

Lecture 24: ASCII ART Overview

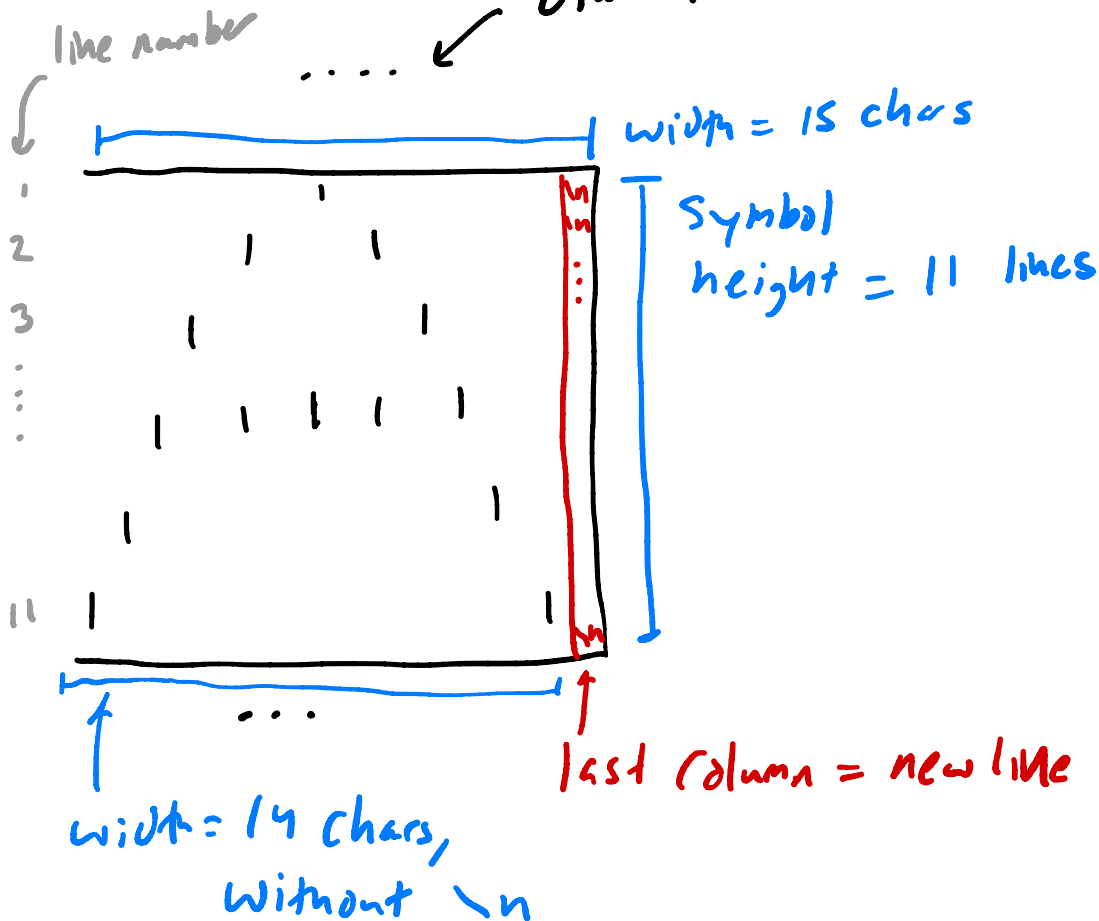
font Alphabet.txt

A B C D ... a b c d ... ? !
64 in total

alphabet: all
Symbols we can
convert to ASCII
art.

font Symbols.txt

Other Symbols



Suggested way to organize each ASCII art

Symbol: **Dictionary**

keys: single characters in ASCII font
e.g. 'A', 'B', 'C', ...

