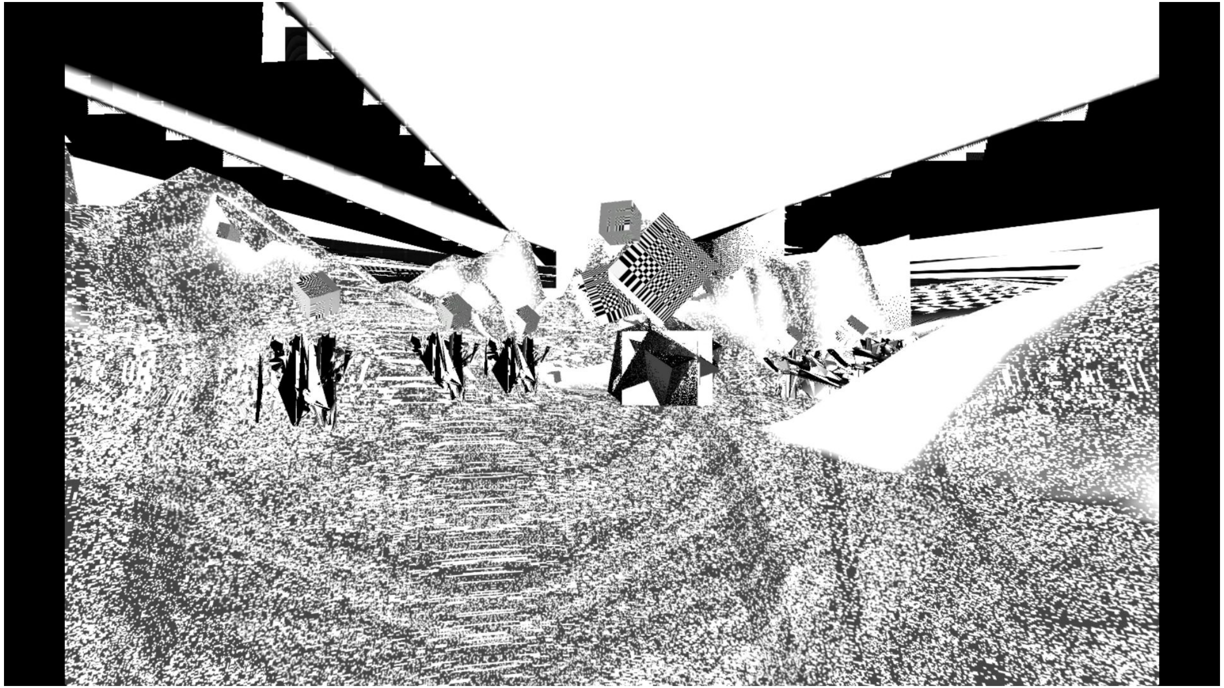
```
class SquareBot{

    //VARIABLES (DATA)

    //PARAMETERS (INPUT)

    //FUNCTIONS (OUTPUT)

  }
```

WRITING A CLASS

```
class SquareBot{

    //VARIABLES (DATA)
  float xpos, ypos, fcolor;

              //PARAMETERS (INPUT)
            SquareBot(float x, float y, float f){
              xpos = x;
              ypos = y;
              fcolor = f;
            }
                        //FUNCTIONS (OUTPUT)

                    void run(){
                      render();
                    }

                    void render(){
                      noStroke();
                      fill(fcolor);
                      for(int i = 0; i < 3; i++){
                        for(int j = 0; j < 3; j++){
                            rect(i * 11 + xpos, j * 11 + ypos, 10, 10);
                        }
                      }
                    }
}
```

```
class Class{

     Class(float x, float y, boolean t){

     }

     void classFunction(){

     }

}
```





Homework: create a custom class which you deploy in a loop to iterate a number of objects. Try to iterate a grid of objects that are spaced out unevenly. I.e., a grid of objects that are not spaced uniformly across the canvas. You could use the random() function or some algorithmic or hand rolled code.