

EECS 598-008 & EECS 498-008: Intelligent Programming Systems

Lecture 1: Course Overview

Clarifications

- EECS 598-008: Graduate, in-person
- EECS 498-008: Undergraduate, in-person
- EECS 598-098: Graduate, remote (for students who cannot arrive in the US)
- EECS 498-081: Undergraduate, discussion section

- If you're a graduate in the US, you should register 598-008, even if you envision you may miss a few lectures (and that's totally fine)
- 498-081 (discussion)
 - Some may convert to remote office hours (see schedule on course page)
 - Optional, but highly recommend
- All office hours, both instructor's and GSI's, will be remote
 - <https://oh.eecs.umich.edu/courses/eecs-498-598-008>

COVID-related Course Policies

- Masks **required**. [1]
- Proof of vaccination/weekly testing **required**. [2]
- Social distancing (6 ft/1.8m) **highly recommended**. [3]

- Whenever not feeling well/uncomfortable attending in-person, skip in-person lectures.
- All in-person lectures are recorded and available afterwards.
- Remote live lectures available.
- Remote student presentations could be an option, but more details forthcoming.
- In case instructor cannot lecture in person, convert to remote or reschedule to another time.

[1] https://ehs.umich.edu/wp-content/uploads/2020/07/U-M-Face-Covering-Policy-for-COVID-19.pdf?utm_source=Engin+Update+Fall+2021&utm_campaign=dde963167e-EMAIL_CAMPAIGN_2018_12_13_02_33_COPY_01&utm_medium=email&utm_term=0_77986d5a13-dde963167e-278510492

[2] <https://record.umich.edu/articles/u-m-outlines-accountability-measures-for-covid-19-vaccine-policy/>

[3] https://ehs.umich.edu/wp-content/uploads/2021/06/COVID-19-Guidelines-for-Campus-Facilities.pdf?utm_source=Engin+Update+Fall+2021&utm_campaign=dde963167e-EMAIL_CAMPAIGN_2018_12_13_02_33_COPY_01&utm_medium=email&utm_term=0_77986d5a13-dde963167e-278510492

Tools Used in This Course

- <https://web.eecs.umich.edu/~xwangsd/courses/f21/index.html>: schedule, dues, recordings, slides, etc.
- Slack: ad-hoc questions/discussions, quick announcements
- <https://oh.eecs.umich.edu/courses/eecs-498-598-008>: office hours
- Canvas: mostly assignment/report submissions
- HotCRP (<https://umich-eecs598-eecs498-fall-21.hotcrp.com/>): offline paper reviews/discussions
- Email (ips2021umich@gmail.com): private messages to teaching staff (instructor & GSI)

- Instructor Email (xwangsd@umich.edu): private messages to instructor
- GSI Email (leozhu@umich.edu): private messages to GSI

Announcements

- First programming assignment A0 out today
- Friday discussion section → remote office hour

Agenda

- **Introduce yourself**
- Course overview
- Logistics overview

Course Staff

- Instructor: Xinyu Wang
 - Assistant Professor in CSE (joined in September 2020)
 - Research: Computer Systems
 - > Programming Languages, Formal Methods, Software Engineering
 - > **Program Synthesis/Analysis/Verification**
- GSI: Yuanli Zhu
 - Masters student in CSE

Introduce yourself

- Name?
- What program? What year?
- What CS area(s) are you interested in?

