Quick notes

- Check Piazza
- Reminders:
  - Assign3 is due friday
- Autograder
  - If you get a high score, that'll be your grade
  - If you don't, I'll look at your code + lab log
- Assignment 4 won't be as difficult
  - Python and diff/patch
  - Due one week after assign3
  - We'll talk about this stuff today/wednesday
- Questions?

Feedback / Office Hours

- Tameez Latib
  - tameezlatib@gmail.com, please add “CS35L” to the subject line
  - Office Hours: Monday **4pm-6pm** (or by appointment)
  - Feedback: https://forms.gle/6kcJ2aJtzAzFMhHQ7 (anonymous google form)

Python

- Interpreted language
- Easy to read/code/etc
  - Beginner friendly
- Numerous libraries
  - used extensively in machine learning
- Don’t need to declare variables
- Whitespace/indentation important, no semicolons
- “And”, “or”, “not” instead of && || !

Python

- If syntax
- Use tabs/whitespace!
- Note the “:” \

```
1   condition = True
2   number = 5
3   string = "this is an if statement"
4   if condition:
5       print(string)
6
7   if number < 0:
8       print("your negative number is: ", number)
9   elif number > 0:
10      print("your positive number is: ", number)
11  else:
12      print("your number is 0")
```

Python

- For syntax
- Similar to bash
- Letter in word:
  - Word string interpreted as array of characters
- len(word)
  - Gets the length
- range(num)
  - Returns array (0, 1, 2, ... num-1)

```
1  for i in range(10):
2      print("num: {}".format(i))
3
4  word = "python"
5  for letter in word:
6      print(letter)
7
8  for i in range(len(word)):
9      print(word[i])
```

Python

- Arrays of anything

```
1
2   array = ['cat', 'dog', 'elephant', 1, 2, range(5)]
3   #this is a comment
4   print(array)
5   for thing in array:
6       #if thing is a string, print it
7       if (isinstance(thing, str)):
8           print(thing)
9       #if thing is a number, tell us
10      elif type(thing) == int:
11          print("num: ", thing)
12      #Otherwise thing is an array, print every member
13      else:
14          print("Members of array: ")
15          for i in thing:
16              print(i)
```

Python examples

```
2   array = ['cat', 'bark', 'elephant']
3   array[1] = 'woof'
4   print(array)
5   array[-1] = 'mouse'
6   print(array)
```

```
['cat', 'woof', 'elephant']
['cat', 'woof', 'mouse']
```

```
2   array = ['cat', 'bark', 'elephant']
3   array.append('woof')
4   print(array)
5   array.sort()
6   print(array)
```

```
['cat', 'bark', 'elephant', 'woof']
['bark', 'cat', 'elephant', 'woof']
```

```
1   #find greatest number in array
2   def greatest_num(arr):
3       current_largest = arr[0]
4       for el in arr:
5           current_largest = (el > current_largest)*el + (el < current_largest)*current_largest
6       return current_largest
7
8   arr = [5,1,6,2,3,9]
9   print(greatest_num(arr))
```

True*num = num
False*num = 0

Python examples

```python
num = 4
arr = [1,5,2,3, 4]
if num in arr:
    print("{} is in your array {}".format(num, arr))
```

```python
substr = "water"
sentence = "he drinks water."
if substr in sentence:
    print("{} is in {}".format(substr, sentence))
```

```python
sentence = "he drinks water."
print(sentence[3:9])
```

Python classes

```
1  #rectangle has spatial x,y (bottom left corner)
2  # length (bottom left to bottom right)
3  # height (bottom left to top left)
4  class Rectangle:
5      def __init__(self, length, height, x, y):
6          self.length = length
7          self.height  = height
8          self.x      = x
9          self.y      = y
10
11     def expand(self):
12         self.length *= 2
13         self.height *= 2
14
15     def __str__(self):
16         return "x,y,h,l: {},{},{},{}".format(self.x, self.y, self.height, self.length)
17
18 shape = Rectangle(5,1,0,0)
19 shape.expand()
20 print(shape)
```

init function
str overrides print
Object oriented, object has methods
Self is first argument

Python classes

```python
class Calculator:
    history = []

    #Cannot use self or modify instance variables, but can use class variables
    @classmethod
    def add_store(cls, n1, n2):
        ret = n1+n2
        cls.history.append(ret)
        return ret

    @classmethod
    def print_history(cls):
        print(cls.history)

    #does not have access to self or cls
    @staticmethod
    def add(n1, n2):
        ret = n1+n2
        return ret

calc1 = Calculator()
calc2 = Calculator()
print(calc1.add(5, 109))
print(calc1.add_store(5, 9))
print(calc2.add_store(15, 0))
calc1.print_history()
```

Regular methods/fns
    Access class and self vars

Class methods
    Only class variables

Static methods
    None

Outputs
114
14
15
[14, 15]

Python classes

```python
class Calculator:
    history = []

    #Cannot use self or modify instance variables, but can use class variables
    @classmethod
    def add_store(cls, n1, n2):
        ret = n1+n2
        cls.history.append(ret)
        return ret

    @classmethod
    def print_history(cls):
        print(cls.history)

    #does not have access to self or cls
    @staticmethod
    def add(n1, n2):
        ret = n1+n2
        return ret

calc1 = Calculator()
calc2 = Calculator()
print(calc1.add(5, 109))
print(calc1.add_store(5, 9))
print(calc2.add_store(15, 0))
calc1.print_history()
```

Regular methods/fns
    Access class and self vars

Class methods
    Only class variables

Static methods
    None

Outputs
114
14
15
[14, 15]

Python import

Import modules/libraries

python test.py --n 1 --m 5

```
1  import argparse
2
3  parser = argparse.ArgumentParser()
4
5  parser.add_argument('--n', default=0)
6  parser.add_argument('--m', default=0)
7
8  args = parser.parse_args()
9  print("n, ", args.n, " m, ", args.m)
```

Python deep vs shallow copy

Copy: b is new object, but a, b reference the same [3,4]
Deepcopy: c is a completely new object, c has a new [3,4]

```
1   from copy import copy, deepcopy
2   a = [1, 2, [3, 4]]
3   b = copy(a)
4   c = deepcopy(a)
5   b[2][0] = 40
6   print(a)
7   print(b)
8   print(c)
```

```
[1, 2, [40, 4]]
[1, 2, [40, 4]]
[1, 2, [3, 4]]
```

Python try / except

```
1   a = "100"
2   try:
3       b = a/100
4   except:
5       print("error! a not a number?")
```

Getting started with hw

- Use argparse
- Print statements to debug (or if you want to research the python debugger)
- Try out shuf command
- Check hint slide