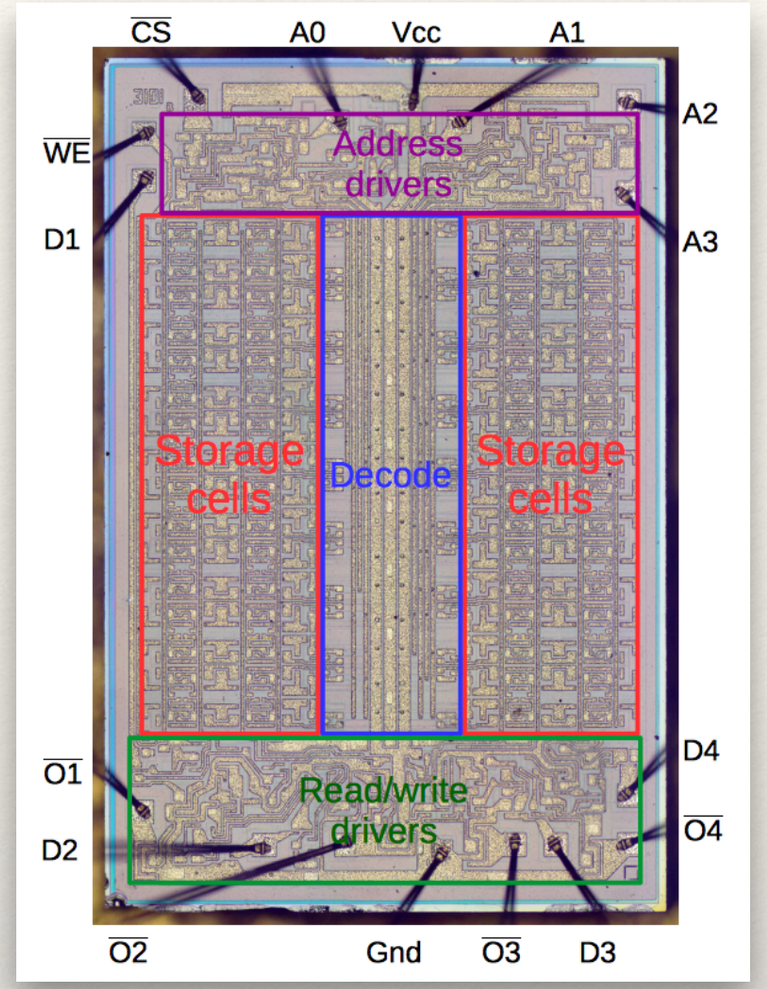
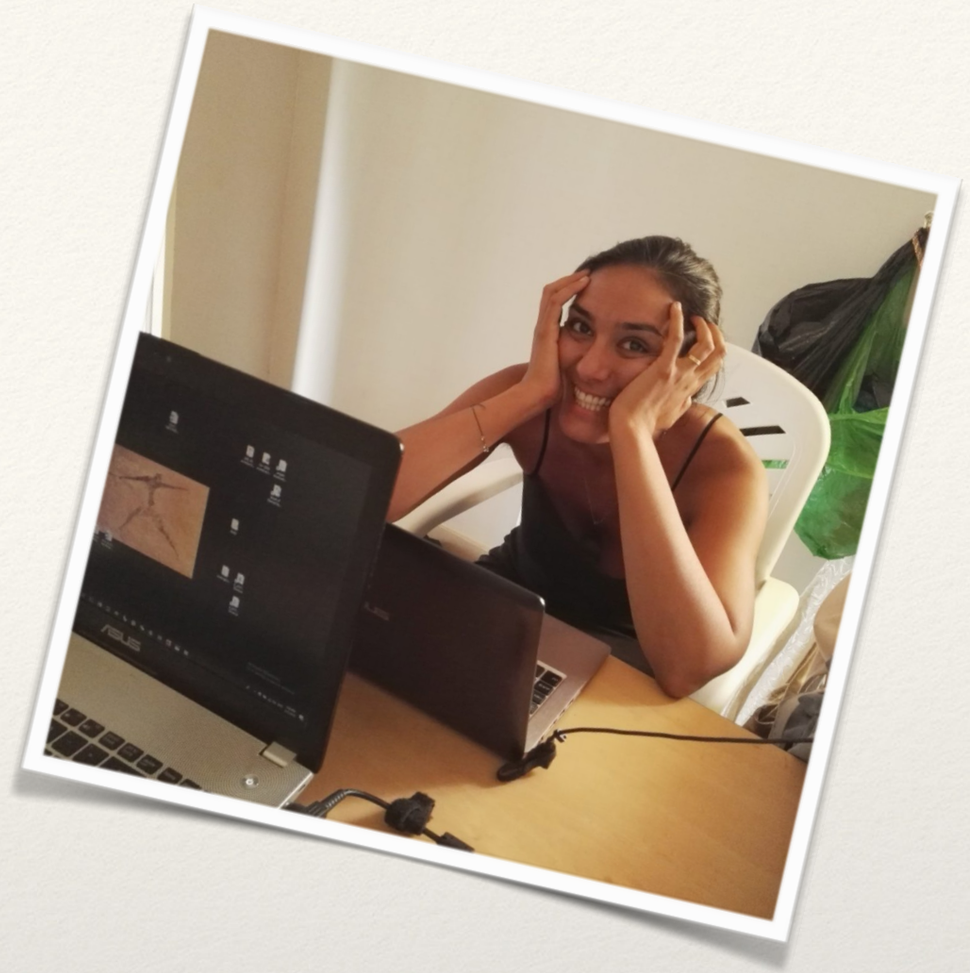