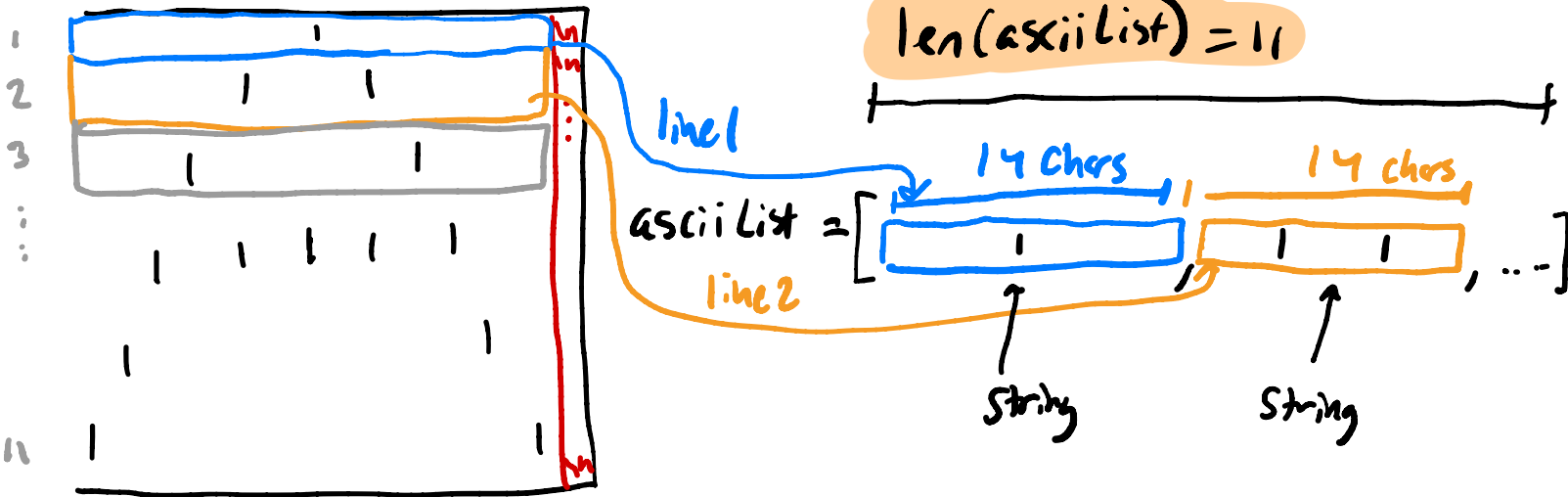
Values: List of 11 strings making up each
of the 11 lines of each symbol.

Step 1 • read in each line of ASCII symbol as a String without new line characters.

- Append 11 consecutive line strings into a list:

Each list has
11 items → all
/ Strings

$\text{len}(\text{asciiList}) = 11$



- Add the list composing the 11 line strings for each ASCII symbol to a dictionary:

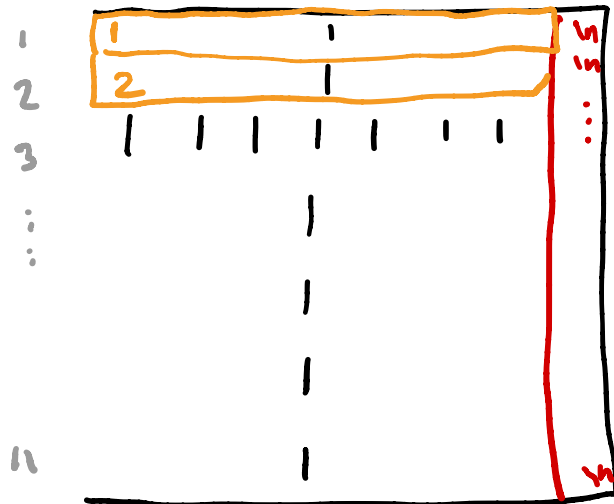
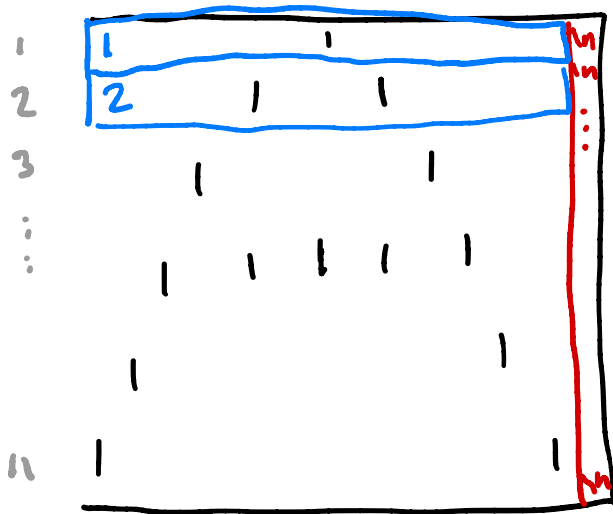
$d['A'] = [\text{14 Chars String}, \text{14 Chars String}, \text{14 Chars String}, \dots]$
 $d['B'] = [\text{14 Chars String}, \text{14 Chars String}, \text{14 Chars String}, \dots]$
 $d['C'] = [\text{14 Chars String}, \text{14 Chars String}, \text{14 Chars String}, \dots]$
 $\vdots$
 $d['!'] = [\text{14 Chars String}, \text{14 Chars String}, \text{14 Chars String}, \dots]$

Keys = # font Symbols in alphabet

$\text{len}(d.\text{keys}()) : 64 \text{ A-Z, a-z, 0-9, ?, !}$

Step 2 Concatenate rows of ASCII symbol strings representing characters of the input string one row at a time.

