

www.cs.cmu.edu/~112/gallery.html

CMU 15-112, Fall 2022

[Home](#)

[Syllabus](#)

[Schedule](#)

[Gallery](#)

[Staff](#)

[Piazza](#)

[Autolab](#)

[OH Queue](#)

[Forms](#)

Term Project Gallery

spring-22



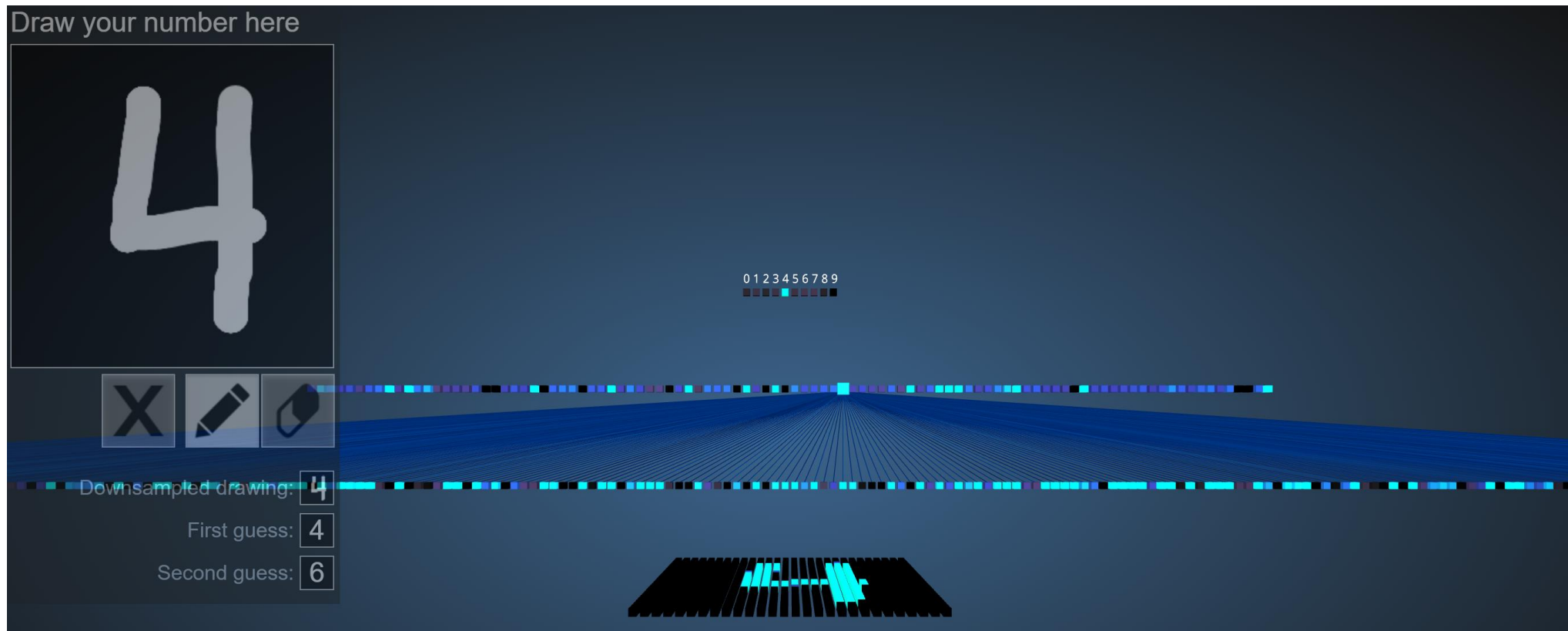
15-112
SPRING22

Term Project Lightning Round Video

As you walk in

Play with this demo of a neural network that classifies handwritten digits:

https://adamharley.com/nn_vis/mlp/3d.html





15-112
Lecture 2

Week 1 Thu
Getting Started

Instructor: Pat Virtue

Walk-through:

Three-neuron neural network

Note: a quick example of real-world functions that we can start to write already!

Design example: Three-neuron neural network

Note:

The following slides were adjusted to match what we ended up covering in lecture.

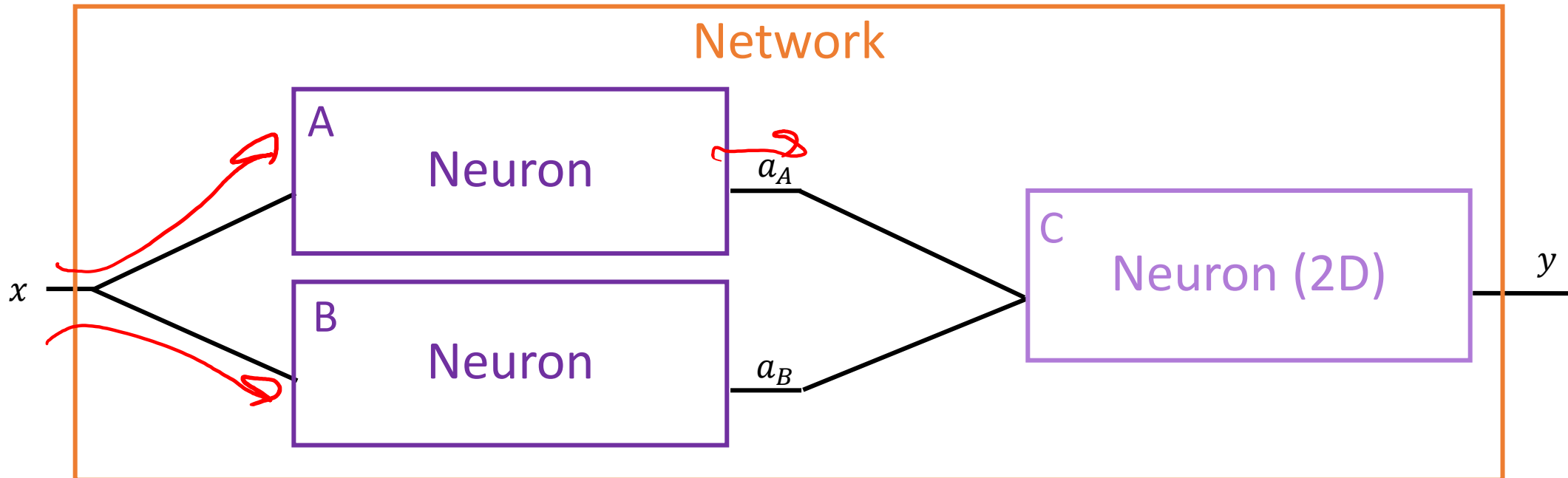
See the [Appendix](#) at the end of these slides for a few more steps to complete the three-neuron network code