W200 Introduction to Data Science Programming

Starting out with Python

Week 02

version: Jan 6, 2020; updated Jan 16, 2020

Agenda

UC Berkeley
SCHOOL OF INFORMATION

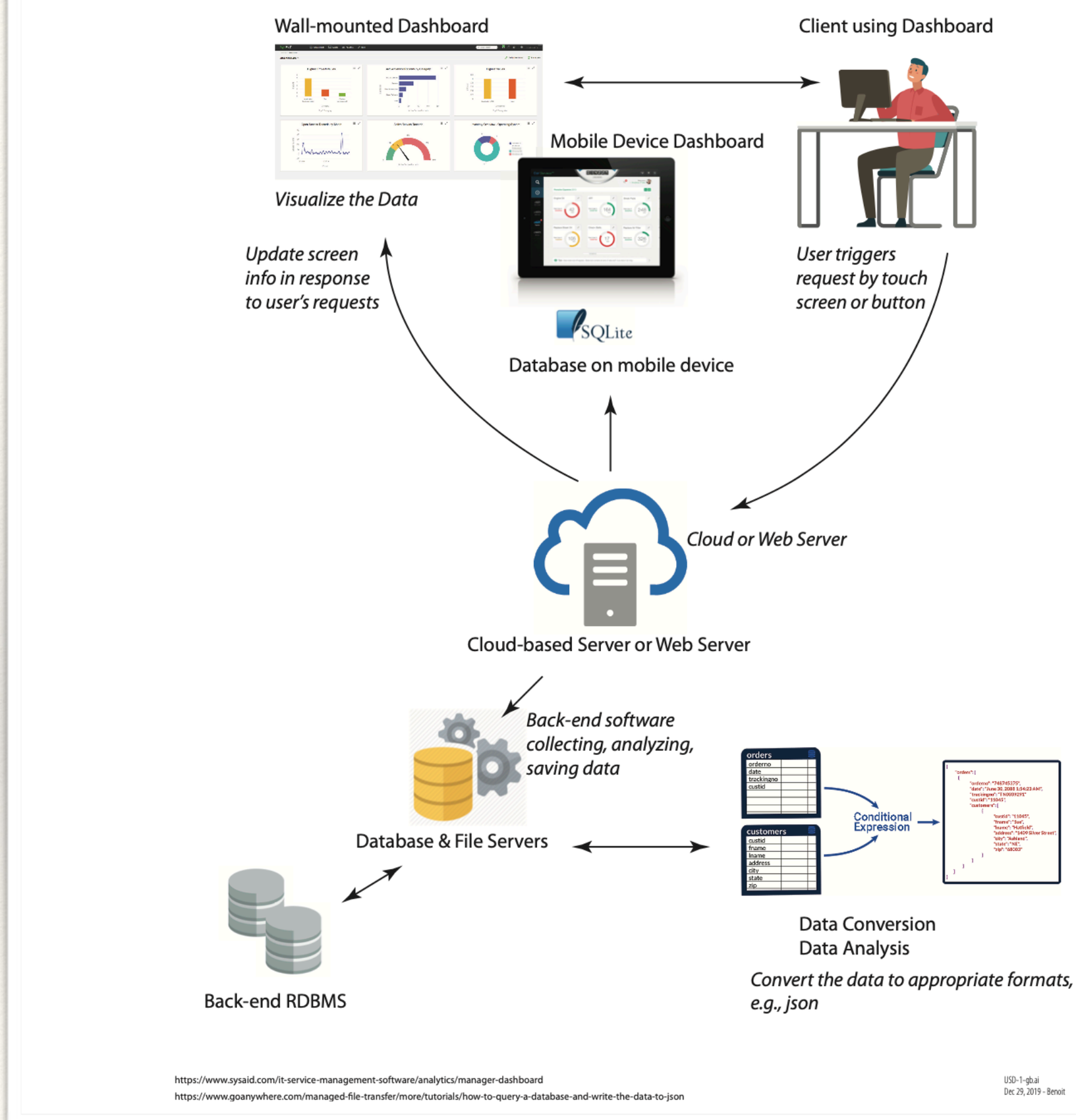
- Checking in for last week; my questions
- Running Python
- Expressions
- Objects
- String Objects
- Control of Flow
- Breakout Activities (several)

Checking in . . .

