Genome Sciences 373: Genome Informatics

Quiz section #1

March 29, 2018

About me

Email: hpliner@uw.edu

Office hours: Thursday right after quiz section (2:20pm) Foege S110

Or by appointment

Other help: I have (I think) enabled discussion on Canvas, you can post questions – if urgent, better to email

My research focuses on designing algorithms for analyzing single-cell genomic data (mostly in R)

Quiz section goals

1. Solidify in-class material
   (if there is something from class you don't understand, let me know)
2. Develop understanding of programming concepts
3. Learn basic Python to write bioinformatics programs

Attendance is not required, but the material covered is required

Homework policies

- No late homework accepted without **prior** arrangements
- Grading is equally about **effort** and **execution**
- Homeworks are assigned on Friday and due the following Friday before lecture (1:30pm)
- First homework will be available tomorrow by 1:30pm on **Canvas**

Homework policies - programming

- Group work/internet searching: You can (and should) use them, **but don't copy.** We can tell!
- The point is to **learn**

- The course is taught in Python 2. Please submit your homework in Python 2 (see me with issues)
- Don't use Python modules that are obvious "shortcuts" to the homework - rule of thumb: don't use any modules we haven't discussed in class

Questions about course logistics?

Today's goals

1. Quick review of alignment

2. Algorithms and programs, what and why?

3. Getting started programming in Python

```
G - A A T T C A G T T A
|     |   | |   |     |
G G - A - T C - G - - A
```

What is an alignment?

- Arrangement of nucleotide (or amino acid) sequences, to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

```
G – A A T T C A G T T A
|     |   | |   |     |
G G – A – T C – G – – A
```

What are some reasons to align sequences?

How would you score this?

Gap = -4

```
C G G A A T C C
| |   | |   | |
C G - A - T T C
```

Substitution matrix

	A	C	G	T
A	10	-5	0	-5
C	-5	10	-5	0
G	0	-5	10	-5
T	-5	0	-5	10

Algorithms and Programming

Programming

This class is designed for you to **understand** and **use** bioinformatics algorithms

You won't learn to implement all of them, but understanding them requires programmatic thinking

Plus, if you do want to implement an algorithm or otherwise code anything, you will be off to a good start!

A few notes on programming in this class

If you have previous programming experience:

- This may be trivial, but be sure you understand how to program well enough *in Python* to understand class topics and complete homework and tests

If you have no previous programming experience:

- We will start at the beginning, but 1) most people learn programming best by doing and 2) we will move quickly
- So I recommend spending a bit of time doing some interactive exercises online: examples, codecademy.org, coursera.org, interactivepython.org (watch out for python2 versus python3)

What is an algorithm?

- A set of detailed instructions that solve a problem/accomplish a task

Example task:

Cook pasta

Algorithm:

```
1. Fill pot with water
2. Put on stove
3. Turn stove to high
4. While water is not boiling:
    - Wait
5. Add pasta to water and stir
6. Wait 8 minutes
```

What is an algorithm?

- A set of detailed instructions that solve a problem/accomplish a task

Example task:

Given three numbers, find the largest

Algorithm:

Properties of algorithms

- Unambiguously defined series of steps
- Works for all inputs of a defined set
- Is guaranteed to produce a correct result for those inputs

Often written in “pseudocode”

Pseudocode

Example task:

Given three numbers, find the largest

Pseudocode:

```
Input: three numbers, A, B, and C
Output: the largest number

current_largest = A
if B > current_largest:
     current_largest = B
if C > current_largest:
     current_largest = C
return current_largest
```

Pseudocode – try it!

Example task:

Given three numbers, find the largest

Pseudocode:

```
Input: three numbers, A=7, B=3, and C=9
Output: the largest number

current_largest = A
if B > current_largest:
      current_largest = B
if C > current_largest:
      current_largest = C
return current_largest
```

What is a program?

