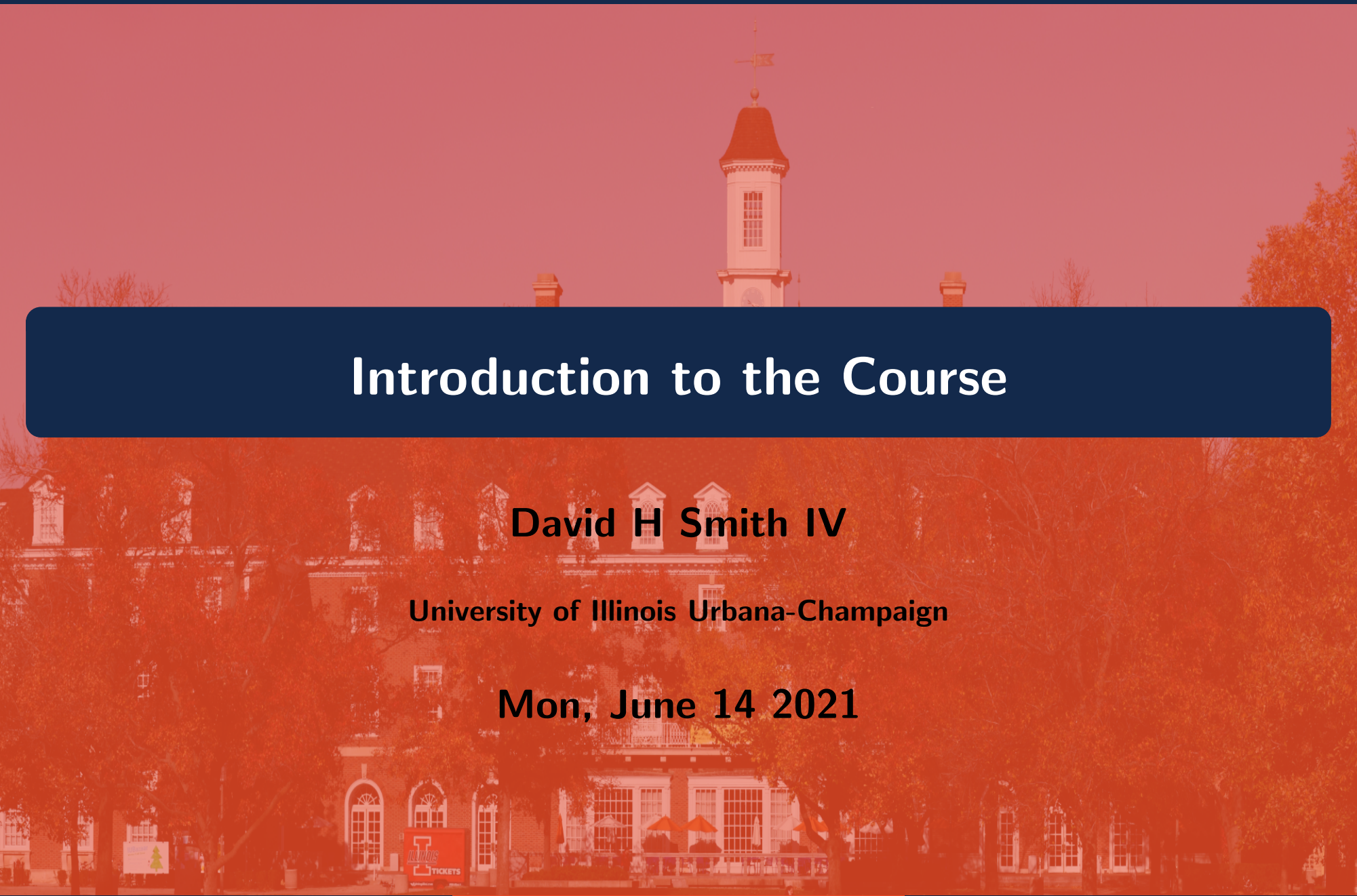


Introduction to the Course

David H Smith IV

University of Illinois Urbana-Champaign

Mon, June 14 2021

Course Outcomes

- Given a small section of code you should be able to:
  - Trace through and predict it's output.
  - Describe, in plain English, what it does.
- Write a small python program some, or all, of the fundamentals you will learn in the course.
- Beginner and intermediate spreadsheet operations.
- Beginner level understanding of how the internet works and how to write basic HTML documents.

What is Computer Science?

- CS is concerned with understanding:
  - What is computable
  - How to compute it in one or more of the following in mind:
    - Speed
    - Reliability
    - Security
    - Resource cost

What is Programming?

- Programming $\neq$ Computer Science
  - Rather, programming is a subset of Computer Science
- "Computer Science is no more about computers than astronomy is about telescopes" - Edsger W. Dijkstra
  - A bit of an overstatement but still a useful thing to keep in mind.
- **High Level Definition:** A series of instructions that a computer carries out.

Where did this all come from?

Early "Computers"

The Difference Engine

Alan Turing, Alonzo Church, and Turing Machines

1. **Church-Turing Thesis** - Any effective calculation can be performed by a mechanical computer (e.g., a Turing Machine).
2. **Turing Machines** - An abstract, mathematical model of a machine that moves up and down a strip of paper, one step at a time, and performs operations based on a set of rules (Figure 1).

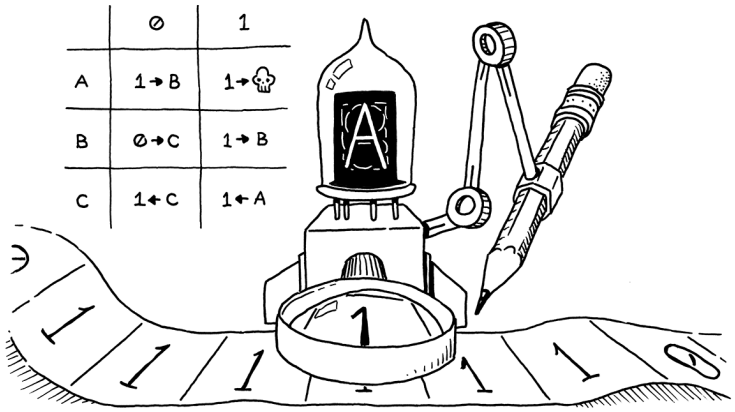


Figure: Artists Representation of a Turing Machine

Von Neumann Architecture

1. **Input Devices:** Something with buttons and knobs.
2. **Central Processing Unit:**
   1. **Control Unit:** Manages other parts.
   2. **Arithmetic/Logic Unit:** Does math on integers.
3. **Memory Unit:** Supplies info to CPU (i.e., Random Access Memory).
4. **External Storage Unit (Not Pictured):** We often need larger storage for data that isn't needed immediately (e.g., Hard Drive).
5. **Output Devices:** Flashing lights, monitor, etc.

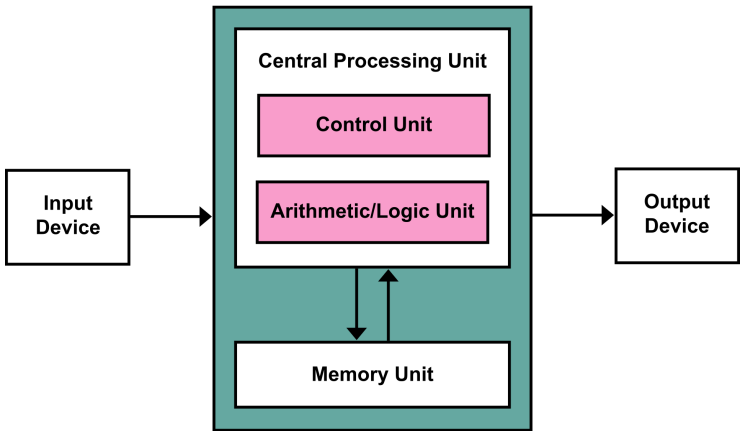


Figure: Von Neumann Architecture

How Programs are Constructed

Computers Run Low-Level Instructions

- Computers don't "Play a video"
- They...
  1. Move some numbers into memory.
  2. Do some math, or a comparison, or both
  3. Make a decision based on the results
  4. Rise a and repeat a few million time before something useful happens
- Programming at this level is:
  - Tedious and error prone.
  - Needs **A LOT** of code to do anything useful.
  - Isn't portable. Each processor is it's own machine and will require a different set of instructions that works with it's parts.

Enter Python (And Other High-Level Languages)

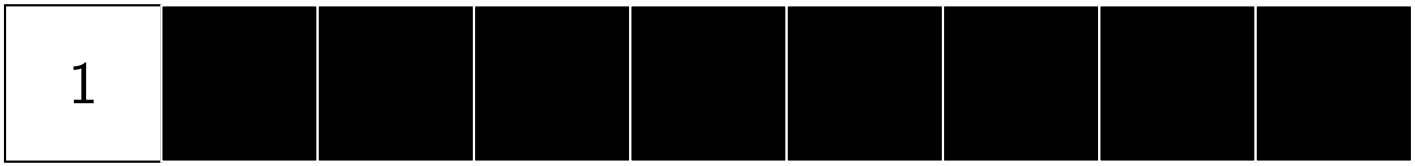
- They are:
  - **Productive** →A few lines do a lot and it's easy to debug.
  - **Safer** →Less likely to write code that is insecure or damaging.
  - **Portable** →Works on all systems that support the Python interpreter.
- Used for everything from machine learning to general scripting.
- We use Python 3.x (not Python 2.x)

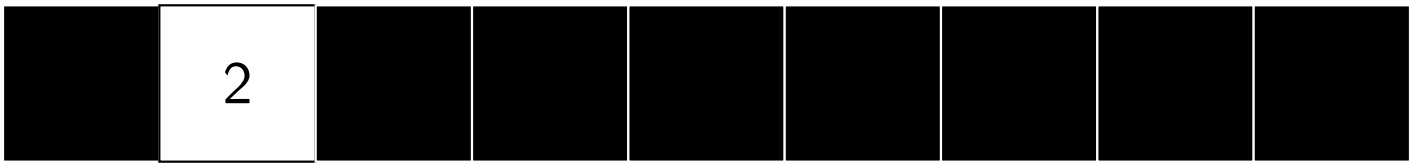
How programs are Constructed

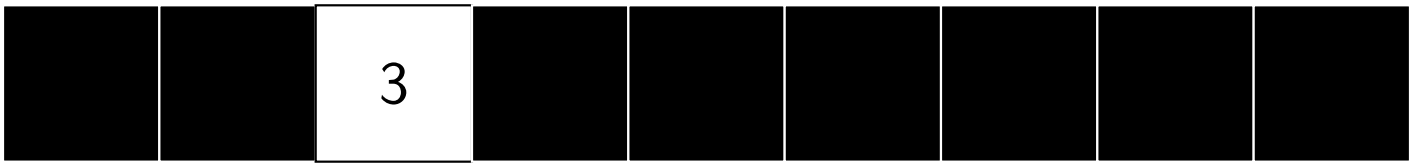
- **Algorithms** →A step-by-step process for achieving a result
  - Often Written in pseudo-code
- **Programming** →Express the commands in a form the computer understands.
- **Testing** →Designing inputs that test specific behaviours of the code.
- **Debugging** →Finding errors in the code based on the results of your tests and fixing them.

A Search Algorithm

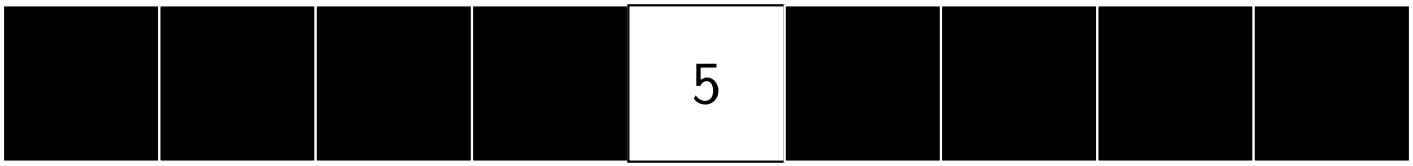
- **Input:** (1) An array of numbers and (2) number for which we're searching.
- **Output:** Whether or not the number exists in the list of numbers.
- **Algorithm:**
  1. Look at first item.
  2. Check if it's the item we're looking for.
  3. If it's the item we're looking for stop otherwise go to next item.
  4. Repeat steps 2 & 3 if there are still items in the list.

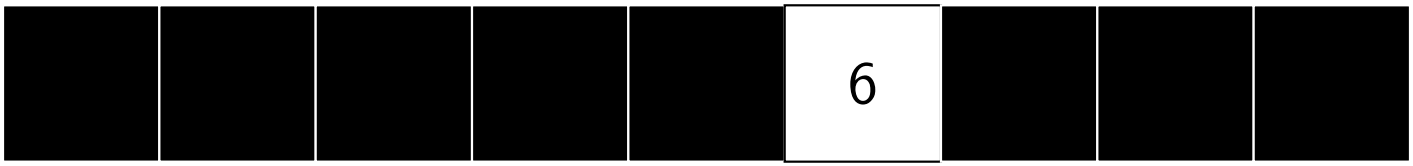

1

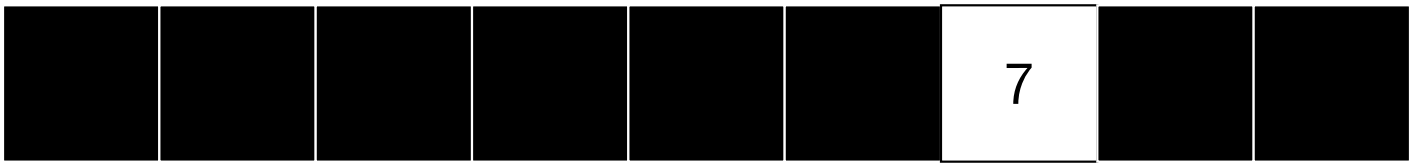

2

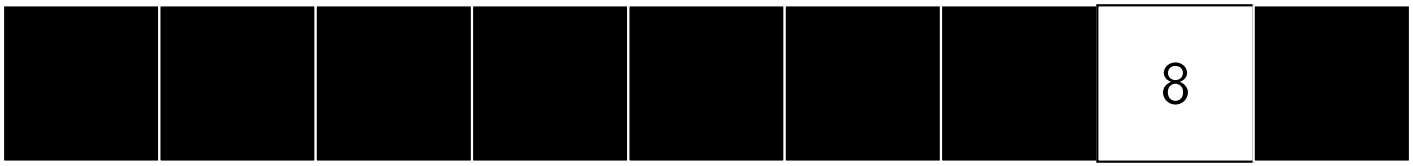

3


4


5


6


7


8

Algorithm:

1. Look at first item.
2. Check if it's the item we're looking for.
3. If it's the item we're looking for stop otherwise go to next item.
4. Repeat steps 2 & 3 if there are still items in the list.

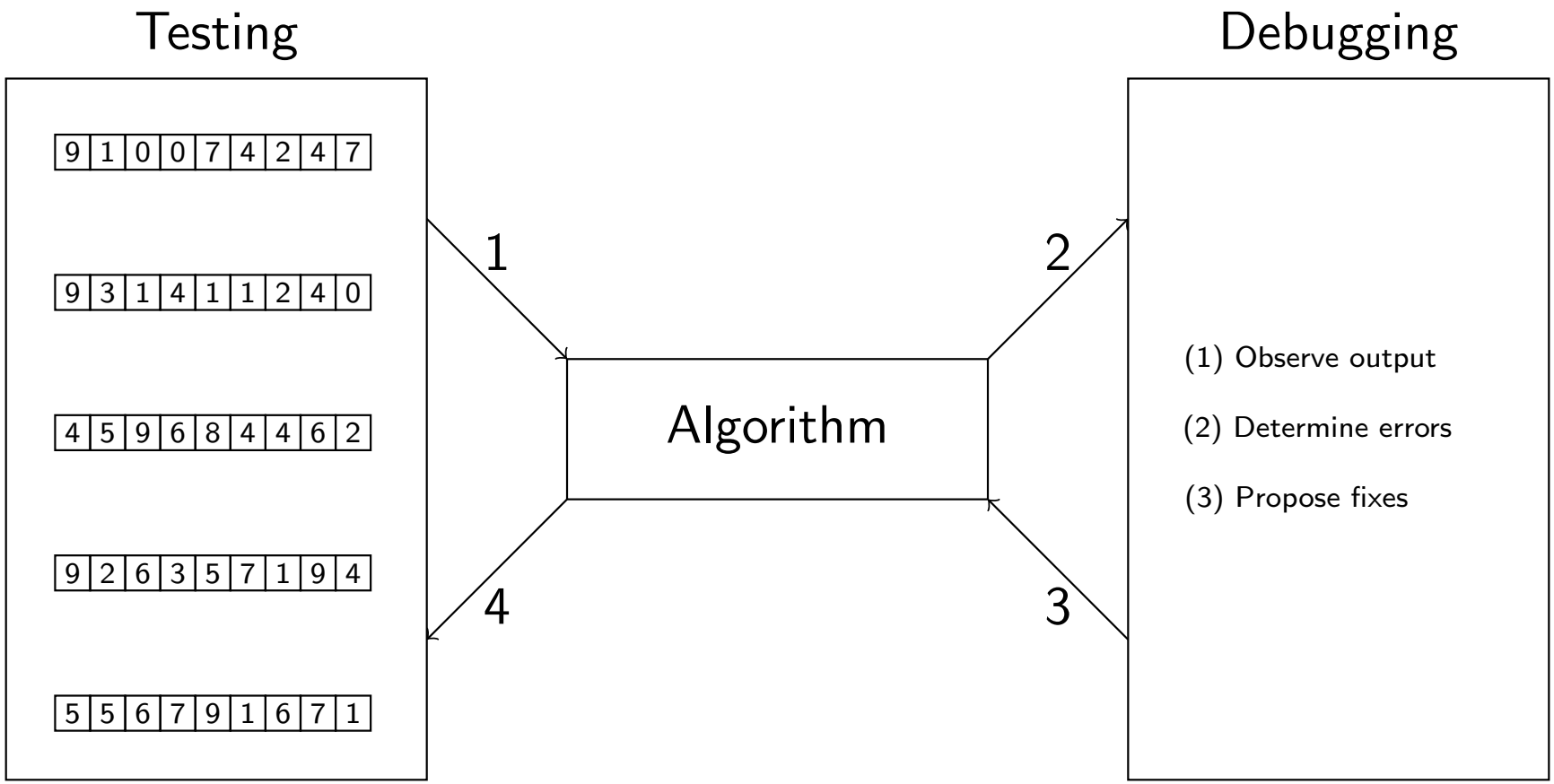
Code:

```
def search(list, searchitem):
    for item in items:
        if item == searchitem:
            return True
    return False
```

How Languages are Constructed

- **Syntax** →Rules of how programs should be constructed (think grammar).
- **Semantics** →Rules specifying what a program does.

Testing and Debugging



Next Lecture Input and Print

- **The Input Function**: A builtin function that gets a string from the user.
  - **input()** →Doesn't give a prompt.
  - **input("A test input: ")** →Will output the message *"A test input:"* to the screen and let the user type their input in after it.
- **The Print Function**: A builtin function that takes a string as a parameter (in between the parentheses) and outputs that string
  - **print("Hello, World!")** →Will output the "Hello, World!".

Next Lecture: Data Types in Python

- Two types we'll need to know now:
  - **Strings:** A long list of characters accompanied by surrounding quotes (e.g., "Hello, CS 105!").
  - **Integers:** Whole numbers, both positive and negative.
- You can check the type of a variable or expression with the `type()` function.
- You can convert between them with the `str()` and `int()` functions.
- It's important to keep the type of your variables in mind when programming.
- Keeping types in mind when attempting to deduce what a program is doing is very important.