UC Berkeley
SCHOOL OF INFORMATION

- Are your environments set up okay?
  - Homework assignments are posted on GitHub … upstream on Monday evenings.
  - TA grade and provide feedback; Instructors may review; grade & provide feedback on projects.
- Make sure you're using Python 3.x (3.6 or 3.7). We ask that you don't (yet) use PyCharm or Spyder.
- GitHub ready to go? Ask Rob for help.
- Bash info: [http://hyperpolyglot.org/unix-shells]
- Resources

Myriad data sources, that we'll have to inspect, so understanding Strings, splicing, how data are read by chunks or line-by-line, here's an architecture of these sources. You'll practice string slicing next!



https://www.sysaid.com/it-service-management-software/analytics/manager-dashboard
https://www.goanywhere.com/managed-file-transfer/more/tutorials/how-to-query-a-database-and-write-the-data-to-json

USD-1-gb.ai
Dec 29, 2019 - Benoit

Where to go with questions?

UC Berkeley
SCHOOL OF INFORMATION

Near Real-Time Help:

1. Communication to instructors & other students: w200-python-2020-spring@googlegroups.com
2. Communication with instructors only: mids-python-instructors@googlegroups.com
3. Slack Channels for information communication: ucbischool.slack.com channel #w200-python

Extra "Live" Help:

1. Three instructor-led office hours each week (3-hours of personalized instructor time) See Syllabus.
2. One dedicated TA-led office hour each week (+1 more hour of personalized time); contact Mark B.

Extra "Practice" With Solutions:

1. Drills, with answers!
2. Q&A from breakouts (posted to assignments_upstream)
3. Course Syllabus, with all Async Notebooks! https://github.prod.oc.2u.com/UCB-INFO-PYTHON/Course-Syllabus
4. Homework solutions (posted within 1-week to assignments_upstream)

Other Classmates:

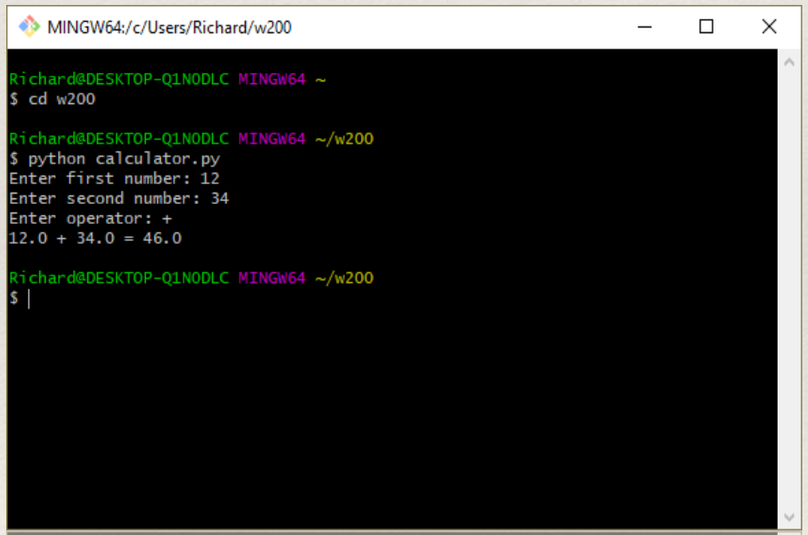
1. Introductions show students with a vast range of experience in data science, analysis, programming … just as in medicine we "see one, do one, teach one" - so it is here - share your experiences.

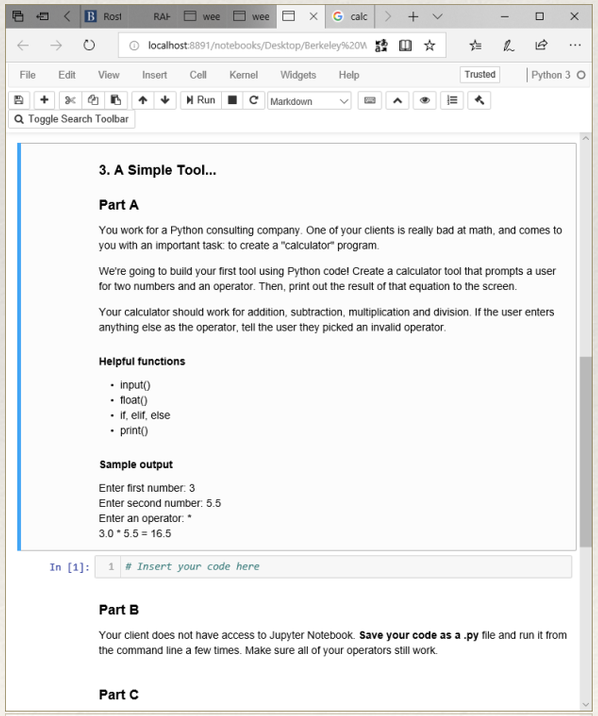
Running Python (reminder)

UC Berkeley
SCHOOL OF INFORMATION

- **Command line** Use a text editor [Atom, BBEdit, NotePad++; and save as a .py file. You *may* need to specify >python3 myfile.py if you have multiple versions installed.