A series of instructions that performs a specific task when executed by a computer

The translation of steps from pseudocode or English to language a computer can understand

Get python going on your computer

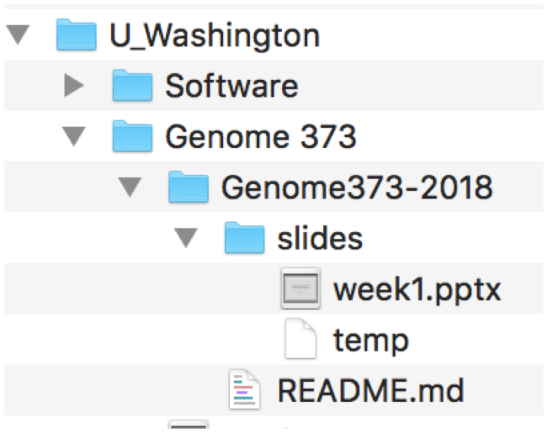
- Open the terminal (Macs) or command prompt (Windows)

- Open a text file, copy below, and save as ‘myfirstprogram.py’

```
print ‘hello world’
```

- In the terminal/command prompt, navigate to where you saved your program and type python myfirstprogram.py press enter

Navigating in terminal/command prompt



	Mac/Unix	Windows	Example output if I start in "U_Washington"
See where you are	pwd	cd	/Users/hpliner/Documents/U_Washington
See what's in your current directory	ls	dir	Software Genome 373
Move into a directory inside your current directory	cd Genome\ 373	cd Genome\ 373	No output, now in /Users/hpliner/Documents/U_Washington/Genome 373
Move out one level	cd ..	cd ..	No output, now back in U_Washington

Python can be used interactively

From the terminal/command prompt, type python

```
>>> print 'hello world'
```

Python can be used interactively

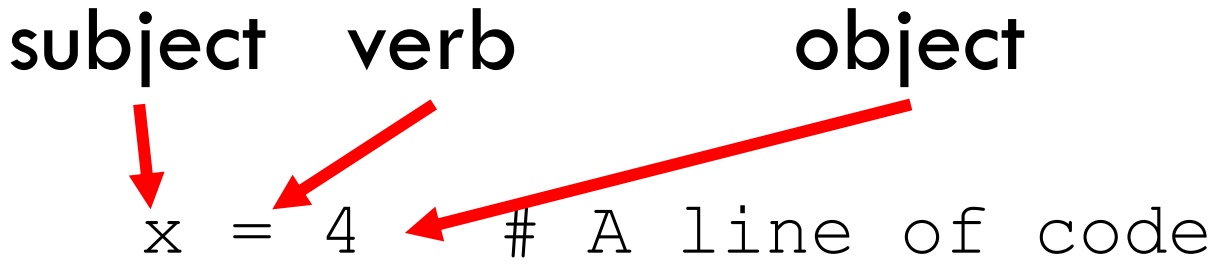
From the terminal/command prompt, type python

```
>>> print 'hello world'
```

One of the main (and possibly only differences we will come across) differences between Python 2 and Python 3 is the print statement. In Python 3, you need parentheses:

```
print ('hello world')
```

Variables and operators



Variables and operators

An operator is a 'verb'

= is Python's assignment operator, it assigns what's on its right to what's on its left

