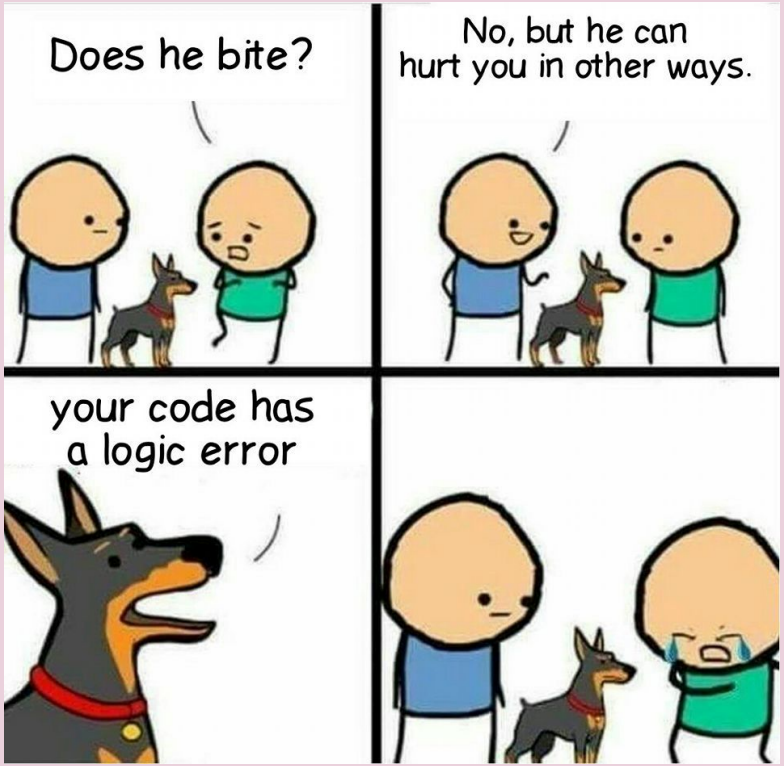
Day 2: Using Logic: If/Else

July 11th 2023



bit.ly/3O7T5nL

Previous Day Clarifications

- using quotes in environment diagrams
- mango-mango example error
- additional print statements problem 1
- homework review
- terms:
  - **script** = list of code instructions
  - print statements appear in the **shell**
- plagiarism chat

Environment Diagrams Online!

https://pythontutor.com/python-debugger.html#mode=edit

^^ an online resource for visualizing your code step by step!

example

https://replit.com/@camp-code-summer-2023/Day2Workspace

https://replit.com/@camp-code-summer-2023/Day2Workspace

Write a script that starts with Boolean variables `b1, b2, b3` and `b4`. Your task is to translate the following English expressions into Python expressions, and assign your Python expressions to variables `A1` and `A2`. For example, to assign an expression such as “either `b1` or `b2` is true” to variable `A0`, would be translated to “`A0` = `b1` or `b2`”.

Print the values `A1` and `A2` at the end of the script.

Assign `A1` to "At least one of `b1` or `b2` is true, and at least one of `b3` or `b4` is true"

Assign `A2` to "Either both `b1` and `b2` are true, or both `b3` and `b4` are true"