- **Jupyter Notebook** App that interprets and provides error msgs. Useful for testing - be sure to reload variables and remember that instructors don't have access to your hard drive.





Check out Jupyter Notebook in class; PyCharm & Spyder websites

Expressions

UC Berkeley
SCHOOL OF INFORMATION

Assignment		`x = 5`
Addition	`+`	`x + 5`
Subtraction	`-`	`x - 5`
Division	`/`	`x / 5`
Multiply	`*`	`x * 5`
Exponent	`**`	`x ** 5`
Cumulative	`+=`	`x += 5` `x = x + 5`

Equivalency		`x == 5`
Cumulative	`-=`	`x -= 2`
Integer division	`//`	`x // 5`
Modulus	`%`	`x % 2`

Objects

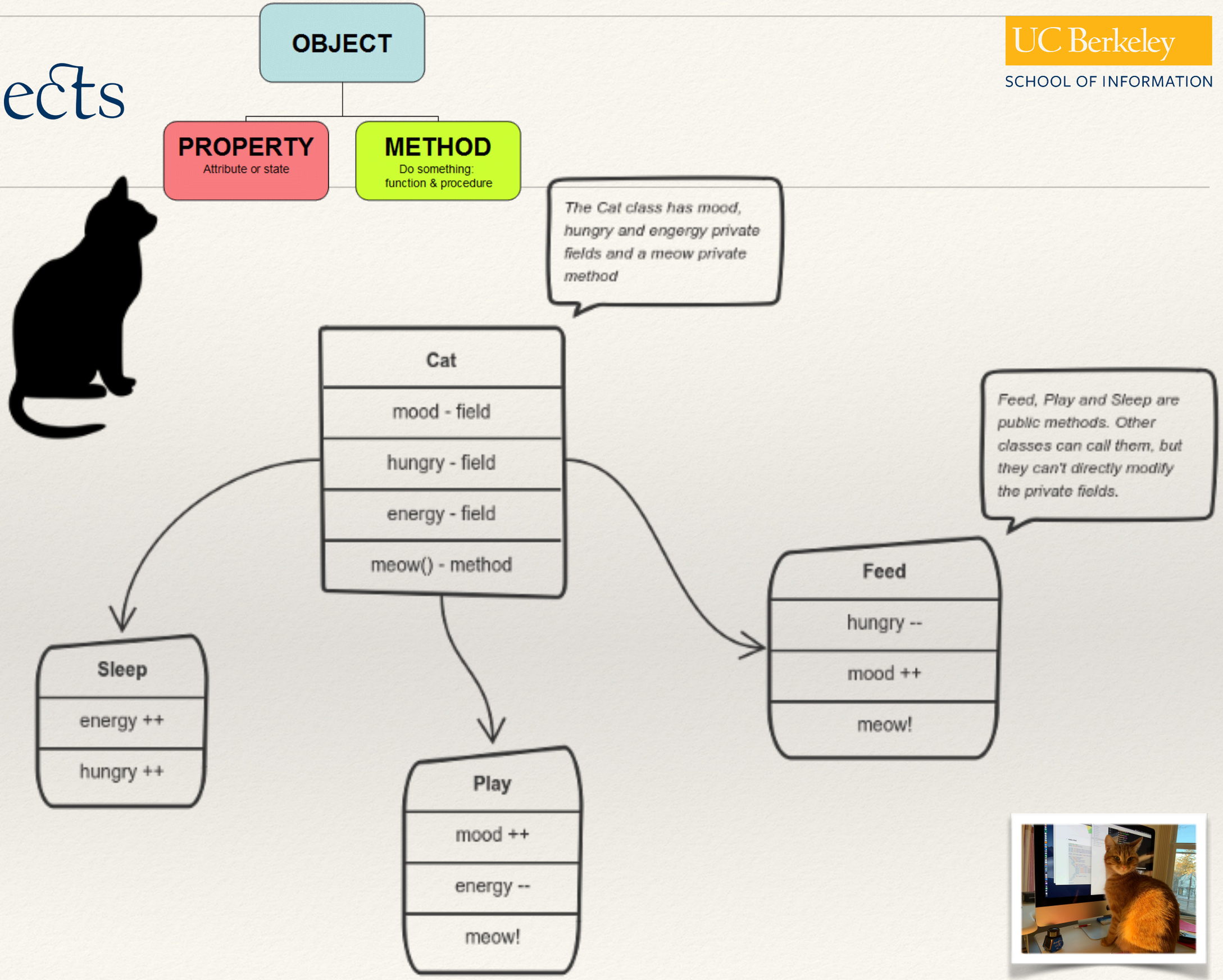
UC Berkeley
SCHOOL OF INFORMATION

- What is an "object" - a programmatic way of expressing things in the physical world* of the time you'll have to discover answers on your own (that's how it is in computing).
- For OOP languages - *the "object" is the parent, from which child objects are copied and used; these child copies are the instantiation of the Object. In coding the "Object" becomes a "Class."*
- Objects (a) contain all their variables and behaviors [methods] [encapsulation], (b) can accept multiple parameters [polymorphism], and (c) can be copied and new properties added [inheritance].
- Objects aren't necessarily linear: they can send messages to/from each other, given appropriate triggers.
- In OOP there's a hierarchy of libraries; going up the chain of a command leads ultimately to the base Object; from there we can traverse a chain back down to another type of object (hence "type-casting").