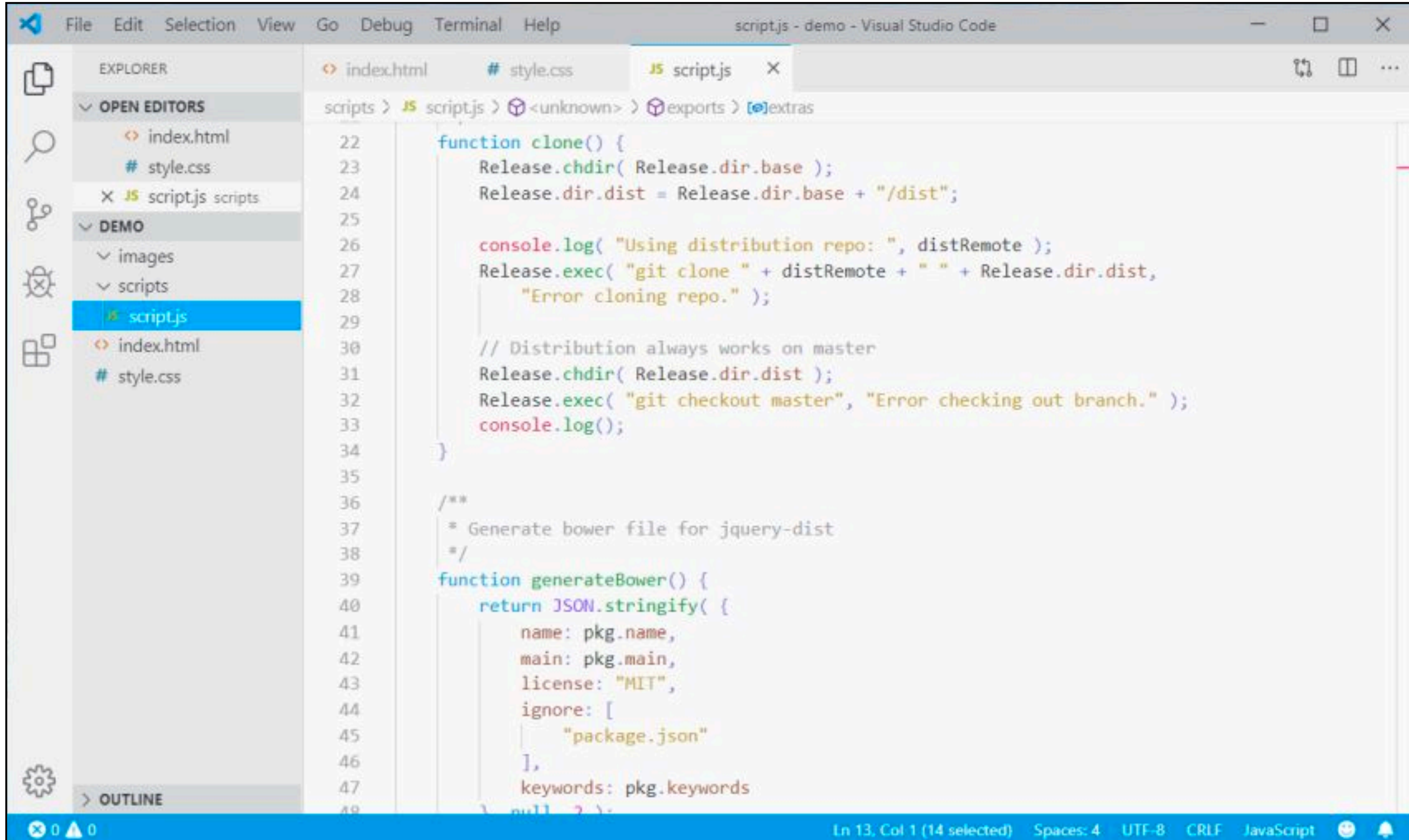
Agenda

- Introduce yourself
- **Course overview**
- Logistics overview

Course Overview

- What's this course about?
 - Intelligent programming systems, powered by program synthesis
 - Make programming systems more intelligent
- What's "programming systems"?
 - Broad: programming languages, compilers, runtime systems, ...

E.g., IDEs (VS Code, Eclipse, ...)



The screenshot shows the Visual Studio Code IDE interface. The Explorer sidebar on the left displays the file structure, including 'index.html', 'style.css', and 'script.js'. The main editor window shows the content of 'script.js', which includes a 'clone' function and a 'generateBower' function. The 'clone' function uses 'Release.chdir' and 'Release.exec' to clone a repository and checkout the master branch. The 'generateBower' function returns a JSON object representing a Bower package configuration.

```
function clone() {
  Release.chdir( Release.dir.base );
  Release.dir.dist = Release.dir.base + "/dist";

  console.log( "Using distribution repo: ", distRemote );
  Release.exec( "git clone " + distRemote + " " + Release.dir.dist,
    "Error cloning repo." );

  // Distribution always works on master
  Release.chdir( Release.dir.dist );
  Release.exec( "git checkout master", "Error checking out branch." );
  console.log();
}

/**
 * Generate bower file for jquery-dist
 */
function generateBower() {
  return JSON.stringify( {
    name: pkg.name,
    main: pkg.main,
    license: "MIT",
    ignore: [
      "package.json"
    ],
    keywords: pkg.keywords
  } );
}
```

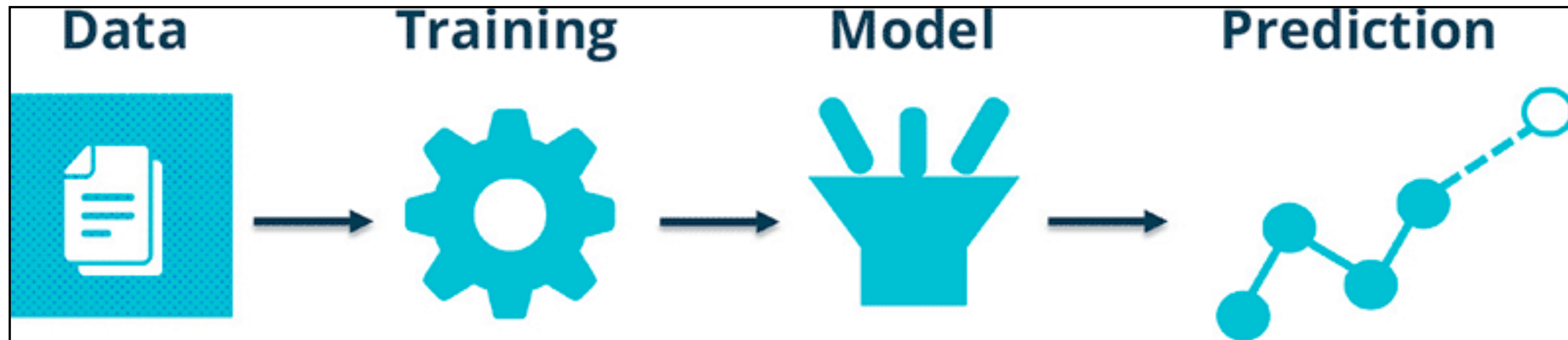
E.g., Compilers (gcc, llvm, ...)



E.g., Spreadsheets (Excel, ...)