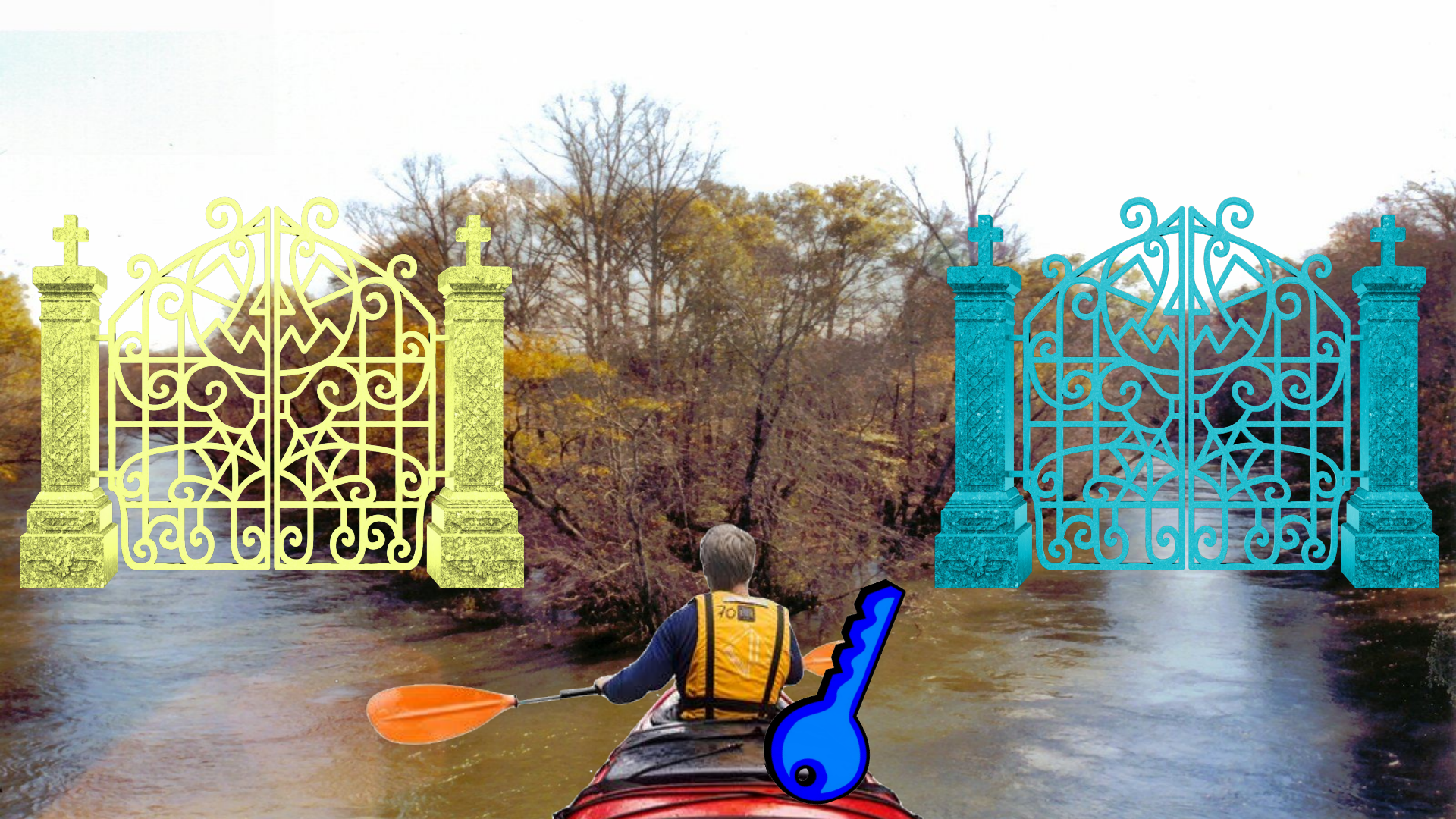
Conditionals

now that we know about booleans (making things TRUE or FALSE) we can use them to decide **whether to run code**

these are called **conditional statements** and in python they are `if`, `else`, and `elif` (else+if)

until now we have run all code we see – this will allow us to run some code only if certain conditions are true!

this will also introduce **indentation** which helps the computer know what to run with each statement.



Pseudo-Code

(english, not python, description of what happened)

if you bring a yellow key, you pick up a pig

if you bring a blue key, you pick up a sheep

after gates, you have a passenger

Conditionals: `if`

if is our first building block. this checks whether the code after it is true or false

let's look at a script to see if it is snowing (if it is below 32 F AND there is precipitation)

you can follow along on replit!

take note of indentation and colon: – most code editors will automatically help

```
# exercise 1: snow or no?
temperature_fahrenheit = 40
precipitation = True

if (temperature_fahrenheit < 32) and (precipitation == True):
  snow_or_no = True
if (temperature_fahrenheit >= 32) and (precipitation == True):
  snow_or_no = False
if (temperature_fahrenheit < 32) and (precipitation == False):
  snow_or_no = False
if (temperature_fahrenheit >= 32) and (precipitation == False):
  snow_or_no = False
print('is it snowing? ')
if snow_or_no == True:
  message = 'yes!'
if snow_or_no == False:
  message = 'no :('
print(message)
```

indentation should be a **tab** key or four spaces

Pseudo-Code to code

(english, not python, description of what happened)

if you bring a yellow key, you pick up a pig

if you bring a blue key, you pick up a sheep

after gates, you have a passenger

```
key_color = 'blue'
if key_color == 'blue':
    passenger = 'sheep'
if key_color == 'yellow':
    passenger = 'pig'
print(passenger)
```

```
# indentation introduction
# your teachers go to a coffee shop
# how would you change the following code
# so that we all order a scone?
instructor_name = 'haley'
print("I'd like to order a coffee ")
if instructor_name == 'haley':
    print('with almond milk!')
if (instructor_name == 'kei') or (instructor_name == 'annika'):
    print('with soy milk!')
    print('and a scone.')
print('please!')
```

use parentheses to help organize your goal in the code – remember, as programmers we want to communicate as clearly as possible with the machines!

```
# notice !! this will not work!
if instructor_name == 'kei' or 'annika':
    print('with soy milk!')
```

