Selection Statements

by Ahmet Sacan

selection statements, branching statements, condition, action, temporary variable, error-checking, nesting statements, cascading if-else, “is” functions

Selection Statements

- Selection (aka Conditional or Branching) Statements: statements that conditionally execute a block of code.
  - **if** somethingAistrue, dosomethingX; **end**
  - **if** somethingAistrue, dosomethingX; **else** dosomethingY; **end**
  - **if** somethingAistrue, dosomethingX; **elseif** anotherthingBistrue, dosomething Y; **else** dosomethingZ; **end**
- Conditions in if statements are relational (aka Boolean or logical) expression that evaluate to either true or false.

if statement

```
if condition
    action
end
```

- condition is any expression that can be evaluated to true/false.
- action is any number of statements that get executed if and only if the condition evaluates to true.

Examples

- if true; fprintf('this always prints\n'); end
- if 5; fprintf('this always prints\n'); end
- a=3; if a+1; fprintf('this always prints\n'); end

- if false; fprintf('this never prints\n'); end
- if 0; fprintf('this never prints\n'); end
- a=3; if a-3; fprintf('this never prints\n'); end

Examples

```
if num < 0
        num = num + 1;
end
```

```
num = -5
if num < 0
        num = num + 1;
end
disp(num)
```

```
num = 5
if num < 0
        num = num + 1;
end
disp(num)
```

Exercise

- Write a matlab function **div23.m** that will take an integer number, and display 'yes' if this number is divisible by 2 and separately by 3; 'no' otherwise.
- Sample Run:

```
>> div23( 16 )
divisible by 2: yes
divisible by 3: no
```

Exercise

- Write a function sqrtif.m that takes a number x, and
  - returns its square root if x is positive.
  - if x is negative, notifies the user that taking square root of negative numbers is not allowed and returns NaN.

if-else statement

```
if condition
    action1
else
    action2
end
```

- if condition is true, action1 is performed. Otherwise, action2 is performed.

Exercise

- Write a function sqrtif.m that takes a number x, and
  - returns its square root if x is positive.
  - if x is negative, notifies the user that taking square root of negative numbers is not allowed and returns NaN.
  - Use if-else

Nested if-else statements

- Example: calculate y using the following:

$$y = \begin{cases} 1 & if\ x < 5 \\ x^2 & if\ 5 \le x \le 7 \\ 4 & if\ x > 7 \end{cases}$$

```
if x < 5
   y=1;
end
if 5 <= x && x <= 7
   y = x^2;
end
if x > 7
   y = 4;
end
```

```
if x < 5
   y=1;
else
   if 5<=x && x <= 7
      y = x^2;
   elseif x>7
      y = 4;
   end
end
```

```
if x < 5
   y=1;
elseif x <= 7
   y = x^2;
else
   y = 4;
end
```

Exercise

- Write a function getdayname(x) that takes an integer x from 1 to 7, and returns the corresponding day name.
- Sample Run:

```
>> getdayname(3)
ans =
Tuesday
```

switch statement

```
switch switchexpression
        case caseexpression1
                action1
        case caseexpression2
                action2
        ...
        otherwise
                defaultaction
end
```

Exercise

- Write a function getdaynameswitch(x) that takes an integer x from 1 to 7, and returns the corresponding day name.
- Sample Run:

```
>> getdaynameswitch(3)
ans =
Tuesday
```

Exercise

- Write a function printnumber(n) that prints 'good' if n is 0, 1, or 2; prints 'not good' if n is 3 or 4, prints nothing if n is 5, and prints 'cannot decide' otherwise.

```
switch(n)
  case 0
    disp('good')
  case 1
    disp('good')
  case 2
    disp('good')
  case 3
    disp('not good')
  case 4
    disp('not good')
  case 5
  otherwise
    disp('cannot decide')
  end
```

```
switch(n)
  case { 0, 1, 2 }
    disp('good')
  case { 3, 4 }
    disp('not good')
  case 5
  otherwise
    disp('cannot decide')
  end
```

