Pseudocode

Ju An Park
MASc Candidate
Civil and Environmental Engineering
University of Waterloo, Canada

AE121: Computational Method

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Last updated: 2019-07-21

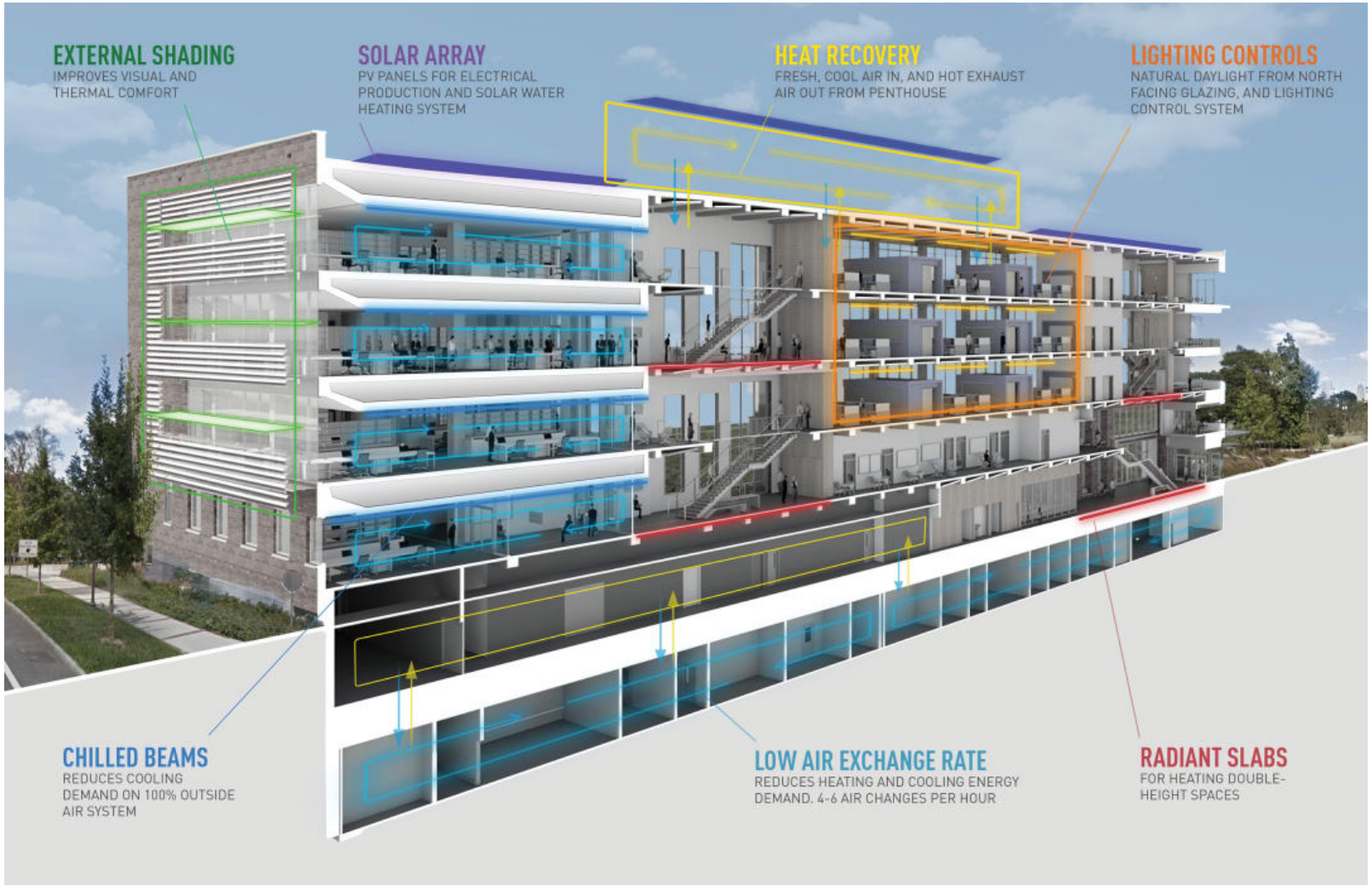
Good job!

- We have almost covered the fundamentals of MATLAB!

You deserve a good pat on the back ☺

- So, you now know code can automate boring stuff or do really cool data stuff!
- Before we dive into pseudocode, I'd like to show you some cool-ish stuff we can do with programming!

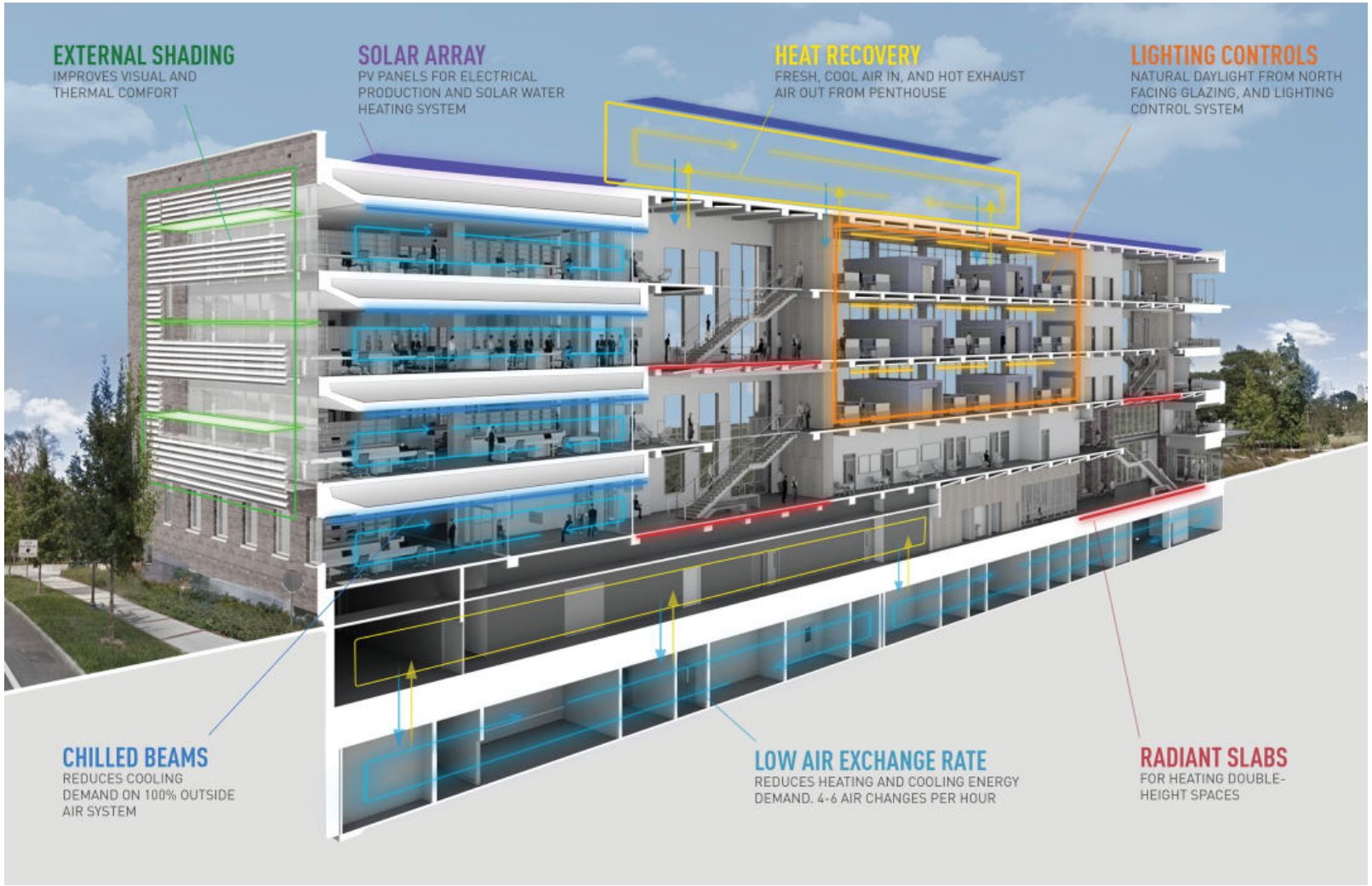
Coding Application Examples



Using code, you can:

- Model different building properties
  - Heat exchange rates[1]
  - Energy usage[2]/optimization
- Generative Design[3]
- Preliminary floor plan design[4]
- Interior Design Assistant[5]

Coding Application Examples



Using code, you can :

- Model different building properties
  - **Heat exchange rates**[1]
  - Energy usage[2]/optimization
- Generative Design[3]
- Preliminary floor plan design[4]
- Interior Design Assistant[5]

Coding Application Examples - 1: Hospital Heat Distribution Modelling (CFD)