Remember: computers are not very smart!

what happens in this code if you bring a green key? what will the print line result in?

it’s important to write code which can deal with situations you may not have anticipated – which is one way the next feature can help catch

```
key_color = 'blue'
if key_color == 'blue':
    passenger = 'sheep'
if key_color == 'yellow':
    passenger = 'pig'
print(passenger)
```

Conditionals: `else`

if statements are powerful, but it would be tedious to write out the opposite of every if statement. so, we can use **else** to tell the computer what to do when our first condition is false!

the **else** gets unindented and matches with the closest 'if' containing statement

where else could we use an **else** in this code?

```
# exercise 2: snow or no?
temperature_fahrenheit = 20
precipitation = True

if temperature_fahrenheit < 32 and precipitation == True:
  snow_or_no = True
else:
  snow_or_no = False
print('is it snowing? ')
if snow_or_no == True:
  message = 'yes!'
if snow_or_no == False:
  message = 'no :('
print(message)
```

environment diagram: the else does not get evaluated when the if is true!

Conditionals: `elif`

we might have more than one thing to check for. now I'm going to make 'snow or no' into a percent chance instead of a boolean, and if it's below 32 degrees but no precipitation yet, we should make it a 50% chance.

```
# exercise 3: snow or no?
temperature_fahrenheit = 20
precipitation = False

if temperature_fahrenheit < 32 and precipitation == True:
    snow_or_no = 100
elif temperature_fahrenheit < 32 and precipitation == False:
    snow_or_no = 50
else:
    snow_or_no = 0
print('percent chance of snow: ')
print(snow_or_no)
```

Conditionals: elseif

elseif is really powerful to let you not type out so many conditional statements

let's say you are building a computer vision algorithm which takes in pictures of animals to decide if they can fly. you know if the animal doesn't have wings, it definitely can't fly. but, some animals with wings can't fly (sorry, penguins). we can use elif to code this situation!

elif ONLY runs when if statement is false

```
# exercise 4: flying animals
has_wings = True
is_penguin = False
if not has_wings:
    can_fly = False
elif is_penguin:
    can_fly = False
else:
    can_fly = True
print('can my animal fly?')
print(can_fly)
```

notice – in this code, we use variables directly and don't say == True – if the variable is true, that will be true!

Conditionals: elseif

you can have multiple elseif under the same if statement – it is special

only one if / else statement can go together without starting a new sequence. but you can have multiple elseif if you want!

```
# multiple elif
instructor = 'haley'
if instructor == 'haley':
    print('go bears!')
elif instructor == 'kei':
    print('here we go kohawks!')
elif instructor == 'annika':
    print('go crimson!')
else:
    print('impostor!! 👽')
```

Common Bug: Conditionals

remember that once a conditional ends, it doesn't remember what was going on above unless you save the information into a variable. a very common bug (even for experienced programmers!) is to chain two if statements instead of remembering elif.

what would happen if we passed a **dog** into this script?

```
# exercise 5: dog
has_wings = False
is_penguin = False
if not has_wings:
    can_fly = False
if is_penguin:
    can_fly = False
else:
    can_fly = True
print('can my animal fly?')
print(can_fly)
```

Group Problem: Conditionals

uh oh! you forgot about ostriches which also have wings but can't fly. how would you change the code to include them?

what about flying squirrels, who have no wings but can fly?

add new variables or change the logic structure however you want!

```
has_wings = True
is_penguin = False
if not has_wings:
    can_fly = False
elif is_penguin:
    can_fly = False
else:
    can_fly = True
print('can my animal fly?')
print(can_fly)
```

Group Problem: Conditionals

uh oh! you forgot about ostriches which also have wings but can't fly. how would you change the code to include them?

what about flying squirrels, who have no wings but can fly?