Design example: Three-neuron neural network

High-level design of a small neural network

Input x : _____

Output y : _____

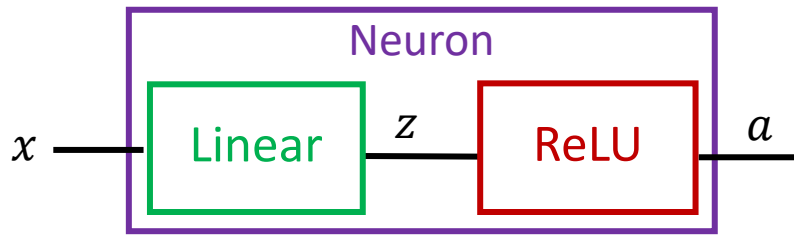
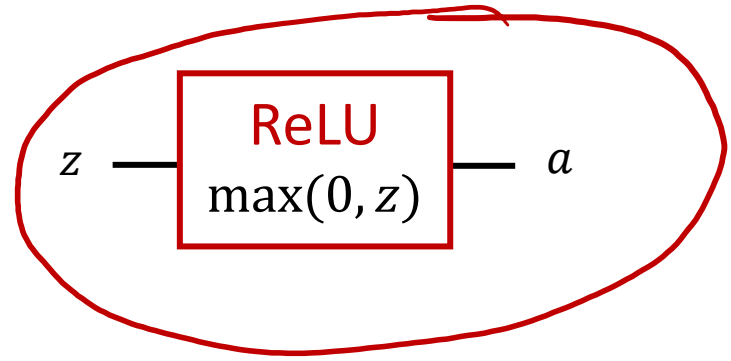
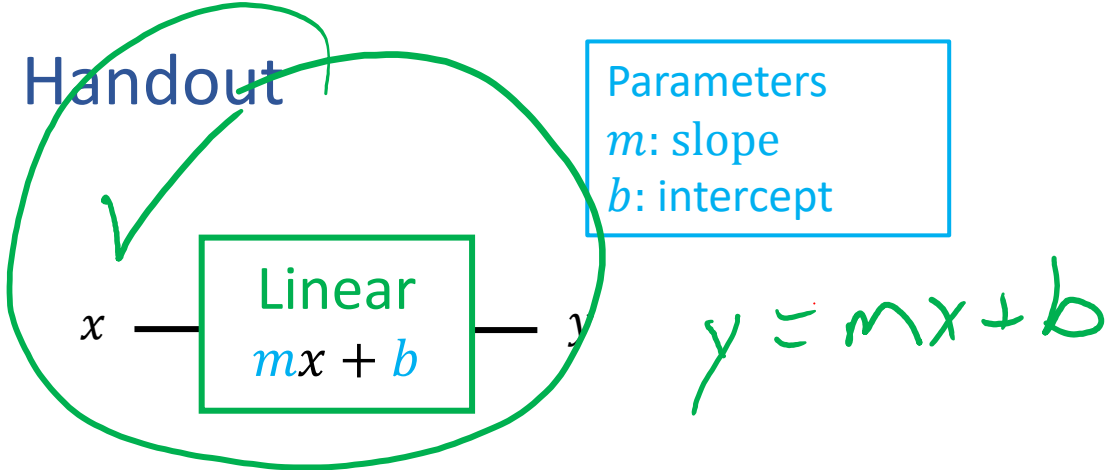


Temporary, intermediate values:
 a_A : _____
 a_B : _____

Design example: Neural Network Components

Handout

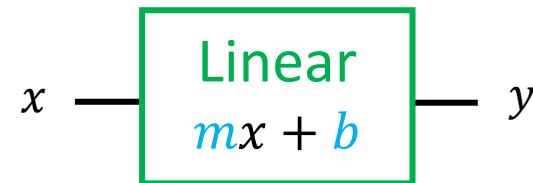
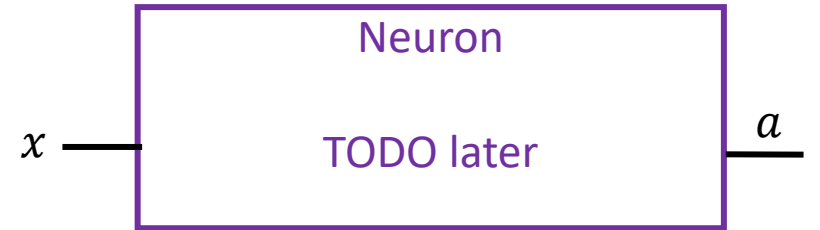
Parameters
 m : slope
 b : intercept



Example: Three-neuron neural network

Code – first pass: moving top-down

```
def network(x):  
    # Return y, the output of the network  
    pass # for now  
  
def neuron(x, m, b):  
    # Return a, the output of one neuron  
    pass # for now  
  
def linear(x, m, b):  
    # print(f'Inside linear({m}, {x}, {b}'))  
    return m*x + b
```



Example: Three-neuron neural network

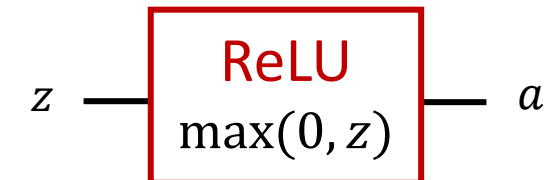
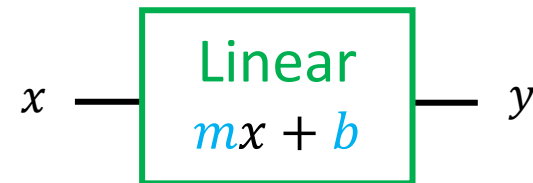
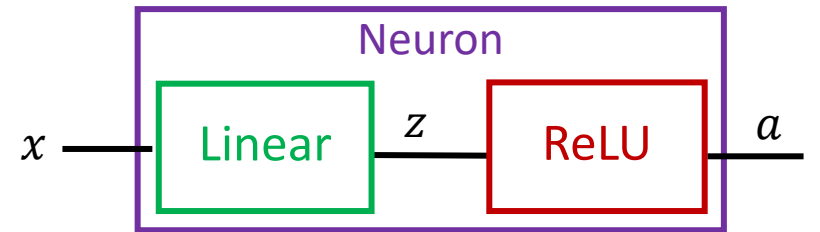
Code – move back up to implement neuron

```
def network(x):  
    # Return y, the output of the network  
    pass # for now
```

```
def neuron(x, m, b):  
    z = linear(x, m, b)  
    a = relu(z)  
    return a
```

```
def linear(x, m, b):  
    return m*x + b
```

```
def relu(z):  
    return max(0, z)
```