"is" functions

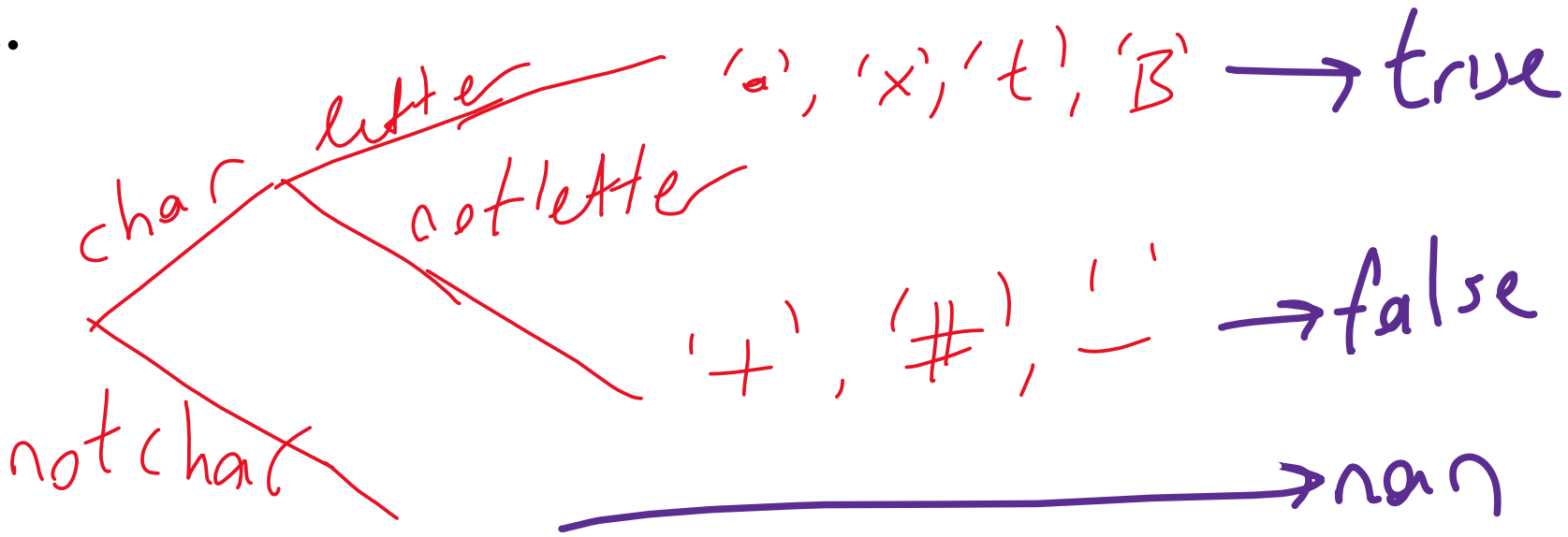
- "is" functions return true if the argument is of a given type.

```
>> ischar('hello')
ans =
1  (logical)
>> ischar([4 5 6])
ans =
0 (logical)

>> isletter('h')
ans =
1 (logical)
>> isletter('hi !# hi !')
ans = (logical)
    1    1    0    0    0    0    1    1    0    0
```

Exercise

- Assume x is a scalar value. Write a function myisletter(x) that returns true if x is a letter, and false otherwise. If x is not a character, return NaN. Assume x is a scalar. Do not use the built-in isletter function.



exist(…,'var')