add new variables or change the logic structure however you want!

```
has_wings = False
is_penguin_or_ostrich = False
if_flying_squirrel = True
if is_penguin_or_ostrich:
    can_fly = False
elif if_flying_squirrel:
    can_fly = True
else:
    can_fly = has_wings
print('can my animal fly?')
print(can_fly)
```

New tool: +=

If you want to add a value to a variable, you don't need to write out the variable twice – a shortcut in python

a = a + 1

a += 1

both of the above lines do the same thing! you can use the same *syntax* (meaning how code is written) with other operators like minus and times

Descriptions of all python operators: https://www.w3schools.com/python/python_operators.asp

New types of variables

- empty strings: literally ‘’ or “” with no space. can be useful to begin a string you want to add things to as your code runs.
- NaN: not a number, to represent numerical data which is **undefined**. typically this is something like 10/0 which does not have a value a computer can store / process. sometimes used as a placeholder before you set a value into a variable, or if a value is missing (example: you want to save a table out and not all students have a middle name)

Group Problems: Conditionals 2

assume you have a variable CEL which stores the temperature in celsius, write code to:

- converts the value to fahrenheit
- prints that value
- then decides if the temperature is low enough for it to be snowing (<=32 fahrenheit)
- if it is cold enough, prints as many snowflakes (‘*’) as it is degrees fahrenheit outside
    - you can multiply strings by ints! (hint: not floats)

The temperature *T* in degrees Celsius (°C) is equal to the temperature *T* in degrees Fahrenheit (°F) minus 32, times 5/9:

$$T_{(°C)} = (T_{(°F)} - 32) \times 5/9$$

or

$$T_{(°C)} = (T_{(°F)} - 32) / (9/5)$$

or

$$T_{(°C)} = (T_{(°F)} - 32) / 1.8$$

```
> squiggle = '~'
> squiggle * 6
'~~~~~~'
> (squiggle * 6) + ' ' + '^'*7
'~~~~~~ ^^^^^^^'
>
```

Group Problems: Conditionals

assume you have a variable CEL which stores the temperature in celsius, write code to:

- converts the value to fahrenheit
- prints that value
- then decides if the temperature is low enough for it to be snowing (<=32 fahrenheit)
- if it is cold enough, prints as many snowflakes (‘*’) as it is degrees fahrenheit outside
    - you can multiply strings by ints! (hint: not floats)

```
CEL = -1
temp_fahrenheit = (CEL * 1.8) + 32
print(temp_fahrenheit)
snow_weather = temp_fahrenheit <= 32
if snow_weather:
    snow_string = '*' * int(temp_fahrenheit)
    print(snow_string)
```

New python feature: user input

- so far, we have only used information in our scripts which is “hard coded” meaning the code itself contains a value
- but, many times we want to wait for a user to input information so that we can run different things depending on what input we get
    - example: password checker. if typed correctly, login, otherwise ask again. need to wait for what the user types!
- to do this, the simplest thing in python is to use the input() function. this generates a popup or shell line (depending on how you are coding) which the user can type in.
- we want to assign what is typed to a variable so the computer remembers the information
- you can add a message to the input call to tell the user what you are asking for (end space helps!)

```
true_password = 'I_l0ve_c0d!ng'
guessed_password = input('please type your password ')
if guessed_password == true_password:
    print('welcome to your account!!')
else:
    print('try again')
```

New python feature: user input

- user input is **always** of type string when it is first entered
  - even if it only contains numbers!
- if you would like to use the input as a number, you will need to **cast** it using the same commands we saw before for floats and ints
- this code will never enter the if statement – what can you do to fix it?

```
want_number = input('what is your favorite number? ')
if want_number == 4:
    print('right choice!')
else:
    print("why don't you recognize 4 is the best number??")
```

A note on quotes

some of our examples use double quotes (“ “) and some of the examples use single quotes (‘ ‘) when we are “defining” ‘strings’

in python, this is user preference, but in other languages you might learn they have different meanings

one time it helps to pay attention is if you need to use one or the other in your string!

```
quote = 'They just said "Hi!" to me, can you believe?'
mystr = "I can't fail"
mystr = 'I don't work'
```

Group problems: diner discounts

A diner gives discounts for certain customers. Your task is to write a script that prints the final bill after considering discounts given a total amount.