Walk-through: Three-neuron neural network

Concepts highlighted

- ✓ Defining functions
 - ✓ Calling functions
 - ✓ Print vs return
 - ✓ None type
 - ✓ Helper functions
 - ✓ Variable scope
 - ✓ if-else statement
 - ✓ Built-in functions (print, max)
 - ✓ Function composition
 - ✓ Top-down design (brief introduction)
- Handwritten notes: `float` (with arrow pointing to `type(0.63)`), `abs`

Announcements

Assignments:

112 student contract

- Due already

Week 2 Pre-reading Checkpoint

- Due Fri 9/2, 8 pm

HW1

- Due Saturday 9/3, 8 pm

Announcements

Recitation

Friday

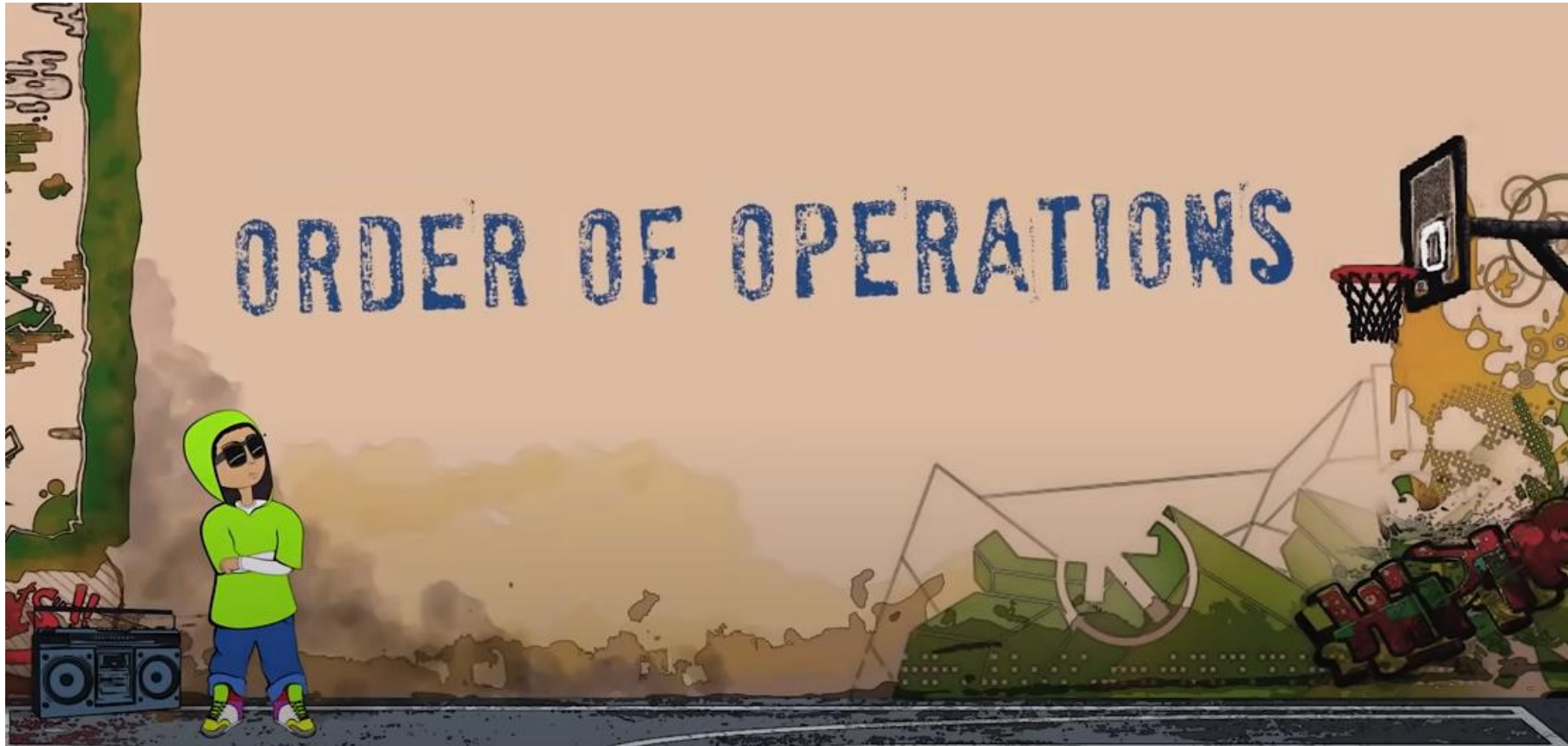
- Time reserved to work on **Pre-reading** and associated **Checkpoint**
- (Fix-it Fridays, but not in week 1)
- GHC 5th Floor Clusters