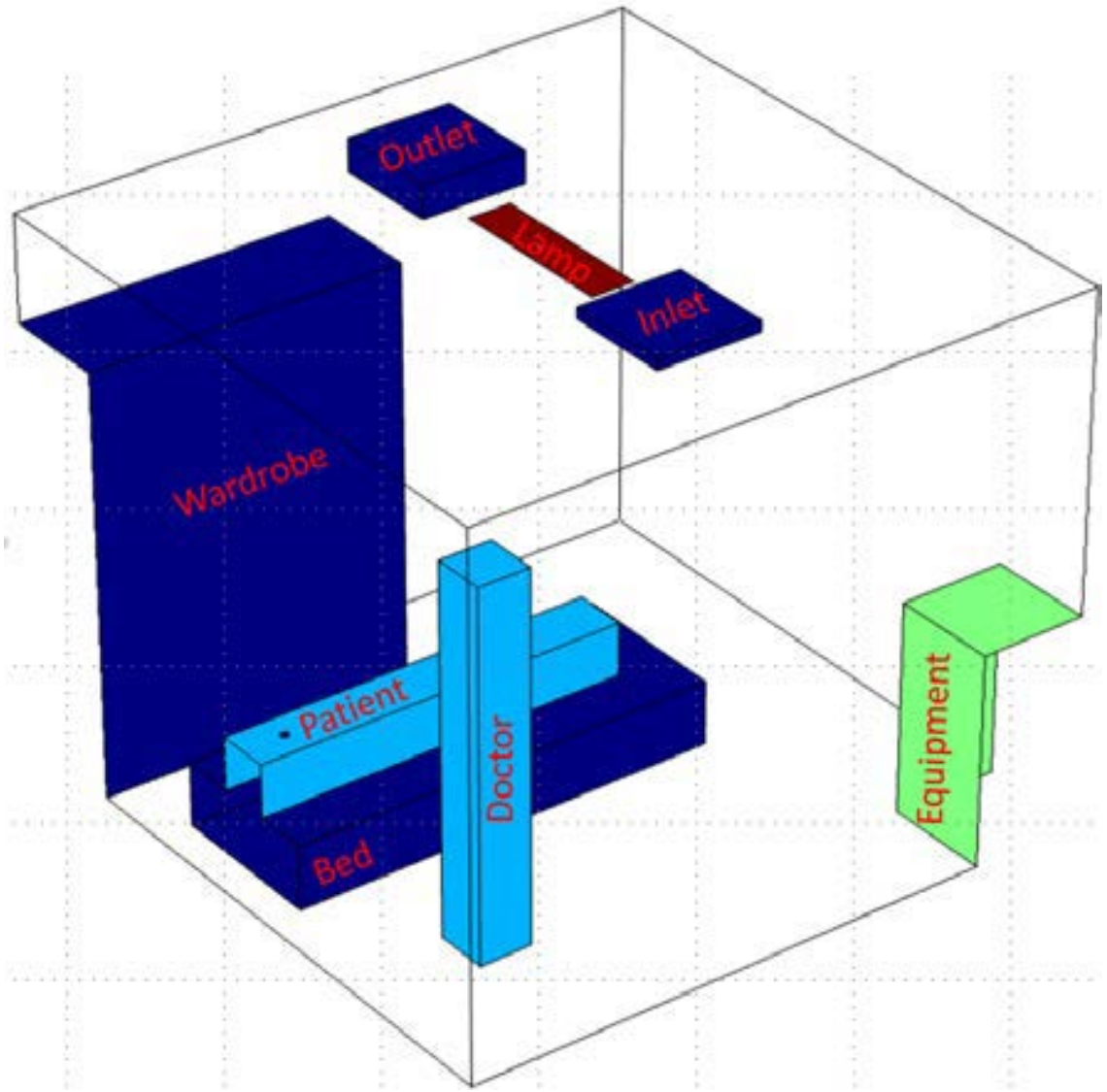


Figure 1. Room layout

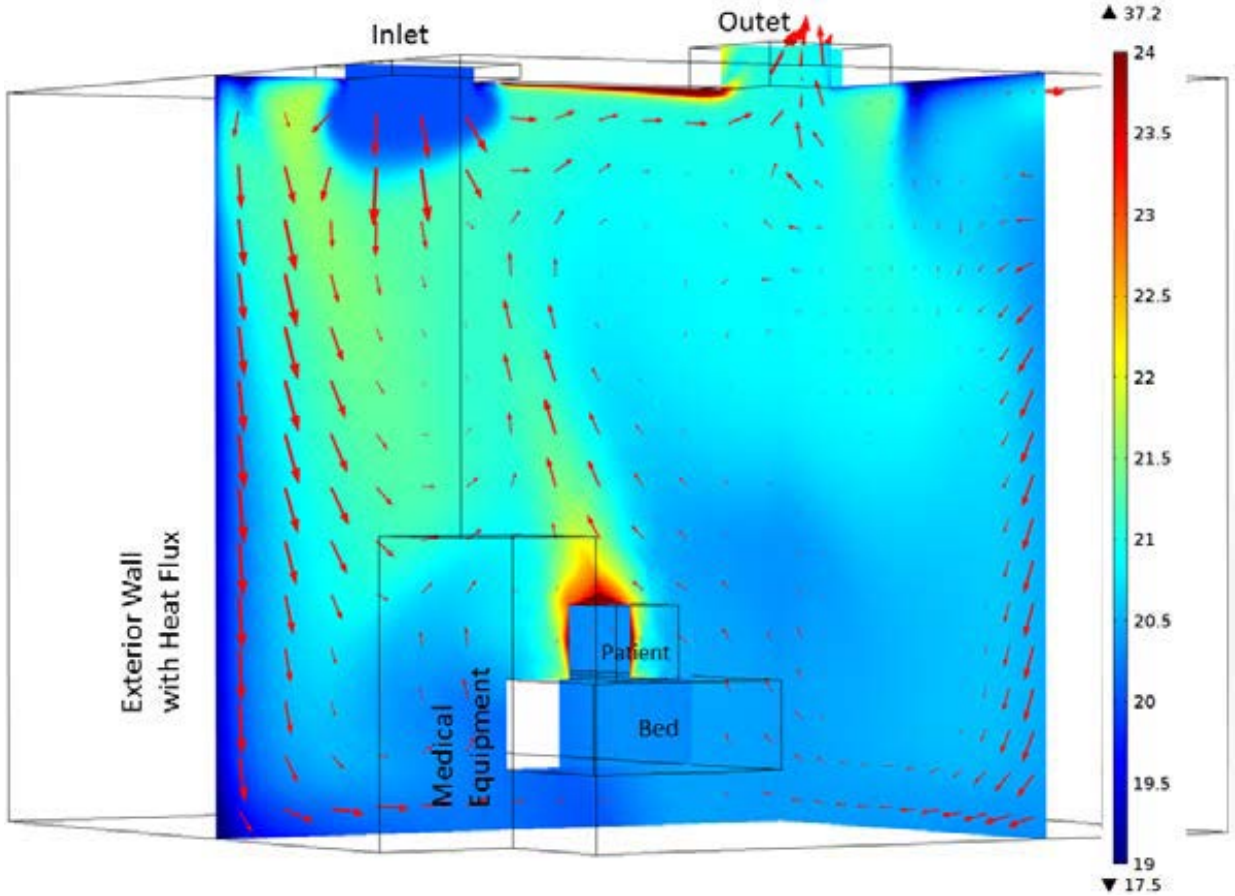


Figure 2. Temperature distribution and velocity vector in the room

Coding Application Examples - 1: Hospital Air Particle Movement Modelling (CFD)

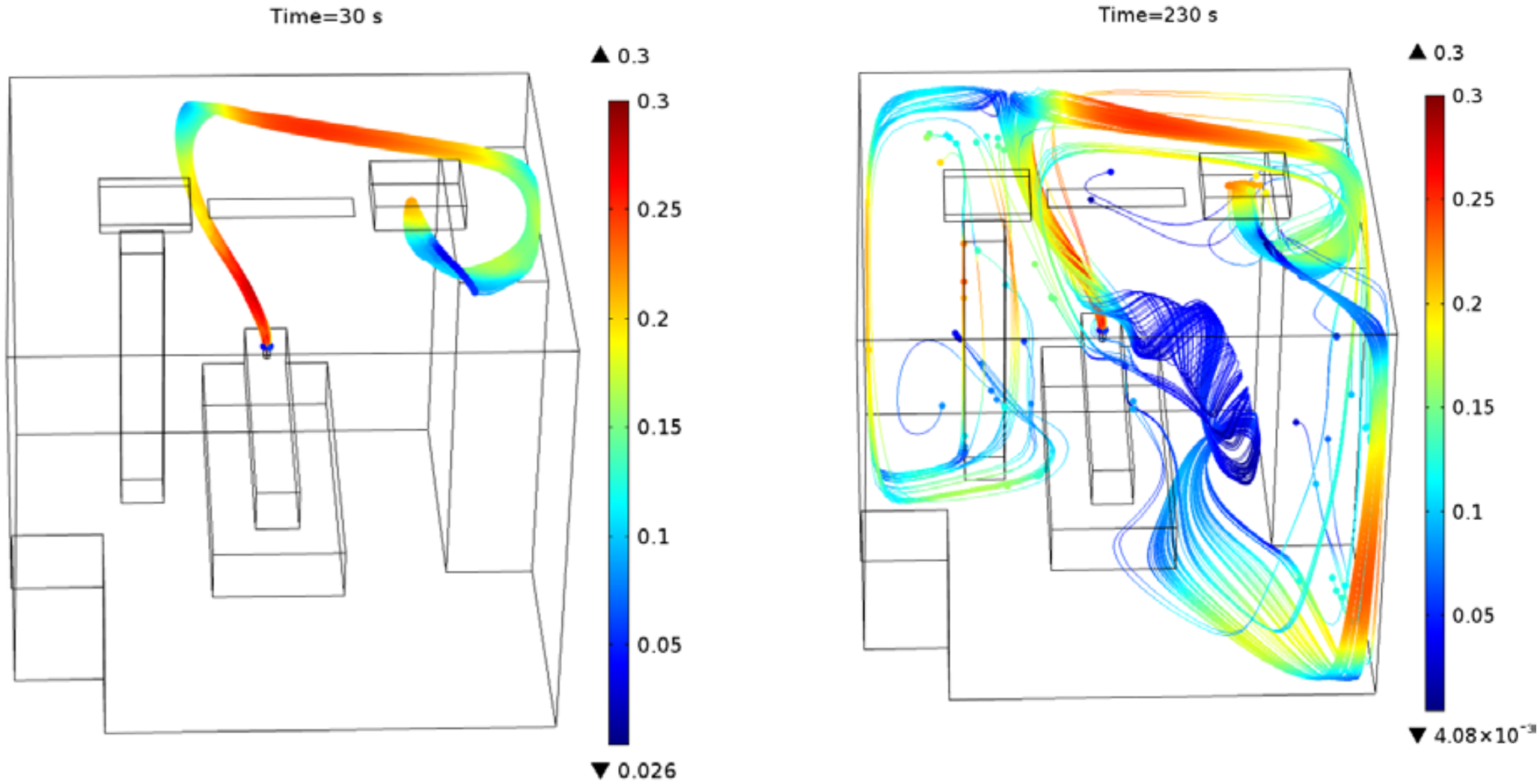
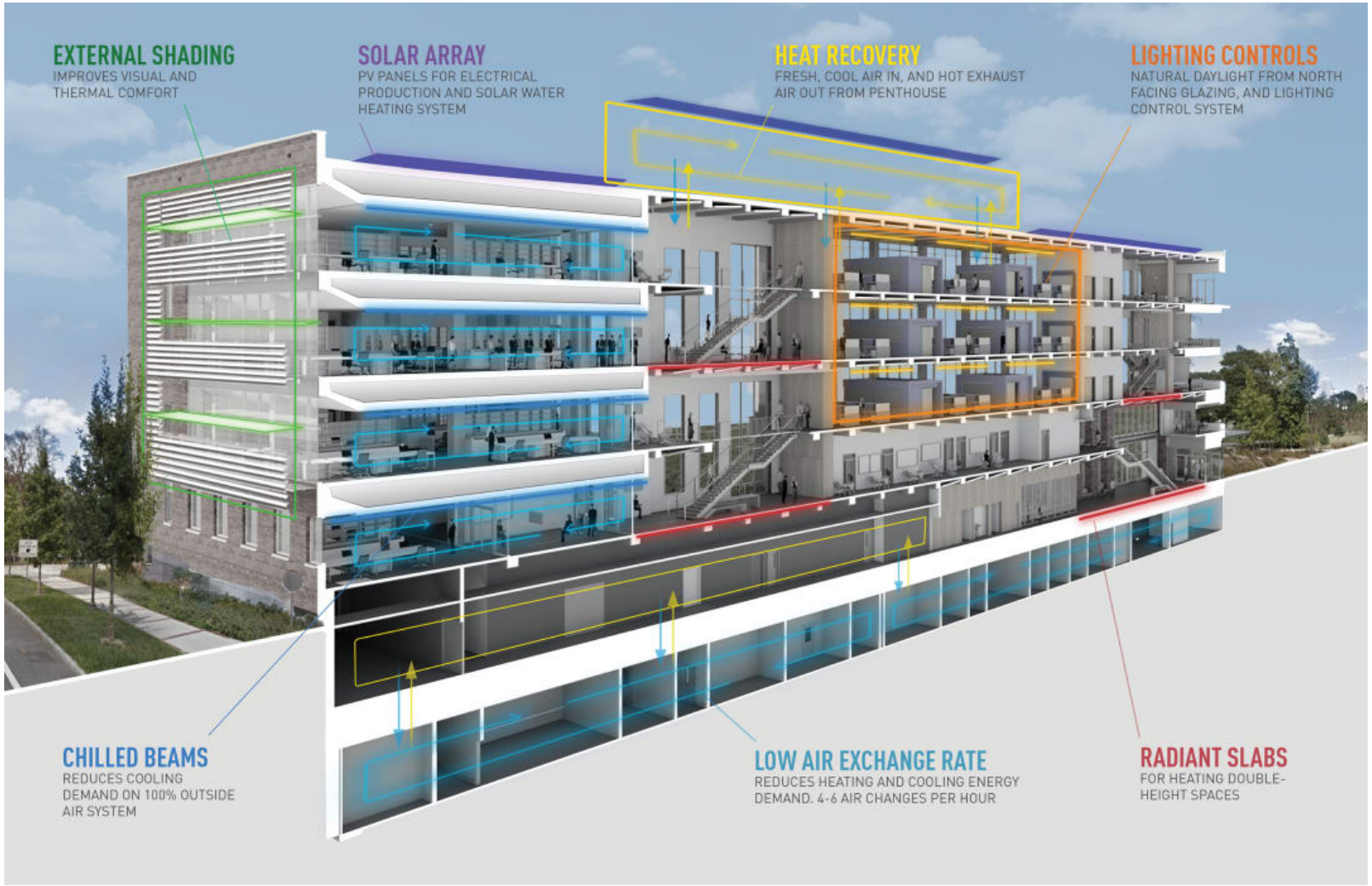


Figure 4. Particle tracing showing the motion of bacteria particles at 30 and 230 seconds after patient coughing.

Coding Application Examples - 1: Applications?