- 10% discount for senior citizens (65 or older)
- 5% discount for veterans

assume you have a variable **bill** with the initial total, and ask your user if they are a veteran and how old they are

```
# approach 1: calculate final bill as you go though the rules
bill = 100  # you can change the value
vet = input("Are you a veteran (Y/N):")
age = int(input("How old are you:"))

finalBill = bill
if age >= 65:
  finalBill *= .9
if vet == "Y" or vet == "y":
  finalBill *= .95
print(finalBill)
```

```
# approach 2: calculate the discount first
bill = 100  # you can change the value
vet = input("Are you a veteran (Y/N):")
age = int(input("How old are you:"))

discount_pct = 0
if vet == "Y" or vet == "y":
  discount_pct += .05
if age >= 65:
  discount_pct += .1
finalBill = (1 - discount_pct) * bill
print(finalBill)
```

Nested if / else

although we can string together long logical commands, it is often difficult to keep track of how each statement affects the others. luckily, we can **nest** if/elif/else statements by using additional indentation!

```
non_dairy_here = True
temp_fahrenheit = 76
order = "I'd like a "
if non_dairy_here:
    order += 'latte'
else:
    if temp_fahrenheit < 40:
        order += 'green tea'
    else:
        order += 'black iced coffee'
order += ', please!'
print(order)
```

when you go out for coffee, you like lattes but only if non-dairy milk is available. when it isn't, you order tea when it is cold and iced coffee when it's hot

Nested if / else

although we can string together long logical commands, it is often difficult to keep track of how each statement affects the others. luckily, we can **nest** if/elif/else statements by using additional indentation!

Problem: Nested loops

You love playing basketball, where you can either make shots worth two points or difficult shots worth three points. You have a strategy you use to decide when you should shoot from inside or outside the three point line.

When the game is close, meaning the teams are less than ten points apart, you don't want to risk a three point shot and will make a two point one. If you are winning the game, you also don't try to make three point shots. However, when you are losing by more than ten points, you only go for three point shots to try to catch up.

Write code **using a nested loop** which starts with two variables representing each team's score, and prints what type of shot you should make. Remember to track which team is which!