Operators and Order of Operations

Order of operations

PEMDAS

<https://www.youtube.com/watch?v=ZzeDWFhYv3E>

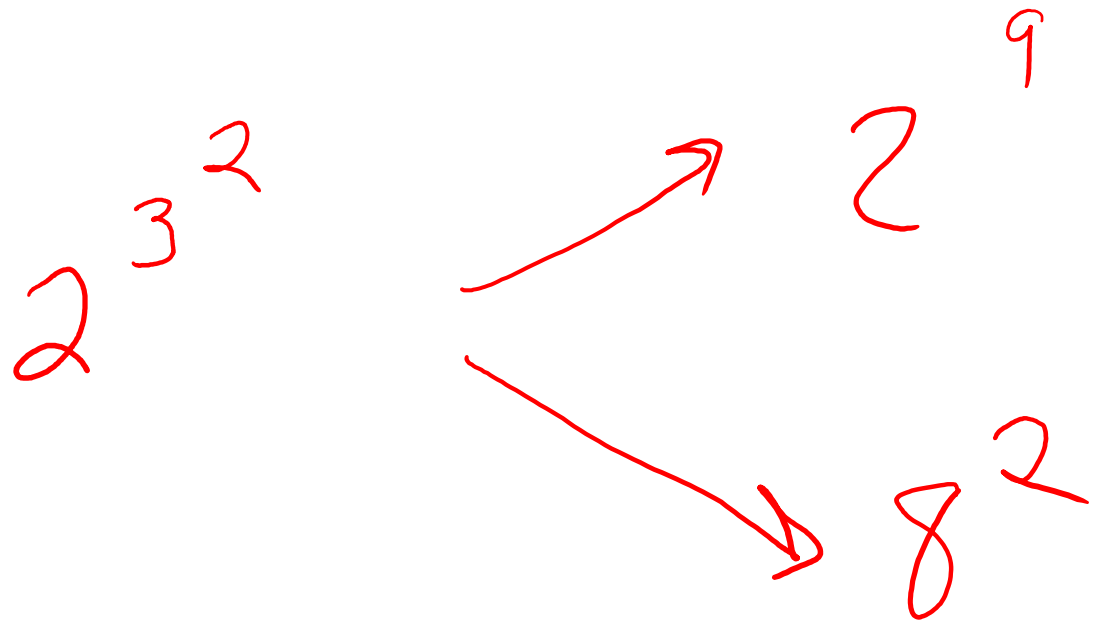


Poll 4 (unused)

What does this print?

```
print(2**3**2)
```

- A) 7
- B) 64
- C) 512
- D) Error



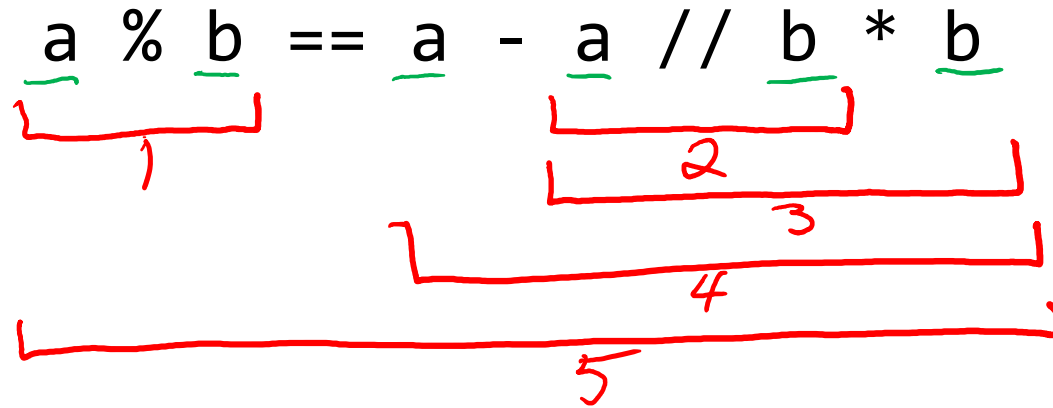
Debugging tip!

Expressions are things in Python that evaluate to a value

- 1) Save expressions (of all sizes) to variables
- 2) Use `print(expr)` to confirm values and order of operations

Poll 5

How many expressions are there in:



A) 4

B) 5

C) 6

D) 11

E) 21

F) Other

G) I have no idea

$ex1 = a \% b$
 $ex2 = a // b$
 $ex3 = ex2 * b$
 $ex4 = a - ex3$
 $ex5 = (ex1 == ex4)$

$print(ex1)$
 $print(ex2)$
 \vdots
 $print(ex5)$

Operators

Arithmetic

- +, -, *, /, //, **, %, - (unary), + (unary)

Relational

- <, <=, >=, >, ==, !=

Assignment

- +=, -=, *=, /=, //=, **=, %=

Logical

- and, or, not

Note: not covering the bitwise operators (for now at least)

<<, >>, &, |, ^, ~, &=, |=, ^=, <<=, >>=

Strings and Comments

Poll 6 (unused)

Which one does the right thing?

- A) `print("Have you read "Pride and Prejudice" by Jane Austen?")`
- B) `print("Have you read 'Pride and Prejudice' by Jane Austen?")`
- C) `print('Have you read "Pride and Prejudice" by Jane Austen?')`
- D) `print('Have you read "Pride and Prejudice" by Jane Austen?')`

Poll 6 (unused)

Which one does the right thing?

- A) `print("Have you read "Pride and Prejudice" by Jane Austen?")`
- B) `print("Have you read 'Pride and Prejudice' by Jane Austen?")`
- C) `print('Have you read 'Pride and Prejudice" by Jane Austen?')`
- D) `print('Have you read "Pride and Prejudice" by Jane Austen?')`

```
print("Have you read "Pride and Prejudice" by Jane Austen?")
```

```
print("Have you read 'Pride and Prejudice' by Jane Austen?")
```

```
print('Have you read 'Pride and Prejudice" by Jane Austen?')
```

```
print('Have you read "Pride and Prejudice" by Jane Austen?')
```

Comments

Notes for humans (really important!)

```
# Comments can go on their own line
i = 0 # Comments can go at the end of a line

def squared(x):
    """ This is technically a multiline string
        but is often used as a comment
    """
    return x**2
```

Strings

Single or double quote are fine

- Can be useful for quotes withing strings (but alternated correctly)
- Escape characters are needed sometimes

```
print('Have you read Jane Austen\'s "Pride and Prejudice" recently?')
```

- There are also triple quotes for multiline strings (actually, often used for comments)

f-Strings

- Really useful to print a combination of strings and expressions

```
x = 42
```

```
y = 99
```

```
print(f'Did you know that {x} + {y} is {x+y}?')
```

42 99 141

Errors

Prev Poll 3

What college are you in?

- A) Letss eat Grandma
- B) Letss eat, Grandma
- C) Lets eat Grandma
- D) Lets eat, Grandma
- E) Let's eat Grandma
- F) Let's eat, Grandma

Lessons learned

- Sensitive to small things
 - Like spelling, grammar, usage
 - Different kinds of error
- Different from language to language
- Be patient while you learn
 - With yourselves
 - With each other
- Commas save lives
- Don't consume your relatives

Errors

Syntax error

```
print("100") # Never prints  
1 ? 0  
print("200") # Never prints
```

Runtime error

```
print("100") # Prints!  
1 / 0  
print("200") # Never prints
```

Logical error

```
print(f"100:, {x}") # Prints!  
if x % 2 == 1:  
    print(f"{x} is even") # Prints?  
print("200") # Prints!
```

Debugging tip!

- Use print functions to help learn where runtime errors are happening

Debugging tip!