- Better understanding of different ventilation options leads to:
  - Increased energy efficiency
  - Reduce risk of airborne infection

Coding Application Examples



Using code, you can :

- Model different building properties
  - Heat exchange rates[1]
  - **Energy usage[2]/optimization**
- Generative Design[3]
- Preliminary floor plan design[4]
- Interior Design Assistant[5]

Coding Application Examples - 2: Building Energy Modelling

Fig. 1. Eletrosul building (10 000 m_2).
Modelling Software: DOE-2, BLAST, ESP-r, ENERGY-PLUS

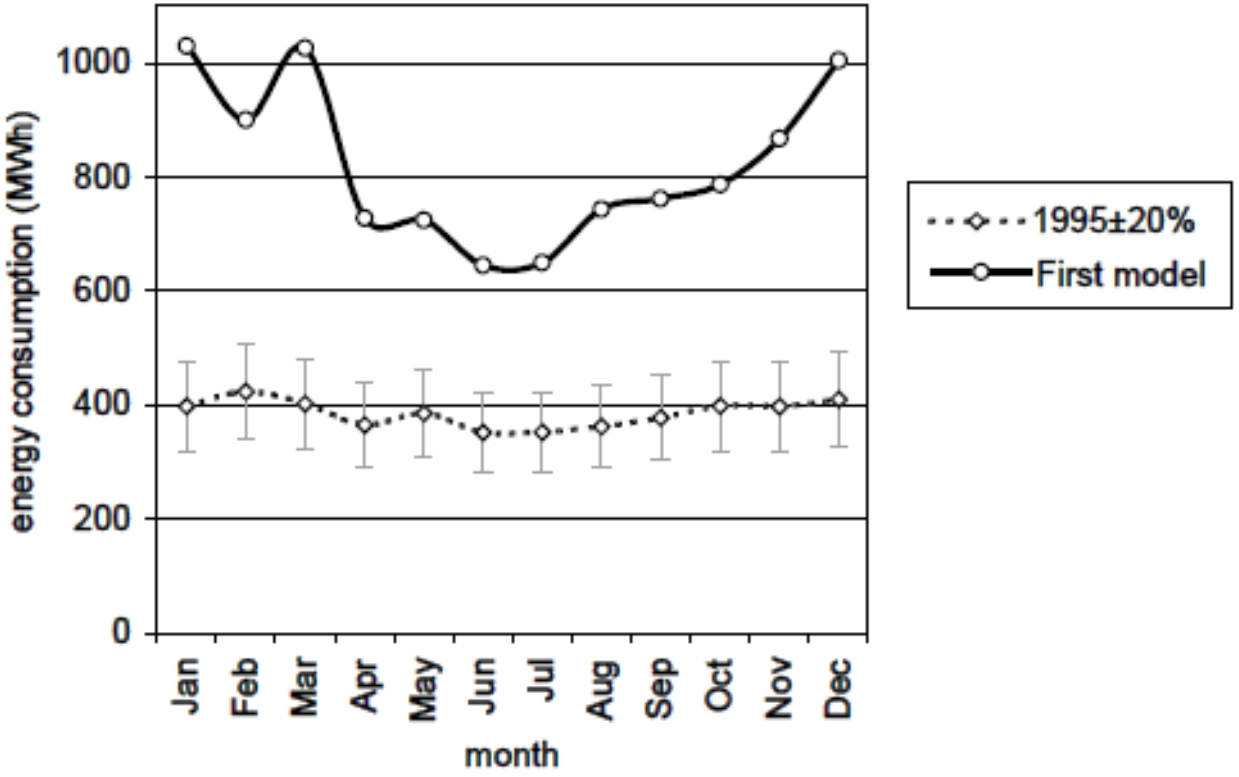


Fig. 2. First model of Eletrosul building.

Coding Application Examples - 2: Building Energy Modelling

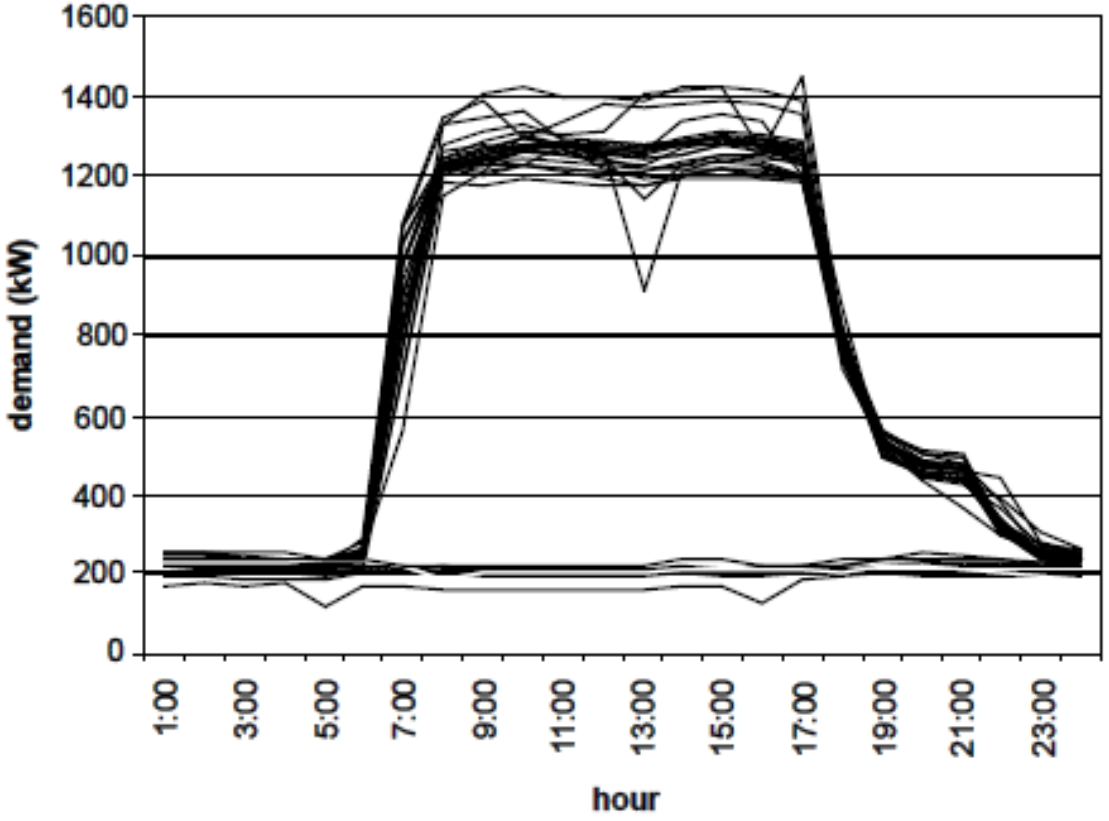


Fig. 3. Recorded demand.

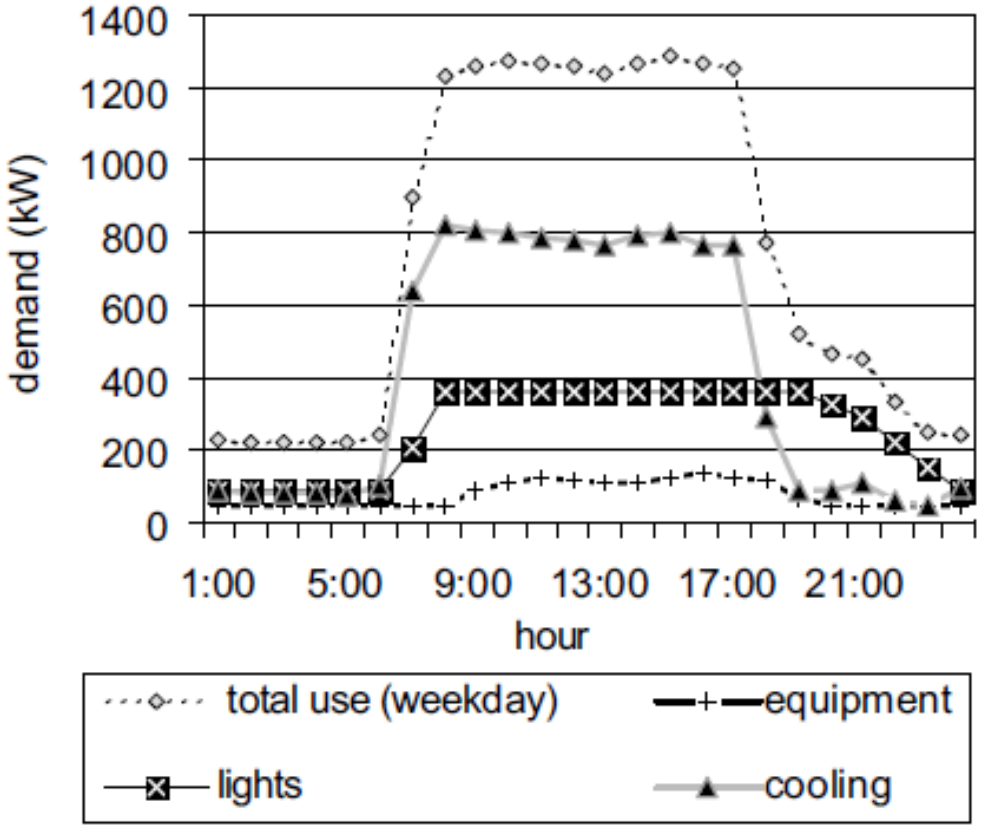


Fig. 9. Total and end-use energy demand.

Coding Application Examples - 2: Building Energy Modelling

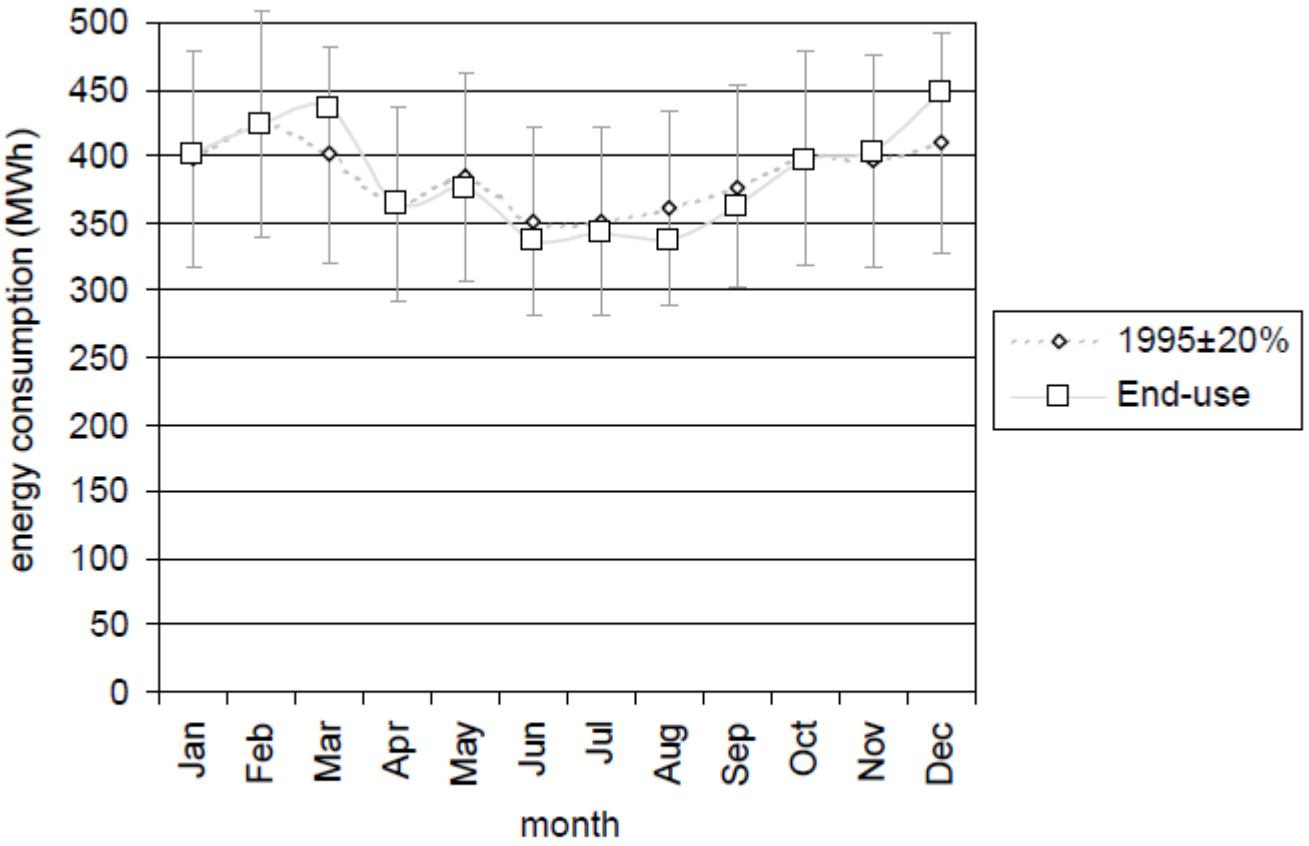


Fig. 10. Third calibrated model: end-use stage.

