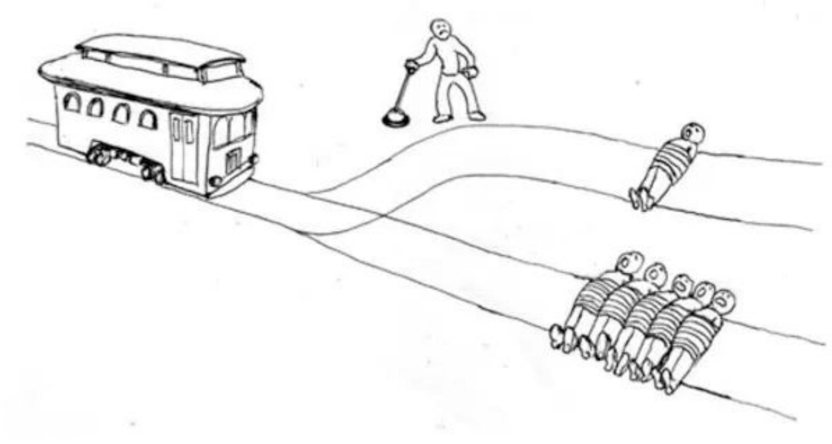


Conditionals

Motivation

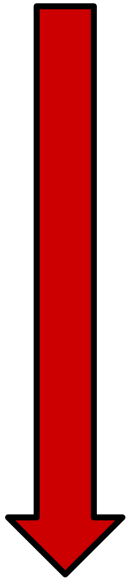
Motivation

So far our code is BORING

Motivation

So far our code is BORING

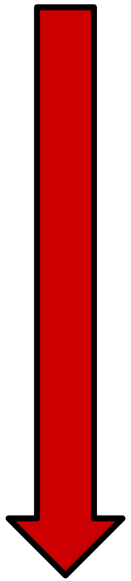
Execution (so far)



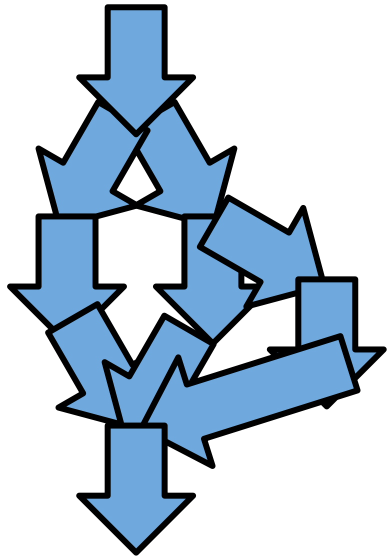
Motivation

So far our code is BORING

Execution (so far)



Execution (aspirational)



Computational thinking

Some classes at GV have many sections (like this class!)

Computational thinking

Some classes at GV have many sections (like this class!)

Think of an algorithm to choose which section to sign up for!

Computational thinking

Some classes at GV have many sections (like this class!)

Think of an algorithm to choose which section to sign up for!

If section is at 8am -> nope!

Computational thinking

Some classes at GV have many sections (like this class!)

Think of an algorithm to choose which section to sign up for!

If section is at 8am -> nope!

If section is taught by the worst prof -> nope!

Computational thinking

Some classes at GV have many sections (like this class!)

Think of an algorithm to choose which section to sign up for!

If section is at 8am -> nope!
If section is taught by the worst prof -> nope!
If section conflicts with another class you want -> nope!

etc!

Branching code

We can make our code do different things based on yes/no questions

Branching code

We can make our code do different things based on yes/no questions

We can have branching execution, not just top-down!

Uh oh!

Uh oh!

So far we’ve talked about variables, functions,
ints, floats, strings, etc.

Uh oh!

So far we’ve talked about variables, functions,
ints, floats, strings, etc.

MANY things in CS are named poorly or named
after someone

Uh oh!

So far we’ve talked about variables, functions,
	ints, floats, strings, etc.

MANY things in CS are named poorly or named
	after someone

We need to talk about **boolean** variables
	Named after George Boole

Booleans

We’re going to be asking yes/no questions

Booleans

We’re going to be asking yes/no questions

However, we’ll refer to True (yes) or False (no)

Booleans

We’re going to be asking yes/no questions

However, we’ll refer to True (yes) or False (no)

These are booleans (bools)!

Which of the following are true?

7 < 12

6 > 90

7 * 2 = 14

60 <= 200

Which of the following are true?

7 < 12 True

6 > 90 False

7 * 2 = 14 True

60 <= 200 True

Which of the following are true?

7 < 12 True

6 > 90 False

7 * 2 = 14 True

60 <= 200 True

Basic boolean expressions are just like what you’ve seen in math classes!

If statements

If statements allow us to start branching our code!

If statements

If statements allow us to start branching our code!

Syntax:

If statements

If statements allow us to start branching our code!

Syntax:

```
if expression:
   # do stuff if True!
   …
```

Example: warn user if input integer is negative

Example: warn user if input integer is negative

```
user_val = int(input())
```

Example: warn user if input integer is negative

```
user_val = int(input())
if ???:
   ???
```

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
    ???
```

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
```

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
print(user_val)
```

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
print(user_val)
```

What is printed when user_val is -5?

What is printed when user_val is 5?

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
print(user_val)
```

What is printed when user_val is -5? The warning, then -5

What is printed when user_val is 5?

Example: warn user if input integer is negative

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
print(user_val)
```

What is printed when user_val is -5? The warning, then -5

What is printed when user_val is 5? 5

What if we want one or the other?

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use <u>else</u>!

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use <u>else</u>!

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
print(user_val)
```

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use else!

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
else:
   print(“Confirmed, positive number!”)
print(user_val)
```

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use else!

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
else:
   print(“Confirmed, positive number!”)
print(user_val)
```

What is printed when user_val is -5?

What is printed when user_val is 5?

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use else!

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
else:
   print(“Confirmed, positive number!”)
print(user_val)
```

What is printed when user_val is -5? The warning, -5

What is printed when user_val is 5?

What if we want one or the other?

If (expresion) is true do A, *otherwise* do B -> use else!

```
user_val = int(input())
if user_val < 0:
   print(“Warning, value is negative!”)
else:
   print(“Confirmed, positive number!”)
print(user_val)
```

What is printed when user_val is -5? The warning, -5

What is printed when user_val is 5? The confirmation, 5

What if we want more than two options?

What if we want more than two options?

`elif`

What if we want more than two options?

`elif` - short for “else if”

What if we want more than two options?

`elif` - short for “else if”

```
if expression_A:
   # Code that only runs if A is True
elif expression_B:
   # Code that only runs if A is False and B is True
else:
   # Code that only runs if A and B are both False
```

What if we want more than two options?

`elif` - short for “else if”

Only one option will execute!

```
if expression_A:
    # Code that only runs if A is True
elif expression_B:
    # Code that only runs if A is False and B is True
else:
    # Code that only runs if A and B are both False
```

Work through some examples of what this code is doing

```
temp = float(input())
if temp < -460:
   print(“Impossible!”)
elif temp < 32:
   print(“Solid”)
elif temp < 212:
   print(“Liquid”)
else:
   print(“Gas”)
```

Comparison operators

There are many ways to compare two values!
All these expressions will evaluate to True or False

For numbers:

- a < b : Less than
- a <= b : Less than or equal to
- a > b: Greater than
- a >= b: Greater than or equal to
- a == b: Equal to
- a != b: Not equal to

A note on floats

Floats are weird

Try not to ask if a float == another float

Either use `math.isclose()`

Or use some epsilon

```
if abs(x - 0.3) < 0.001:
    # Do stuff
```

Comparison operators

For strings:

- a == b: Equal to
- a != b: Not equal to
- a < b, a > b, a <= b, a >= b: Comparing lexicographically
  - a < b implies a would come before b in the dictionary
    - (extended for numbers and symbols)
- a in b: True if a is a substring of b
  - E.g., “the” in “get these out of here” evaluates to true

Boolean operations

Imagine we want to bring an umbrella if

it’s cloudy AND there’s a >25% chance of rain

Boolean operations

Imagine we want to bring an umbrella if

    it’s cloudy AND there’s a >25% chance of rain

We often do this in code. We want to *combine* two boolean statements

Boolean operations

Imagine we want to bring an umbrella if

    it’s cloudy AND there’s a >25% chance of rain

We often do this in code. We want to *combine* two boolean statements

A and B is only true if both A and B are true!

Boolean operations

Imagine we want to bring an umbrella if

it’s cloudy AND there’s a >25% chance of rain

We often do this in code. We want to *combine* two boolean statements

A and B is only true if both A and B are true!

```
if is_cloudy == True and rain_chance > 0.25:
    bring_umbrella()
```

Boolean operations

A and B is True if **both** are True!

Value of `A`	Value of `B`	Value of `A and B`
False	False	

Boolean operations

A and B is True if **both** are True!

Value of `A`	Value of `B`	Value of `A and B`
False	False	False
False	True	

Boolean operations

A and B is True if **both** are True!

Value of `A`	Value of `B`	Value of `A and B`
False	False	False
False	True	False
True	False	

Boolean operations

A and B is True if **both** are True!

Value of `A`	Value of `B`	Value of `A and B`
False	False	False
False	True	False
True	False	False
True	True	

Boolean operations

A and B is True if **both** are True!

Value of `A`	Value of `B`	Value of `A and B`
False	False	False
False	True	False
True	False	False
True	True	True

Boolean operations

A <u>or</u> B is True if **either** is True!

Value of `A`	Value of `B`	Value of `A or B`
False	False	

Boolean operations

A <u>or</u> B is True if **either** is True!

Value of `A`	Value of `B`	Value of `A or B`
False	False	False
False	True	

Boolean operations

A or B is True if **either** is True!

Value of `A`	Value of `B`	Value of `A or B`
False	False	False
False	True	True
True	False	

Boolean operations

A or B is True if **either** is True!

Value of `A`	Value of `B`	Value of `A or B`
False	False	False
False	True	True
True	False	True
True	True	

Boolean operations

A or B is True if **either** is True!

Value of `A`	Value of `B`	Value of `A or B`
False	False	False
False	True	True
True	False	True
True	True	True

Boolean operations

not A just flips the boolean!

Value of `A`	Value of `not A`
False	True
True	False

Order of operations

not comes before and, and comes before or

Order of operations

not comes before and, and comes before or

When in doubt, just use parentheses!

Order of operations

not comes before and, and comes before or

When in doubt, just use parentheses!

```
class_is_fine = (prof_is_cool and class_is_hard) or class_is_easy
```

Order of operations

not comes before and, and comes before or

When in doubt, just use parentheses!

```
class_is_fine = (prof_is_cool and class_is_hard) or class_is_easy
```

All comparisons (==, <, in, etc.) come before not/and/or

https://docs.python.org/3/reference/expressions.html