	A	B	C	D
1	Source of Leads	November Leads	December Leads	Total Leads
2	Blog Post 1	10	15	=SUM(B2, C2)
3	Blog Post 2	4	12	
4	Blog Post 3	11	7	
5	Blog Post 4	2	8	
6	Blog Post 5	12	19	
7	Blog Post 6	6	11	
8	Blog Post 7	8	8	
9	Blog Post 8	17	19	
10	Blog Post 9	3	6	
11	Blog Post 10	8	14	
12				

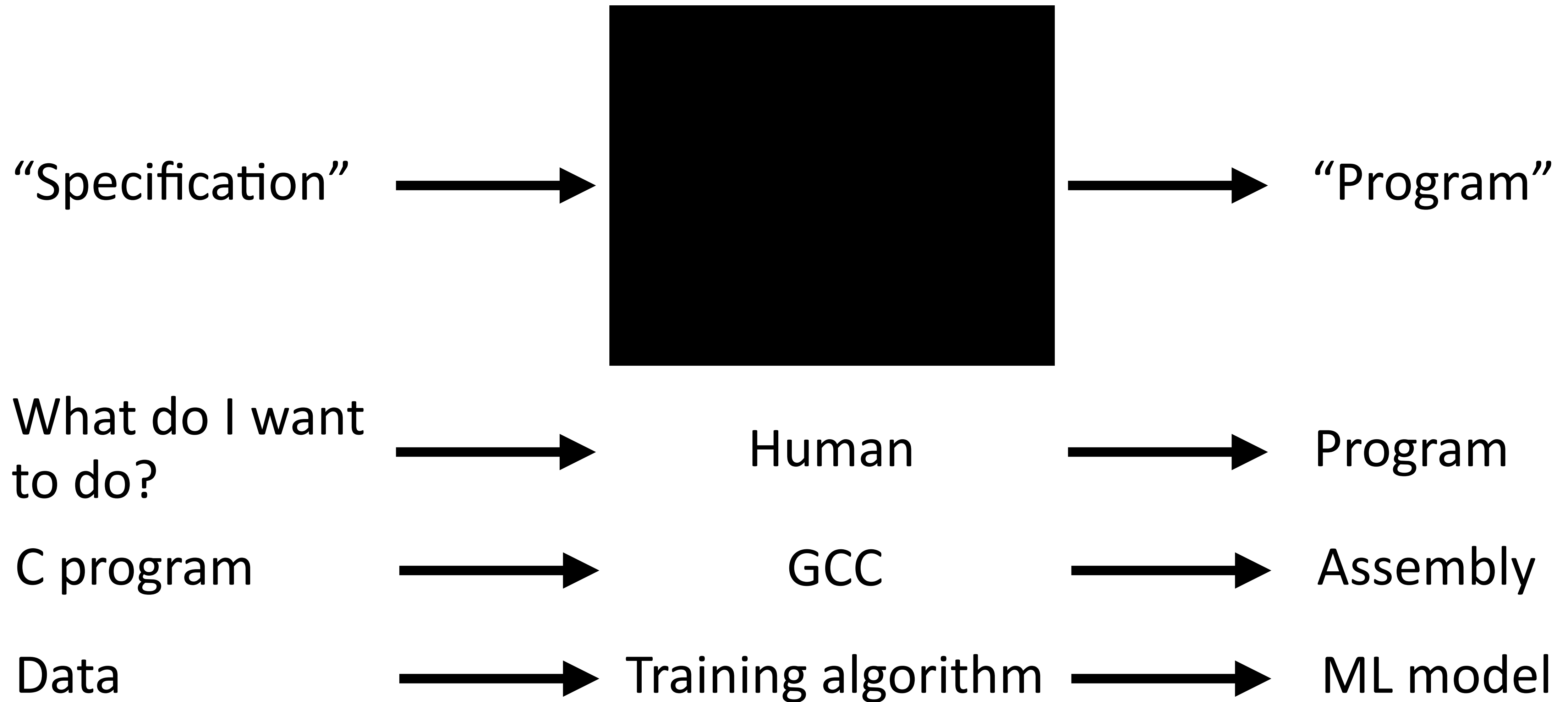
E.g., Machine Learning Pipelines



Course Overview

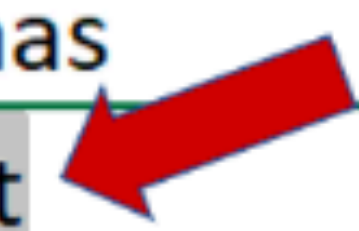
- What's this course about?
 - Intelligent programming systems, powered by program synthesis
 - Make programming systems more intelligent
- What's "programming systems"?
 - Broad: programming languages, compilers, runtime systems, ...
 - They have to do with programming
 - What's "programming"?

