



THE UNIVERSITY OF  
CHICAGO

**MASTERS IN  
COMPUTATIONAL  
SOCIAL SCIENCE**  
THE UNIVERSITY OF CHICAGO

MACS 30111

Principles of Computing 1:  
Computational Thinking for Social Scientists

Lecture: T/Th 12:30-1:50pm

# Agenda

- ▶ Background / course info
- ▶ Diving into content!
  - ▶ Writing programs
  - ▶ Variables
  - ▶ Expressions and operators

# About me

**Jean Clipperton** [clipperton@uchicago.edu](mailto:clipperton@uchicago.edu)

- Jean or Prof. Clipperton

## **Research:**

- Ph.D. in Political Science
- Agent Based Modeling
- Data Visualization

## **Current research interests:**

- Text analysis of Taylor Swift + Eras tour
- Political candidates and campaign music

<https://jmclip.github.io/>

# Today's class

- ▶ Course management
- ▶ Course Structure and Syllabus
- ▶ Difference between 30111 and 30121
- ▶ Grading
- ▶ Policies
- ▶ Challenges
- ▶ Course topics

# Course Management

## ▶ Canvas

- ▶ Course landing page

## ▶ Ed discussion

- ▶ Announcements
- ▶ Asking questions (anonymous option also!)

## ▶ GitHub classroom

- ▶ Distribute assignments
- ▶ Accept first assignment (very simple to complete!)