* Rather like Plato's Theory of forms, *De re publica,* Book 10.

Objects

UC Berkeley
SCHOOL OF INFORMATION



Read more about objects and the scope & visibility of variables (static, protected, private, etc.)

Define a bicycle object prototype

attributes:

- speed
- gear

behaviours:

- speed up
- apply brake
- change gear

```
class bicycle:
    ''' properties'''
    # Class variables.
    gear = 1
    speed = 0

    def __init__(self, gear, speed):
        self.gear = gear
        self.speed = speed

    def speedUp(self, increase):
        self.speed += increase

    def changeGear(self,newGear):
        self.gear = newGear

    def applyBrake(self, decrease):
        self.speed -= decrease
```

https://www.bournetocode.com/projects/AQA_A_Theory/pages/OOP.html

Objects: Types

UC Berkeley
SCHOOL OF INFORMATION

- Python lets us assign values (as all languages do) and we can test what data type we're using, e.g., `x = 5`

  → `type(x)` → returns `int`

```
>>> x = 5
>>> type(x)
<class 'int'>
>>>
```

- Boolean [`bool`, true/false; 0/1]
- Integers (`int`) counting numbers
- Floating point (`float`), e.g., 3.14159
- String (`str`), *a sequence of contiguous characters referenced by the first character's address in RAM*

Type cast: convert data from one type to another, e.g., `float(x)`

```
st = input("Enter some text:") #Default is string
fl = float(input("Enter data:")) #Convert to float
```

vari-able

name space

object space

Variables go into a special object called a “name space.” Think of “scope & visibility” of vars and how they’re accessed in the code.