Programming

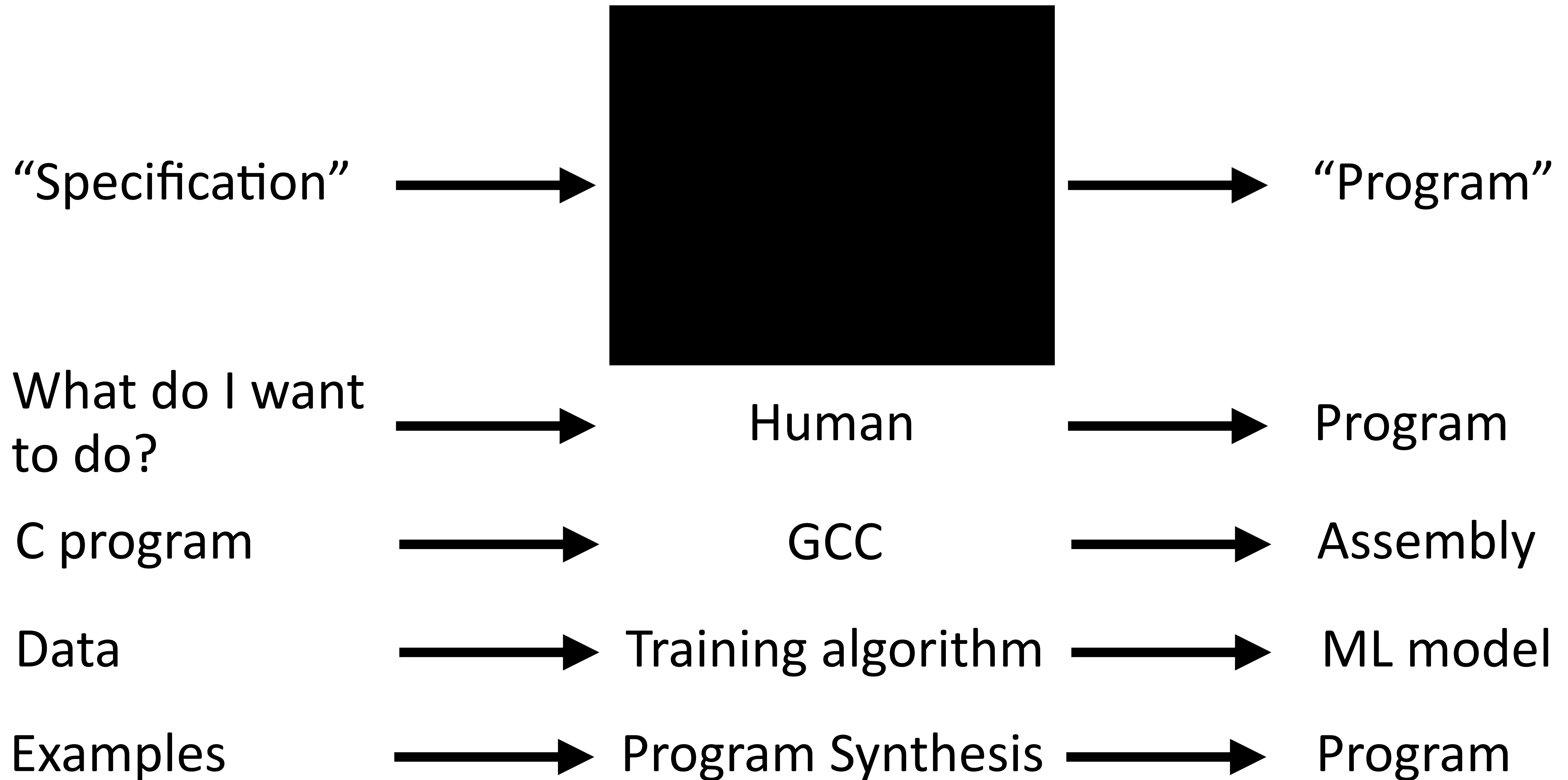


How about this (FlashFill feature in Excel)?

	A	B	C
1	Name and ID	First name and last name	ID #
2	Thomas, Rhonda 82132	Rhonda Thomas	
3	Emmett, Keara 34231	Keara Emmett	
4	Vogel, James 32493	James Vogel	
5	Jelen, Bill 23911	Bill Jelen	
6	Miller, Sylvia 78356	Sylvia Miller	
7	Lambert, Bobby 25900	Bobby Lambert	
8	Sweet, Julie 65477	Julie Sweet	
9	Williams, Don 43920	Don Williams	
10	Spake, Deborah 33488	Deborah Spake	