## ▶ Gradescope

- ▶ Submit assignments and get feedback

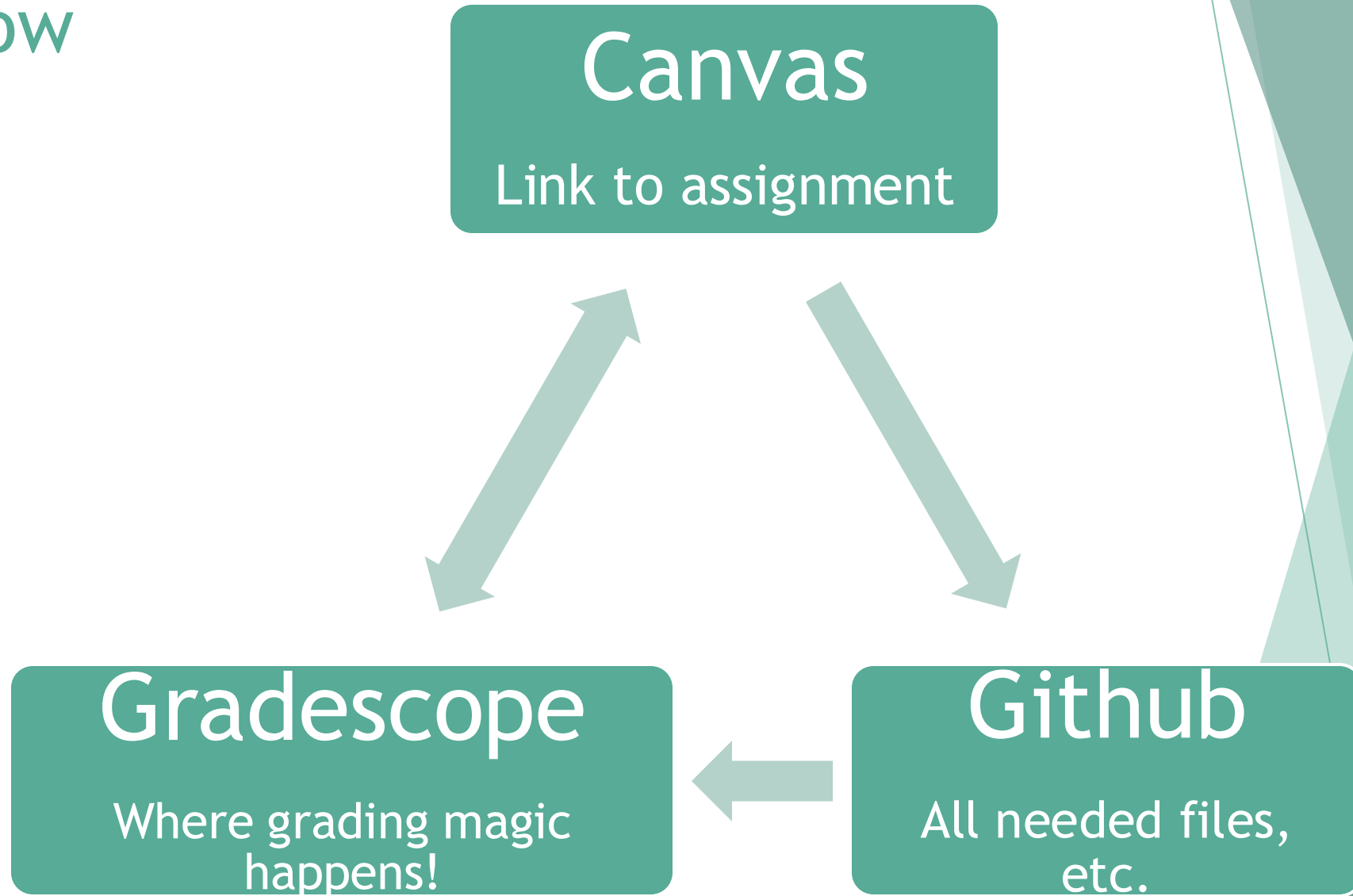
## • Course website

- <https://classes.ssd.uchicago.edu/macss/macss30121/>
- On-campus
- VPN for outside campus

## • Textbook

- Online: <https://book.cs-apps.org/>
- pdf

# Workflow



# Course Structure and Syllabus

- T/Th lectures
- 6 modules (9 weeks)
- **Assignments:**
  - Team tutorial
  - Short exercise
  - Programming assignment
- **Exams**
  - Timed-assessments: **contact Student Disabilities Services if you need accommodations for this -- ASAP**
  - Midterm and Final (paper-and-pencil + take-home)

# Grading

- **Assignments:**
  - Team tutorial (graded: completion)
  - Short exercise (auto-grading)
  - Programming assignment (auto + manual grading)
    - Style + correctness + completeness + efficiency
- **Exams**
  - Midterm and Final (paper-and-pencil)
- **Goal**
  - Help you understand how much you truly understand at a practical and theoretical level



# Grading: different from what you may be used to

- **TT: Teaching Tutorials**, 1 point - these are completed independently and outside of class.
  - Grading is 1 point for a completed TT; 0 otherwise. Please be sure to fully complete the exercises.
  - You need to complete five of these (lowest score dropped, six are available).
- **SE: Short exercises**, 10 points - these are 'autograded' within gradescope.
  - Grading is (essentially) based on completion, although I STRONGLY suggest you look at the feedback you receive as it will help for the PAs.
- **PA: Programming Assignments**, 25 points each -- these are graded on meeting criteria and your overall coding quality. If you meet the autograder criteria + have excellent coding, you can earn a bonus point for quality of code.
  - Grace period: two days for each assignment
  - Grace period OR eligible for resubmission. Can earn up to 20 points.
- **Exams**: ~100 points each -- these are points-based in grading and will occur in-class + have a take-home component. See schedule for more details.

# Grading: Expectations

- ▶ Do your best work
- ▶ Complete the assignments
- ▶ Should be able to expect a B in the course. From there, it depends on effort and how much you are able to challenge yourself with the material
- ▶ **Goal is that this is a supportive learning environment: you can make mistakes and learn.**
- ▶ Grace period is automatic - no penalty, no need to inform us, etc. Please try to be timely but we get it and life happens!

# Policies

- Submit your own work: you are **\*\*encouraged\*\*** to work with others, especially on the team tutorial. However, **everything you submit should be the product of your own intellectual contribution. If in doubt, over-document!**
- **This is a learning environment.** We need to make mistakes so that we can grow. I challenge you to try to ask questions that you feel maybe others already know, to check in when you are feeling confused, and to ask for help if you made a mistake.
- I would much rather you do your best and have it be imperfect than have you submit perfect work that is not your own / not reflective of what you can do
- Finally, be here on time, do the work, and be committed to a growth mindset.

# TAs and OH!

	Name	Email	Office Hours	Location
Instructor	Jean Clipperton	clipperton@uchicago.edu	<u>Wednesdays 11-12 (zoom) and</u>  <u>Thursdays 10:00am-11:00am</u>	<u>Weds zoom</u>   <u>#219: use signup link please!</u>
TA	Qilong Bi	bql20@uchicago.edu	Wednesday 3-4 pm	
TA	Wendi Xue	wendixue@uchicago.edu	Friday 1:30 pm-3pm	
TA	Pierre Loertscher	ploertscher@uchicago.edu	TBD	

**NOW FOR THE FUN STUFF!**

# Preparing for success

- Have VS Code up
- Have the textbook up for easy copy/paste
- Look over slides in advance (I always update right before class, but they are tweaks for the most part)
- **Be ready to participate!**



# What is a computer program?

A collection of instructions for the computer to perform.

Ask the computer to

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

```
8+6
```

```
def calculate_salary(hourly_pay, working_hours):  
    salary = 0  
    for h in working_hours:  
        salary = salary + hourly_pay * h  
  
    print("Bob earned $%d in last week" % salary)  
    return salary
```

```
salary = calculate_salary(hourly_pay=15, working_hours=[8,6])
```

Bob earned \$210 in last week

Human language: English, Chinese, French, Spanish, Germany, etc.

Computers, **programming language**: Python, Java, C, C++, R, etc.

# Python 3

- Widely-used modern programming language
- Simple for beginners to get started
- Strong support from large community
- Python as a tool
- The concepts and skills you learn with Python will carry over to many other languages.

Think computationally



# Topics:

- ❑ Writing and running your first program
- ❑ Variables and Types
- ❑ Expressions and Operators

# Topics:

- ❑ **Writing and running your first program**
- ❑ Variables and Types
- ❑ Expressions and Operators

# Your first program

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

# How do we run a Python program?

1. In the Python interpreter
  - Small piece of code
  - Interactive
2. Saving the code to a file and *running* that file (you also got training on opening files in the terminal in the bootcamp)
  - Large piece of code
  - Reusable

# Run a Python program in Python Interpreter

```
Last login: Thu Sep 21 13:03:43 on ttys001
(base) jeanclipperton@MAPSS-35004L Examples % python3
Python 3.9.12 (main, Apr 5 2022, 01:53:17)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print("Hello World!")
Hello World!
>>> █
```

```
Last login: Thu Sep 21 13:03:43 on ttys001
ipython
(base) jeanclipperton@MAPSS-35004L macs30111 % ipython
Python 3.9.12 (main, Apr 5 2022, 01:53:17)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.2.0 -- An enhanced Interactive Python. Type '?' for help.

[In [1]: print("Hello World!")
Hello World!

In [2]: █
```

# Run a Python file

A **.py** file that contains a sequence of Python instructions

```
Last login: Thu Sep 21 13:01:11 on ttys001
[(base) jeanclipperton@MAPSS-35004L Examples % echo "print('Hello world!')" > test.py
[(base) jeanclipperton@MAPSS-35004L Examples % python3 test.py
Hello world!
(base) jeanclipperton@MAPSS-35004L Examples % █
```

# Interpreter vs File

1. In the Python interpreter
  - Small piece of code
  - Interactive
  - “REPL” (Read-Evaluate-Print-Loop)
2. Saving the code to a file and *running* that file
  - Large piece of code (e.g., 1000 lines)
  - Re-usable

## **Workflow:**

1. Use interpreter to play around
2. Add code to the *.py* file
3. Test the program
4. Polish and repeat

# Coding practice:

- Follow instructions from: **course website** → resources → running Python
  - Terminal + Python Interpreter
  - Terminal + IPython Interpreter
  - Terminal + Text Editor (*.py* file)

Test out your own “hello world!”