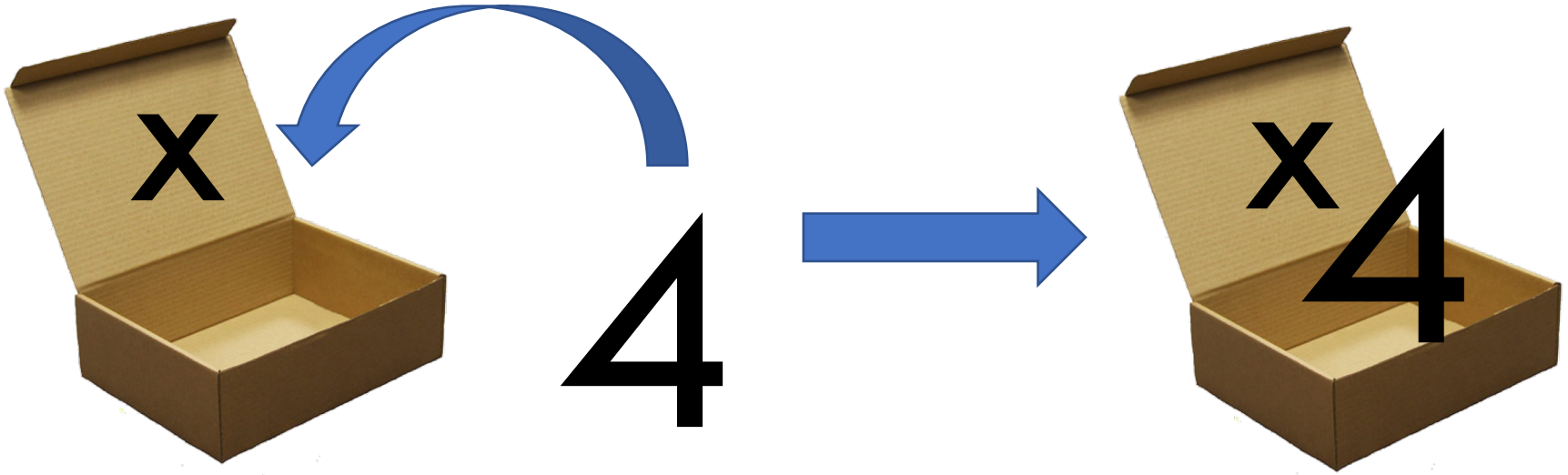
Variables and operators

subject verb object

```
x = 4   # A line of code
```

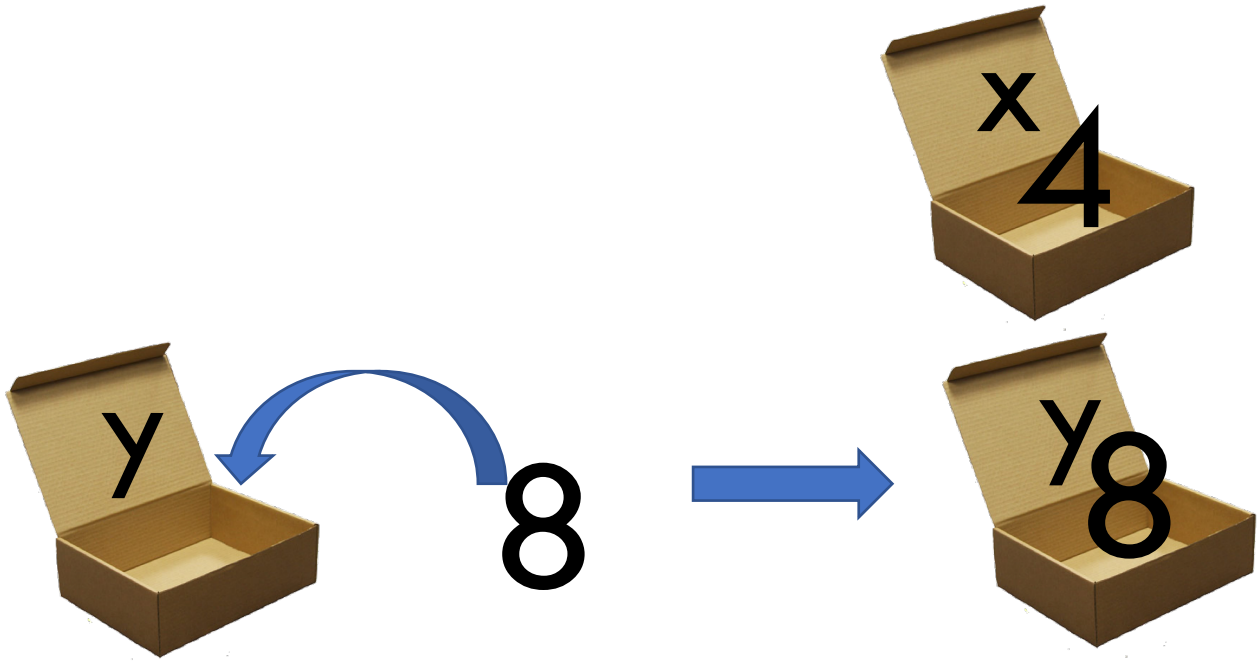
An operator is a 'verb'