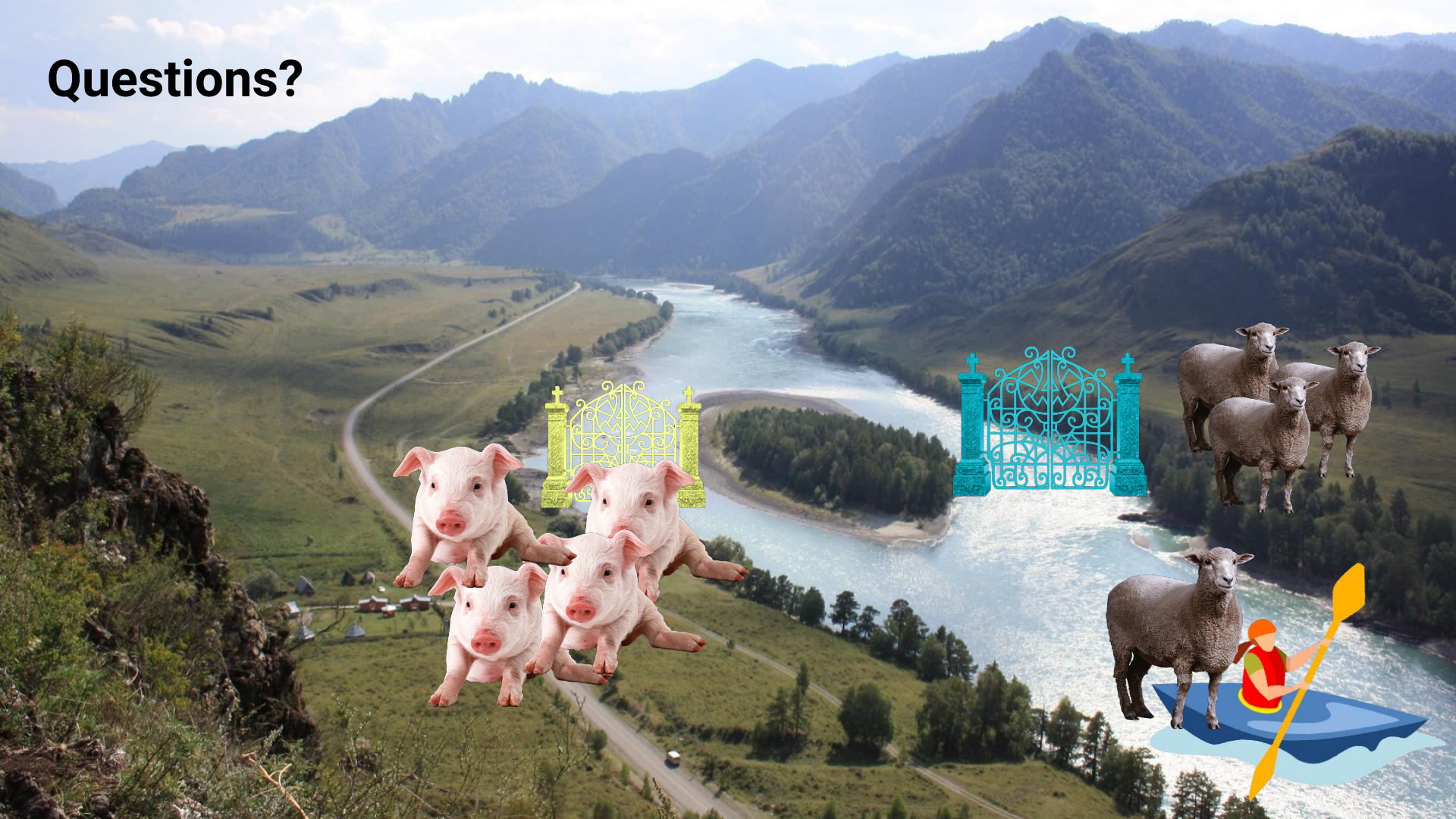
Problem: Nested loops

When the game is close, meaning the teams are less than ten points apart, you don't want to risk a three point shot and will make a two point one. If you are winning the game, you also don't try to make three point shots. However, when you are losing by more than ten points, you only go for three point shots to try to catch up.

```
myteam = 87
otherteam = 66
if myteam > otherteam:
  print('shoot a two, we are winning')
else:
  if myteam < (otherteam - 10):
    print('go for three!!')
  else:
    print('shoot a two, things are close')
```

note the vertical lines replit (and other editors) add to help you track the nesting!

Questions?



Calculator Problem 1

Your task is to write a script that accomplishes the following things:

1. Asks the user to input two variables, and assign them to values x1 and x2
2. Creates a variable x3 which takes the value:
    a. -2, if the sum of x1 and x2 is less than -10
    b. -1, if the sum of x1 and x2 is between 0 and -10
    c. 0, if the sum of x1 and x2 is between 0 and 10
    d. 1, otherwise
3. Prints x3

Calculator Problem 2

Your task is to write a script that accomplishes the following things:

1. Creates a first variable (with your choice of name) and assigns it the value 102.5
2. Creates a second variable which is equal to the square of the first variable.
3. Creates a third variable which is equal to 3 times the second variable plus the square root of the first variable.
4. Prints the statement: “My final answer is: “ followed by the third variable.
5. Challenge: can you use only one print statement to complete the last task?

Ibuprofen Problem

It is important for us to be able to understand what dosage of medicine should be given. To assist with this, the manufacturer has issued the following guidelines:

- Adult Dose for Fever: 200 to 400 mg orally every 4 to 6 hours as needed.
- Pediatric (6 months - 12 years) Dose for Fever:
  - For temperatures less than 102.5° F (39.2 C), 5mg/kg/dose orally every 6 to 8 hours as needed.
  - For temperatures 102.5° F (39.2 C) or higher, 10mg/kg/dose orally every 6 to 8 hours as needed.
  - Warning: Do not exceed the maximum adult dosage.
- Infant (less than 6 months) Dose for Fever: Do not use!

Write a script that prints the recommended amount (in mg) of ibuprofen that should be given to an individual based on inputs (you need to ask for inputs!).

Hint: try not to ask for things you don't need, like the weight of an adult!

EEG Research Problem

you work in a research lab where participants play a computer game while you record EEG signals. on each trial, you know how fast the participant answered in ms and whether they got the question right or wrong. before you look at the EEG data, you want to separate the types of trials to use – the brain signal you're looking for only happens on **correct** trials. additionally, you know that if the subject took less than 200ms to respond they didn't have time to actually do the task (impossibly fast) and that if they took more than 4000ms, they weren't paying close attention, so you also want to exclude the trials outside these bounds.

write a script which takes in trial information (speed in ms and correct/incorrect as a boolean) and prints a boolean variable which tells you if a trial is good to use or not.