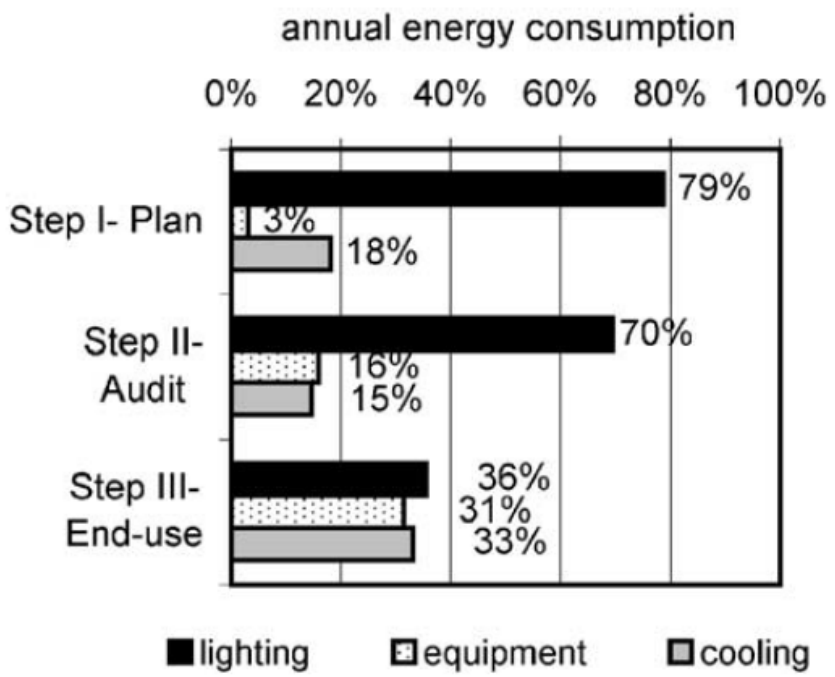
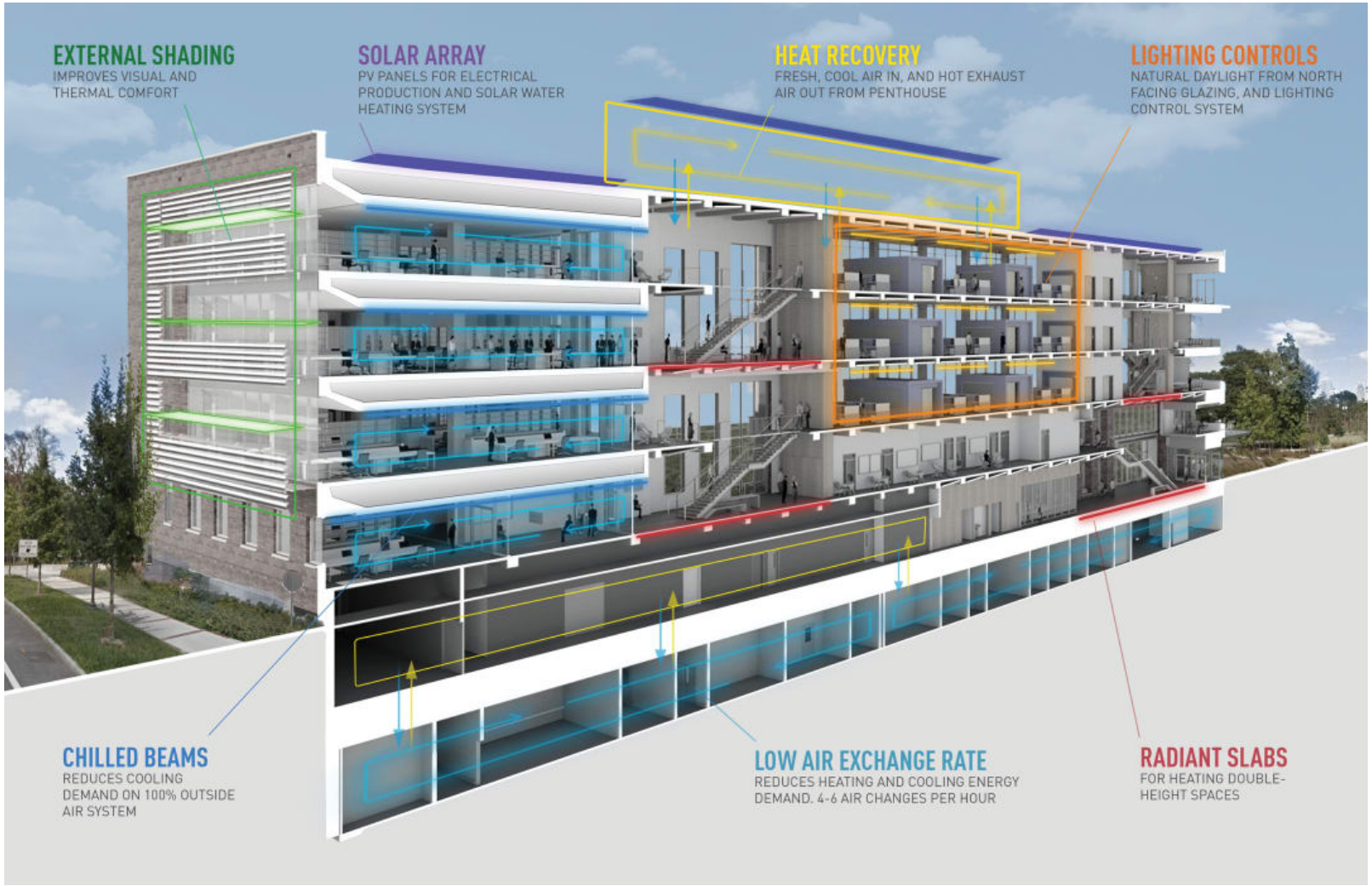


Fig. 11. End-use energy for different levels of modelling.

Coding Application Examples - 2: Applications?

- Better understanding of building energy usage
- Better deal with peak energy demand situations
- Determine CO2 emissions
- Better compare different energy options (life cycle analysis)

Coding Application Examples

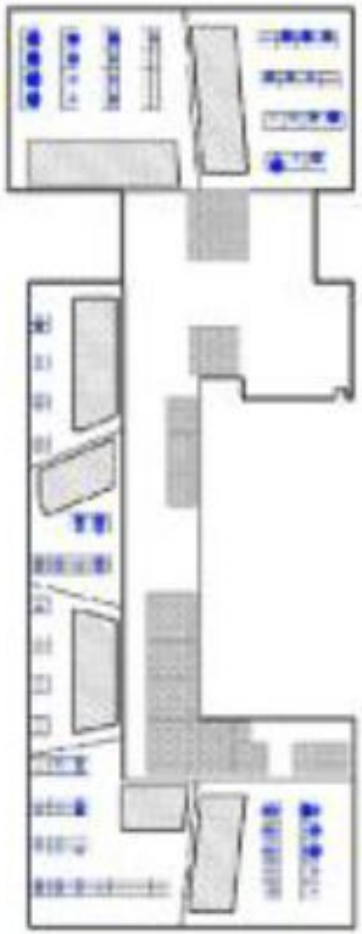


Using code, you can :

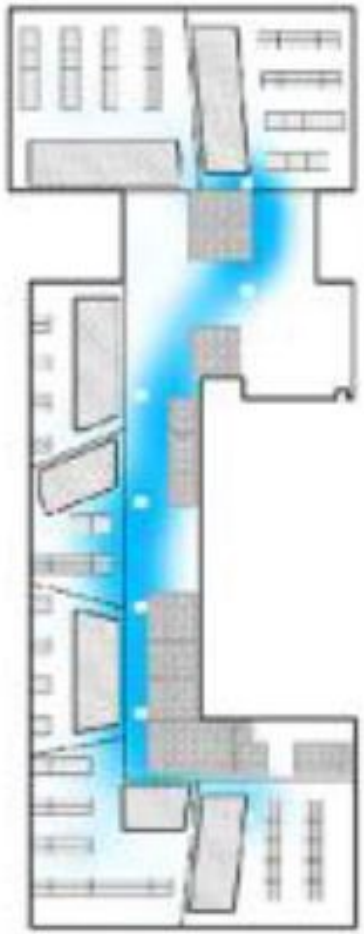
- Model different building properties
  - Heat exchange rates[1]
  - Energy usage[2]/optimization
- **Generative Design[3]**
- Preliminary floor plan design[4]
- Interior Design Assistant[5]