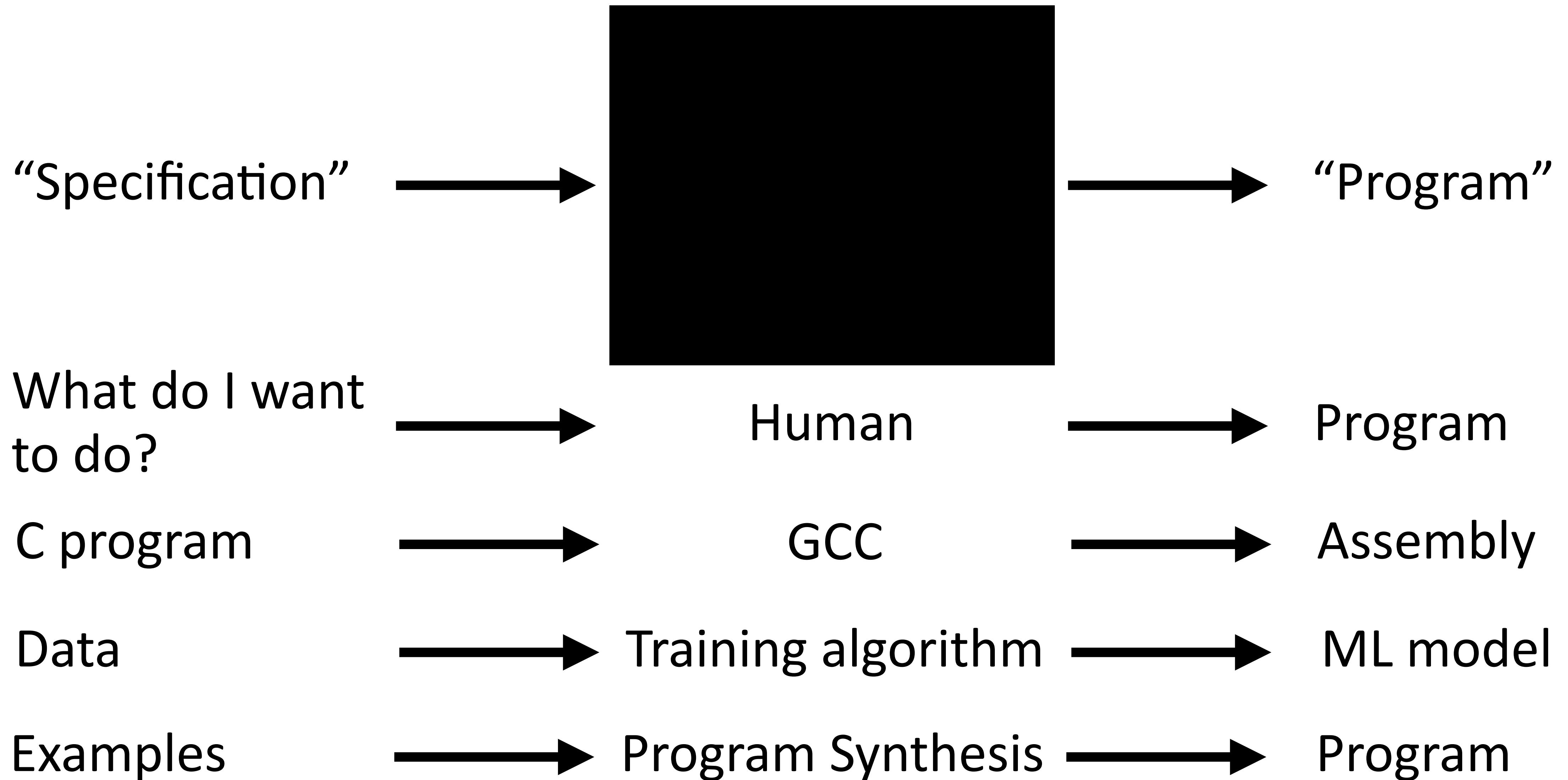
Programming



Course Overview

- What's this course about?
 - Intelligent programming systems, powered by program synthesis
 - Make programming systems more intelligent
- What's "programming systems"?
 - Broad: programming languages, compilers, runtime systems, ...
 - They have to do with programming
 - What's "programming"? Specifications → Programs
- How to make it more "intelligent"?
 - Specifications: more "high-level"
 - Programming process: more automated

Programming



E.g., FlashFill [Gulwani et al. 11]

- Synthesize Excel macros for string processing from input-output examples ([video](#))



	A	B
1	Names	Initials
2	Neil Lieber	N L I
3	Mathew Prisco	
4	Althea Bertin	
5	Kelly Gamblin	
6	Chandra Valenzula	
7	Cody Castillon	
8	Tyrone Brazier	
9	Althea Buhl	
10	Dollie Munsey	
11	Allyson Phou	



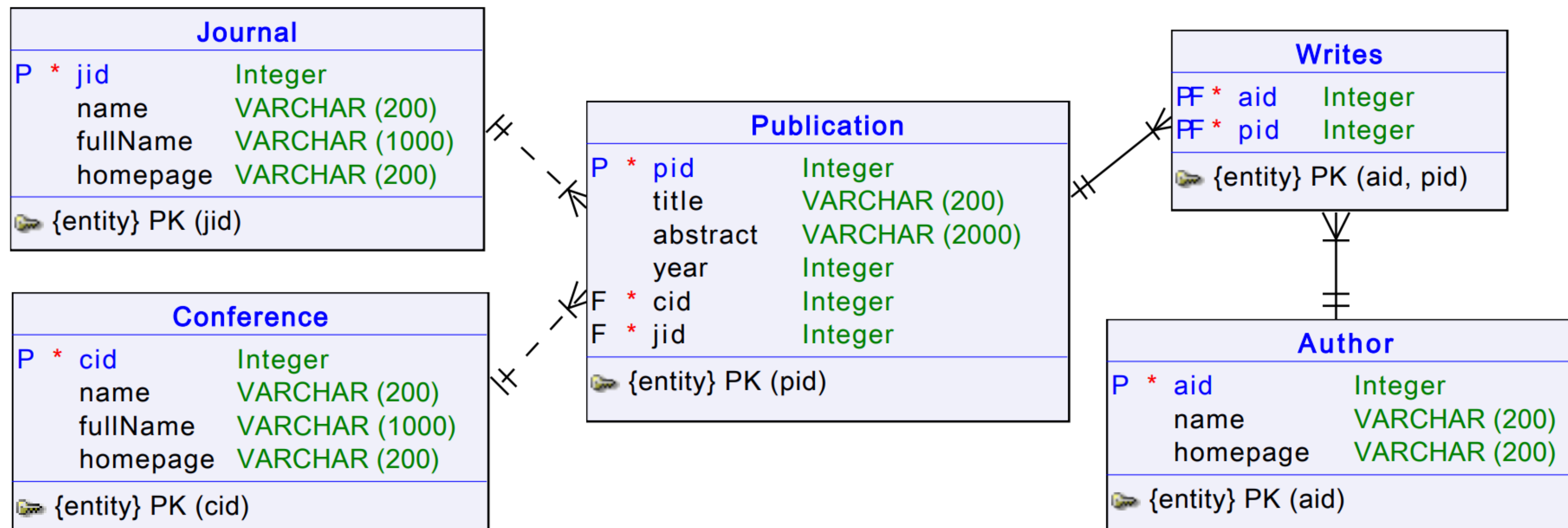
	A	B
1	Names	Initials
2	Neil Lieber	N L
3	Mathew Prisco	M P
4	Althea Bertin	A B
5	Kelly Gamblin	K G
6	Chandra Valenzula	C V
7	Cody Castillon	C C
8	Tyrone Brazier	T B
9	Althea Buhl	A B
10	Dollie Munsey	D M
11	Allyson Phou	A P

E.g., SQLizer [Yaghmazadeh et al. 17]

- Synthesize SQL queries from natural language (given schema)

NL: *“Find the number of papers in OOPSLA 2010”*

Schema:



SQL query:

```
SELECT count(Publication.pid)
FROM Publication JOIN Conference ON Publication.cid = Conference.cid
WHERE Conference.name = "OOPSLA" AND Publication.year = 2010
```

E.g., Rousillon [Chasins et al. 18]

- Synthesize web scraping scripts from example demonstrations ([video](#))

The image shows a Google Scholar interface. On the left, a search bar contains 'label:computer_science' and a 'Profiles' section lists several researchers, including Vapnik, Geoffrey Hinton, DEYWIS MORENO, David S. Johnson, and David Haussler. On the right, a detailed profile for Vapnik is displayed, including his title, affiliations, and a list of his works with citation counts and years. A bar chart on the far right shows the number of citations for Vapnik's work from 2011 to 2018.

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	77066 *	1995
Statistical Learning Theory VN Vapnik Wiley-Interscience	76088 *	1998
Support-vector networks C Cortes, V Vapnik Machine learning 20 (3), 273-297	31294	1995
Pattern classification and scene analysis RO Duda, PE Hart A Wiley-Interscience Publication, New York: Wiley, 1973	20912	1973

	All	Since 2013
Citations	257774	105623
h-index	124	81
i10-index	465	347

Year	Citations
2011	~15000
2012	~16000
2013	~16500
2014	~17000
2015	~17500
2016	~18000
2017	~18500
2018	~10000

E.g., Rousillon [Chasins et al. 18]

- Synthesize web scraping scripts from example demonstrations ([video](#))

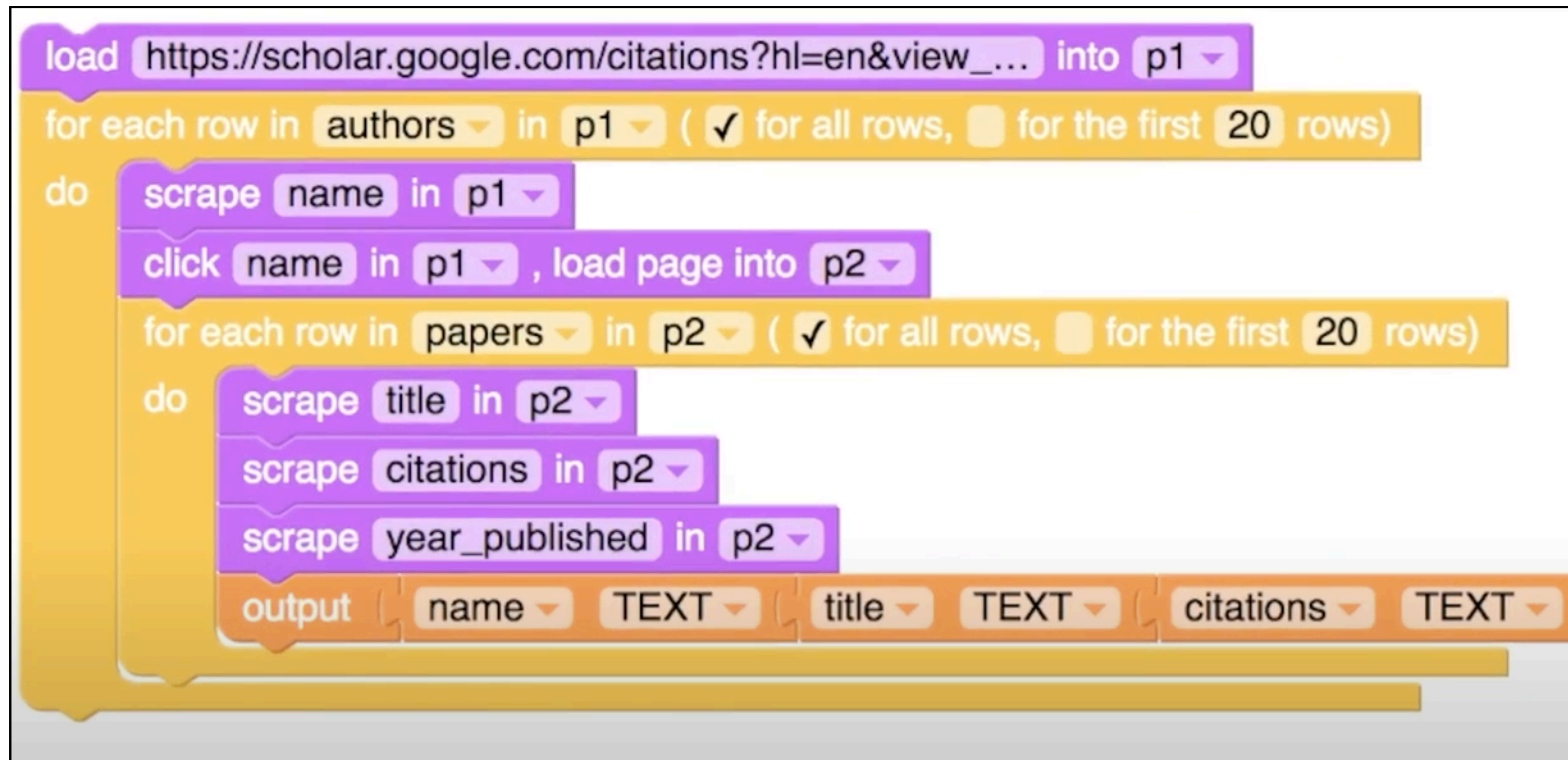
The image shows a side-by-side comparison of a web scraping tool interface and a target website. On the left, the tool's interface includes tabs for 'Current Script', 'Saved Scripts', and 'Scheduled Runs'. A text box contains the instruction: 'Awesome. We're recording, so go ahead and collect that first row of data. When you're ready to add a new cell to the first row of your dataset, just hover over the text you want to add, then press **ALT** + click. We'll show the data you've collected right here:'. Below this, a data row is displayed: 'vapnik The Nature of Statistical Learning Theory 77066 1995'. At the bottom of the tool interface is a red 'Stop Recording' button. On the right, a browser window shows a Google Scholar profile for 'vapnik', a Professor of Columbia. Below the profile, a table of articles is visible:

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	77066 *	1995
Statistical Learning Theory VN Vapnik Wiley-Interscience	76088 *	1998
Support-vector networks C Cortes, V Vapnik Machine learning 20 (3), 273-297	31294	1995
...	20912	1973

A purple text box at the bottom of the screenshot reads: 'demonstrate how to collect the first row'.

E.g., Rousillon [Chasins et al. 18]

- Synthesize web scraping scripts from example demonstrations ([video](#))



```
load https://scholar.google.com/citations?hl=en&view_... into p1
for each row in authors in p1 ( ✓ for all rows, for the first 20 rows)
do
  scrape name in p1
  click name in p1 , load page into p2
  for each row in papers in p2 ( ✓ for all rows, for the first 20 rows)
  do
    scrape title in p2
    scrape citations in p2
    scrape year_published in p2
  output name TEXT title TEXT citations TEXT
```

The image shows a sequence of code blocks for web scraping. It starts with a 'load' block for a Google Scholar URL into variable 'p1'. A 'for each row' loop iterates over 'authors' in 'p1', with a checked option for 'for all rows' and a limit of 20 rows. Inside this loop, a 'do' block contains: a 'scrape' block for 'name' in 'p1'; a 'click' block for 'name' in 'p1' to load a new page into 'p2'; another 'for each row' loop for 'papers' in 'p2', also with 'for all rows' checked and a limit of 20 rows; and a nested 'do' block with three 'scrape' blocks for 'title', 'citations', and 'year_published' in 'p2'. Finally, an 'output' block concatenates 'name', 'title', and 'citations' as TEXT fields.

This Course

- How to “automatically” generate “programs” from “specifications”?
 - Program synthesis, in different application domains
- Learn fundamental concepts in programming languages, automated reasoning, formal methods, formal languages, etc.
- Build a real program synthesizer (for R language)
- Read papers on program synthesis
 - How to apply program synthesis to solve interesting problems in many other areas: data science, data wrangling, databases, web automation, etc.
 - How to develop better program synthesis techniques: using HCI, ML, NLP, etc.
- Work on your own project and present it to others

Working Definition of Program Synthesis

High-level intent

Specification

Program synthesis



Lower-level code

Program

Typically involves search

*I/O examples, demonstrations,
natural language, reference
implementation, etc.*

*In some programming language
(grammar + semantics)*

Program Synthesis vs. Machine Learning/Deep Learning

- ML/DL is also program synthesis?
 - ML/DL: data is spec, model is program, try to learn a model that matches data
 - At a high-level, yes
 - Not the focus of this course
 - Definitions of “programs” are very different (e.g., grammar vs. neural nets)
 - Data is noisy whereas spec is less noisy (but there is a trend in program synthesis to tolerate noise in spec)
 - Typically continuous in ML/DL vs. discrete search space in program synthesis
 - The line is getting blurry

Program Synthesis vs. Compilers

- Program synthesizers are compilers? Compilers are synthesizers?
 - Compilers also convert high-level intent (code) to lower-level code
 - At a high-level, yes
 - Not the focus of this course
 - Compilers translate (well, not really nowadays) whereas synthesizers discover
 - Compilers apply predefined transformations (again, not really nowadays) whereas synthesizers perform search
 - The line is getting blurry

Working Definition of Program Synthesis

High-level intent

Specification

Program synthesis



Lower-level code

Program

Typically involves search

*I/O examples, demonstrations,
natural language, reference
implementation, etc.*

*In some programming language
(grammar + semantics)*

Agenda

- Introduce yourself
- Course overview
- **Logistics overview**

Two Cohorts of Students

- EECS 598 (3 credits): Graduate Students
- EECS 498 (4 credits): Undergraduate Students

Lectures, Discussions, Office Hours

- Lectures: 1010 DOW, 3-4:30pm Tuesdays and Thursdays
 - Learn fundamentals
 - Can attend remotely live via zoom
 - Recordings available afterwards
- Discussions: 3-4pm Fridays, see schedule for details re. location
 - Tutorials of necessary tools, explain assignments, etc.
- Instructor Office Hours: 3-4pm Wednesdays
- GSI Office Hours: 4:30-5:30pm Monday

Schedule

- Weeks 1-6: fundamentals
 - Lectures
 - Assignments
- Weeks 7-14
 - Research papers (reading, reviews, presentations)
 - Final project (proposal, two checkpoints, final project report & presentation)
- Detailed schedule online

What Do You Need To Do?

- Rationale: graduate students should focus on research, undergraduate students should focus on learning fundamentals but also get exposed to research as possible

	EECS 598	EECS 498
Class Participation	Yes (10%)	Yes (10%)
Assignments	Optional (15%)	Yes (30%)
Paper Reviews	Yes (20%)	Optional but highly recommended (10%)
Paper Presentations	Yes (25%)	Optional but highly recommended (20%)
Final Project	Yes (50%)	Yes (50%)

Class Participation

- **Try to** attend lectures/presentations/discussions, in person or remote
- Actively participate in **offline** paper discussions
 - Share your ideas/thoughts/comments
 - Ask questions
 - Answer questions

Assignments

- Four programming assignments
 - Progressively build a (simple) program synthesizer for R language
 - **A0 is out today**
- Due dates online

Paper Reviews

- Starting week 7, we will discuss a paper in each lecture
- Write a paper review before lecture
 - Short summary of paper
 - Strengths
 - Weaknesses
 - Questions/thoughts/limitations/etc.
- Submit review via HotCRP by **noon the day before lecture**
- Once you submit your review, you can see other reviews and discuss on HotCRP

Paper Presentations

- A list of papers will be made available online
- Each student will present one paper (likely with a co-presenter)
 - Read the paper (may need to pick up necessary background knowledge)
 - Prepare presentation
 - At least slides, demo if possible
 - Present the paper (45min talk + 30mins Q&A)
 - Thorough, cover background (45m is quite a long time)
 - Give high-level ideas as well as important lower-level technical details
 - Show one concrete example illustrating how everything works

Final Project

- 50% of final grade, very important
- Start early! Call For Proposals out **September 28th**
- Multiple steps
 - Find your teammates (solo okay; typically 2-3; if more than 3, talk to instructor)
 - Proposal
 - Checkpoint 1
 - Checkpoint 2
 - Final Project Presentation
 - Final Project Report

Final Project: Find Your Teammates

- On your own? That's okay, but consider looking for at least a collaborator!
- Work with 1-2 other students? Great!
- Form a bigger group? Sure, but the project scope will need to be bigger, too.
 - Discuss with the instructor

Final Project: Proposal

- 2-3 page that covers:
 - Statement of the problem you plan to investigate, including:
 - Definition of the problem
 - Concrete examples illustrating the problem
 - Explanation why this problem is interesting and why is it worth solving
 - E.g., how existing techniques are not sufficient in solving this problem
 - Description of your proposed approach
 - Can be a rough idea (proposal is not final report!)
 - Outline of important milestones
 - How to evaluate your proposed approach
 - On what benchmarks do you plan to evaluate your approach?
 - What are the success metrics? How do you know your approach works?

Final Project: Proposal

- What is a good topic for final project?
 - Type 1: Extending an approach from a paper
 - Their solution has certain limitations. Your approach extends their idea.
 - Type 2: Propose new ideas for an old problem
 - Replace (instead of extending) their solution with a better one
 - Type 3: Solve a new problem using standard techniques
 - Apply an existing solution (with tweaks) to a new problem
 - Type 4: new problem, new solution
 - Great!
- **Work on what's most exciting to you!**

Final Project: Two Checkpoints

- “Partial reports”
 - Accomplish important milestones
 - Eventually lead to Final Project Report

Final Project: Final Project Presentation

- Just like a paper presentation, but it's for your "paper"
 - That means, you can just follow how you present other people's work, but this time you're presenting your own work!

Final Project: Final Project Report

- 6-8 pages (in ACM's double-column conference format)
 - Introduction: what problem you're solving and why it's important (1 page)
 - Motivating example: use one concrete example to illustrate the problem as well as your solution (1-2 pages)
 - Your approach: explain in detail how your approach works (2-3 pages)
 - Evaluation: how you evaluate your approach, what the results are, and why they are what they are (1-2 pages)
 - Related work: how your work differs from prior work (1 page)

What Is This Course About?

- This course is about program synthesis, including both techniques and applications
 - Techniques: general synthesis algorithms not necessarily tied to a specific application
 - Applications: novel application of program synthesis techniques
- Beyond acquiring knowledge about program synthesis, also:
 - PL/formal thinking
 - Practice other skills: writing, presentation, etc.
 - Get exposed to research: through reading papers, doing a mini research project

Why This Course?

- Very hot research topic at the interaction of AI/ML and Systems/PL
- Useful in practice, e.g., FlashFill in Excel
- Technically very challenging
 - Essentially, a very hard search problem
- .. if you plan to explore possibilities of applying PL in your own research
- .. if you plan to pursue research career in PL/Formal Methods
- .. if you are interested in programming languages research
- .. if you just want to learn about the topic!

Survey (optional)

- Send me a brief email with:
 - Name
 - I am a [CS/___] [PhD/Masters/undergrad] in year [1/2/3/4/5/...]
 - Write one reason why you are taking this class or one thing you want to get out of it
 - One thing you would like the instructor to do in this class
 - One fun fact about you, or what you like to do in your spare time, or whatever