## Course website

- <https://classes.ssd.uchicago.edu/macss/macss30121/>
- **On-campus**
- **VPN for outside campus**

## Textbook

- Online: <https://book.cs-apps.org/>
- pdf



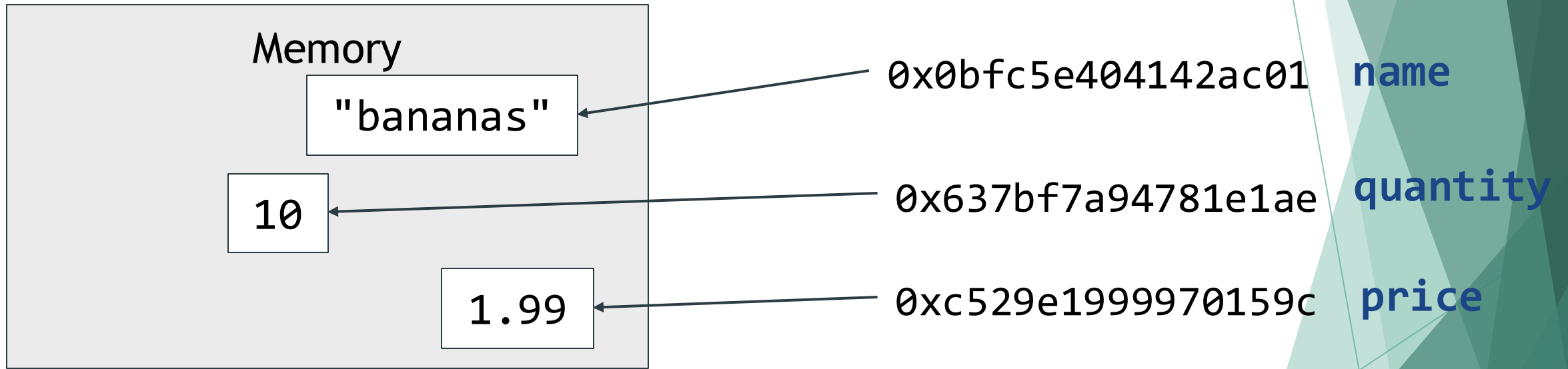
# Topics:

- ❑ Writing and running your first program
- ❑ **Variables and Types**
- ❑ Expressions and Operators

# Variables

Keep track of certain information while your program is running.

10 bananas, \$1.99/pound



A variable is a symbolic name representing a location (the information stored in this location) in the computer memory.

# Variables

Assign a value to a variable:

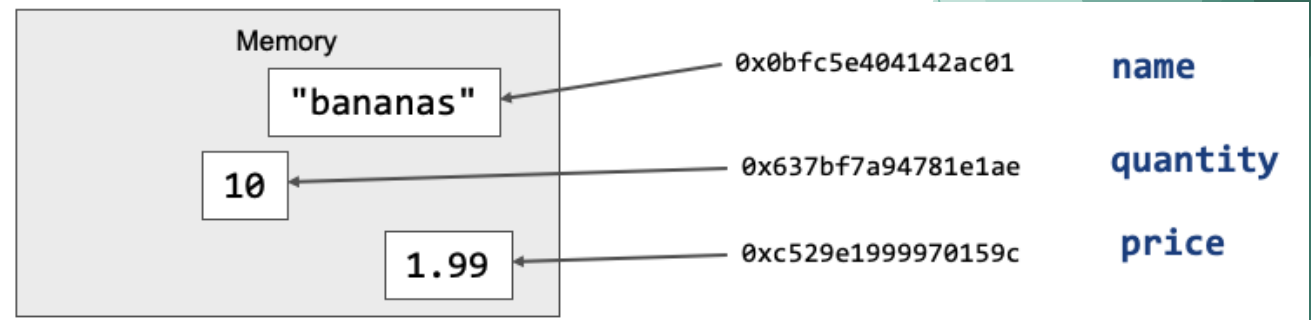
variable = value  
Assignment operator

Assignment statement:

message = "Hello, world!"  
Variable Value

# Variable Types

- Integers (e.g., quantity = 10)
- Real numbers, or “floats” (e.g., price = 1.99)
- Strings (e.g., name = “banana”)
- Booleans (e.g., is\_fail= True/False)