Coding Application Examples - 3: Generative Design

adjacency preference

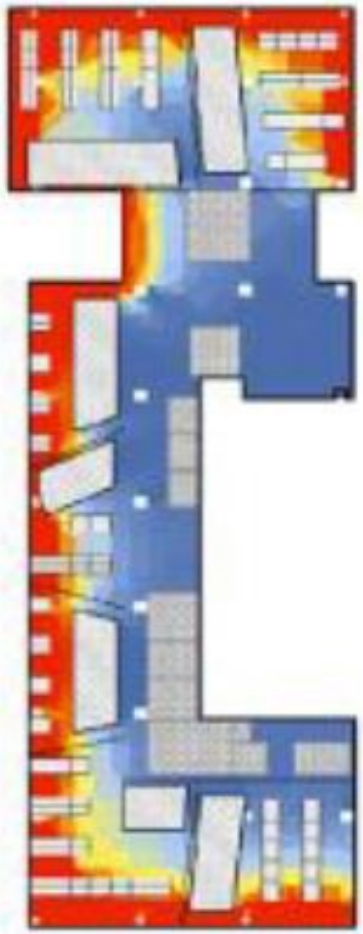


work style preference

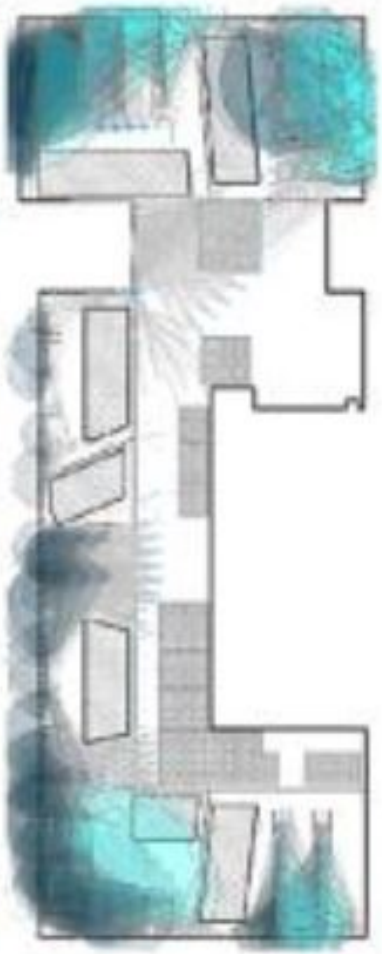


buzz

productivity

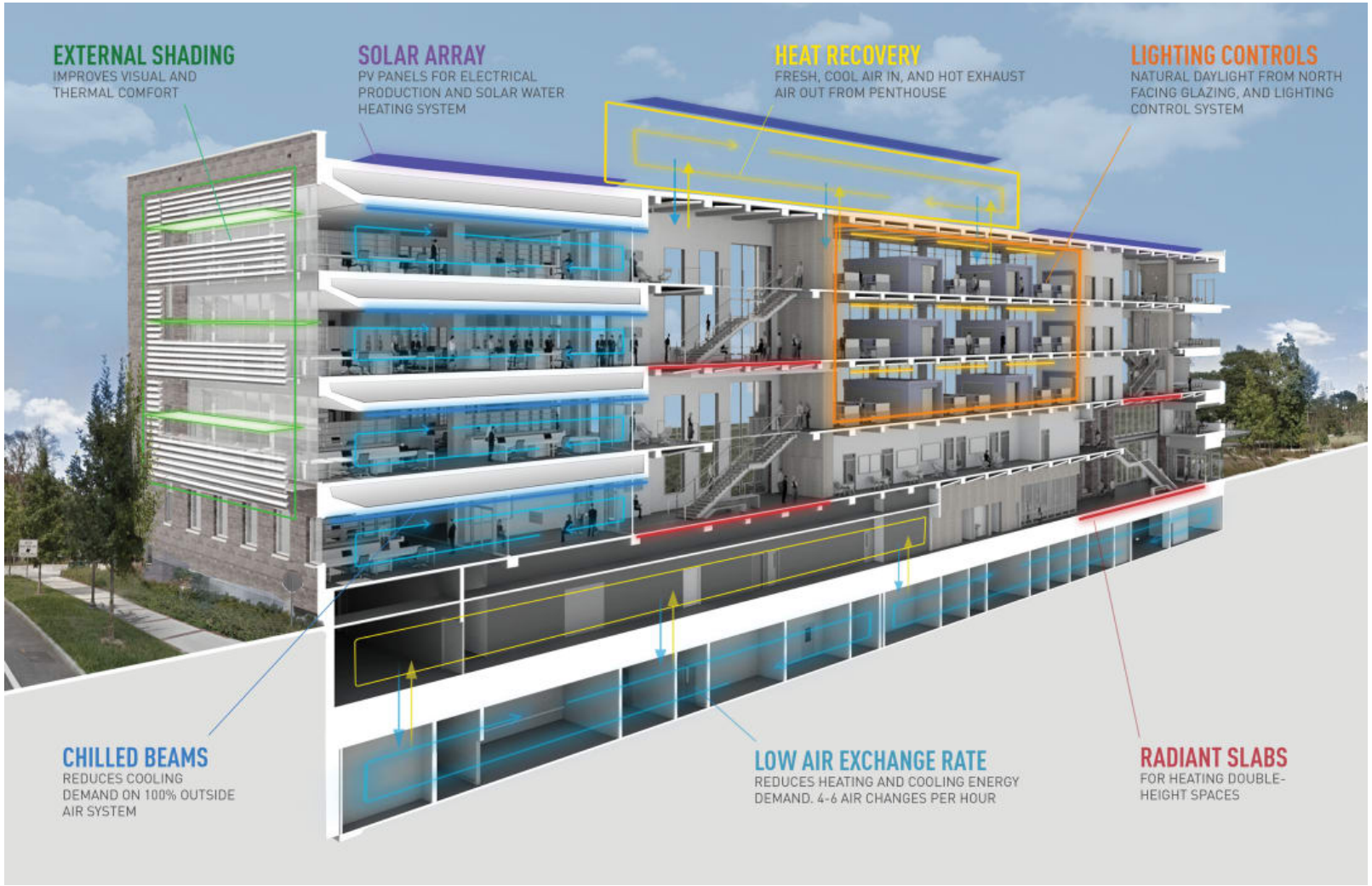


daylight



views to outside

Coding Application Examples

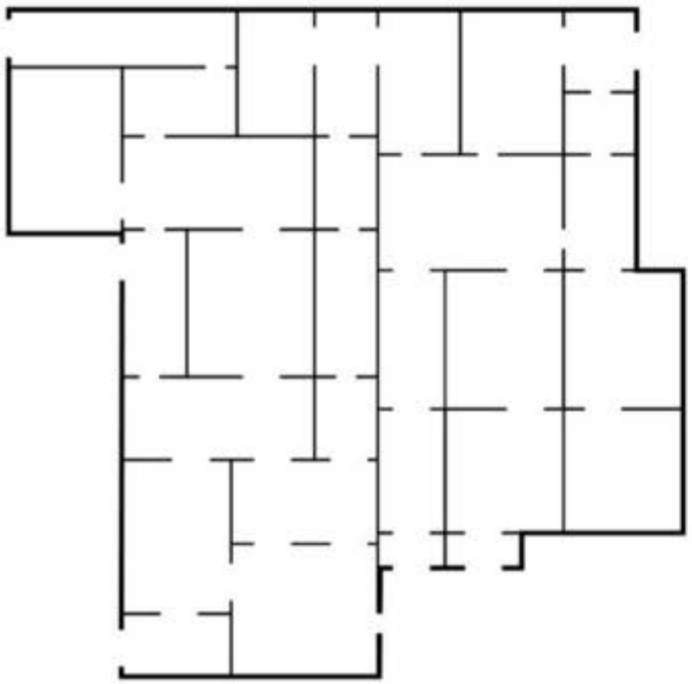


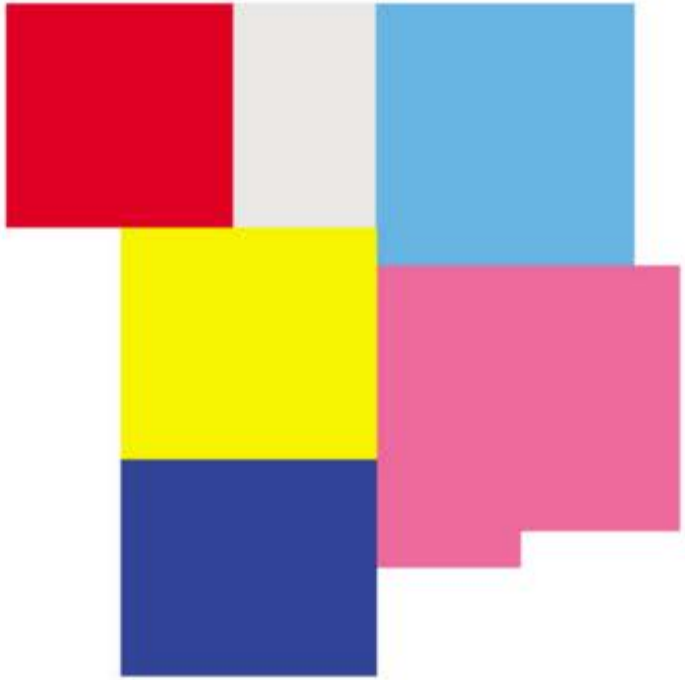
Using code, you can :

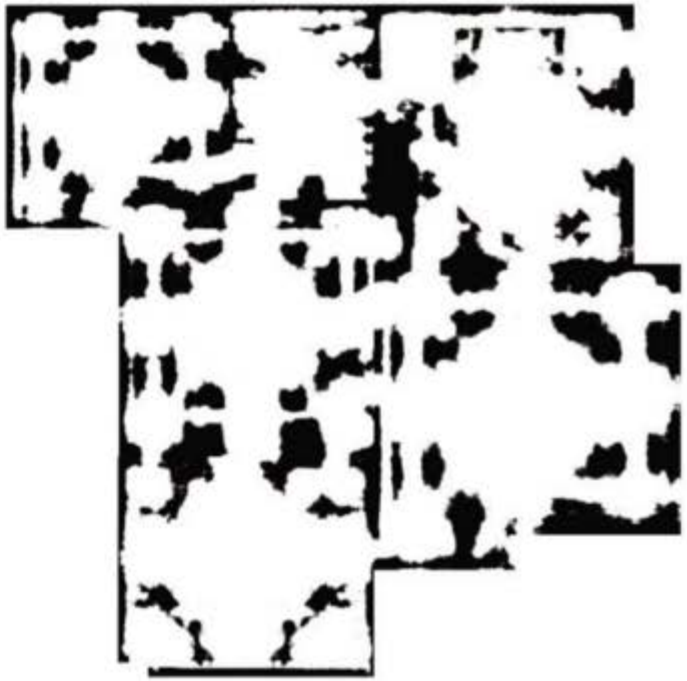
- Model different building properties
  - Heat exchange rates[1]
  - Energy usage[2]/optimization
- Generative Design[3]
- **Preliminary floor plan design[4]**
- Interior Design Assistant[5]

Coding Application Examples - 4: Modern-to-Baroque Style Floor Plan Transfer

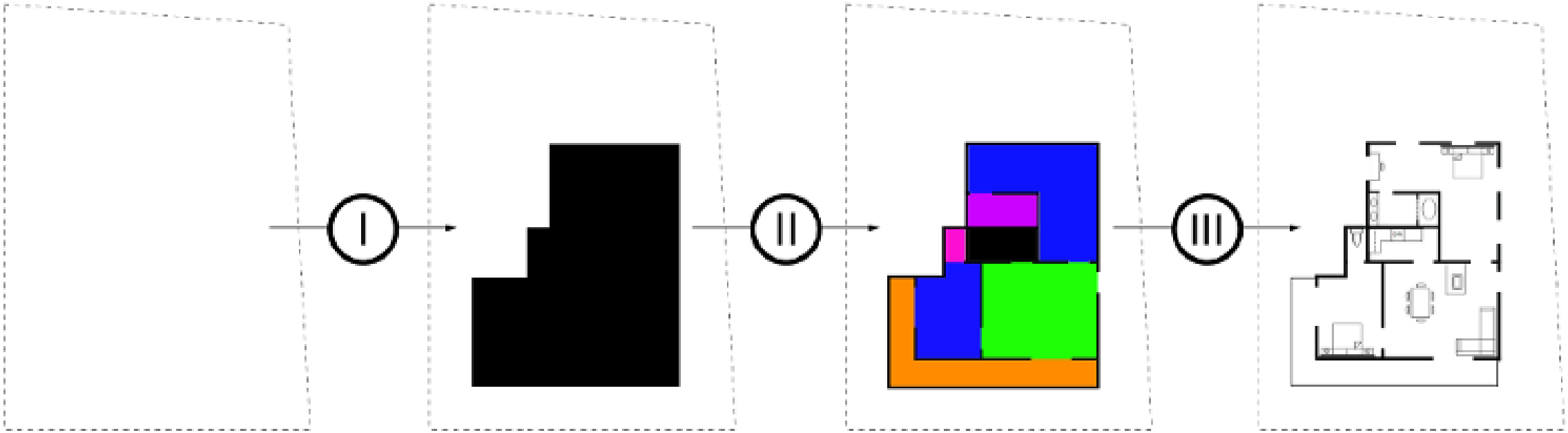
Coding Application Examples - 4: Modern-to-Baroque Style Floor Plan Transfer





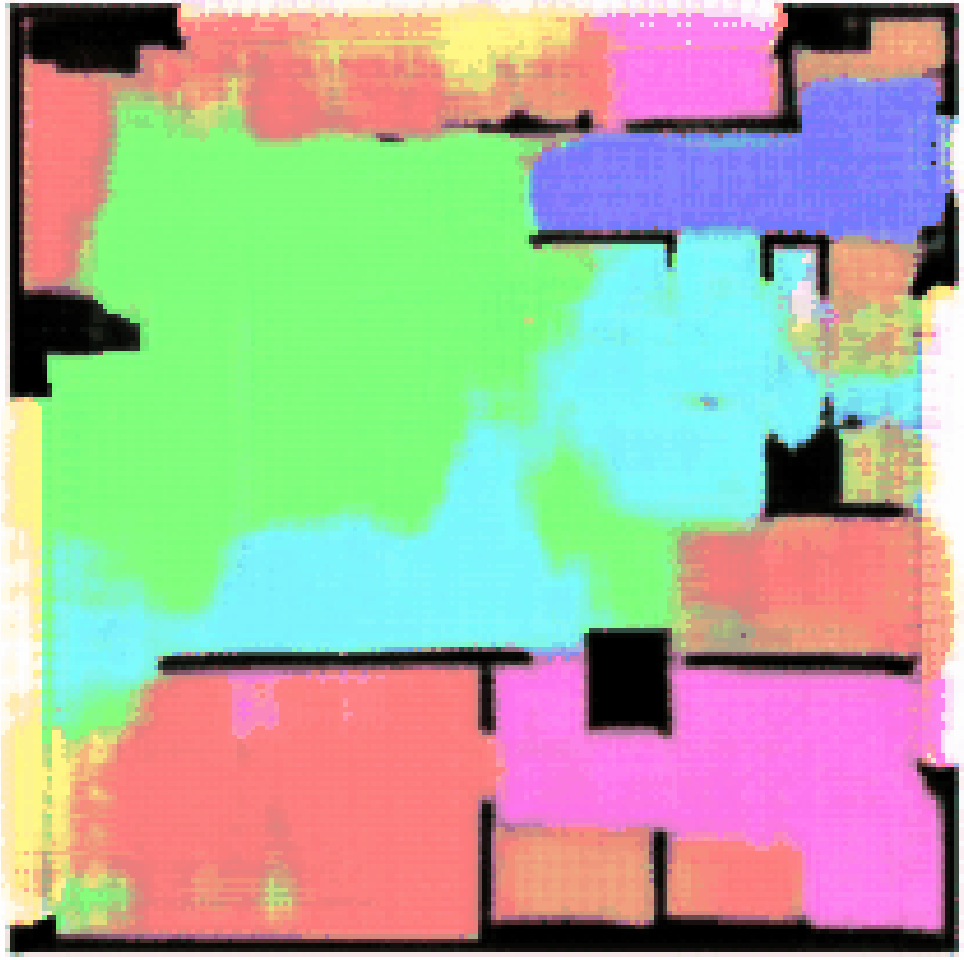


Coding Application Examples - 4: Automatic Layout Design



Parcel → Building Layout → Room Layout → Furniture Layout

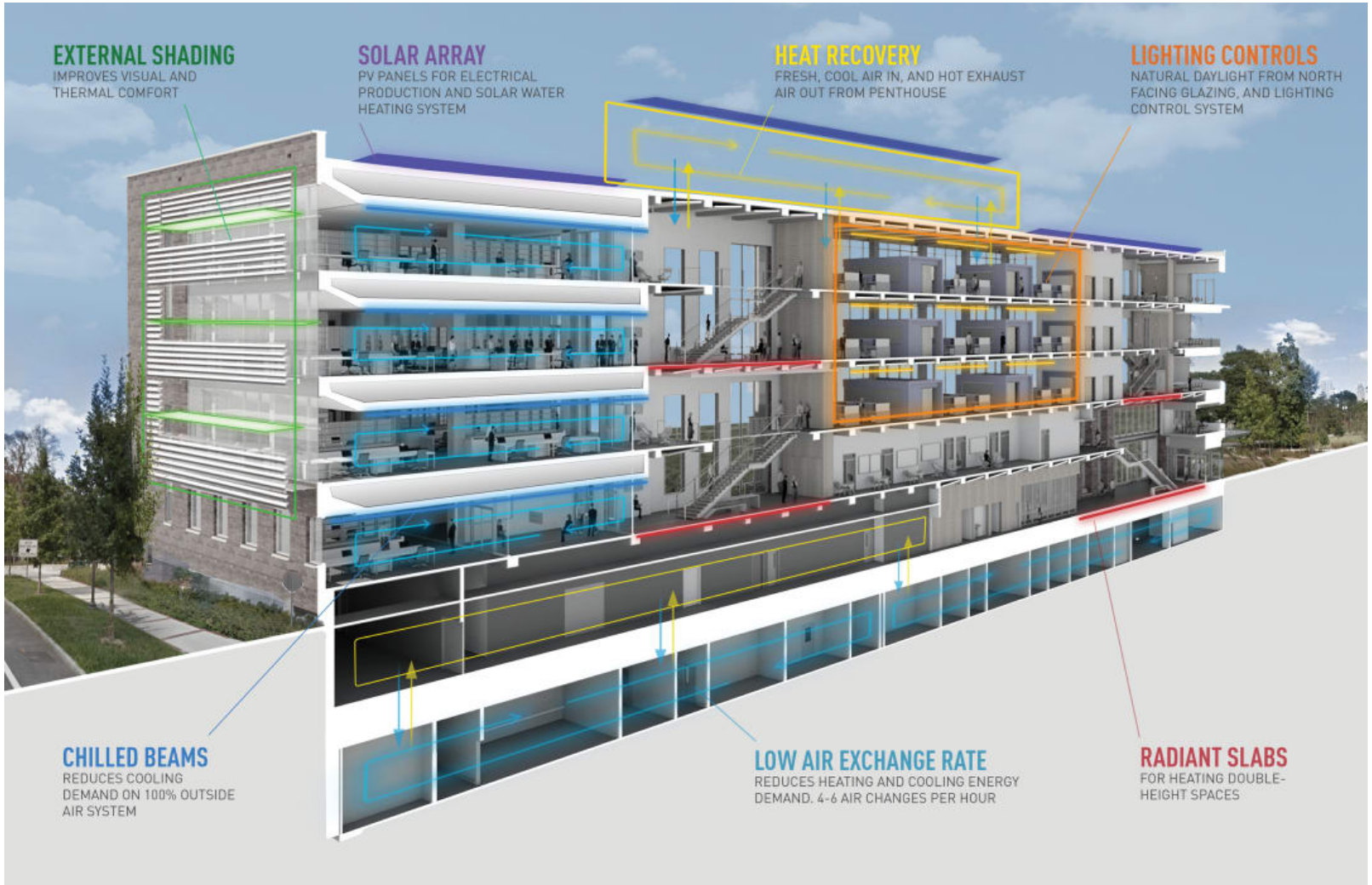
Coding Application Examples - 4: Automatic Layout Design Inception