= is Python's assignment operator, it assigns what's on its right to what's on its left



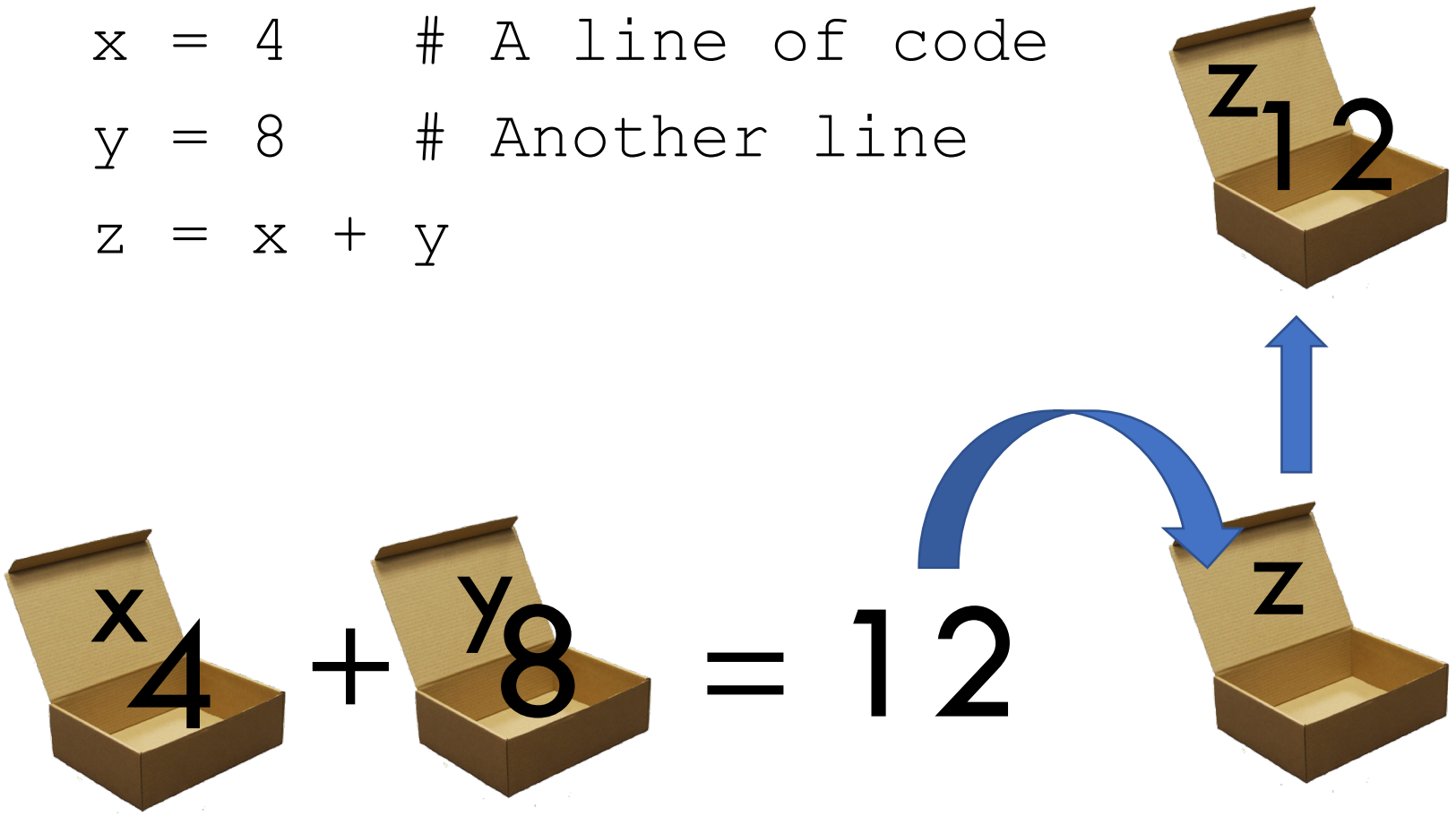
Variables and operators

```
x = 4    # A line of code
y = 8    # Another line
```



Variables and operators

```
x = 4    # A line of code
y = 8    # Another line
z = x + y
```



Comments

```
x = 4    # A line of code
```

In Python, the comment character is #

Anything after #, the computer ignores – it's for the humans

Comments are important! Why?