var

name space

object space

`x`

x ⋯⋯⋯⋯⋯▸ 5 → *int*

`b_int`

`c_int`

`greet`

b_int → 36 → *int*

c_int

greet → print(“Bonjour”) → *function*

file

But without a namespace we can’t reach elements in the object space

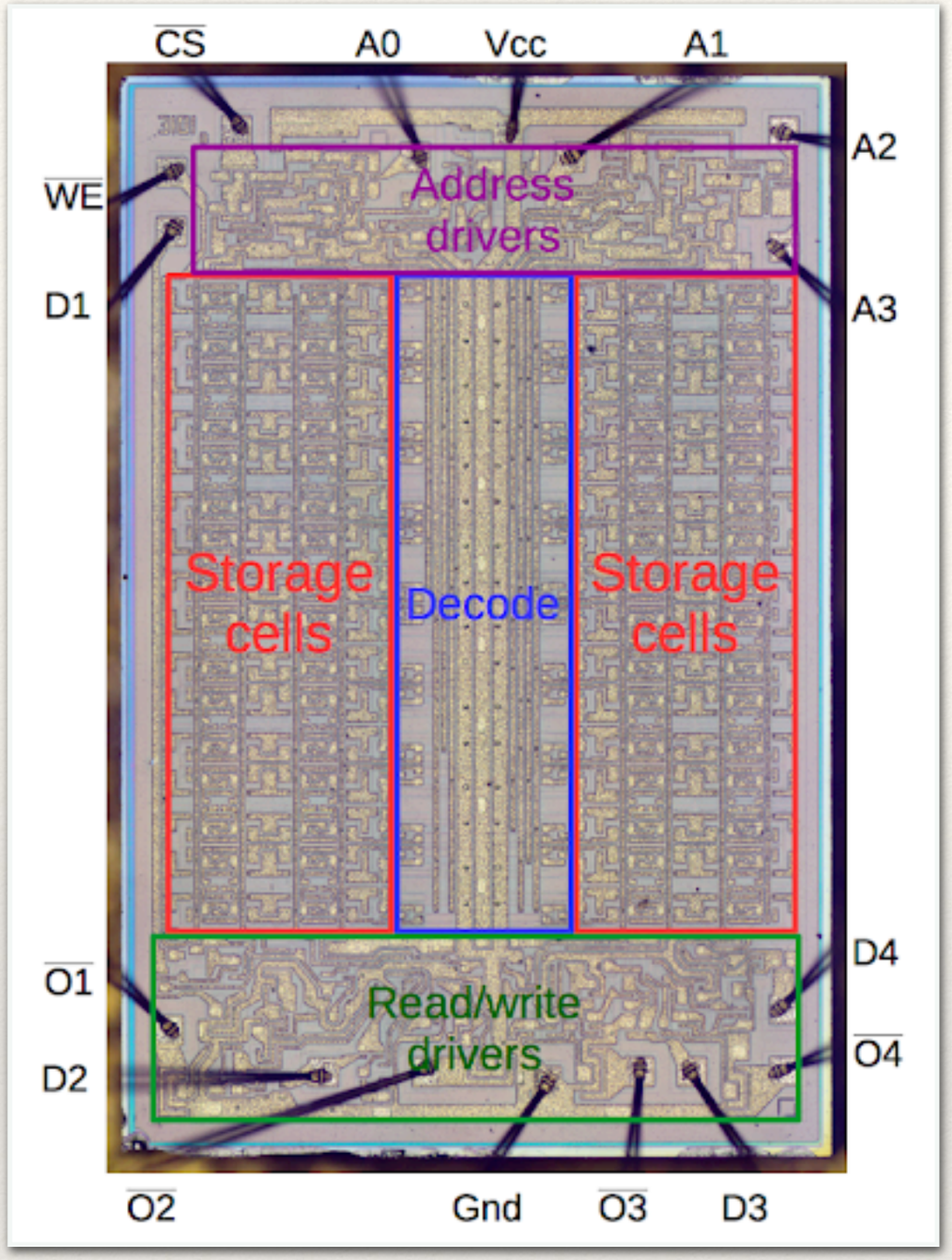
String Objects

UC Berkeley
SCHOOL OF INFORMATION

- A string is a "sequence object" [imagine the memory address of this string is 0 and is n bytes long. - Locate entire strings, individual letters, range of letters]

- Strings can be joined (or "concatenated") and manipulated:
  1. concatenation: `"cat" + "dog"` outputs "catdog"
  2. multiplication of chars: `"-"*10` outputs `----------`
  3. <*string*>`.upper()`, e.g., `x = "cat"  print(x.upper())`
  4. <*string*>`.lower()`
     1. s = input("Type here:")
     2. s_cast = input(str("Cast to string:"))

Python gains a *lot* of speed by using "pointers" and like many languages, the pointers are hidden from the programmer. [No issues of malloc and referencing points!] Notice that as an OOP language, Python often uses String methods (hence the `.` and `()`) to copy, adjust, and then replace the object in RAM, using the same variable name.



Strongly recommended: read Sipser, M., *Introduction to the theory of computation*. (3rd ed). A pdf earlier version is here.

http://www.righto.com/2017/07/inside-intels-first-product-3101-ram.html

Optional: A side note.

RAM (random access memory) looks like a "screen door." Every wire on the x-axis that crosses the wire on the y-axis creates a memory address. There are two registers: address and data.

The data are stored by being *encoded* and *decoded* (pointers and dereferencing pointers; may include casting object into other data type, often as String).