Room Layout Generation

Detailed Layout Generation

Coding Application Examples



Using code, you can :

- Model different building properties
  - Heat exchange rates[1]
  - Energy usage[2]/optimization
- Generative Design[3]
- Preliminary floor plan design[4]
- **Interior Design Assistant[5], [6]**

Introduction

- We want code to be easy to read, organized, concise, and efficient

Example: Compute sine of 1,000 values ranging from 0 to 10:

Option 1 (For-loop)

```
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end
```

Option 2 (Vectorized)

```
t = 0:.01:10;
y = sin(t);
```

- We will learn techniques/guidelines to write clean, easy to understand code!

Pseudocode - Overview

Our learning outcomes for this lecture is:

- **Pseudocode**
- **Flowcharts**
- **How to debug code!**
- If time permits, we can cover the following:
  - Coding Guidelines
  - Commenting Guidelines
- Extra-resources:
  - Googling Guidelines

    (Cause googling, is a skill.)

Pseudocode - Analogy

- An analogy to pseudocode is a cooking recipe (lol)
- To cook a dish, you need to know what materials to use, and how to cook it.

Ingredients

For the dough

400g strong plain flour (preferably Italian 00 flour), plus extra for dusting

330g pumpkin

50ml tepid water, plus 50ml extra as required

1½ tablespoons olive oil

7g dried yeast

1 teaspoon caster sugar

½ teaspoon salt

For the toppings

670g pumpkin

2 tablespoons olive oil

280g persian feta

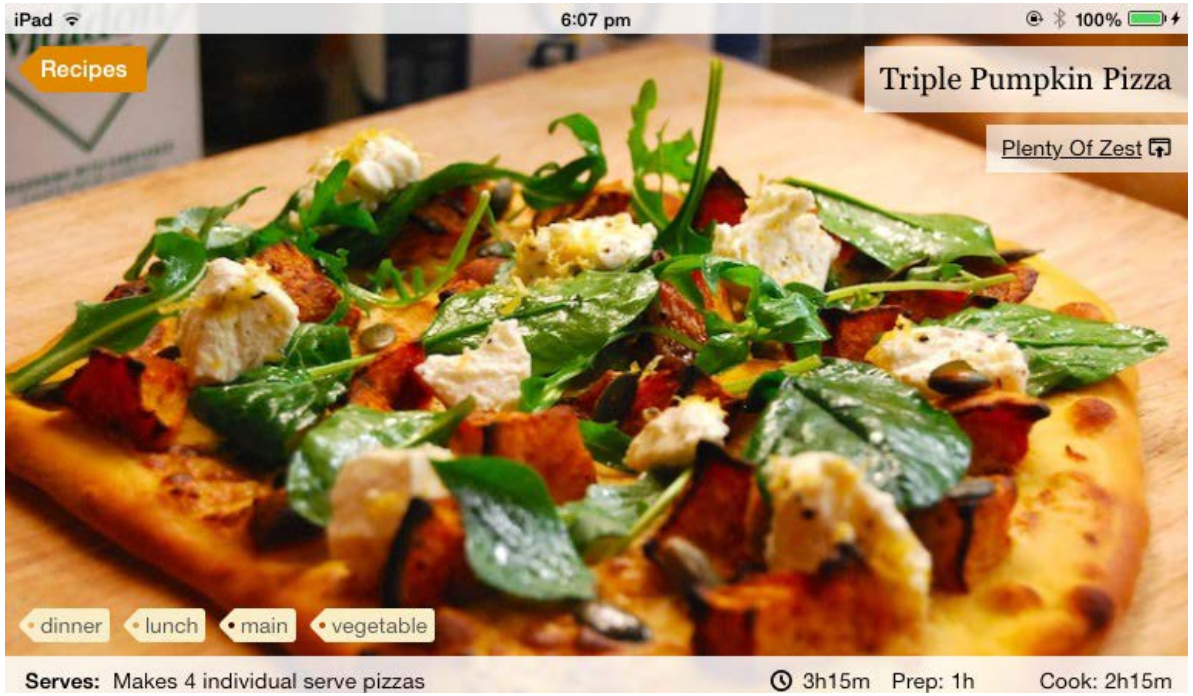
Zest of 1 lemon

Instructions Method Notes

1. Preheat the oven to 200°C.
2. Keep 330g of your pumpkin, for the dough, in one large piece. Wrap tightly in foil, place on a baking tray and into the oven until cooked through, about 1 hour.
3. Meanwhile peel and cut the remaining topping pumpkin into a 1-1.5cm dice. Toss in 2 tablespoons of olive oil and season well with salt and pepper. Spread the pieces on a lined baking tray and add to the oven. Roast until almost cooked, about 25 minutes, then set aside until assembling the pizzas.
4. When the large chunk of pumpkin is totally soft, remove from the oven and allow to cool slightly. Discard the seeds and scoop the flesh out from the peel and into a small food processor. Process into a smooth puree.
5. Mix 50ml of tepid water with the sugar and dry yeast, and allow to stand for a few minutes so it starts to foam.

Sift the flour and ½ teaspoon of salt into a large mixing bowl. Add the pumpkin puree, 2 tablespoons of olive oil

Cook Share Edit Delete



Huzzah!

Pseudocode - Introduction

- What is **pseudocode**?
- A **general**, **step-by-step description** of your code that you can transcribe into actual code.

Find the sum of 5 numbers

Pseudocode:

1. Initialize sum and count variables
2. While count < 5
   a) Store user input number
   b) Add user input number to total sum
   c) Add 1 to count
3. Print total sum

Pseudocode - Steps

- **Pseudocode** summarizes a program's flow but excludes the details
- We need to know:
  - **general step-by-step instructions**
  - **input/process/output variables**

General Thought Process (IDEAS):

1. Identify the problem
2. Define the input and output variables
3. Establish solution steps
4. Analyze solution
5. Solve

Pseudocode – 2_example_1.m

Calculate the sum & average for array containing ‘n’ numbers:

1. **I**dentify the problem
   a) Calculate sum and average for an array of variable size
2. **D**efine the input and output variables
   a) Input: **numbers** (n-sized array)
   b) Output: **summed**, **averaged**
3. **E**stablish solution steps
   a) Count how many elements there are in an array
   b) Loop through array to sum up numbers to get sum
   c) Divide sum by total number of arrays to get average
   d) Return the variables
4. **A**nalyze solution
   a) Need sum variable (**accumulator**), need numel function (**array_length**), need “for”, to loop through array
5. **S**olve (write the code)

Pseudocode – 2_example_1.m

Calculate the sum & average for array containing ‘n’ numbers:

<u>Pseudocode</u>

Given: n-sized numerical array

Output: Calculate sum and average

Steps:

1. Initialize accumulator to 0
2. Count elements in array
3. Loop through array
   - Accumulate numbers
4. Calculate average
5. Return sum and average

```
function [summed, averaged] = CalculateSumAndAverage(numbers)
    accumulator = 0;
    array_length = numel(numbers);
    for i = 1:array_length
        accumulator = accumulator + numbers(i);
    end
    summed = accumulator;
    averaged = accumulator / array_length;
end
```

Pseudocode – 2_example_1.m

- Benefits of pseudocode:
  - You can debug code faster (resolving logical and syntax errors)
  - Everyone can understand code faster to use/help
  - Pseudocode can be used as comments to explain the code functionality:

Pseudocode put into script as **comments** to give readers a human-level description

Pseudocode

Given: n-sized numerical array
Output: Calculate sum and average
Steps:
1. Initialize accumulator to 0
2. Count elements in array
3. Loop through array
   - Accumulate numbers
4. Calculate average
5. Return sum and average

```
function [summed, averaged] = CalculateSumAndAverage(numbers)
% Calculate the sum & average for calculate the sum & average
% for 'n' numbers.

    % Initialize accumulator to 0
    accumulator = 0;
    % Find how many numbers there are
    array_length = numel(numbers);
    % Loop through array and accumulate numbers
    for i = 1:array_length
        accumulator = accumulator + numbers(i);
    end
    % Assign acummulator to output variable
    summed = accumulator;
    % Calculate average
    averaged = accumulator / array_length;

end
```

Pseudocode – Some guidelines

- Some guidelines to writing good pseudocode:
  1. Use control structures
     - Loops (for and while), if/elif/else statements, functions, or switch statements

```
IF spaceship sprite touches asteroid sprite THEN
   show explosion sprite
   play explosion sound
   subtract a life
END IF

IF lives = 0 THEN
   stop game
   show game over screen
ELSE
   restart game
ENDIF
```

Pseudocode – Some guidelines

- Some guidelines to writing good pseudocode:
  2. Clear naming convention
     - **num_rows** vs **a**? Which variable gives more information?
     - Or how about **a, b, c** vs **R, PI, Area**?

```
1. START
2. READ R
3. SET PI=3.142
4. CALCULATE AREA OF CIRCLE
   4.1.    FORMULA:
           Area= PI * R * R
5. DISPLAY Area
6. END
```

Pseudocode – Some guidelines

- Some guidelines to writing good pseudocode:
  3. Indenting when using control structures – it makes a big difference!

```
for i = 1:n
    DoThis()
    for j = 5:x
        DoThat()
        for k = n:x
            AndThis()
        end
        DoThisAfterLoop()
    end
end
```

vs

```
for i = 1:n
DoThis()
for j = 5:x
DoThat()
for k = n:x
DoThis()
end
DoThisAfterLoop()
end
end
```

Which one is more easy to understand?

Pseudocode

4. Simplicity when writing pseudocode/code
   - Maximize code efficiency by using vector operations or logical masks instead of for loops, and using minimal number of control structures)

```
% Sum all numbers that are greater than 5
numbers = randi(100, [10 10]);
[row, col] = size(numbers);
sum_of_numbers = 0;
for i = 1:row
    for j = 1:col
        if numbers(i, j) > 5
            sum_of_numbers = sum_of_numbers + numbers(i, j);
        end
    end
end
```

vs

```
% Sum all numbers that are greater than 5
numbers = randi(100, [10 10]);
sum(numbers(numbers > 5))
```

11 lines vs 3 lines of code!!