- Use print functions to see if branches of code are being entered

Poll 7 (unused)

What happens when we run the following line?

```
x = 3(2+7)
```

- A) x takes on the value 27
- B) Syntax error
- C) Runtime error
- D) Logical error
- E) I have no idea

Errors

Tip

Keep a list of errors that you encounter along with what they might mean

TypeError: 'int' object is not callable

→ Hmm, I probably have number, variable, or expression followed by a (

e.g., `x = 3(2+7)` should be `x = 3*(2+7)`

NameError: name 'total' is not defined

→ Hmm, I probably have variable named total that I never assigned a value

```
num = 10
```

```
mean = total/num
```

Functions

Variables and Types

type vs isinstance

Both work to check

More complex

Poll 8 (unused)

What is this?

- A) Apple
- B) Banana
- C) Fruit
- D) Food



type vs isinstance

Both work to check

More complex

Poll 9 (unused)

What should this print?

```
print(0.1 + 0.1 + 0.1 == 0.3)
```

- A) bool
- B) True
- C) False
- D) Yes
- E) No

Issues with floats

Equality

```
x = 0.1 + 0.1 + 0.1
```

```
y = 0.3
```

```
x == y # Doesn't work well with floats
```

- Use 112: `almostEqual(x, y)`

Rounding

```
round(x) # Doesn't work as you might expect
```

- Use 112: `roundHalfUp(x)`

Conditionals

Appendix

Additional examples with neural networks

Three-neuron neural network

Handout

Input x : _____

Output y : _____

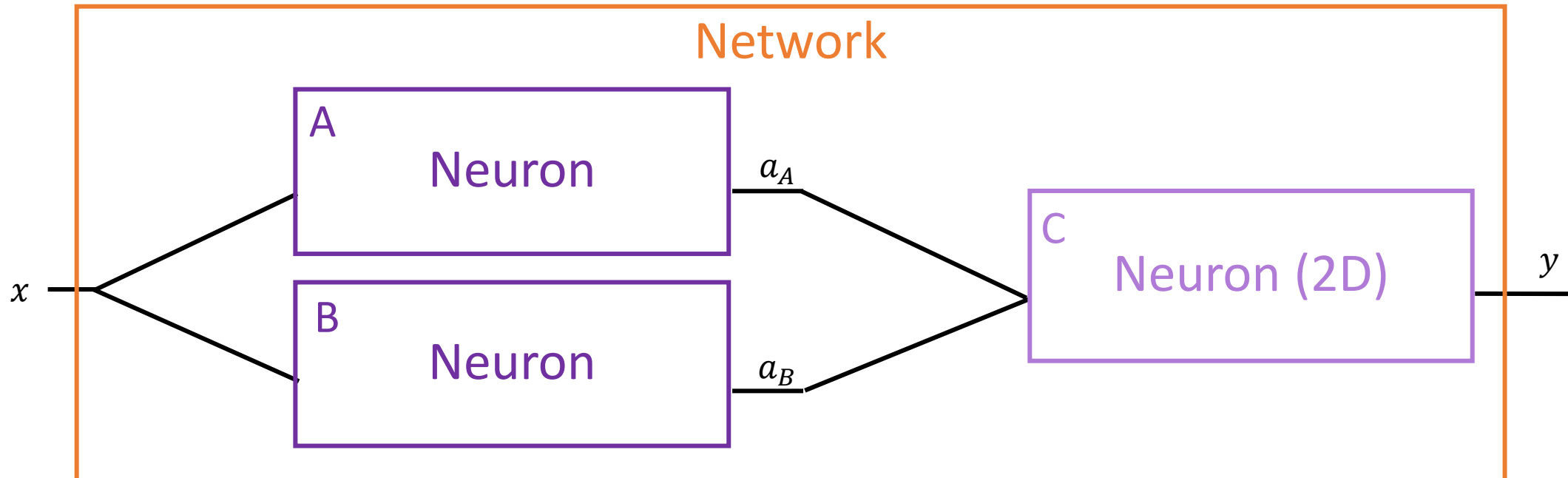
Parameters

m_A : -0.01 b_A : 0.7

m_B : 0.05 b_B : -3.5

$m_{C,1}$: 1.0 b_C : 1.5

$m_{C,2}$: 1.0

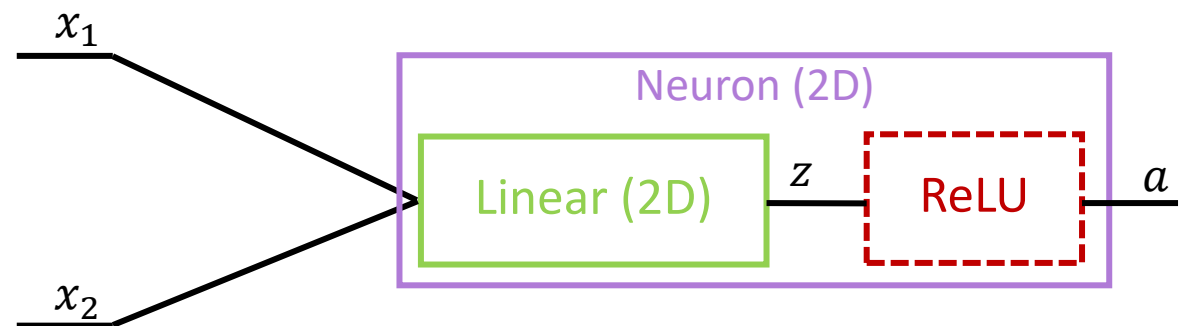
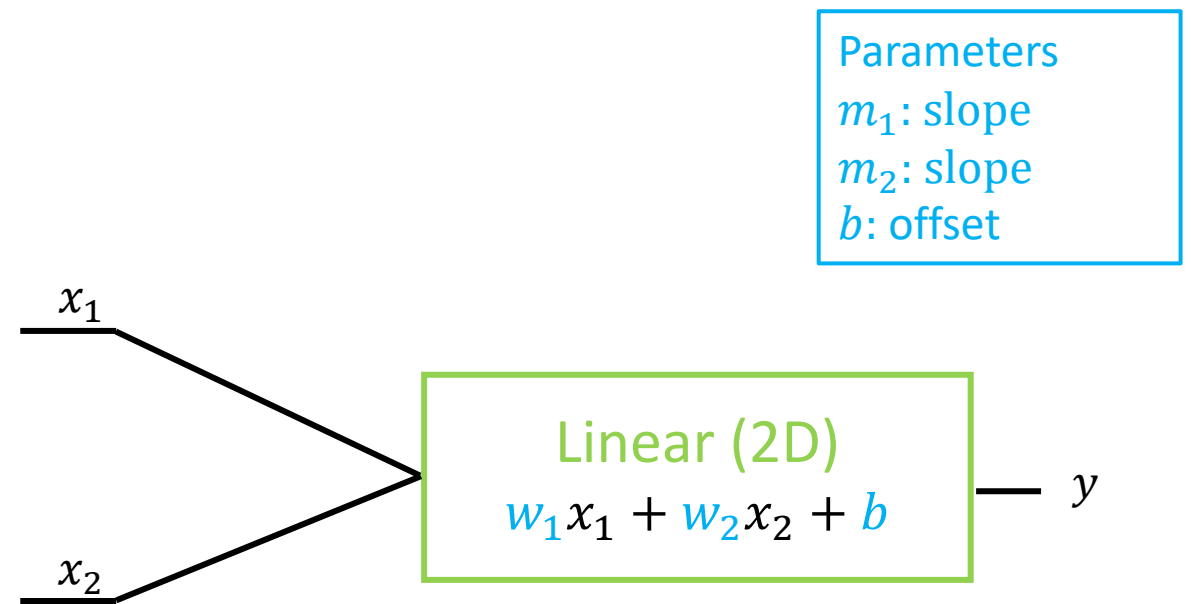
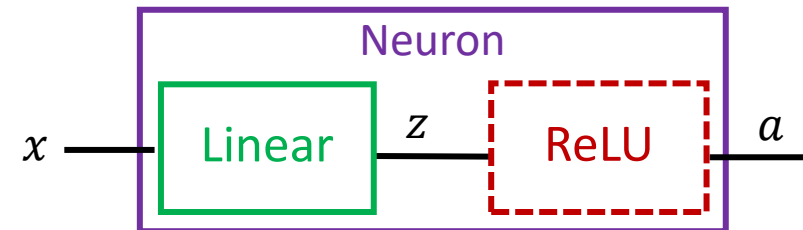
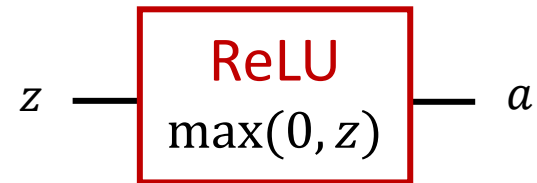
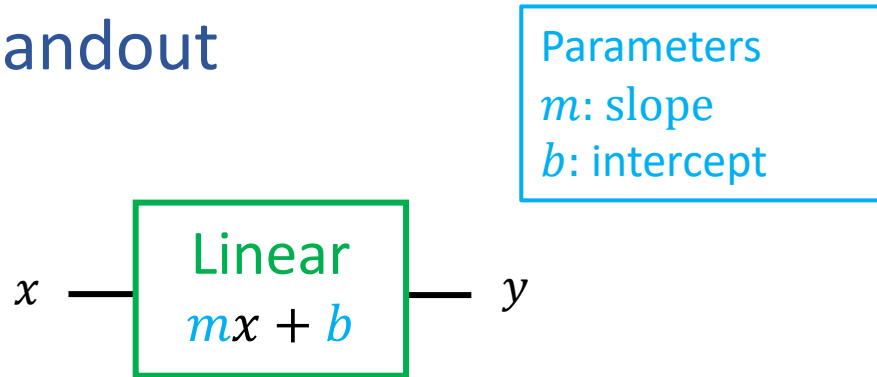


a_A : _____

a_B : _____

Neural Network Components

Handout



Three-neuron neural network

Handout

Input x : _____

Output y : _____

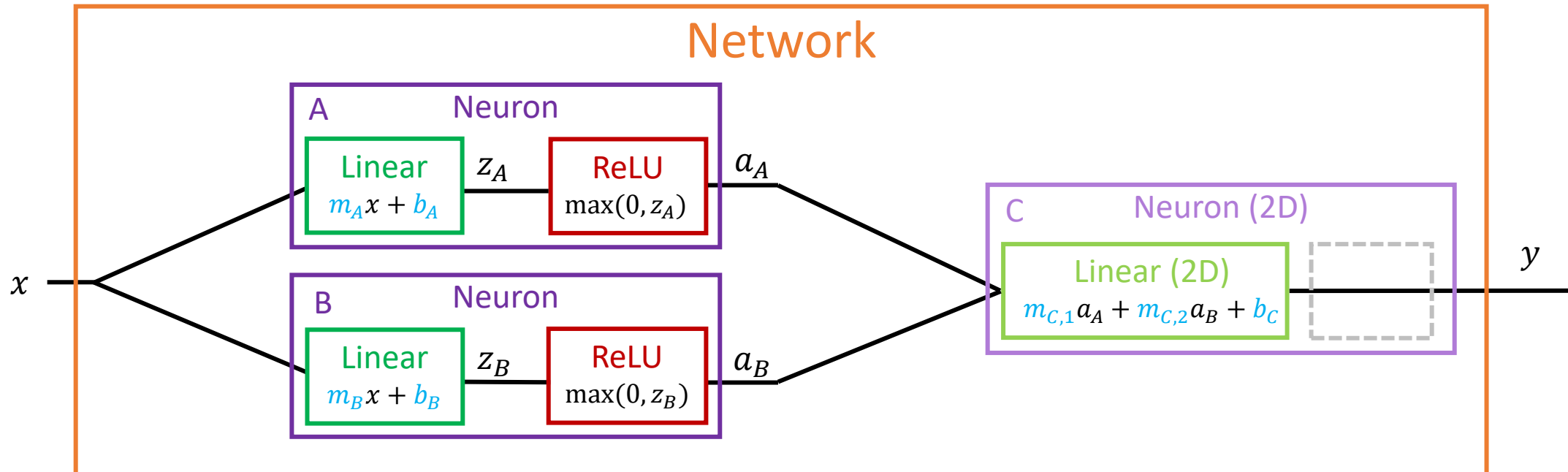
Parameters

m_A : -0.01 b_A : 0.7

m_B : 0.05 b_B : -3.5

$m_{C,1}$: 1.0 b_C : 1.5

$m_{C,2}$: 1.0



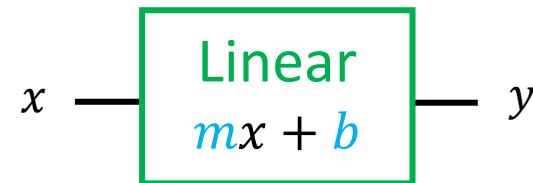
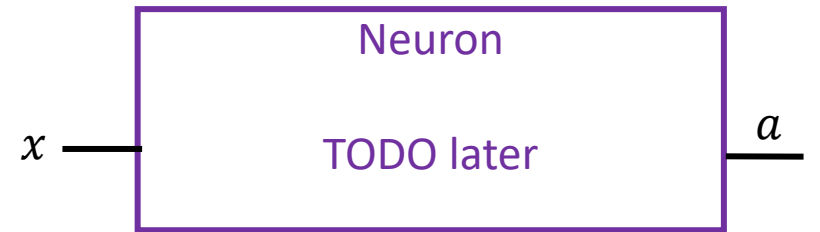
z_A : _____ a_A : _____