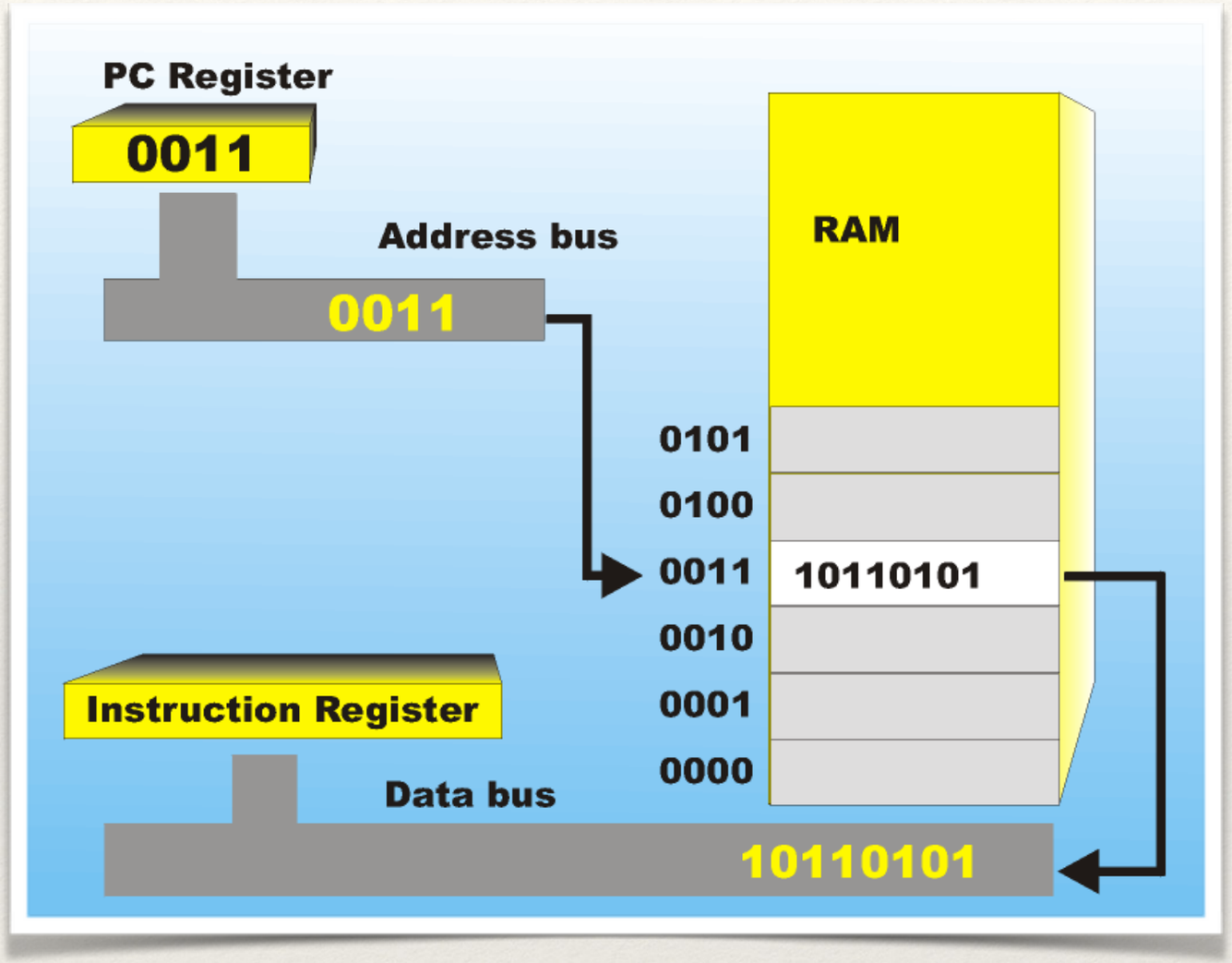
So . . . when a Python variable (object) is stored in RAM, our variable "point" to the object's memory address in RAM. A *copy* of the variable has a different name but *still points to the same memory address*. This is important.

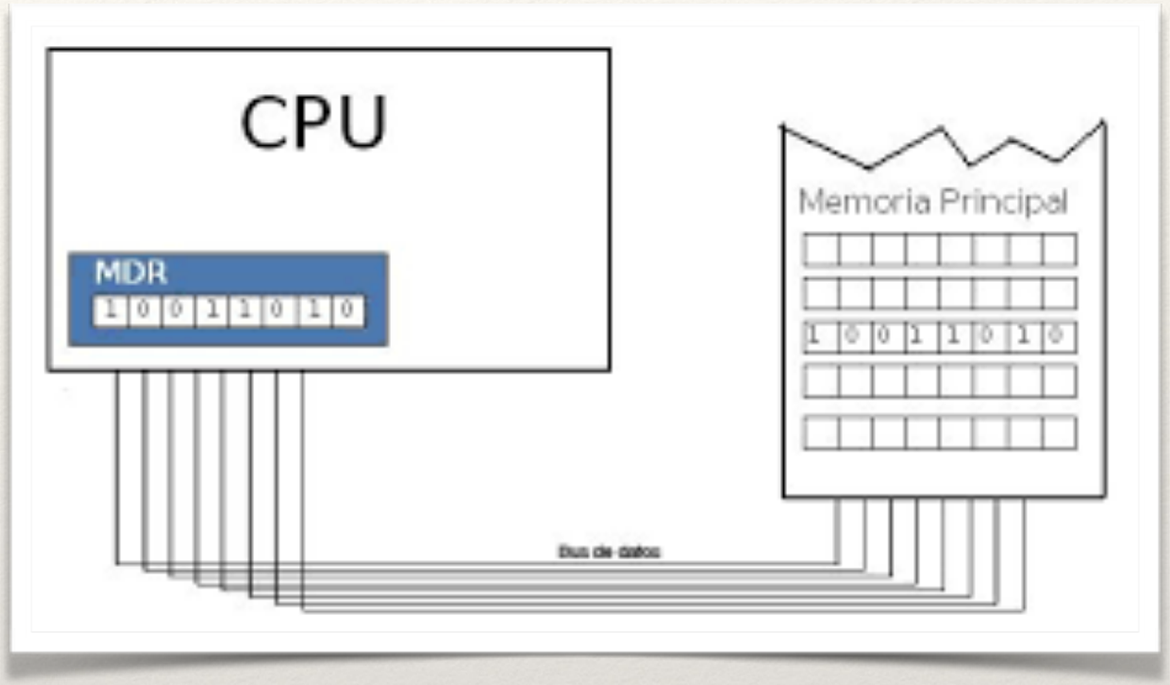
If we have `a = 5` and then `b = a`, both `a` and `b` refer (point) to the same memory address. Consequently, changing the value of "`b`" changes the value of "`a`".

This might cause a problem in your coding. We'll see this addressed in the idea of "scope and visibility" of our variables and in certain Python commands, such as `copy` versus `deepcopy`.

(But it is cool how our code is so intimate with the chip!)

CPU loads commands for itself in the instruction register; one of those commands is to store the data in the data register - but first we need the address in RAM.





This examples shows 8 bits going from memory (on the right; Memory Address Register MAR) via the “data bus” to be feed into the CPU’s registers.
MBR Memory Buffer Register; MDR Memory Address Register

String Objects

UC Berkeley
SCHOOL OF INFORMATION

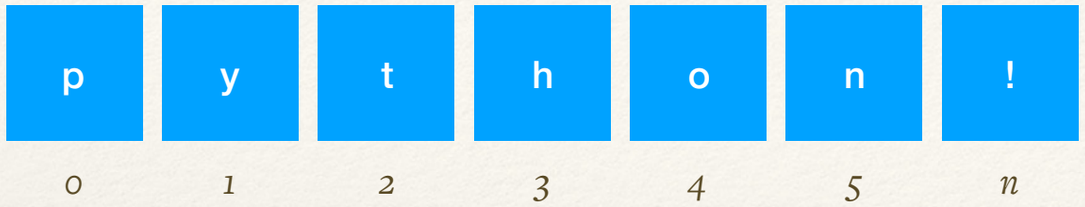
- Yikes, those darn single- and double quotes!
- Python, like other loosely-typed languages allows using single- and double-quotes around Strings. Keep in mind that often when concatenating data we use a mix of " and ' . . . the use of these *must* be balanced, else a hard to find error.
- Comments in python code use *three* ", e.g., """
- \ means an "escape sequence", note: `\n, \r\n, \t`

```
' '       # single quotes (the dagger kind)
"  "      # double quotes (notice the pairing)

t = input("Welcome, you're here! ")
          # double quotes surrounding string
          # so the single quote is safe
s = input('Welcome, you\'re here!')
          # single quotes, so middle one needs
          # to be "escaped"
```

String Objects: Slicing

UC Berkeley
SCHOOL OF INFORMATION



`[ 0 ]`	from the start of the string
`[ -1 ]`	one letter from the end of the string
`[ 0:3 ]`	from the start of the string to the 4th letter
`[ 1:-1 ]`	from 1 to the next to last letter
`[ 1:5:2 ]`	from 1 to 5, by 2 *at a time*
`[ :-1 ]`	beginning (:) to second to last letter (*n* - 1)
`[ : ]`	entire string
`[ ::-1 ]`	entire string, reversed

"Slicing" is surprisingly important in data sci and processing of data, during "data cleansing."

We rely in splicing to test data before ingesting, to test data for certain conditions . . .

Breakout Activities:

1) Tribbles, 2) Spiders, and 3) Calculator

UC Berkeley
SCHOOL OF INFORMATION

1. Fire up what you downloaded for week 2 in Jupyter and create a string variable that prints exactly

```
    The "trouble with
 Tribbles" is that they
\\\EAT/// too many MREs.
```

2. Using one line of Python code, slide your variable from part 1 to print "selbbirT" 300 times.

```
selbbirT selbbirT selbbirT …
```

Okay, we're nerds who like StarTrek! And we'll drop into your breakout rooms for Q&A.

UC Berkeley
SCHOOL OF INFORMATION

Breakout Activities: Tribbles & Spiders

```
Do you have 8 legs?no
Do you have 4 legs?no
You are a bicycle
```

https://www.youtube.com/watch?v=DVAkVwy4TD4

Breakout Activity: calculator

UC Berkeley
SCHOOL OF INFORMATION

Make a calculator - the purpose is to practice conditions (“if” statements) and saving your work as a .py and executing it.

```
Enter first number: 7
Enter second number: 4
Enter an operator: %
7.0 % 4.0 = 3.0
```

```
i = 0
j = 1
if (i < 12):
    if (j < 60):
        j++
    else
        i++
        j = 1
```

`if` *statements*
Nested if
    [No switch statement!]
`while / for` *loops*
`continue / break`
Comprehensions

https://www.clayallsopp.com/posts/definition-of-design-terms/

That's it!

UC Berkeley
SCHOOL OF INFORMATION

- That's it! Enjoy a great week and keep up with your drills and homework assignments.