Pseudocode – more examples

- There is no end-all standard to pseudocode!

```
IF spaceship sprite touches asteroid sprite THEN
   show explosion sprite
   play explosion sound
   subtract a life
END IF

IF lives = 0 THEN
   stop game
   show game over screen
ELSE
   restart game
ENDIF
```

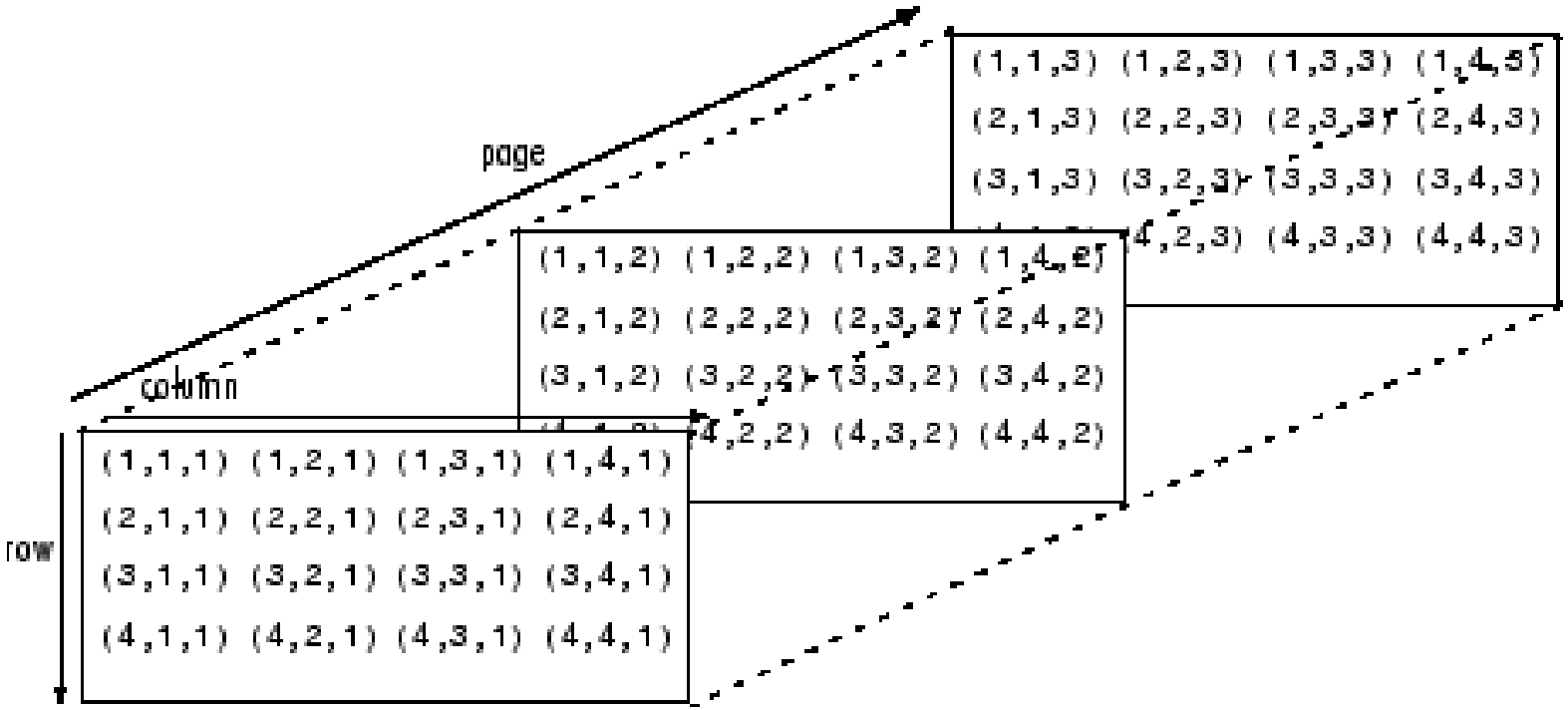
```
1. START
2. READ R
3. SET PI=3.142
4. CALCULATE AREA OF CIRCLE
   4.1.    FORMULA:
           Area= PI * R * R
5. DISPLAY Area
6. END
```

```
Begin
   Initialize swarm, velocities and best positions
   Initialize external archive(usually empty)
While(stopping criterion not be satisfied)
   For each particle
      Select a member for the external archive(if needed)
      Update velocity and position
      Evaluate new position
      Update best position and external archive
   End for
End While
```

- Pseudocode should be as concise, easy to understand, and clear, as possible.

Pseudocode – 2_example_2.m

- **Given an N-dimensional array, find the maximum value.**
- So given an array that could be length, n, or n x m, or n x m x o, and so on, how can we find the max without using the function max?



Example: a 5-dimensional array

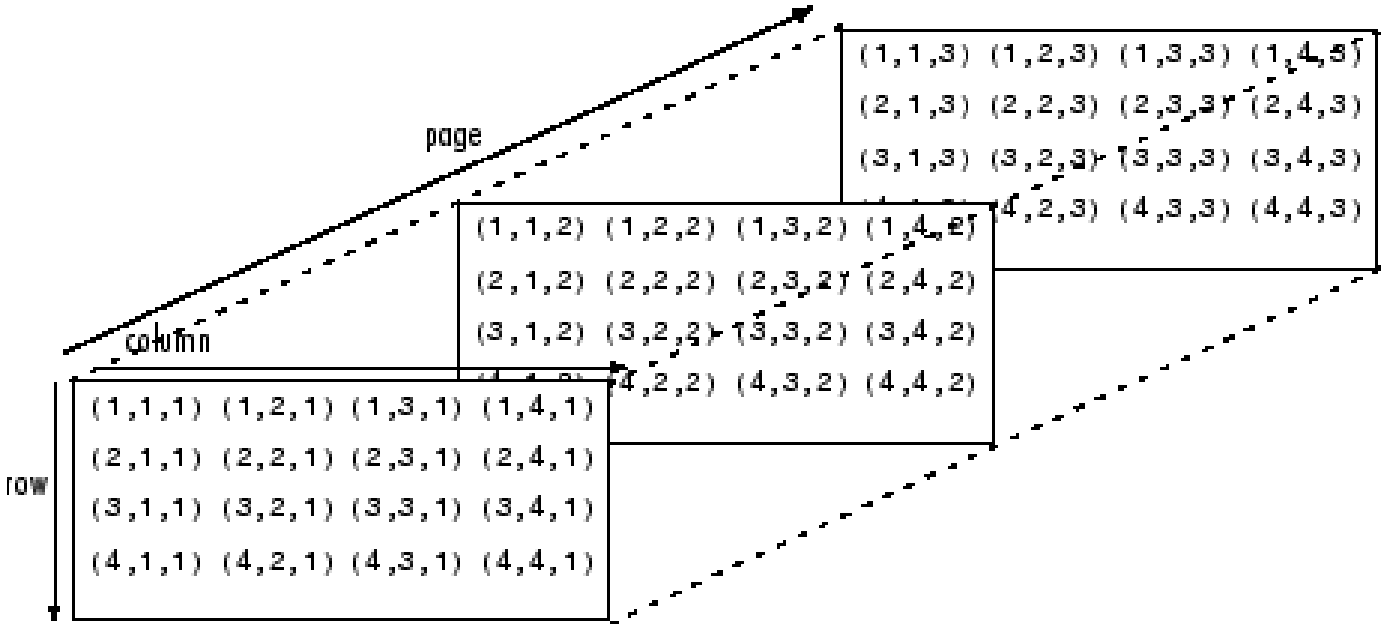
Pseudocode – 2_example_2.m

- **Given an N-dimensional array, find the maximum value**

1. Identify the problem
   a) Find the maximum value given a. n-dimensional array
2. Define the input and output variables
   a) Input: **numbers** (n-dimensional array)
   b) Output: **max_number** (scalar variable)
3. Establish solution steps
   a) Initialize **max_number** and set first element in **numbers** as the max variable
   b) For-loop through rest of the array using linear indexing
      I. If **max_number** < **number**
         max_number = **number**;
4. Analyze solution
5. Solve

Pseudocode – Simple Example Solution

- **Given an N-dimensional array, find the maximum value.**



<u>Example Pseudocode (more code-like):</u>
Function: Find Max of a N-dimensional array
Input: numbers (the n-dimensional array)
Output: max_number (the maximum number)
Pseudocode:

1. **max_number = numbers(1)**
2. for i = 2:numel(**numbers**)
   I. if max_number < numbers(i)
      - max_number = numbers(i)
3. Return max_number

Pseudocode – Simple Example Solution

- **Given an N-dimensional array, find the maximum value.**

Code:

```
function max_number  = FindMax(numbers)
% Find the maximum value in a n-dimensional array
    % Initialize variable
    max_number = numbers(1);
    % Loop through values in the array
    for i = 2:numel(numbers)
        if max_number < numbers(i)
            max_number = numbers(i);
        end
    end
end
```

Indents!

Less-ambiguous variable names

Pseudocode – Quiz

Given a 2-D array, create a function that sorts each row or column of the array in ascending order and gives its sorted linear indices.

Sample array:

Linear Index!

3^{1}	1^{4}	2^{7}
5^{2}	6^{5}	1^{8}
1^{3}	8^{6}	4^{9}

Output (sorted array, linear index):

Sorted Array:

1^{4}	2^{7}	3^{1}
1^{8}	5^{2}	6^{5}
1^{3}	4^{9}	8^{6}

Sorted Array's Linear Index:

```
[4, 8, 3, 7, 2, 9, 1, 5, 6]
```

Feel free to share your pseudocode with your classmates here

Pseudocode – Quiz Answer

Keep in mind there are many right answers!
Your solutions may differ.

<u>Example Pseudocode:</u>
Function: Sort each row or column of the array
Input: array_2D, sort_option (‘r’ for rows, ‘c’ for columns)
Output: sorted_2D_array and linear_indicies
Pseudocode:

1. Initialize an array of zeros of shape array_2D and assign to variable ‘linear_indices’
2. Initialize an array of zeros of shape array_2D and assign to variable ‘sorted_2D_array’
3. If sort_option is ‘r’
    I. Loop through each row
        a) Initialize row vector called “row” that contains its row values
        b) Initialize row vector called “row_indices” that contains its linear indices
        c) Sort row and update row_indices (Any sorting algorithm can be chosen)
        d) Assign the sorted row and row_indices to the corresponding row in sorted_2D_array and linear_indices, respectively
4. If sort_option is ‘c’
    I. Loop through each column
        a) Repeat steps like sort_option “r” but for columns

Pseudocode – Quiz Answer (3_quiz_answer.m)

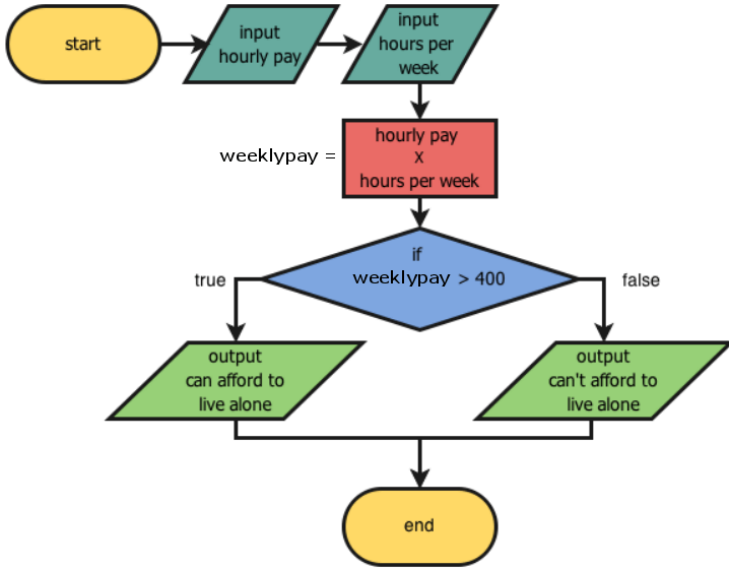
- **Given a 2-D array, create a function that sorts each row or column of the array in ascending order and gives its sorted linear indices.**

Code:

```
function [sorted_2D_array, linear_indices] = Sort_2D_Array(array_2D, sort_option)
    % Get Array Dimensions
    [r, c] = size(array_2D);
    % Declare empty variables
    linear_indices = zeros(r, c);
    sorted_2D_array = linear_indices;
    % Sort each row in the array
    if sort_option == 'r'
        % Loop through each row
        for i = 1:r
            % Initialize row and its indices
            row = array_2D(i, :);
            row_indices = i:c:numel(array_2D);
            % Sort the array and its indices by finding the smallest number
            % and placing it in the first element of the array, then
            % finding the second smallest numeber and placing it in the
            % second element of the array, and so on...
            for j = 1:numel(row)-1
                smallest_number = row(j);
                smallest_array_index = row_indices(j);
                switch_index = j;
                for k = j+1:numel(row)
                    if smallest_number > row(k)
                        smallest_number = row(k);
                        smallest_row_index = k;
                        smallest_array_index = row_indices(k);
                        switch_index = k;
                    end
                end
                % Perform switch if a smaller number has been found
                if j ~= switch_index
                    row(smallest_row_index) = row(j);
                    row_indices(smallest_row_index) = row_indices(j);
                    row(j) = smallest_number;
                    row_indices(j) = smallest_array_index;
                end
            end
            sorted_2D_array(i, :) = row;
            linear_indices(i, :) = row_indices;
        end
    % Sort each column in the array
    elseif sort_option == 'c'
        % Loop through each column
        for i = 1:c
            % Initialize column and its indices
            col = array_2D(:, i);
            col_indices = c*(i-1)+1:c*i;
            % Sort the array and its indices by finding the smallest number
            % and placing it in the first element of the array, then
            % finding the second smallest numeber and placing it in the
            % second element of the array, and so on...
            for j = 1:numel(col)-1
                smallest_number = col(j);
                smallest_array_index = col_indices(j);
                switch_index = j;
                for k = j+1:numel(col)
                    if smallest_number > col(k)
                        smallest_number = col(k);
                        smallest_col_index = k;
                        smallest_array_index = col_indices(k);
                        switch_index = k;
                    end
                end
                % Perform switch if a smaller number has been found
                if j ~= switch_index
                    col(smallest_col_index) = col(j);
                    col_indices(smallest_col_index) = col_indices(j);
                    col(j) = smallest_number;
                    col_indices(j) = smallest_array_index;
                end
            end
            sorted_2D_array(:, i) = col;
            linear_indices(:, i) = col_indices;
        end
    end
end
```

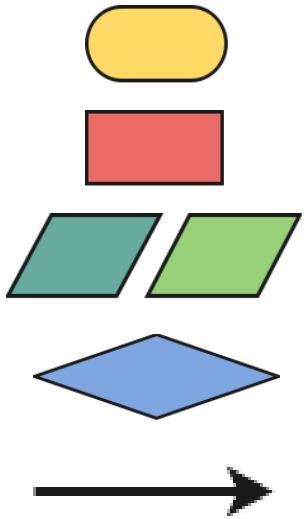
(Optional) Flowcharts

- An alternative to pseudocode is a flowchart!
- A flowchart is a visual representation of pseudocode.
- This example calculates weekly pay and determines if one can live alone:



Note: The shapes and colors of the blocks indicate if it is the:

1. start/end of code
2. a computation
3. input/output
4. a conditional/loop structure
5. Connection from one part of the code to the next



- Flowcharts are good to visualize and understand code flow, but cannot be used as comments in code.
- A good flowchart maker is LucidChart.