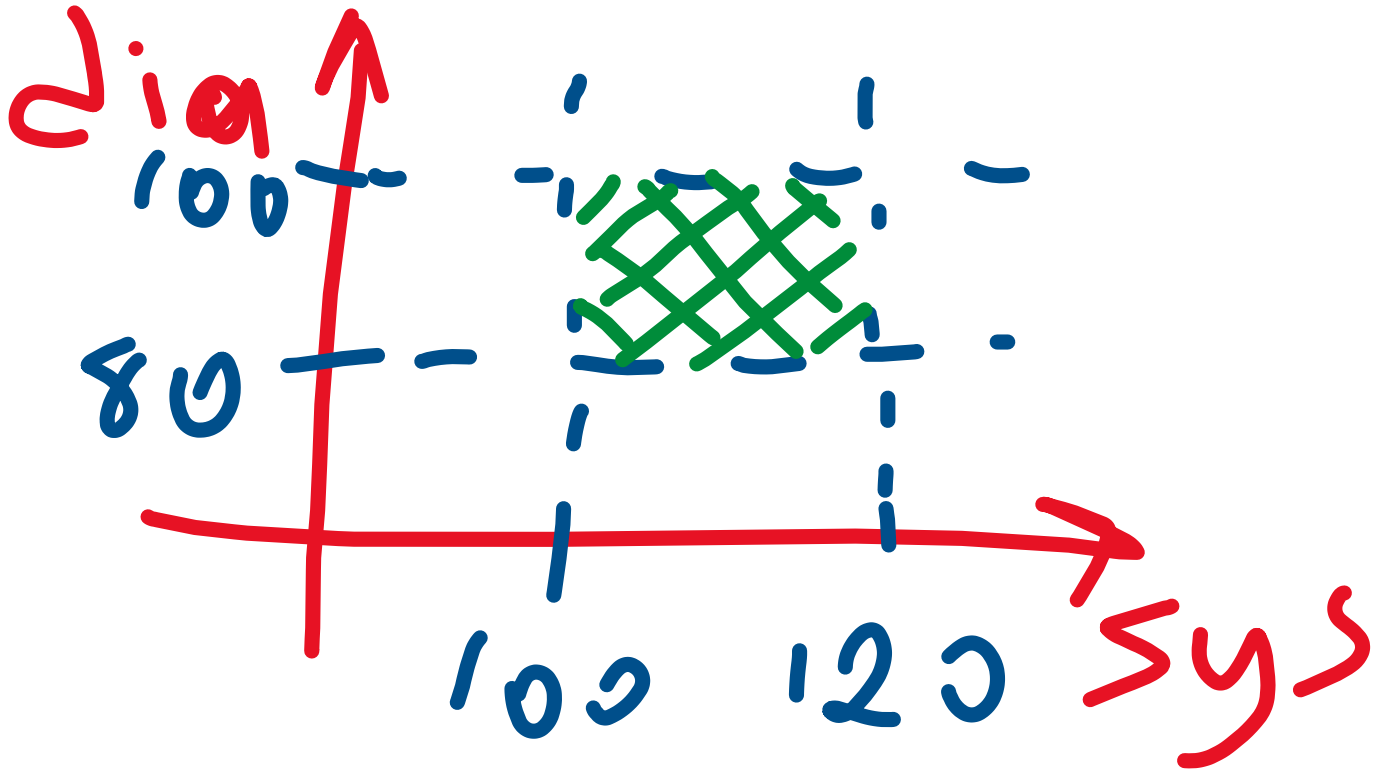
- The most common use of the **exist(…,'var')** construct is to support default input arguments to functions.
- Exercise: Write a function divby that takes a vector v and a number n and returns the sum of elements in v that are divisible by n. **If n is not given, use n=2.**

Common Mistakes

- Using = instead of == to test for equality
- Not using quotes when comparing a text variable to a text, such as:
  choice == y
- Not spelling out an entire logical expression:
  if radius || height <= 0; disp('Need positive values'); end
- Unnecessarily testing if something is 0 or 1:
  - if (x < 5) == true
  - if (x < 5)
- Unnecessary elseif statements
  if x==5; action1
  else~~if x~=5;~~ action2; end