Example: Convert input string 'A+' to ASCII art.



String

Art = [1] + [1] + '\n' + [2] + [2] + '\n' + ...

How to build these rows of the ASCII art?

$d['A'] = [11 \text{ lines (strings) making up } A \text{ Symbol}]$,

$d['t'] = [11 \text{ lines (strings) making up } t \text{ Symbol}]$,

$$\begin{aligned} \text{art} &= d['A'][0] + d['t'][0] + '\n' \\ &+ d['A'][1] + d['t'][1] + '\n' \\ &+ \dots \\ &+ d['A'][10] + d['t'][10] + '\n' \end{aligned}$$

Lecture 25: classes

A class allows us to create our own object types.

e.g. NOT just use what Zelle, turtle have to offer — opens up tons of possibilities!

Let's say we wanted to make a type of object for a Colby Student — Student

Having Student class allows us to make objects of that type:

Our class like calling
rect = gr. Circle(gr. point(0,0), 10)

jane = Student('Jane')

bob = Student('Bob') name = 'bob'

joe = Student('joe') name = 'joe'

Constructor: Special method that returns a new object — e.g. of type Student.
e.g. jane, bob, joe
3 new objects

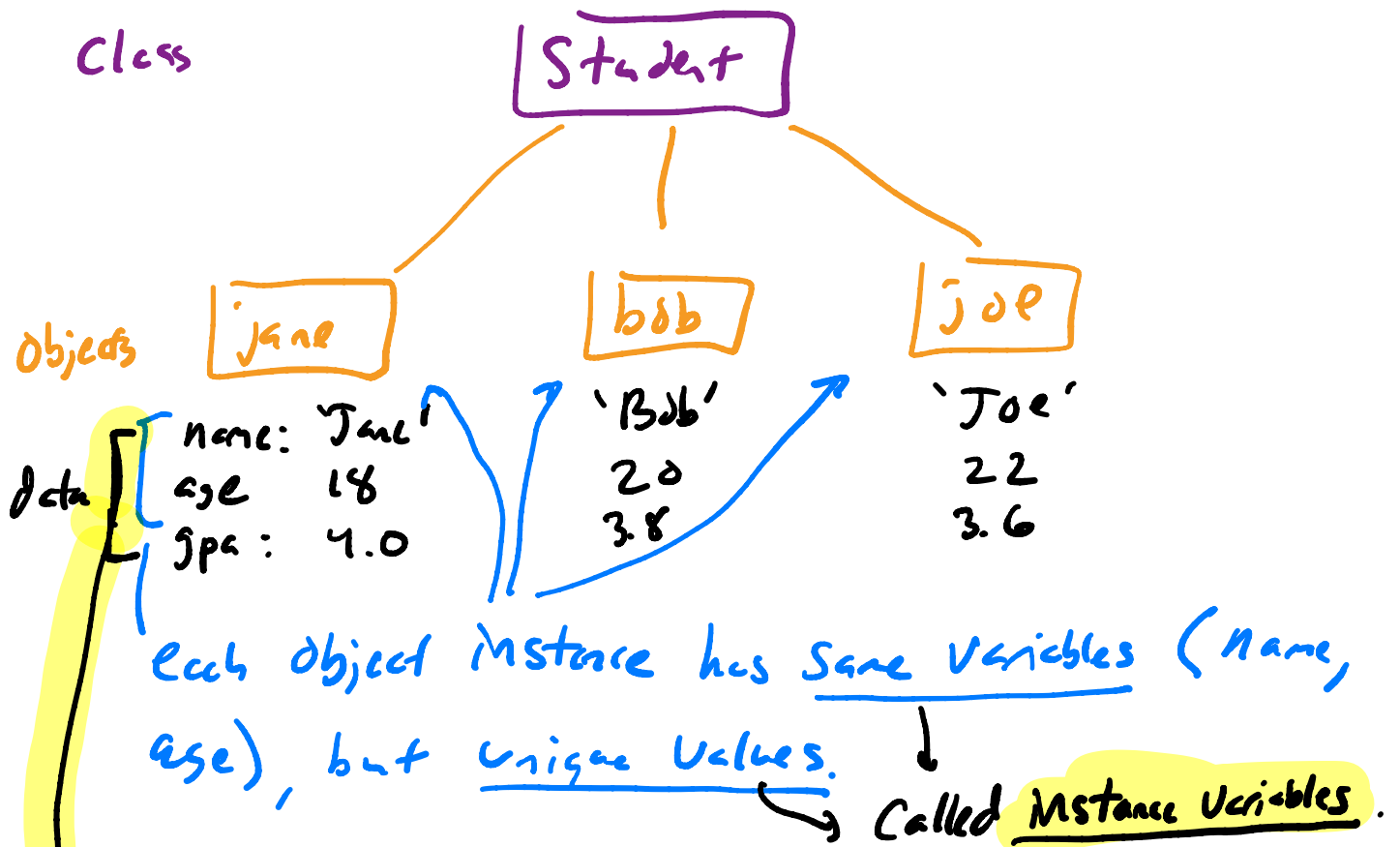
each object has unique data

• What is a good thing to create objects of?

nouns : things — e.g. Student is a thing
rectangle is a thing
turtle is a thing

Why make a class for a thing?

1) Groups data / information related to unique instances of that thing

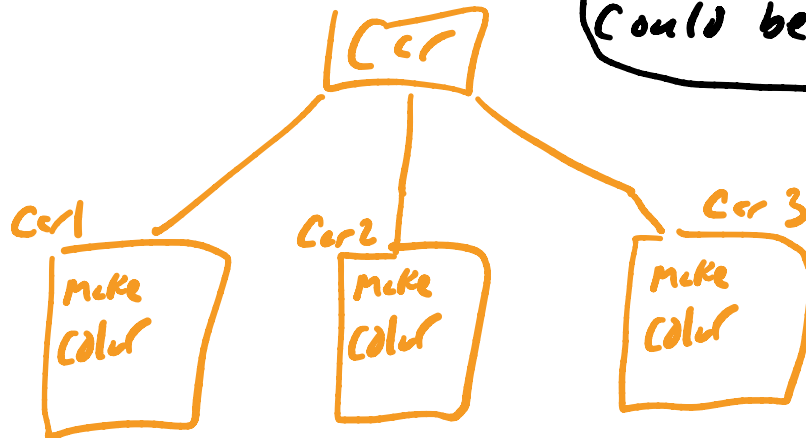


• An Instance Variable could be basic data type or an object
⇒ will be defined for any object you make of a class
e.g. any Student object will have name, age, gpa Variables.

More examples:

e.g. a Car class would probably have instance variables for: Make and Color

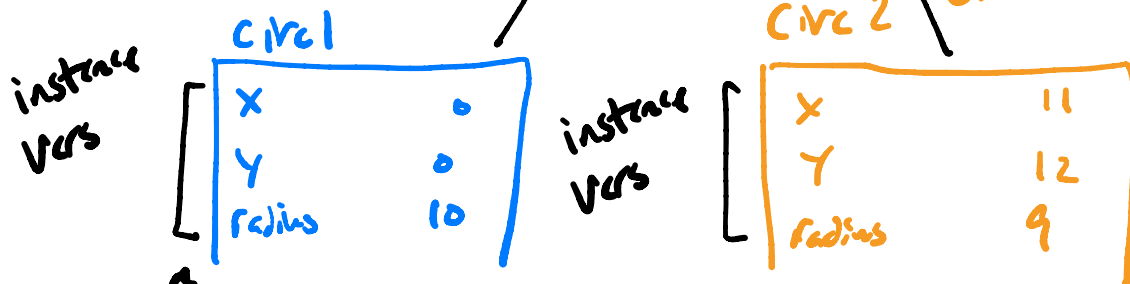
all cars have these properties
but of course specific
values across cars
could be different



- all cars have a make and color
- These 3 cars may have different specific colors and makes