Pseudocode vs Flowchart Comparison

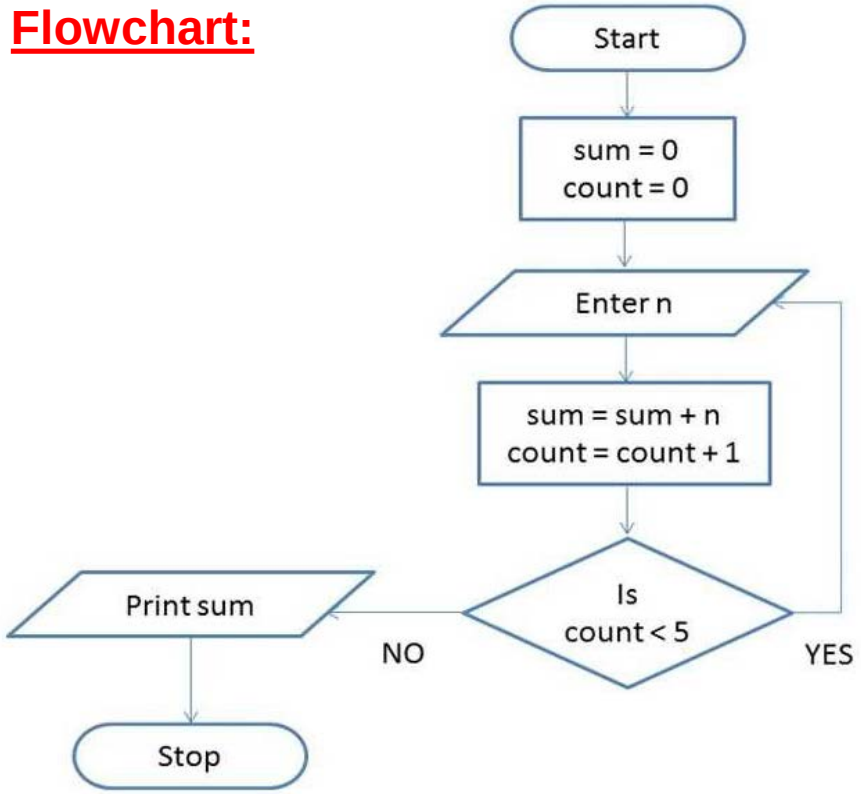
Find the sum of 5 numbers

Pseudocode:

1. Initialize sum and count variables
2. While count < 5
   a) Store user input number
   b) Add user input number to total sum
   c) Add 1 to count
3. Print total sum

VS

Flowchart:



Debugging Tutorial

In this section, we will learn how to:

- Debug logical errors
- Debug syntax errors (most frequent)!
- Refer to **this**.

Clean Coding Guidelines

- Clean, clear, and readable code is really important!!
- "Imagine you have written a script that finds the max of an n-dimensional array, and you must modify the code to find the 2nd highest number instead."
- Which version of code is easier to understand?

```
function m = FindMax(arr)
arr = arr(:);  % Flatten array
m = arr(1);    % Get first element
for i = 2:numel(arr)
if m < arr(i)  % Compare max to element
m = arr(i);  % Assign new max value
end
end
end
```

VS

```
function max_number = FindMax(n_array)
    % Flatten n_array to 1-dimension
    flat_n_array = n_array(:);
    % Assign first element of flat_n_array to variable current_max
    current_max = flat_n_array(1);
    % Loop through all the values of the flat_n_array excluding the first element
    for i = 2:numel(flat_n_array)
        % Compare flat_n_array(i) with current_max and keep the bigger
        % value
        if current_max < flat_n_array(i)
            current_max = flat_n_array(i);
        end
    end
    % Return the maximum value found
    max_number = current_max;
end
```

- Let's learn to write clean code!

Clean Coding Guidelines – Naming Conventions

- The purpose of naming conventions are to help the reader and programmer.

1. Variable names should describe in short what they are/do:

```
z = x * y
```

VS

```
wage = hourlyRate * nHours
```

2. a. Write variable names in mixed case starting with lower case

   Ie. nNames, town, or isCaseOpen

   b. Write variable names in lower case separated by underscore

   Ie. n_names, town, or is_case_open

3. Use a prefix (typically “n” or “m”) for variables representing number of objects

   Ie. “nFiles”, or “nPoints”, etc…

Clean Coding Guidelines – Naming Conventions

4. Minimize abbreviation in names

   Ie. “compareArrivalTime” vs “cmpAT”

5. Use action verb/state prefixes to describe a variable. For example:
   1. “has/is/can/should” for Boolean variables (“isComplete”, “canRun”)
   2. “find/get/compute” for functions (“findMax”, “getArray”, “computeMatrix”)
6. Use suffixes for dimensioned variables and constants

   Ie. “angleRadians”, “lengthCM”, “weightNewtons”

Clean Coding Guidelines – Statements

- Minimize use of repurposing variables to avoid confusion
- Initialize loop result variables immediately before the loop (more efficient!)

```
result = nan(nEntries,1);
for index = 1:nEntries
    result(index) = foo(index);
end
```

- Avoid complex conditionals – use temporary logical variables instead:

  Instead of this:

```
if (value>=lowerLimit)&(value<=upperLimit)&~…
   ismember(value,… valueArray)
  :
end
```

Clean Coding Guidelines – Statements

- Do this:

```
isValid = (value >= lowerLimit) &…
   (value <= upperLimit);
isNew   = ~ismember(value, valueArray);

if (isValid & isNew)
   :
end
```

Clean Coding Guidelines - Format

- Keep lines of code within 80 columns
- Split lines at commas/operators using the continuation (…) operator

```
totalSum = a + b + c + …
           d + e;
function (param1, param2,…
          param3)
setText (['Long line split' …
          'into two parts.']);
```

- Indent 3 or 4 spaces (or 1 tab) for an indented block of code

```
for i = 1:n
    DoThis()
    for j = 5:x
        DoThat()
        for k = n:x
            AndThis()
        end
        DoThisAfterLoop()
    end
end
```

VS

```
for i =1:n
 DoThis()
 for j = 5:x
  DoThat()
  for k = n:x
   AndThis()
  end
 end
end
```

Clean Coding Guidelines - Format

- Write one executable statement per line of code

This:
```
max_nArray = max(nArray)
mean_max_nArray = mean(max_nArray)
sum_mean_max_nArray = sum(mean_max_nArray)
```
is more readable than this:
```
sum(mean(max(nArray)))
```

- Single statements using if/for/while statements can be written on one line of code

```
if(condition), statement; end

while(condition), statement; end

for iTest = 1:nTest, statement; end
```

```
% Regular if statement
if testScore > meanScore
    pass = true;
end
% One line if statement
if testScore > meanScore, pass = true; end
% Regular for loop
result = nan(n, 1);
for i = 1:n
    result(i) = nTestScore(i) * 100;
end
% One line for loop
for i = 1:n, result(i) = nTestScore(i) * 100; end
```

Clean Coding Guidelines - Format

- Add white space between operators, commas followed by space

  Ie. a+b+c+d vs a + b + c + d **or** a,b,c,d vs a, b, c, d
- Use alignment wherever it enhances readability

```
value = (10*nDimes) + (5*nNickels) + (1*nPennies);
```

VS

```
value = (10 * nDimes) + ...
        (5  * nNickels) + ...
        (1  * nPennies);
```

Commenting (Pseudocode) Guidelines

- Header comments (for the user)
  - At the top of the script, programmers usually place a block of code containing the purpose of the script, input/outputs, and methods of use.

```
function InsertHeader(varargin)
% InsertHeader - Adds a template to the top of current file in editor for documentation.
%
% Syntax: InsertHeader(varargin)
% varargin - optional input of full file name
%   Example 1:  InsertHeader adds header to top of currently selected file in
%   editor
%   Example 2: InsertHeader('InsertHeader') added the current header information to this
%   file. NOTE:  THIS CANNOT BE UNDONE!  THE FIRST TIME YOU USE THIS, PLEASE
%   HAVE A BACK UP COPY OF THE FILE IN CASE IT MAKES A MISTAKE.
%   Custom defined headers can be created in the subfunction defined at
%   the end of the file.  I wish to express thanks to the authors of tedit.m and
%   newfcn.m which helped me to create this routine.
%  See also:  tedit, newfcn
%
% Subfunctions: EditorCurrentFile, parseForSubFcns, insertHeaderTemplateFile
```

  - Note: Write the function name in comments using its **actual case**.

Commenting (Pseudocode) Guidelines

- Use in-line comments to explain blocks of code
- Comments cannot justify poorly written code!
- Indent code comments the same as the statements referred to.
- Minimize use of end of line (EOL) comments.

```
function m = FindMax(arr)
arr = arr(:);  % Flatten array
m = arr(1);    % Get first element
for i = 2:numel(arr)
if m < arr(i)  % Compare max to element
m = arr(i);  % Assign new max value
end
end
end
```

```
function max_number = FindMax(n_array)
    % Flatten n_array to 1-dimension
    flat_n_array = n_array(:);
    % Assign first element of flat_n_array to variable current_max
    current_max = flat_n_array(1);
    % Loop through all the values of the flat_n_array excluding the first element
    for i = 2:numel(flat_n_array)
        % Compare flat_n_array(i) with current_max and keep the bigger
        % value
        if current_max < flat_n_array(i)
            current_max = flat_n_array(i);
        end
    end
    % Return the maximum value found
    max_number = current_max;
end
```

- For more detailed explanations, refer to this document!

Feedback

Any feedback is greatly appreciated!

https://forms.gle/U8cBwQaWVkMuvtfi9

Slide Credits and References

1. https://economictimes.indiatimes.com/definition/pseudocode
2. https://www.unf.edu/~broggio/cop2221/2221pseu.htm
3. https://sites.google.com/a/ismanila.org/oliverab_cp/python/pseudocode
4. https://en.wikipedia.org/wiki/Pseudocode
5. https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/
6. https://davidzych.com/writing-code-using-the-pseudocode-programming-process/
7. https://en.wikipedia.org/wiki/Comment_(computer_programming)#Planning_and_reviewing
8. https://softwareengineering.stackexchange.com/questions/67852/do-you-also-forget-the-code-after-getting-the-task-done
9. https://www.computerhope.com/jargon/s/script.htm

Image Credits

- https://www.clareecho.ie/wp-content/uploads/2017/10/good-job-meme.jpg
- https://www.youtube.com/watch?v=vOEN65nm4YU
- http://plentyofzest.com/static/images/zest-webshot-recipe.jpg
- http://computersciencementor.com/wp-content/uploads/2017/02/pseudocode.jpg
- https://www.computerscience.gcse.guru/wp-content/uploads/2018/03/pseudocode-example.png
- http://4.bp.blogspot.com/_FTeNn51Pac0/TLapqY7DAsI/AAAAAAAAABc/1aYEqOputDA/s320/pseudocode.bmp
- https://www.researchgate.net/figure/Structure-of-MOPSO-a-simple-pseudo-code_fig6_273479296
- https://www.mathworks.com/help/examples/matlab/win64/nddemo_02.gif
- https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/12323/versions/3/screenshot.jpg
- http://beyondefficiency.us/blog/aia-code-ethics-finally-incorporates-sustainability-rule
- https://interestingengineering.com/5-ways-artificial-intelligence-is-changing-architecture
- https://www.comsol.com/paper/download/257311/kermani_paper.pdf
- https://www.sciencedirect.com/science/article/pii/S0360132302000513
- https://www.designingbuildings.co.uk/w/images/9/9b/Corridor.jpg
- https://commons.wikimedia.org/wiki/File:St-Anne_church_Krakow_003.JPG