Variables

Variables don’t have to be digits, they can be:

- characters (like a single letter)
- strings (a series of characters)
- integers
- booleans (True or False)
- floats (numbers with decimal points)

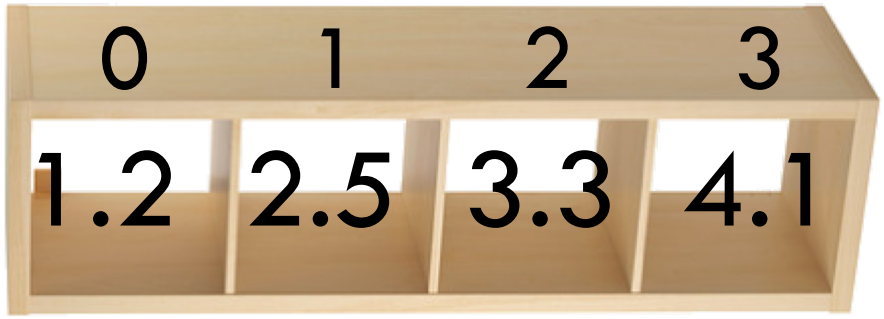
All of these are valid assignments:

```
y = "hannah" #string
Hannah = 3.4 #float
g = True #bool
x = 'h' #char

#What will these do?
print Hannah

print "Hannah"
```

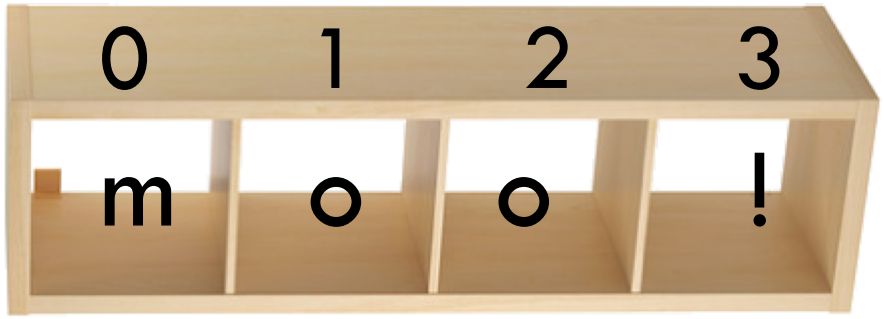
A list can be used to store lots of variables at once



Like a bookshelf, where each item is accessed by its place in the order

```
hp_list = [1.2, 2.5, 3.3, 4.1]
print hp_list[0]
print hp_list[3]
print hp_list[4]  # ?
```

A string is like a list of characters



Like a bookshelf, where each item is accessed by its place in the order

```
moo_list = "moo!"
print moo_list[0]
print moo_list[3]
print moo_list[4]   # ?
```

A hash/dictionary is used to store data accessible by name

```
new_dict = {"zach":30, "pam":64,
"sam":28}


new_dict["zach"]
```

Other operators (Boolean)

Asking True False questions

```
x = 4    # Assignment, not Boolean operator!!


x == 4
x != 4
x > 4
x >=4
x == 4 and x != 3
x != 4 or x != 2
```

Flow control: if, else, elif

Flow control determines which lines the computer looks at in your code. So far, it's been everything.

```
x = 4
if x == 4:
    print "it's a 4"
else:
    print "it's not a 4"
```

Flow control: if, else, elif

```
x = 4
if x == 4:
    print "it's a 4"
elif x == 5:
    print "it's a 5"
else:
    print "it's not a 4 or a 5"
```

Flow control: if, else, elif

```
x = 4
if x == 4:
    print "it's a 4"
elif x == 5:
    print "it's a 5"
else:
    print "it's not a 4 or a 5"
```

- Homework will be posted tomorrow on Canvas