Example: Circle class

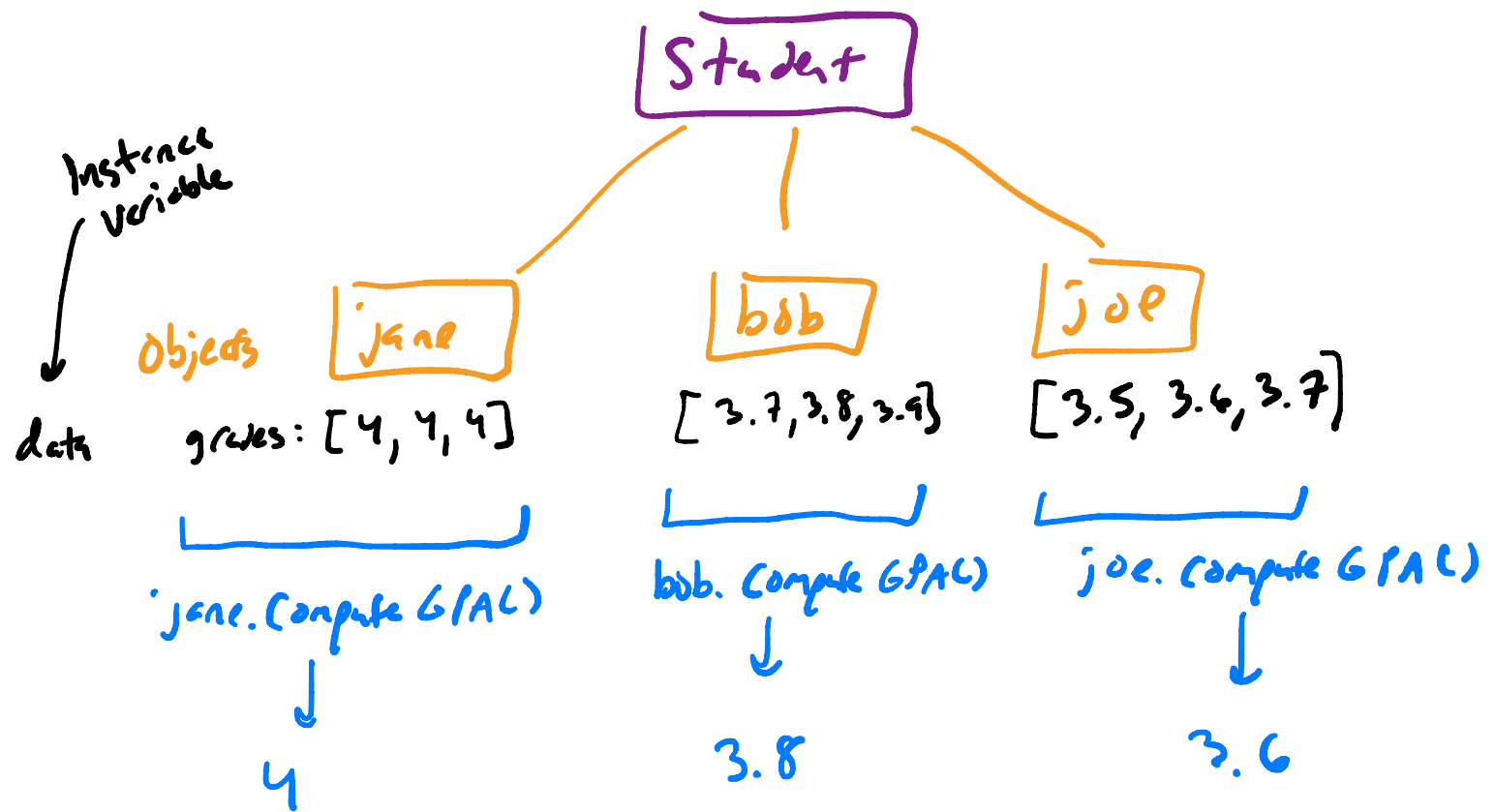
circ 1 = Circle(Point(0,0), 10)
circ 2 = Circle(Point(11,12), 9)



define what data any circle should have.

2) Classes allow us to perform actions on each instance, with its own data: **methods**

- we could add a method to our Student class to compute each student's gpa — `compute GPA()`



Common types of methods:

- Access data from each instance of a class (object)
— **get method** — method has "get" in it.

e.g. `get GPA()`, `get Name()`, `get Age()`

ask a student what his/her gpa is — `jane.get GPA()`

- Change data from an instance — **set method**
— method has "set" in it.

e.g. set Age(), set GPA(), set Name()

It's Bob's birthday, we call bob.set Age(21)
to update age 20 → 21.

- Let's write a simple class for a Student
and play around with it.
- Let's analyze the memory model for a program
involving Student class