Python is a *dynamically-typed* language. It infers the type based on the value being assigned.

the built-in `type()` function

# Variable Types: Casting

Infer the type of a variable by the value it is assigned:

- `x = 14.5`, `x = "hello"`
- `msg = "The value of pi is" + 3.1415`
- Is this correct?

**Casting:** explicitly tell Python to convert a value to a specific type:

- *cast* 3.1415 into a string by using the `str()` function
- `msg = "The value of pi is " + str(3.1415)`
- similarly, there are `int()` and `float()` functions.

# Quiz

Variables allow us to...

- Increase and decrease the size of our memory (because that size is variable)
- Keep track of information in our program
- Specify whether I'm running a Python program or a Java program

# Quiz

What operator do I use to assign a value to a variable?

- =
- ==
- <\_

# Quiz

What are the types we can use in Python?

- Python only provides numerical types
- Integers, floats, strings, booleans, and None
- Real numbers, complex numbers, and text



# Topics:

- ❑ Writing and running your first program
- ❑ Variables and Types
- ❑ **Expressions and Operators**

# Expressions

`a = 5`

`b = 3`

`x = 1.5`

`s = "Hello, world!"`

**Expressions: combinations of variables and operators**

`addition = 2 + 2`

`subtraction_1 = a - 1`

`subtraction_2 = a - b`

# Arithmetic Operators

- Addition: +
- Subtraction: -
- Multiplication: \*
- Division: /
- Integer division: //
- Modulus: % (e.g.,  $8\%3 = 2$ )

The order of evaluation is governed by *precedence rules*, which we can affect using parentheses.

# Comparison Operators

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

return Boolean value

# Boolean operators

a = 5

b = -10

Logical AND: **and**

- True if both operands are True. Otherwise, False.
- a>0 and b>0

Logical OR: **or**

- True if at least one of the operands are True
- False if both are False
- a>0 or b>0

Logical NOT: **not**

- opposite of the boolean value
- not a>0

# Coding practice:

Reference book:

- 1.2.5
- 1.2.6
- 1.2.7

A leap year is a year that...

- Is divisible by 4 but *not* divisible by 100
- Unless the year is divisible by 400

# Topics:

- ❑ Writing and running your first program
  - ❑ Interpreter
  - ❑ File
- ❑ Variables and Types
  - ❑ Types (integer, real number, string, boolean)
  - ❑ Casting
- ❑ Expressions and Operators
  - ❑ Arithmetic
  - ❑ Comparison
  - ❑ Boolean

# Homework and Assignments

- 1) Read 1.3 and 2.1 before next class
- 2) TT: Tuesdays
- 3) SE/PA: Fridays
- 4) Accept first assignment to join GitHub classroom
- 5) Explore the course website

<https://classes.ssd.uchicago.edu/macss/mac30121/>

- 6) Office hours
  - 1) Canvas



# First week: to do

- ▶ **Software and package installation**

- ▶ Anaconda

- ▶ **Tutorials**

- ▶ Git

- ▶ Linux

- ▶ **Course workflow**

- ▶ Git/GitHub

- ▶ Canvas

- ▶ Gradescope

- ▶ Resources

# 30121 VS 30111

	30121	30111
Lectures	M1~M7	missing M6: recursion
Team tutorials	✓	✓
Short exercises	✓	✓ + additional
Programming assignments	7	3
Exams	✓	✓
Goal	computational thinking in various social science applications, which are both helpful for social scientists and potential computer scientist (or CS courses or a tech job (SDE, DS))	computational thinking ability and application in basic social science applications, preparation for a career in social science



A pass in 30121 means you can take advanced CS courses.

A pass in 30111: you still need to take CS placement exam for advanced CS courses.