z_B : _____ a_B : _____

Example: Three-neuron neural network

Code – first pass: moving top-down

```
def network(x):  
    # Return y, the output of the network  
    pass # for now  
  
def neuron(x, m, b):  
    # Return a, the output of one neuron  
    pass # for now  
  
def linear(x, m, b):  
    # print(f'Inside linear({m}, {x}, {b}'))  
    return m*x + b
```



Example: Three-neuron neural network

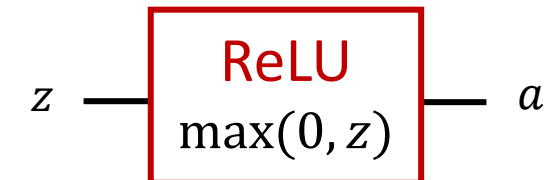
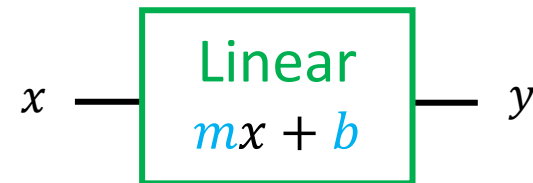
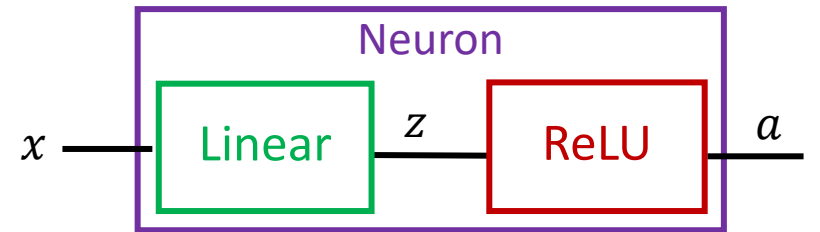
Code – move back up to implement neuron

```
def network(x):  
    # Return y, the output of the network  
    pass # for now
```

```
def neuron(x, m, b):  
    z = linear(x, m, b)  
    a = relu(z)  
    return a
```

```
def linear(x, m, b):  
    return m*x + b
```

```
def relu(z):  
    return max(0, z)
```



Example: Three-neuron neural network

Adding slightly different versions of linear and neuron to handle two input values rather than just one

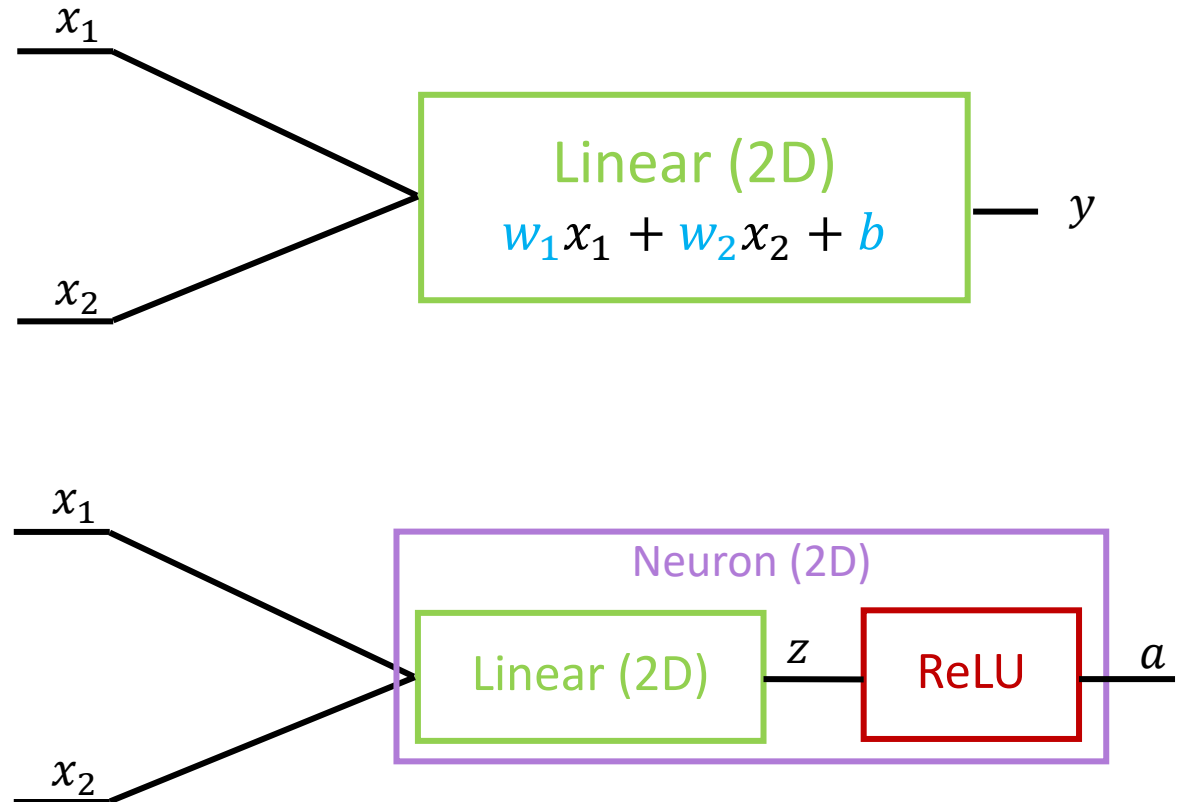
- Side-note: once we get to lists, we won't need different versions

Parameters
 m_1 : slope
 m_2 : slope
 b : offset

```
def linear2D(x1, x2, m1, m2, b):  
    return m1*x1 + m2*x2 + b
```

```
def neuron2D(x1, x2, m1, m2, b):  
    z = linear2D(x1, x2, m1, m2, b)  
    a = relu(z)  
    return a
```

```
# relu still just takes on input
```