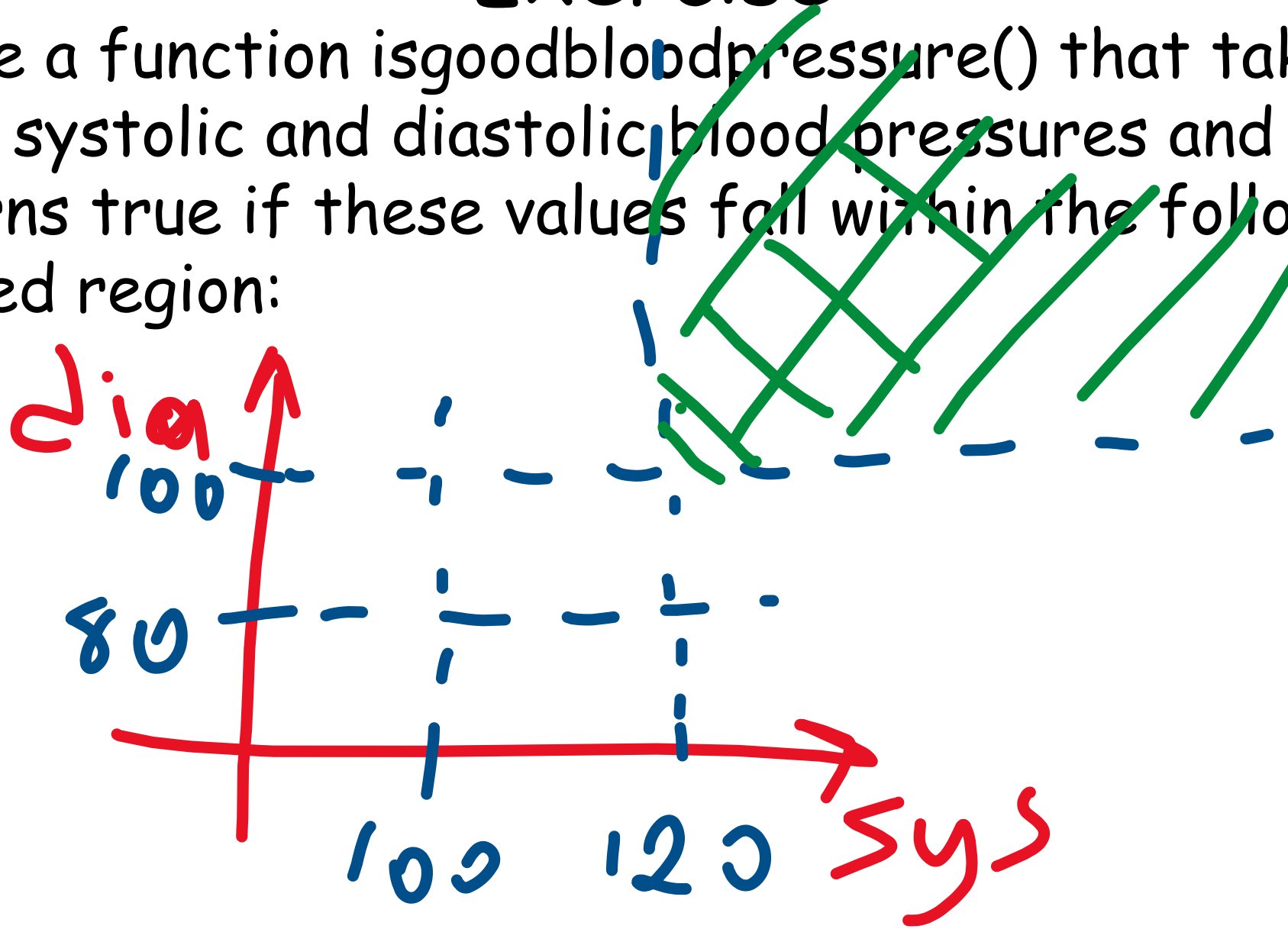
Exercise

- Write a function isgoodbloodpressure() that takes as input systolic and diastolic blood pressures and returns true if these values fall within the following shaded region:



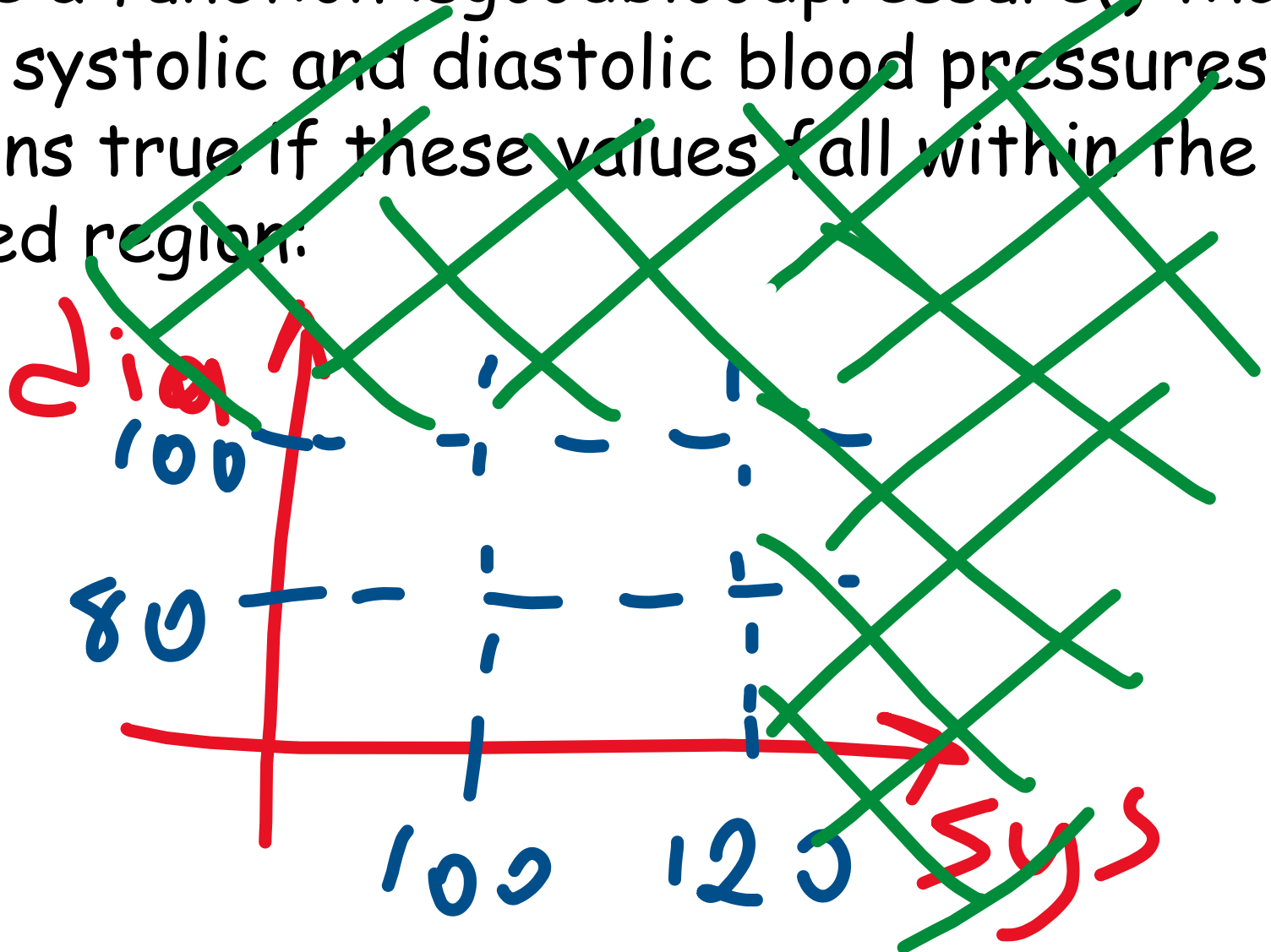
Exercise

- Write a function isgoodbloodpressure() that takes as input systolic and diastolic blood pressures and returns true if these values fall within the following shaded region:



Exercise

- Write a function isgoodbloodpressure() that takes as input systolic and diastolic blood pressures and returns true if these values fall within the following shaded region:



Exercise

- Write a function isgoodbloodpressure() that takes as input systolic and diastolic blood pressures and returns true if these values fall within the following shaded region: