As you walk in

1) Introduce your self to people around you

2) Log into piazza.com (we'll use it for polls in class)
   - Any device is fine
   - On a phone, the browser tends to work better than the app for polls

piazza

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

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

CMU 15-112, Fall 2023

Fundamentals of Programming and Computer Science
Carnegie Mellon University

W
15-112 S23 Term Project Lightning Rou...
Watch later
Share
15-112 SPRING23
TERM PROJECT LIGHTNING ROUND VIDEO
Watch on YouTube

Click here for the Term Project Gallery!

15-112
Lecture 2

Basic Programming Constructs

Instructor: Pat Virtue

Tuesday Logistics

[Practice] Poll 1

Are you new to CMU?

Course Team

https://www.cs.cmu.edu/~112/staff.html

Instructors

Mike Taylor
mdtaylor

Pat Virtue
pvirtue

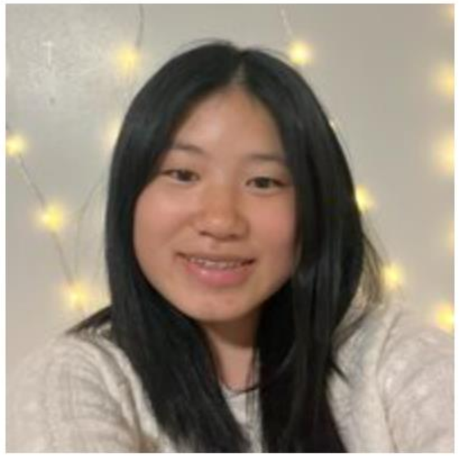
Head Teaching Assistants

Emily
esands

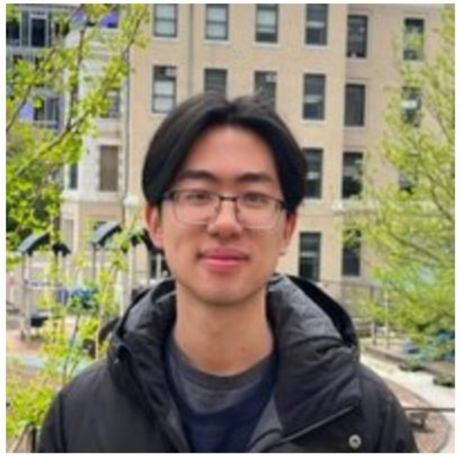
Liv
oduvanic

Lynn
lckim

Teaching Assistants


Andrea
arwang

Andrew
ayoun2


Andrew
acyu

Anna
annashi

Ariel
ychiu3

Arohee
abhoja

Audrey
ahasson

Avi
aarya2

Teaching Assistants

Brontosaurus

Christina
ctavlara

Daphne
daphneh

Emily
ealiu

Emily
emilyjia

Ethan
ethankwo

Gleb
gryabtse

Hugo
hsmartin

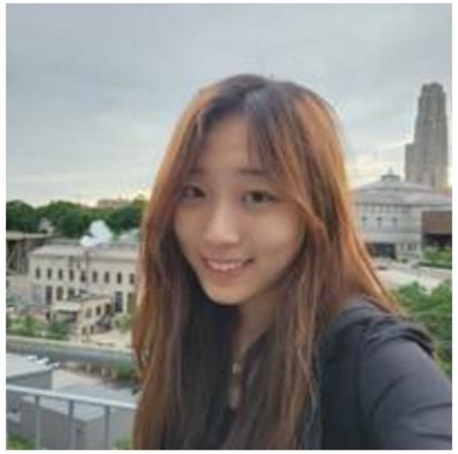
Teaching Assistants

Isaac
isaackap

James
changyaw

Jason
jstentz

Jerry
zhuoranh

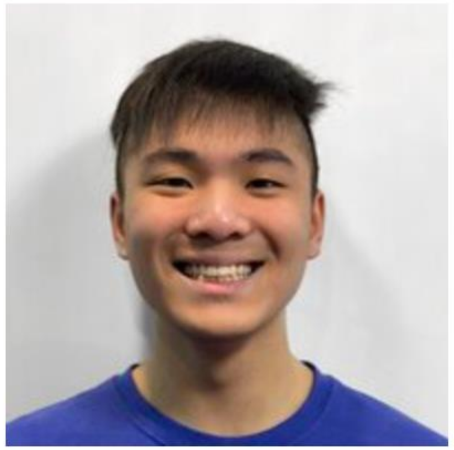

Jieun
jieunlim

Jose
jcestero

Kat
kstudent

Kayla
klei

Teaching Assistants


Kyle
kylechen

Lakshmi
ladiga

Lauren
lsands

Maddie
mrburrou

Maerah
maerahm

Marcus
malenius

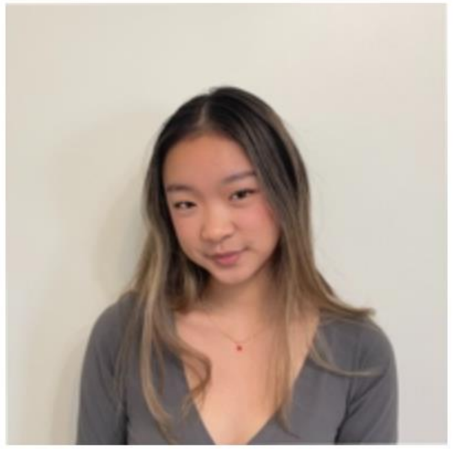
Margaret
mche

Meroushka
mrosner

Teaching Assistants

Mia
shengzhk

Monica
qimow


Orelia
opi

Peter
pkhoudar

Prina
phdoshi

Pterodactyl

Rhea
rsoo

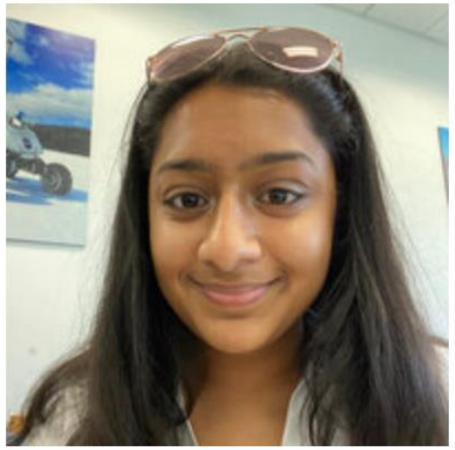
Riley
rkrzywda

Teaching Assistants

Rong
rongyuan

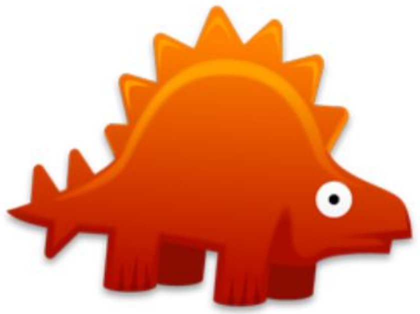
Sam
samuelch

Shawn
sihyunl


Shruti
shrutisr

Sonya
skarnata

Sophia
sophiaho


Stegosaurus

Suanna
suannaz

Teaching Assistants

T-rex

Tiger
rhuo

Timothy
tcarullo

Wen Hui
wleng

Course Team

Course administrative assistant

Marcie!

Course Team

Students!

[Practice] Poll 2

What college are you in?

A) BXA
B) CFA
C) CIT
D) DC
E) MCS
F) SCS
G) TSB
H) MIS/CMU/Other

[Practice] Poll 2 Take 2

What college is a person sitting next to you in?

A) BXA

B) CFA

C) CIT

D) DC

E) MCS

F) SCS

G) TSB

H) Other

Course Team

Students!

Course Information

Website: https://www.cs.cmu.edu/~112

Communication: https://piazza.com

If piazza doesn't work:

E-mail: pvirtue@andrew.cmu.edu, mdtaylor@andrew.cmu.edu

Announcements

Recitation

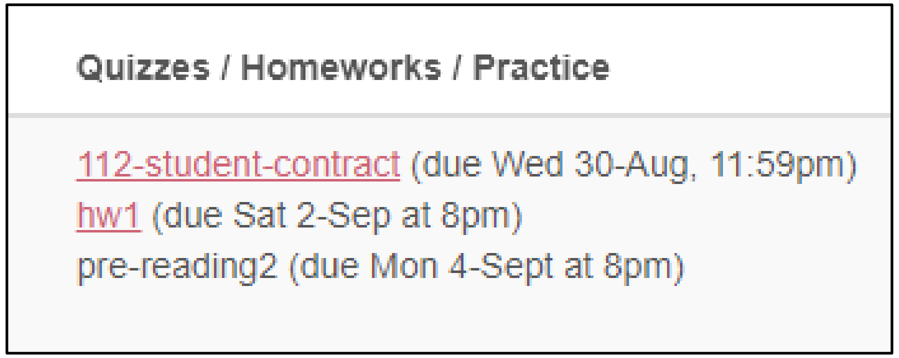
Wednesday & Friday

- Both days required
- Attend your assigned section
- Friday: GHC 5th Floor Clusters

Announcements

Assignments:

https://www.cs.cmu.edu/~112/schedule.html



Quizzes / Homeworks / Practice

112-student-contract (due Wed 30-Aug, 11:59pm)
hw1 (due Sat 2-Sep at 8pm)
pre-reading2 (due Mon 4-Sept at 8pm)

112 student contract

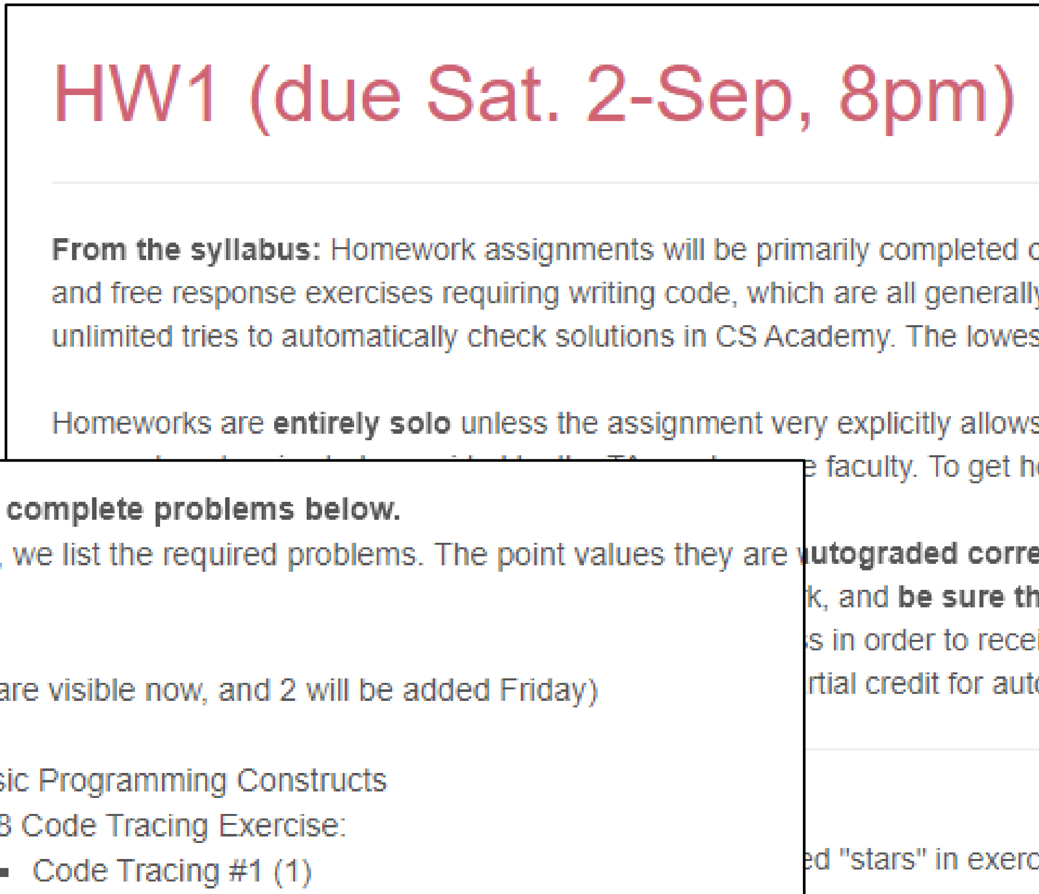
- Due Tomorrow 8/30, 11:59 pm

HW1

- Due Saturday 9/2, 8 pm

Week 2 Pre-reading Checkpoint

- Released by Thursday
- Due Mon 9/3, 8 pm



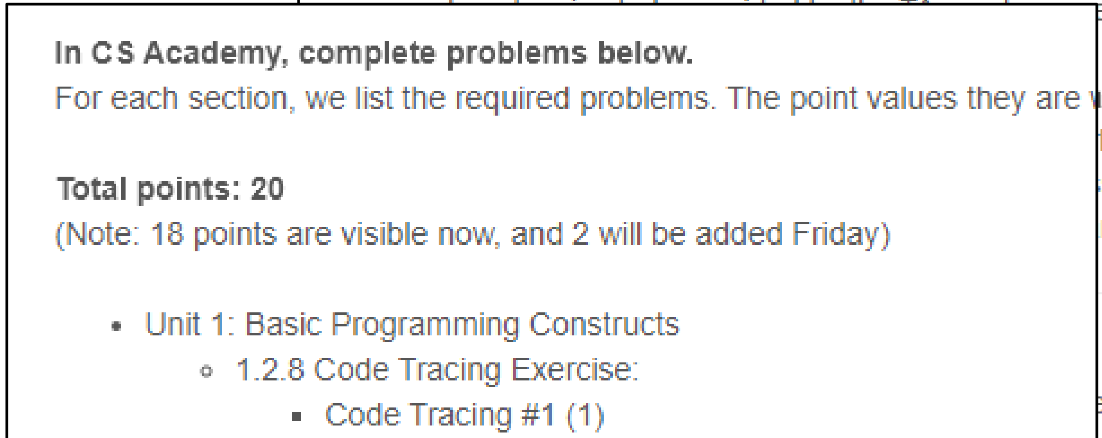
HW1 (due Sat. 2-Sep, 8pm)

From the syllabus: Homework assignments will be primarily completed o
and free response exercises requiring writing code, which are all generally
unlimited tries to automatically check solutions in CS Academy. The lowes

Homeworks are **entirely solo** unless the assignment very explicitly allows
e faculty. To get h

utograded corre
k, and **be sure th**
s in order to recei
rtial credit for auto

ed "stars" in exerc



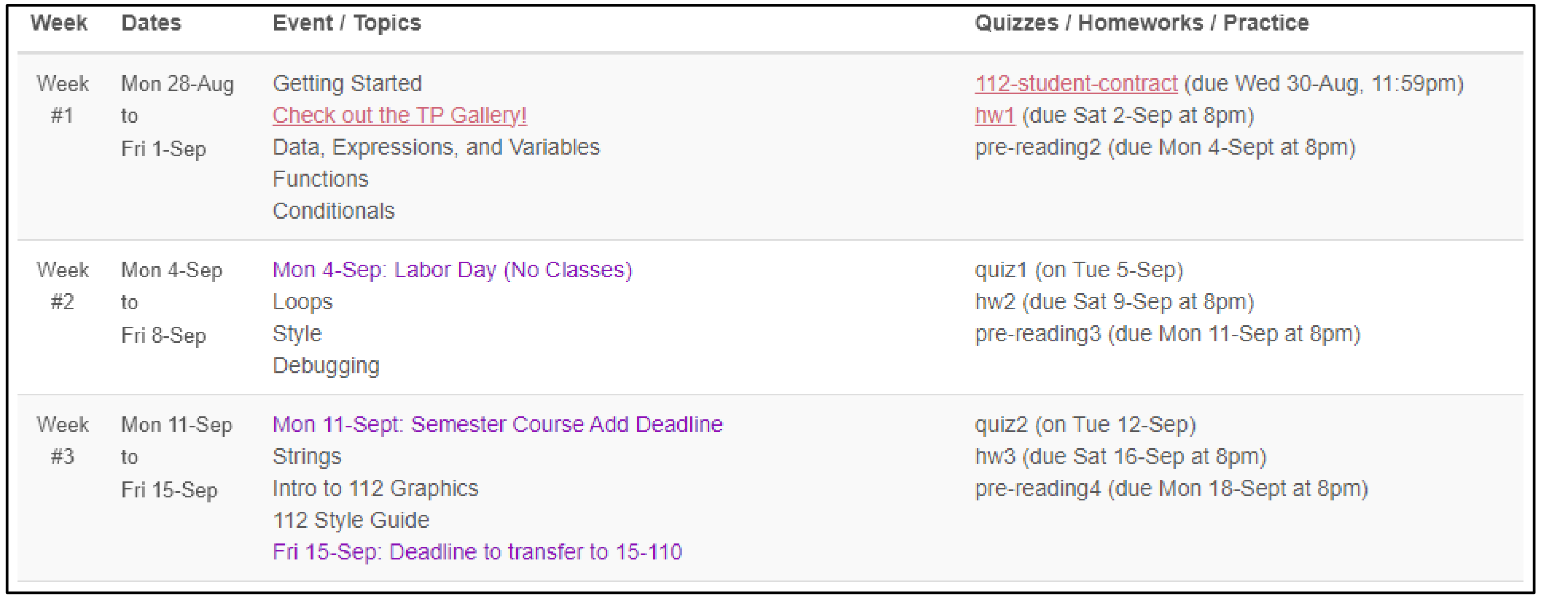
In CS Academy, complete problems below.
For each section, we list the required problems. The point values they are

Total points: 20
(Note: 18 points are visible now, and 2 will be added Friday)

- Unit 1: Basic Programming Constructs
  - 1.2.8 Code Tracing Exercise:
    - Code Tracing #1 (1)

https://www.cs.cmu.edu/~112/schedule.html

Weekly Rhythm

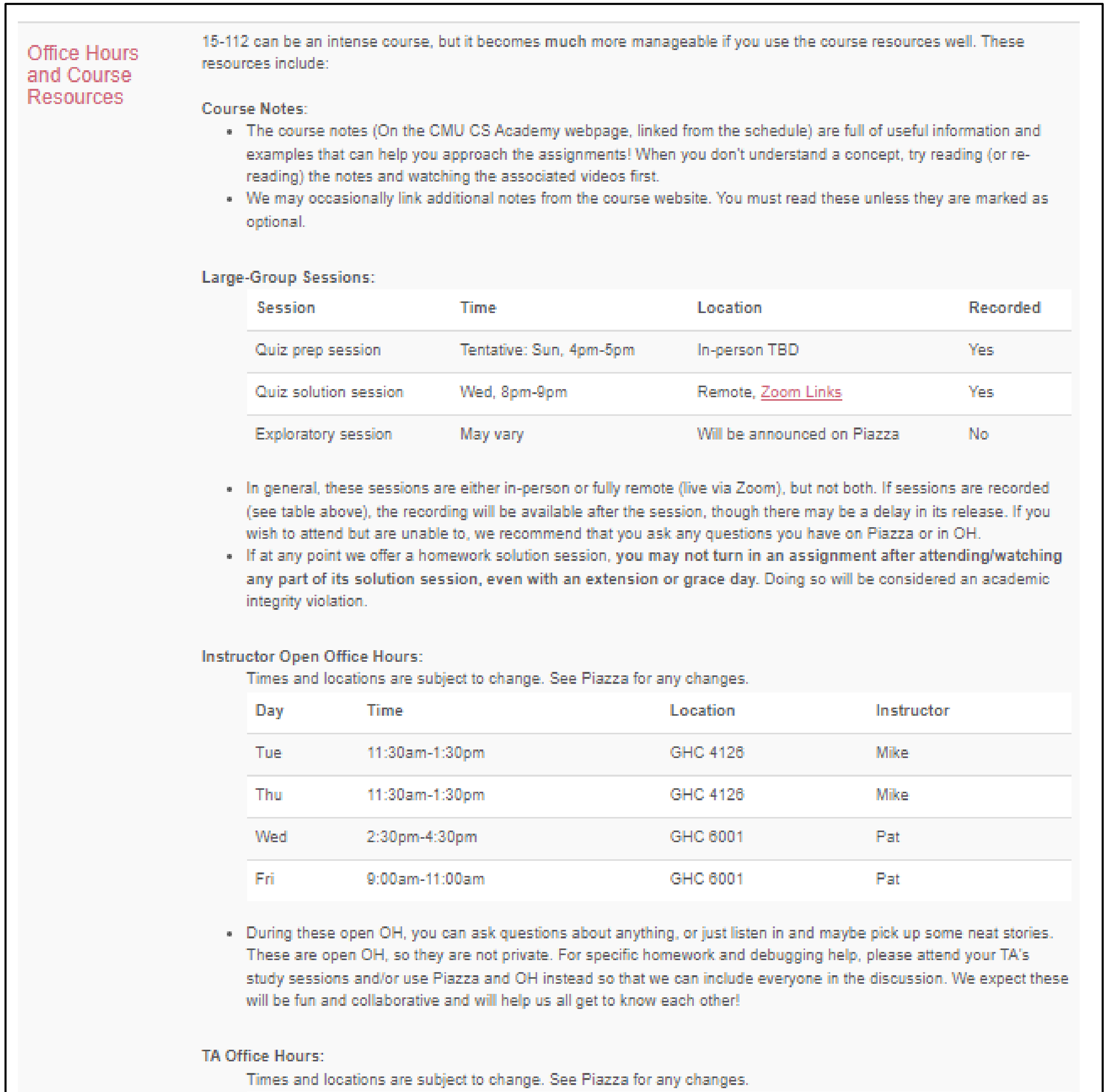


Week	Dates	Event / Topics	Quizzes / Homeworks / Practice
Week #1	Mon 28-Aug to Fri 1-Sep	Getting Started Check out the TP Gallery! Data, Expressions, and Variables Functions Conditionals	112-student-contract (due Wed 30-Aug, 11:59pm) hw1 (due Sat 2-Sep at 8pm) pre-reading2 (due Mon 4-Sept at 8pm)
Week #2	Mon 4-Sep to Fri 8-Sep	Mon 4-Sep: Labor Day (No Classes) Loops Style Debugging	quiz1 (on Tue 5-Sep) hw2 (due Sat 9-Sep at 8pm) pre-reading3 (due Mon 11-Sep at 8pm)
Week #3	Mon 11-Sep to Fri 15-Sep	Mon 11-Sept: Semester Course Add Deadline Strings Intro to 112 Graphics 112 Style Guide Fri 15-Sep: Deadline to transfer to 15-110	quiz2 (on Tue 12-Sep) hw3 (due Sat 16-Sep at 8pm) pre-reading4 (due Mon 18-Sept at 8pm)

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Week 1							
Week 2							
Week 3							

Course Resources

Use 112 resources wisely!



Office Hours and Course Resources

15-112 can be an intense course, but it becomes **much** more manageable if you use the course resources well. These resources include:

Course Notes:

- The course notes (On the CMU CS Academy webpage, linked from the schedule) are full of useful information and examples that can help you approach the assignments! When you don't understand a concept, try reading (or re-reading) the notes and watching the associated videos first.
- We may occasionally link additional notes from the course website. You must read these unless they are marked as optional.

Large-Group Sessions:

Session	Time	Location	Recorded
Quiz prep session	Tentative: Sun, 4pm-5pm	In-person TBD	Yes
Quiz solution session	Wed, 8pm-9pm	Remote, Zoom Links	Yes
Exploratory session	May vary	Will be announced on Piazza	No

- In general, these sessions are either in-person or fully remote (live via Zoom), but not both. If sessions are recorded (see table above), the recording will be available after the session, though there may be a delay in its release. If you wish to attend but are unable to, we recommend that you ask any questions you have on Piazza or in OH.
- If at any point we offer a homework solution session, **you may not turn in an assignment after attending/watching any part of its solution session, even with an extension or grace day.** Doing so will be considered an academic integrity violation.

Instructor Open Office Hours:

Times and locations are subject to change. See Piazza for any changes.

Day	Time	Location	Instructor
Tue	11:30am-1:30pm	GHC 4126	Mike
Thu	11:30am-1:30pm	GHC 4126	Mike
Wed	2:30pm-4:30pm	GHC 6001	Pat
Fri	9:00am-11:00am	GHC 6001	Pat

- During these open OH, you can ask questions about anything, or just listen in and maybe pick up some neat stories. These are open OH, so they are not private. For specific homework and debugging help, please attend your TA's study sessions and/or use Piazza and OH instead so that we can include everyone in the discussion. We expect these will be fun and collaborative and will help us all get to know each other!

TA Office Hours:

Times and locations are subject to change. See Piazza for any changes.

Lecture Logistics

Polls

- One participation point for *each* take
- Correctness of answer doesn’t count
- Profs really do use this as realtime feedback on your understanding
- Don’t stress
- Tech issues
  - One-time issue: no problem, you just need >= 80%
  - Persistent issue: let us know so we can find a solution
- Used for educational technique call Peer Instruction (more on this later

Lecture Logistics

Notes

CS Academy notes

- Required reading (and viewing)

Pat’s Slides

- Additional resource. Helpful for lecture notetaking and review
- Preview version posted before lecture (on website Schedule)
- Inked versions posted later (on website Schedule)

Taking notes

Devices in lecture

Thursday Logistics

Thursday Announcements

Recitation

Friday

- Required
- GHC 5th Floor Clusters (see link to GHC 5 video on syllabus)

Thursday Announcements

Assignments:

https://www.cs.cmu.edu/~112/schedule.html

112 student contract

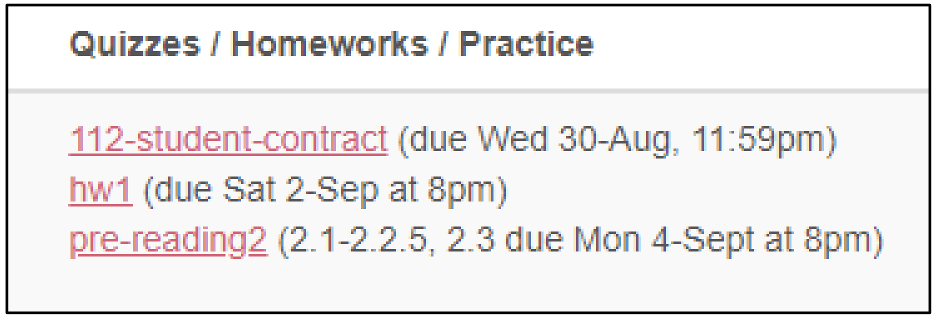
- Due YESTERDAY 8/30, 11:59 pm

HW1

- Due Saturday 9/2, 8 pm

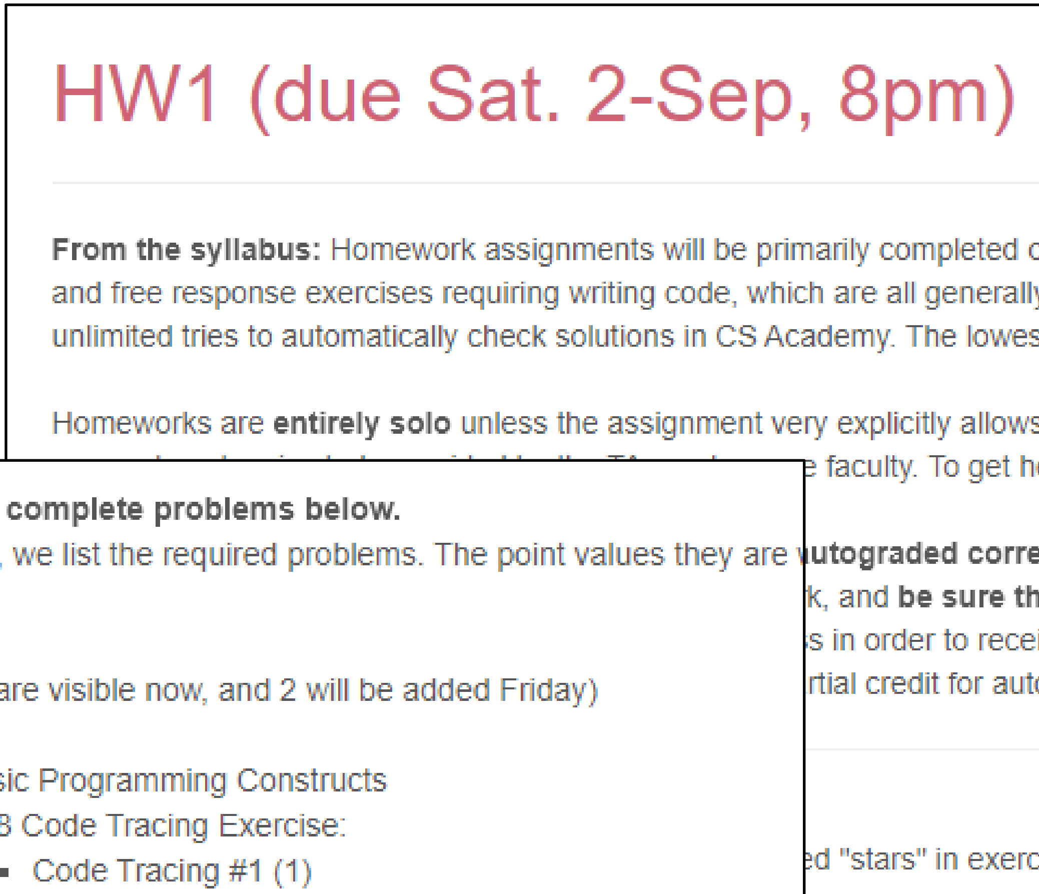
Week 2 Pre-reading Checkpoint

- Released by Thursday
- Due Mon 9/3, 8 pm



Quizzes / Homeworks / Practice

112-student-contract (due Wed 30-Aug, 11:59pm)
hw1 (due Sat 2-Sep at 8pm)
pre-reading2 (2.1-2.2.5, 2.3 due Mon 4-Sept at 8pm)

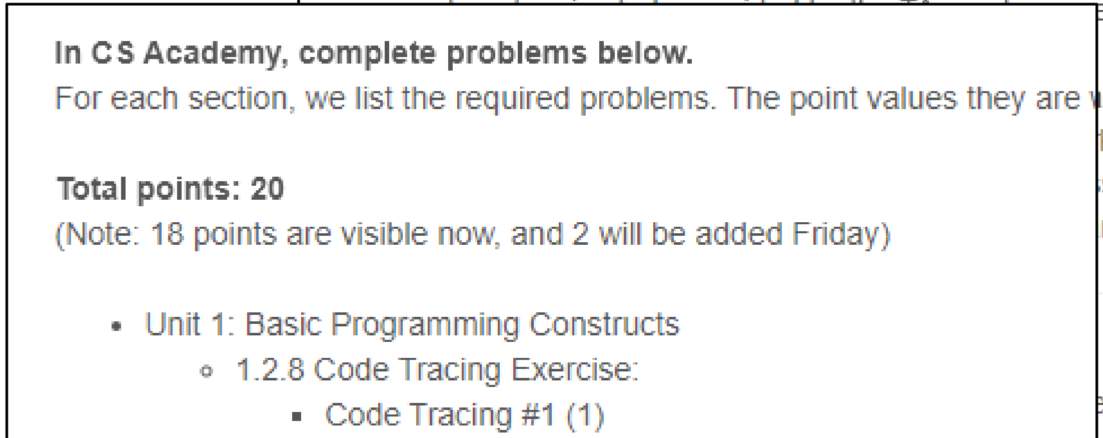


HW1 (due Sat. 2-Sep, 8pm)

From the syllabus: Homework assignments will be primarily completed c
and free response exercises requiring writing code, which are all generally
unlimited tries to automatically check solutions in CS Academy. The lowes

Homeworks are **entirely solo** unless the assignment very explicitly allows
e faculty. To get h
utograded corre
k, and **be sure th**
s in order to recei
rtial credit for aut

ed "stars" in exerc



In CS Academy, complete problems below.
For each section, we list the required problems. The point values they are

Total points: 20
(Note: 18 points are visible now, and 2 will be added Friday)

- Unit 1: Basic Programming Constructs
    - 1.2.8 Code Tracing Exercise:
        - Code Tracing #1 (1)

Weekly Rhythm

https://www.cs.cmu.edu/~112/schedule.html
https://www.cs.cmu.edu/~112/syllabus.html

Support (see syllabus and watch Piazza)

- OH
- Practice Quiz
- Quiz Prep Session

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Week 1			Lec	Rec **Contract**	Lec	Rec	**HW due**
Week 2	Quiz prep	**Pre-reading**	Lec **Quiz in Lec**	Rec	Lec	Rec	**HW due**
Week 3	Quiz prep	**Pre-reading**	Lec **Quiz in Lec**	Rec	Lec	Rec	**HW due**

Lecture Logistics

Polls

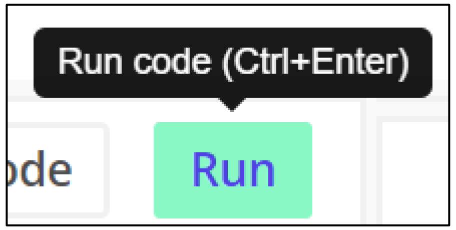
- Polls this week don’t count. Just practicing Piazza.
- Don’t stress
- Tech issues
  - One-time issue: no problem, you just need >= 80%
  - Persistent issue: let us know so we can find a solution

Tips!

Tips for editing code

Run code without clicking Run button

- Ctrl/Cmd + Enter



Comment or uncomment block of code

1. Select multiple lines together
2. Ctrl/Cmd + /

Indent or unindent block of code

1. Select multiple lines together
2. Indent: Ctrl/Cmd + Tab

   Unindent: Ctrl/Cmd + Shift + Tab

Getting Started with Python

Hello World!

Classic start to new tech

```
print("Hello World!")
```

But where can we run this?

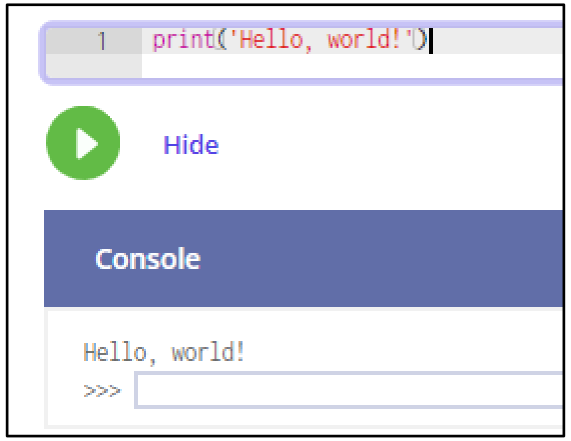
Running Python

CS Academy

- Edit code boxes in notes
- Exercises
- Sandbox!

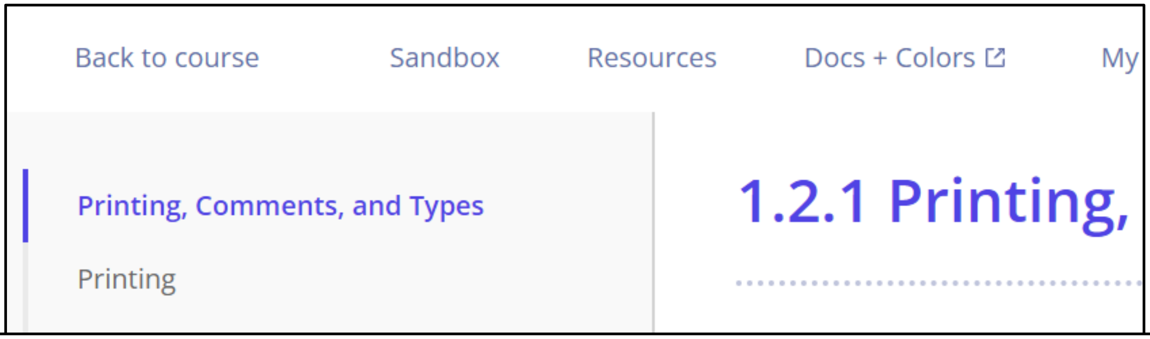
Python file /editor

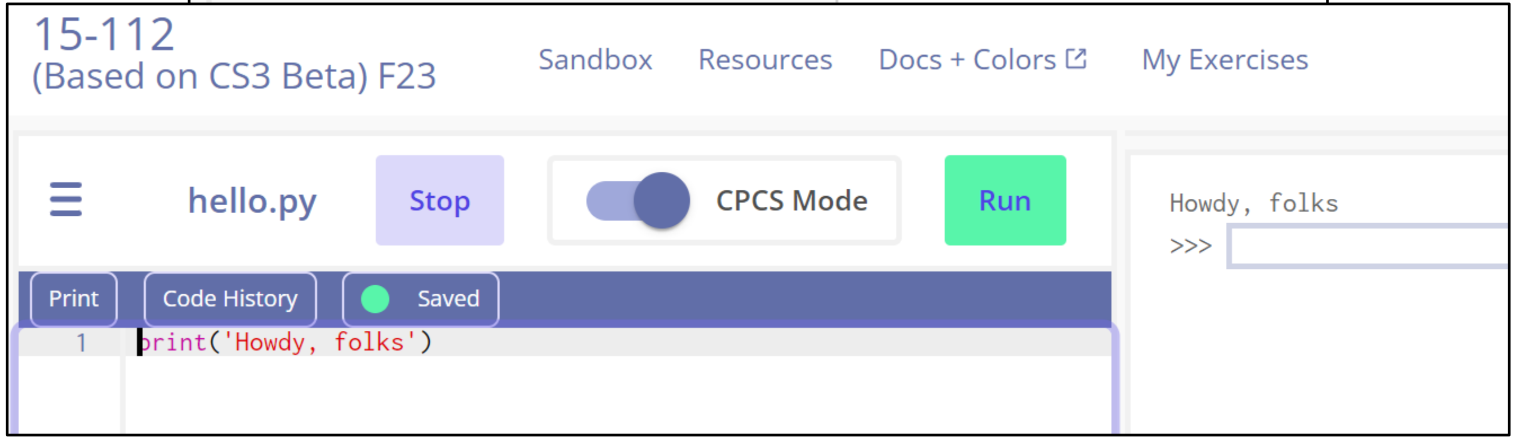
Python interpreter



```
print('Hello, world!')
```

```
Hello, world!
>>>
```




15-112
(Based on CS3 Beta) F23

```
print('Howdy, folks')
```

```
Howdy, folks
>>>
```

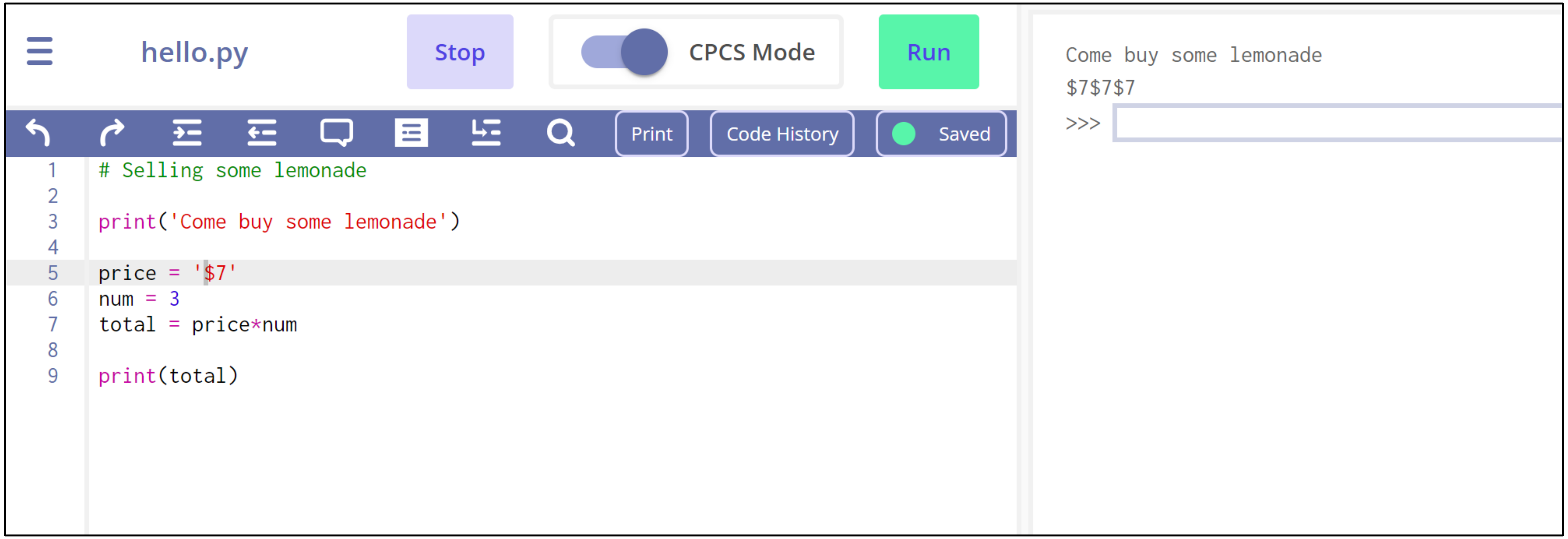
Python files/editor vs Python interpreter

Python files and editor

Write and save code

Python interpreter

Quickly test code and explore



```python
# Selling some lemonade

print('Come buy some lemonade')

price = '$7'
num = 3
total = price*num

print(total)
```

```
Come buy some lemonade
$7$7$7
>>>
```

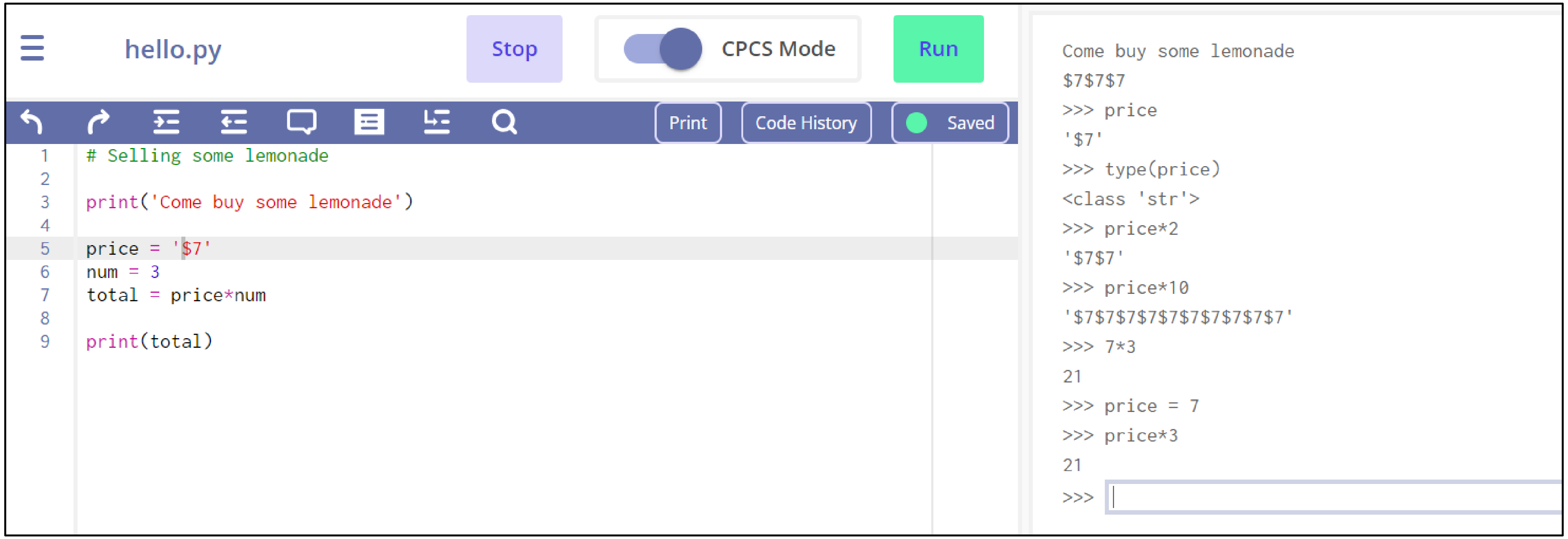
Python files/editor vs Python interpreter

Python files and editor

- Write and save code
- Need to explicitly run code

Python interpreter

- Runs each line when you hit enter
- Auto prints resulting object
- Quickly test code and explore



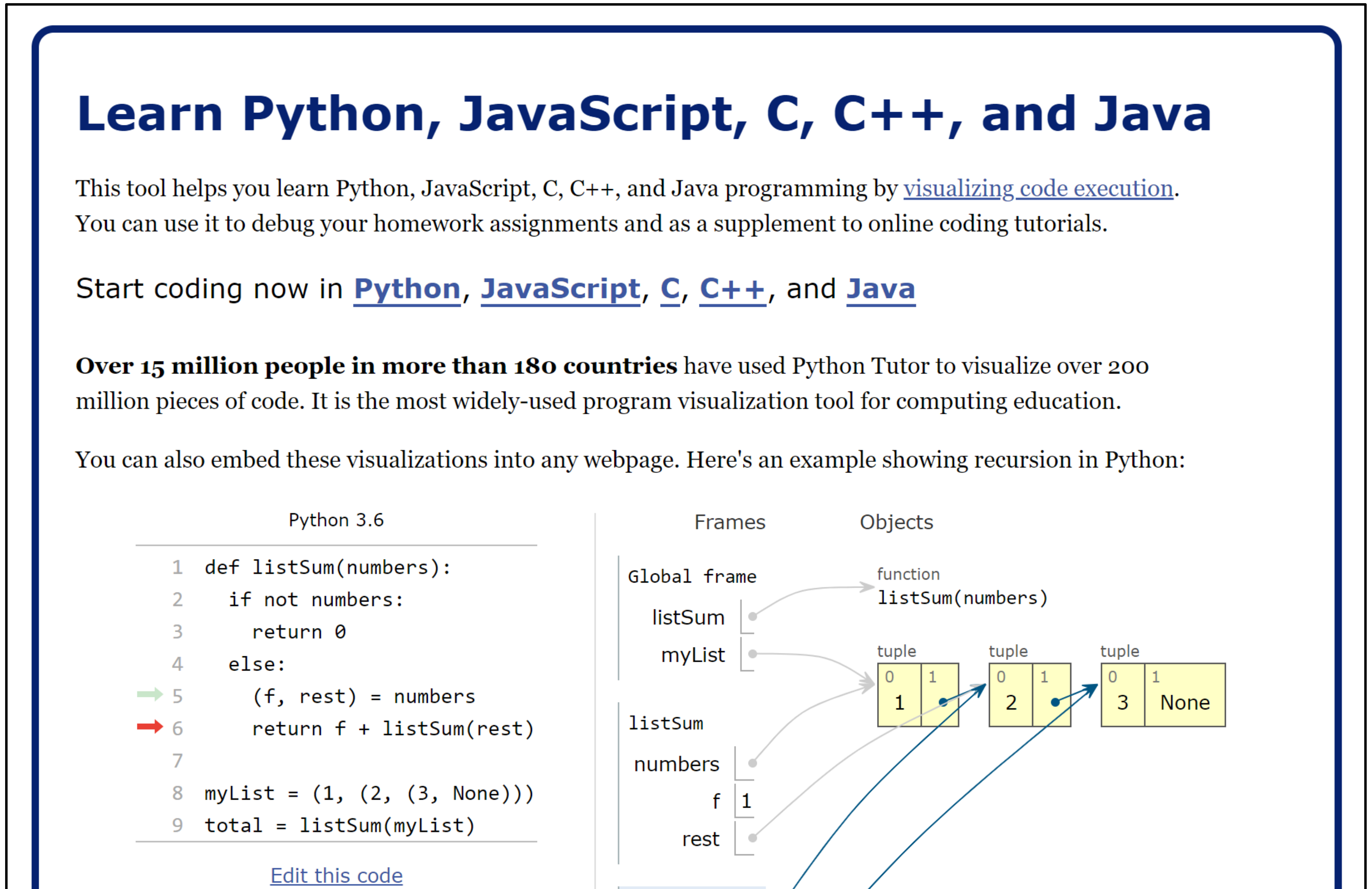
```
# Selling some lemonade

print('Come buy some lemonade')

price = '$7'
num = 3
total = price*num

print(total)
```

```
Come buy some lemonade
$7$7$7
>>> price
'$7'
>>> type(price)
<class 'str'>
>>> price*2
'$7$7'
>>> price*10
'$7$7$7$7$7$7$7$7$7$7'
>>> 7*3
21
>>> price = 7
>>> price*3
21
>>>
```

Running Python

Pythontutor

- Help *see* how Python works



Learn Python, JavaScript, C, C++, and Java

This tool helps you learn Python, JavaScript, C, C++, and Java programming by visualizing code execution. You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in **Python**, **JavaScript**, **C**, **C++**, and **Java**

Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

Python 3.6

```
def listSum(numbers):
  if not numbers:
    return 0
  else:
    (f, rest) = numbers
    return f + listSum(rest)

myList = (1, (2, (3, None)))
total = listSum(myList)
```

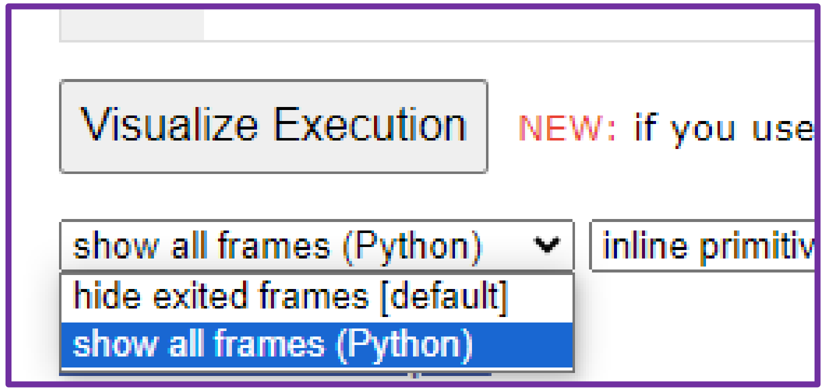
Edit this code

Running Python

Pythontutor

- Help *see* how Python works
- Helpful to learn how to write out work for code tracing

Recommended setting (bottom-left)



Visualize Execution

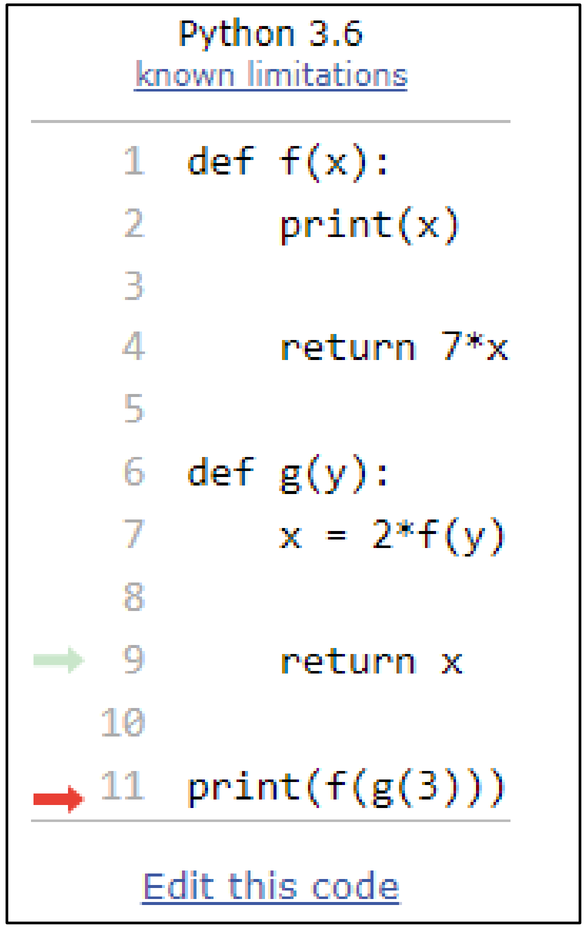
NEW: if you use

show all frames (Python)

inline primitiv

hide exited frames [default]

show all frames (Python)

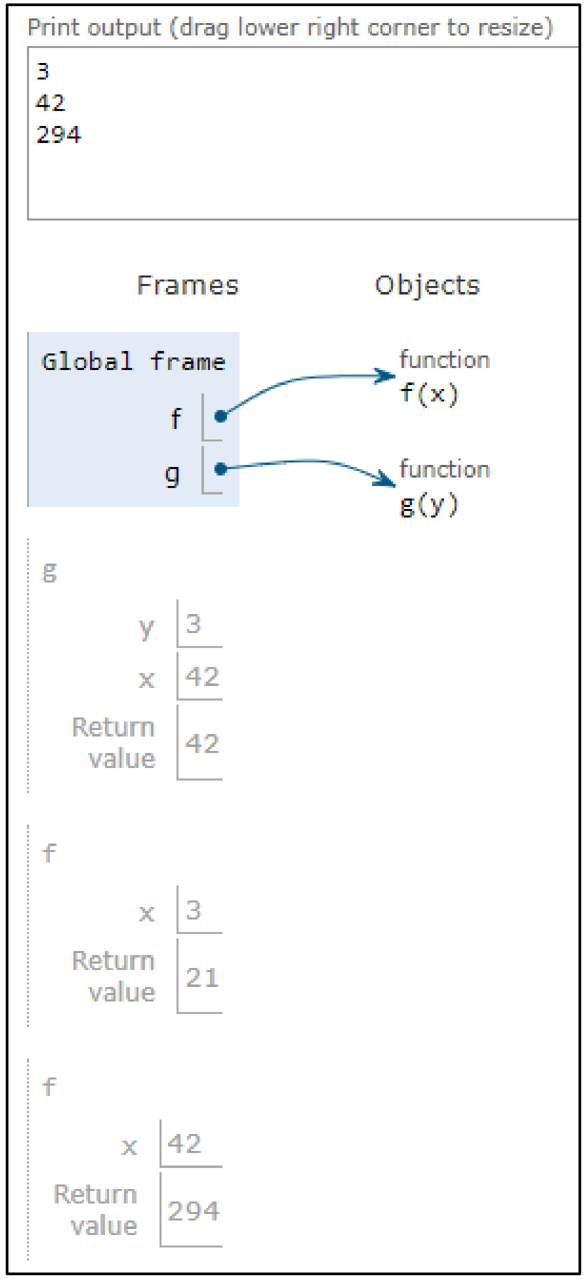


Python 3.6

known limitations

```
def f(x):
    print(x)

    return 7*x

def g(y):
    x = 2*f(y)

    return x

print(f(g(3)))
```

Edit this code



Print output (drag lower right corner to resize)

```
3
42
294
```

Frames / Objects

Global frame: f → function f(x); g → function g(y)

g	
y	3
x	42
Return value	42

f	
x	3
Return value	21

f	
x	42
Return value	294

Running Python

(more details later in course)

Terminal (a.k.a. command line) → `python` → Python interpreter

(Code) Editor `myFile.py` → Terminal: `python myFile.py`

IDE (Integrated development environment)

- Editor connected with terminal/interpreter
- VS Code (more details later in course)

Printing

Printing

We can print a few different types of things in Python:

- Text (which we call a "string")

```
print('Hello World!’)
Hello World!
```

- Numbers (which we'll separate into integers and floating point numbers)

```
print(123)
print(12.3)
```

- Expressions (which evaluate to a value before we print them)

```
print(12+3)
15
```

Printing Multiple Things

Call the print function with multiple arguments separated by commas

(An "argument" is a value that we pass to a function)

```
print('12+3:', 12+3)
12+3 = 15
```

This will print them separated by spaces (not commas)

```
print('Thing1', 'Thing2’)
Thing1 Thing2
```

Printing with f-strings (formatted strings)

By putting the letter f right before a string, you can then

place variable names in {squiggly braces} to print their values, like so:

```
x = 42
y = 99
print(f'Did you know that {x} + {y} is {x+y}?’)
Did you know that 42 + 99 is 141?
```

Since the introduction of f-strings in Python, this has become a popular way to print combinations of text and values.

The print function

`print` is a function. The `print` function will send text to the console output.

Like in math, Python functions return values, and we can assign those values to variables, e.g. `y = abs(-7)`

But, some functions, like print, just return the special Python value `None`

```
y = print('Hello World!’)
print(f'The value of y is {y}.')
Hello World!
The value of y is None.
```

Operators and expressions

Operators Summary

Arithmetic

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

Comparison

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

Assignment

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

Logical

- and, or, not

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

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

Arithmetic Operators

```python
print(6 + 2)
print(6 - 2)
print(6 * 2)
print(6 / 2)
```

```
8
4
12
3.0
>>>
```

Arithmetic operators

Operator	Example Python	Example Result
Addition	3+5	8
Subtraction	3-5	-2
Multiplication	3*5	15
Division	3/5	0.6
Power (Exponent)	3**5	243
Negation	-3	-3
Modulo “Mod” (remainder)	5 % 3	2
“Div” (integer division)	5 // 3	1

Expressions

Expression in Python are just segments of code that evaluate to a value (or more specifically an object)

For arithmetic expressions, we need to pay attention to the order of operations.

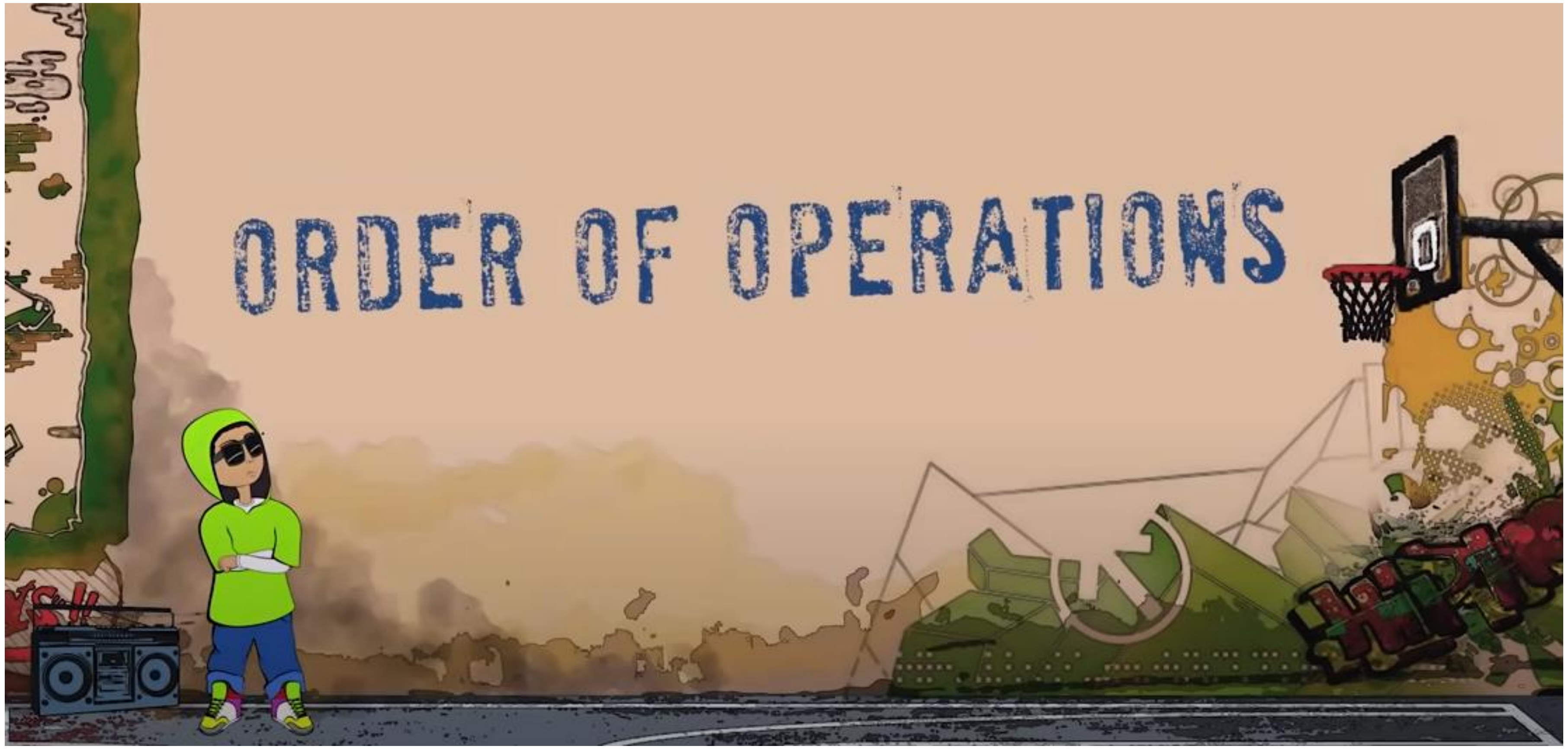
Paratheses

- Can change affect the order of operations, just like in math
- Can help clarify the order of operations, even when not necessary
- In general, don’t add unnecessary paraentheses unless for clarity

Order of operations

PEMDAS

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



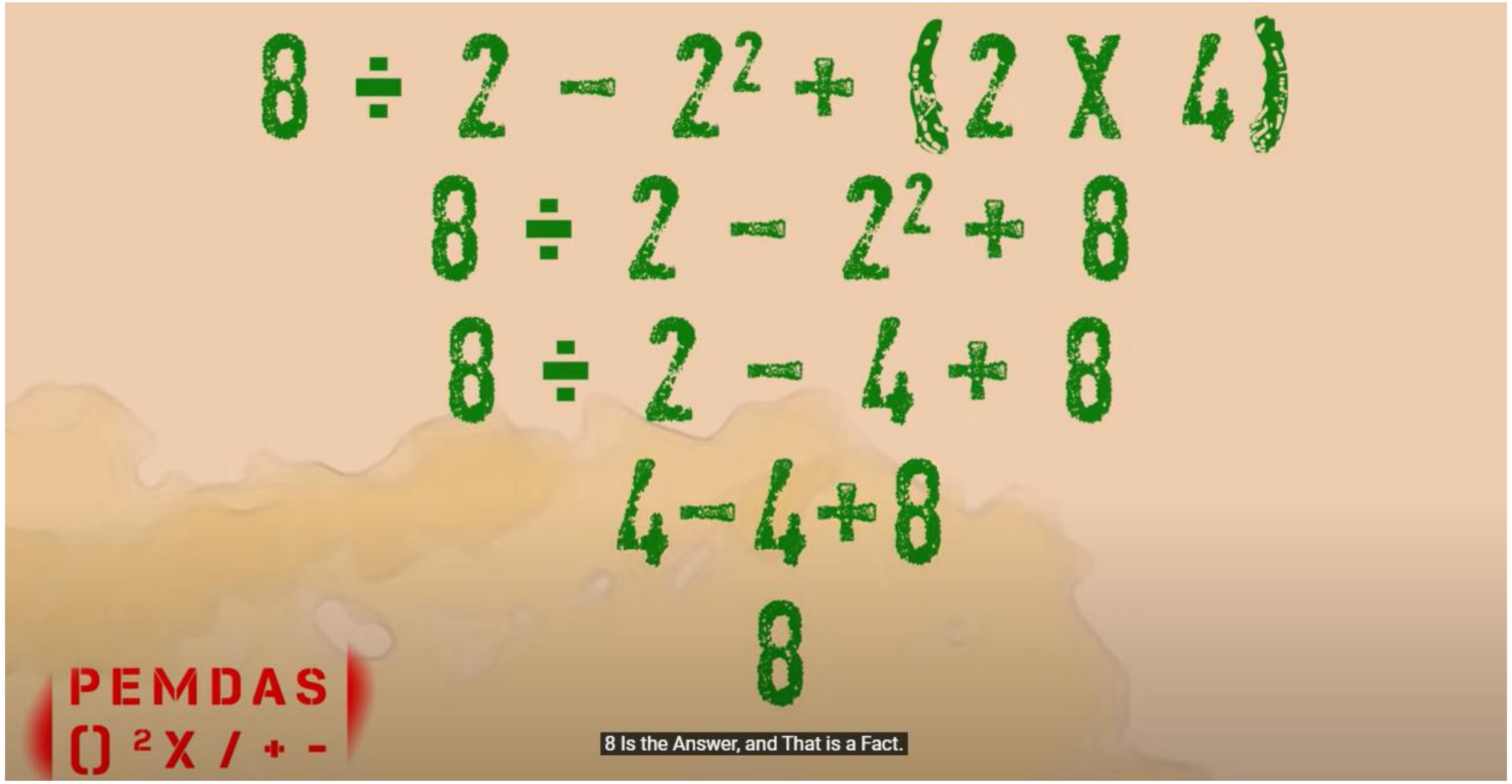
Order of operations

PEMDAS

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

Tip

Be a robot!



$$8 \div 2 - 2^2 + (2 \times 4)$$
$$8 \div 2 - 2^2 + 8$$
$$8 \div 2 - 4 + 8$$
$$4 - 4 + 8$$
$$8$$

Poll 3

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 4

How many expressions are there in:

```
a - a // b * b
```

A) 1

B) 2

C) 3

D) 4

E) 5

F) Other

G) I have no idea

Errors

Natural Language

Which is correct?

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 5 (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
```

Strings and Comments

Poll 6 (Unused)

Which one does the right thing?

Select all that apply

```
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?

Select all that apply

```
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?’)
```

```python
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?')
```

Strings

Single or double quote are fine

- Can be useful for quotes withing strings (but alternated correctly)
- Escape characters are needed sometimes (more on this later in the course)

```
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}?')
```

Comments Summary

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
```

Comments

```python
print("Hello World!") # This is a comment
# print("What will this line do?")
```

Comments are for humans

Comments are sections of text that we can write in Python (and most computer languages) that provide helpful information for humans to understand the associated code.

In Python, a # symbol (also called a "pound sign" or "hash symbol") begins a comment and tells Python to ignore all of the contents from the # sign until the end of the line.

Even though Python ignores the contents of a comment, comments are an essential part of writing clear code!

Comments

```
# Comment on its own line
x = 7 # Comment after code

# Multiline comments can be useful too
# when you have more to say
# or just want to make your comments easier to read
```

Comments

```
"""
Long comments can be written inside triple-quotes.
Either triple-single-quotes or triple-double-quotes
work.

These can save you from writing a # on every line.

(these long quotes are technically strings that are
just ignored by Python.)
"""
```

Poll 7 (Unused)

Which of the following will be printed?

Select all that apply

A. ONE
B. TWO
C. THREE
D. FOUR
E. FIVE
F. SIX
G. SEVEN
H. EIGHT
I. NINE

```
"""
print("ONE")
print("TWO")
"""
print("THREE")
print("FOUR")
# print("FIVE")
print("SIX") #
# print("SEVEN") #
# print("EIGHT") # print("NINE")
```

Poll 7 (Unused)

Which of the following will be printed?

Select all that apply

A. ONE

B. TWO

C. THREE

D. FOUR

E. FIVE

F. SIX

G. SEVEN

H. EIGHT

I. NINE

```
"""
print("ONE")
print("TWO")
"""
print("THREE")
print("FOUR")
# print("FIVE")
print("SIX") #
# print("SEVEN") #
# print("EIGHT") # print("NINE")
```

Variables

Poll 8

Which of the following will result in the variable `x` being `0.4`?

Select all that apply

A. x = 0.4
B. 0.4 = x
C. x = 2/5
D. 2/5 = x
E. 5x = 2
F. 2 = 5x
G. 5*x = 2
H. 2 = 5*x
I. None of the above

Variables Summary

```
x = 4
y = x**2
print(x)
print(y)

# Reassign x to 3
x = 3
print(x)
print(y)
# y is still 16 (not automatically y = 3**2)
# We would have to execute y = x**2 again for y to be 3**2
y = x**2
print(x)
print(y)
```

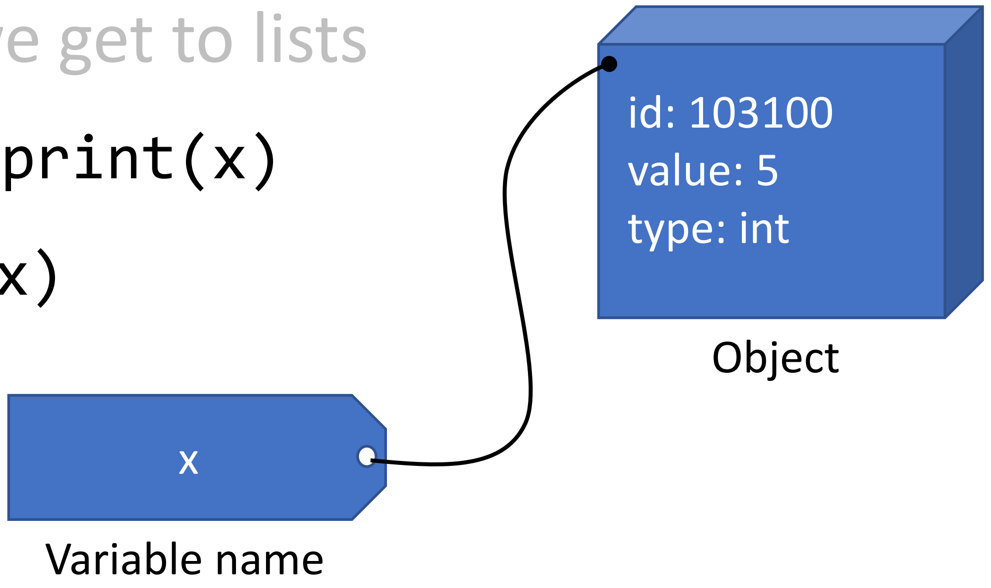
```
# The variable we are assigning has to
# be the ONLY thing on the left
# of the = sign

# 4 = x # Error
# 3*x = 4 # Error
x = 4/3
print(x)
```

Python Objects and Variable Naming

All of the “things” in Python are objects

Python objects all have:

- id More on object ids when we get to lists
- value We can try to see this with `print(x)`
- type We can see this with `type(x)`



Variable naming

Think of a variable name as a gift tag attached to an object

Python keeps track of variable names to allow us to use that object later

Variable Assignment

variable_name = *expression*

Variable name must be the ONLY thing on the LEFT of the =

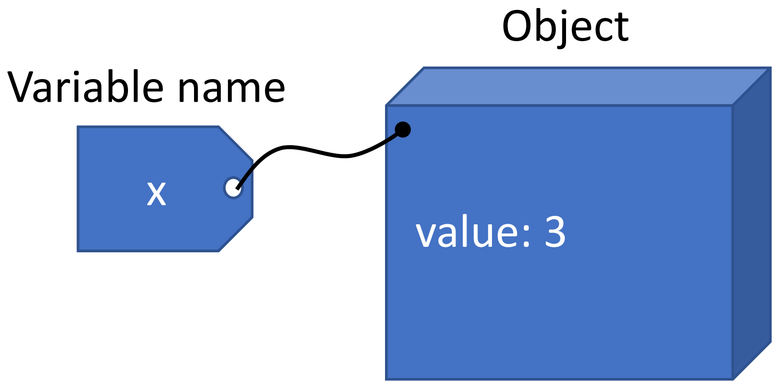
Everything to the RIGHT of the = will be evaluated before the name is assigned

Python code

```
x = 3
```

Python code

```
x = 3 + 2
```



Variable Reassignment

variable_name = *expression*

Python evaluates the right-hand-side to create a single object and then assigns the variable name tag to that object

Python code

```
x = 3
x = x + 2
```

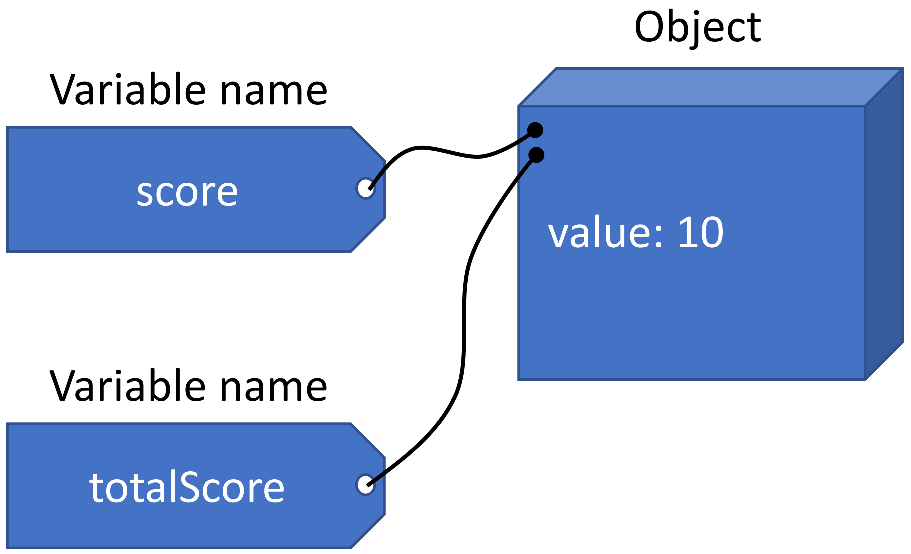
Assigning a Variable to a Variable

another_variable_name = *variable_name*

Multiple variables can point to the same object

For example, after running the following two lines, score and total_score will both be 10

```
score = 10
totalScore = score
```



Variables

Variable names often temporarily point to the same object and are later changed to point to something else

Python code

```
score = 10
total_score = score

score = 20
total_score = total_score + score
```

Poll 9 (Unused)

Which of the are valid variable names in Python

Select all that apply

A. val = 4
B. 4val = 4
C. val4 = 4
D. my4val = 4
E. four = 4
F. value? = 4
G. my value = 4
H. my_value = 4
I. my-value = 4
J. myValue = 4

Arithmetic assignment operators

Operator	Shortcut	Long(cut)
Addition	x += 5	x = x + 5
Subtraction	x -= 5	x = x - 5
Multiplication	x *= 5	x = x * 5
Division	x /= 5	x = x / 5
Power (Exponent)	x **= 5	x = x**5
Modulo “Mod” (remainder)	x %= 5	x = x % 5
“Div” (integer division)	x //= 5	x = x // 5

Functions

Functions

```
def function_name(parameter):
    body_including_return_statements

def myFunctionName(parameter1, parameter2, parameter3):
    # Do something here
    return 42

argument1 = 3
argument2 = 9
argument3 = 27
x = myFunctionName(argument1, argument2, argument3)
```

Poll 10 (Unused)

Which code is better?

A)

```
def distance(x1, y1, x2, y2):
    return ((x1-x2)**2 + (y1-y2)**2)**0.5
```

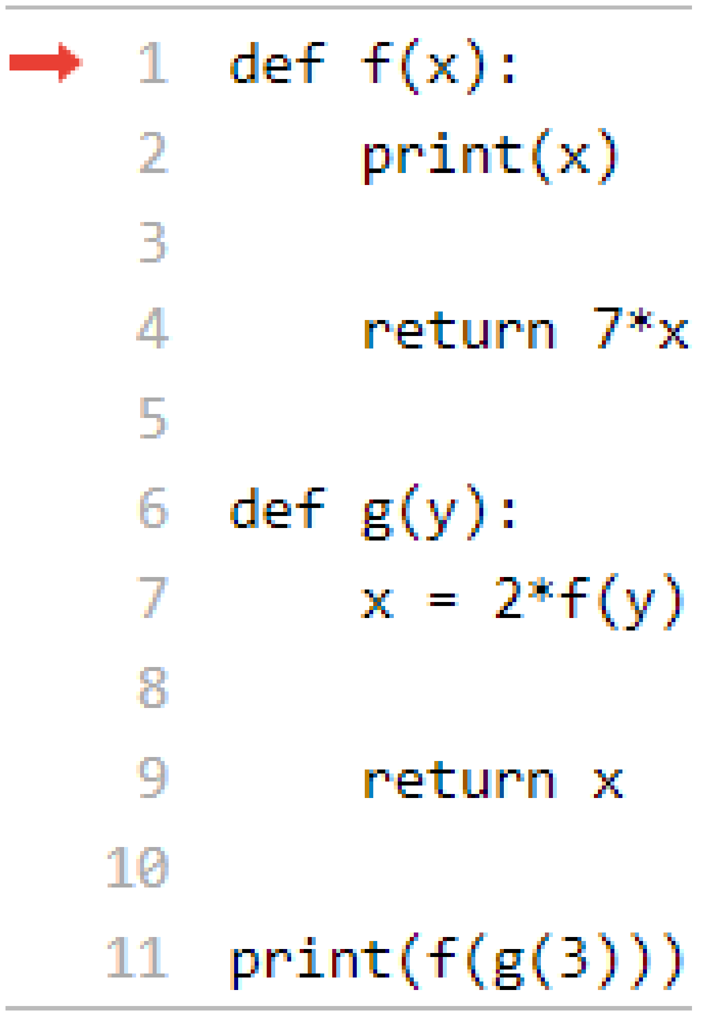
B)

```
def distance(x1, y1, x2, y2):
    xDiff = x1 - x2
    yDiff = y1 - y2

    xDiffSquared = xDiff**2
    yDiffSquared = yDiff**2

    sumOfSquares = xDiffSquared + yDiffSquared

    result = sumOfSquares**0.5
    return result
```

Poll 11 (Unused)

This code just started executing in pythontutor.com, to which will the red arrow move to next?

2.
3.
4.
5.
6.
7.
8.
9.
10.
11.



```
➡ 1  def f(x):
  2      print(x)
  3
  4      return 7*x
  5
  6  def g(y):
  7      x = 2*f(y)
  8
  9      return x
 10
 11  print(f(g(3)))
```

Types

Types

Types we are working with so far

`int, float, str, bool, NoneType (and type)`

Code

```
print(type(3))
print(type(3.0))
print(type("3"))
print(type(True))
print(type(None))
print(type(int))
```

Output

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'NoneType'>
<class 'type'>
```

Types

`type(`*`value`*`)` vs `isinstance(`*`value`*`, `*`type`*`)`

Code

```
s = 'abc'
print(type(s) == str)
print(isinstance(s, str))
print(isinstance(s, int))
```

Output

```
True
True
False
```

Types

Why do we care?

Types affect semantics

(i.e., depending on the type of objects involved, an expression may do different things)

```
x = 4
y = 3
z = x*y
print(f'{x}*{y} = {z}')
print(f'The type of z is {type(z)}.')
```

```
4*3 = 12
The type of z is <class 'int'>.
```

```
x = 4
y = '3'
z = x*y
print(f'{x}*{y} = {z}')
print(f'The type of z is {type(z)}.')
```

```
4*3 = 3333
The type of z is <class 'str'>.
```

```
x = 4.0
y = '3'
z = x*y
print(f'{x}*{y} = {z}')
print(f'The type of z is {type(z)}.')
```

```
Traceback (most recent call last):
  File "g:\My Drive\112\workspace\week1\lec\variables.py", line 41, in <module>
    z = x*y
TypeError: can't multiply sequence by non-int of type 'float'
```

Types Conversions

We can convert between types when necessary

```
n = int('12')
print(type(n))
print(5*n)
```

Output

```
<class 'int'>
60
```

Example with input() function

```
responseStr = input('How many pears to you want to buy? ')
responseInt = int(responseStr)

pricePerPear = 1.5
totalPrice = responseInt * pricePerPear

print(f'That will cost ${totalPrice}.')
```

Comparison operators

Operators Summary

Arithmetic

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

Comparison

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

Assignment

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

Logical

- and, or, not

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

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

Operators with Boolean values

Comparison

- <, <=, >=, >, ==, !=, is
- e.g., x <= y
- Results in Boolean value

Logical

- and, or, not
- Intended to compare two Boolean values (or negate one Boolean value in the case of not)

Poll 12 (Unused)

What will this print?

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

A. True
B. False
C. I don’t know

Issues with floats

Equality

```
x = 0.1 + 0.1 + 0.1
y = 0.3
x == y # Doesn’t work well with floats
```

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

Rounding

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

- Use cmu_cpcs_utils : `rounded(x)`

Poll 13

Which of these won’t crash (i.e., produce a DivByZeroError)?

Select all that apply

```
A. print(1/0)
B. print(True or 1/0)
C. print(True and 1/0)
D. print(1/0 or True)
E. print(1/0 and False)
F. print(False or 1/0)
G. print(False and 1/0)
H. None of the above
```

Conditionals

Conditional statements

```
if boolean_expression:
    body
```

```
if boolean_expression:
    bodyA
else:
    bodyB
```

```
if boolean_expressionA:
    bodyA
elif boolean_expressionB:
    bodyB
else:
    bodyC
```

```
if boolean_expressionA:
    bodyA
elif boolean_expressionB:
    bodyB
elif boolean_expressionC:
    bodyC
else:
    bodyD
```

Nested Conditional Statements

```
if boolean_expression:
    if boolean_expression:
        bodyA
    else:
        bodyB
```

```
if boolean_expression:
    body
```

Serial if statements vs. if elif elif…

```
if boolean_expressionA:
    bodyA
if boolean_expressionB:
    bodyB
if boolean_expressionC:
    bodyC
if boolean_expressionD:
    bodyD
```

- Potentially, all bodies execute
- All four Boolean expressions will definitely be checked

```
if boolean_expressionA:
    bodyA
elif boolean_expressionB:
    bodyB
elif boolean_expressionC:
    bodyC
elif boolean_expressionD:
    bodyD
```

- At most one body executes
- Could be more efficient