Example: Three-neuron neural network

Code – quick adjustment to neuron code: **sometimes ReLU is optional**

```
def neuron(x, m, b, useReLU):
```

```
    z = linear(x, m, b)
```

```
    if useReLU:
```

```
        a = relu(z)
```

```
    else:
```

```
        a = z
```

```
    return a
```

```
def neuron2D(x1, x2, m1, m2, b, useReLU):
```

```
    z = linear2D(x1, x2, m1, m2, b)
```

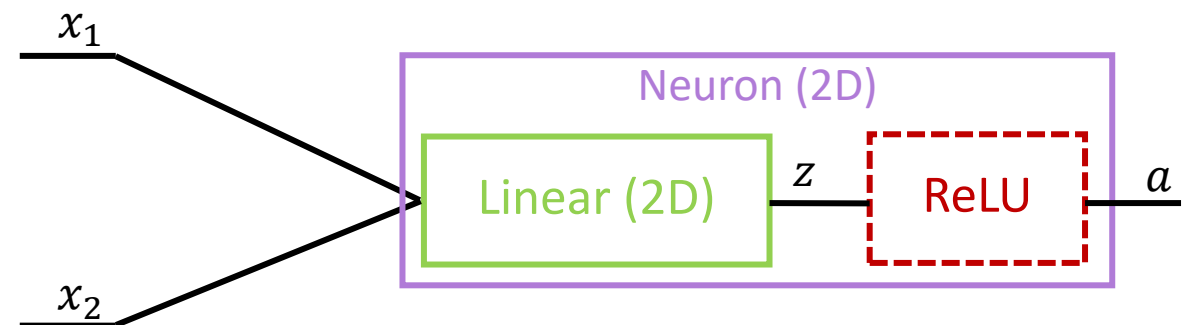
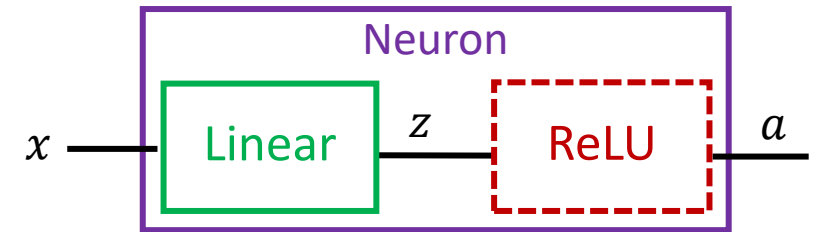
```
    if useReLU:
```

```
        a = relu(z)
```

```
    else:
```

```
        a = z
```

```
    return a
```



Example: Three-neuron neural network

Code – Putting the network together!!

```
def network(x, mA, bA, mB, bB, mC1, mC2, bC):  
    aA = neuron(x, mA, bA, useReLU=True)  
    aB = neuron(x, mB, bB, useReLU=True)  
  
    y = neuron2D(aA, aB, mC1, mC2, bC, useReLU=False)  
    return y
```

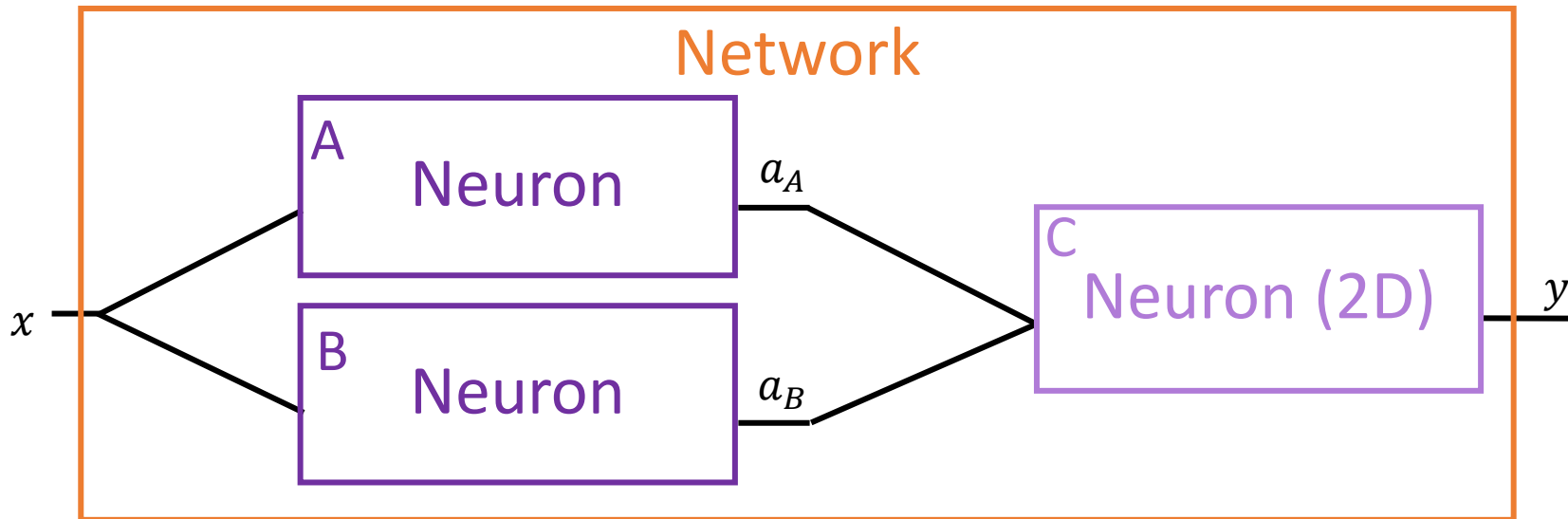
Parameters

m_A : -0.01 b_A : 0.7

m_B : 0.05 b_B : -3.5

$m_{C,1}$: 1.0 b_C : 1.5

$m_{C,2}$: 1.0



Example: Three-neuron neural network

Code – Using to predict traffic in Minneapolis, Minnesota

`xNew = 8` # What is the traffic like at 8 am?

`yPredicted = network(xNew, mA, bA, mB, bB, mC1, mC2, bC)`

Parameters

$m_A: 0.4$ $b_A: -5.9$

$m_B: -0.6$ $b_B: 5.2$

$m_{C,1}: -1.0$ $b_C: 5.1$

$m_{C,2}: -1.0$

