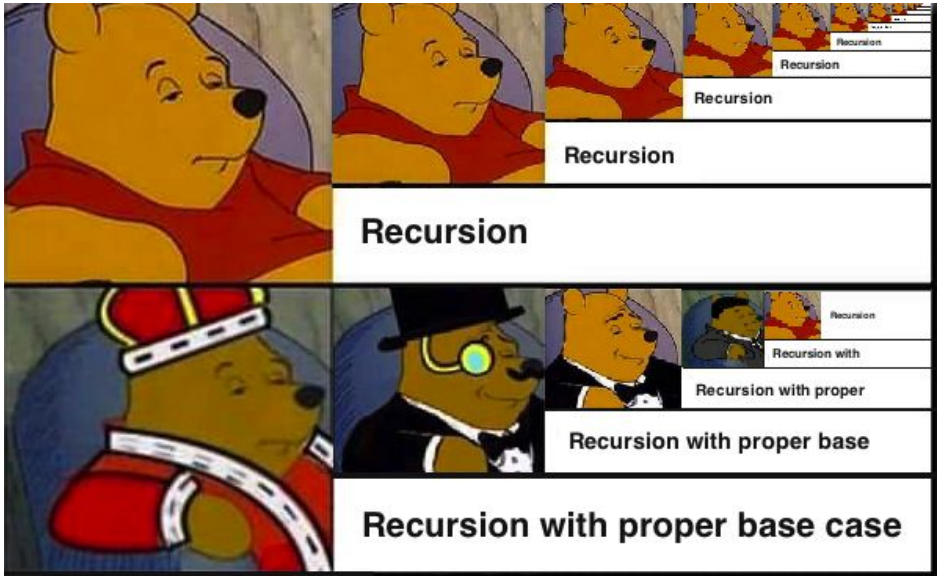


CS 200 Midterm Review

Jonathan Yu and Suba Ramesh

Topics

1. How to get better at UNIX + some practice scenarios
2. Lambda functions
3. List comprehension
4. Recursion and Trees
5. Regex
6. OOP

How can I get better at UNIX?

1) UNIX tutorial on the Zoo! ssh into the Zoo; then in your home folder, type the following command: `python3 /c/cs200/www/unixtutorial.py`

2) Practice by typing commands on the Zoo

General tips:

- Be familiar with the *output* of each command (important in context of the transcript!)

Example 1 from practice midterm

```
-bash-4.2$ pwd
/home/accts/sbs5/cs201
-bash-4.2$ ls
bin  class  graded  handouts  hws  previous-years  README  SUBMIT
TESTS  www
-bash-4.2$ mkdir mt
-bash-4.2$ [redacted]
-bash-4.2$ pwd
/home/accts/sbs5/cs201/mt
```

Answer 1

```
-bash-4.2$ pwd
/home/accts/sbs5/cs201
-bash-4.2$ ls
bin  class  graded  handouts  hws  previous-years  README  SUBMIT
TESTS  www
-bash-4.2$ mkdir mt
-bash-4.2$ cd mt
-bash-4.2$ pwd
/home/accts/sbs5/cs201/mt
```

Example 2

```
-bash-4.2$ ls -l
total 8
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:12 f1
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:13 f2
-bash-4.2$ diff f1 f2
-bash-4.2$ [redacted]
-bash-4.2$ ls -l
total 4
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:13 f2
-bash-4.2$ [redacted]
11 11 72 f2
```

Answer 2

```
-bash-4.2$ ls -l
total 8
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:12 f1
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:13 f2
-bash-4.2$ diff f1 f2
-bash-4.2$ rm f1
-bash-4.2$ ls -l
total 4
-rw-rw-r-- 1 sbs5 cs201ta 72 Oct  6 16:13 f2
-bash-4.2$ wc f2
11 11 72 f2
```

Other commands to be familiar with

- **touch file_name** — creates an empty file with name file_name
- **date > file_name** — redirection operator; creates a file with name file_name that contains the current date
- **cmd1 | cmd2** — pipe operator; redirects output from **cmd1** as input into **cmd2**
- **cp foo bar** — copies file contents of foo into the file bar
- **mv *.txt folder** — moves all .txt files into the directory named folder
- **cd ..** — navigate one directory up
- **ls -l** — list all files in the current directory in long form and with permissions

Lambda functions

```
(lambda x: x * 3 + 1 if x % 2 else x / 2)(11) =>
```

This is the same as:

```
def f(x):
        if x % 2:
                return x*3+1
        else:
                return x/2

f(11) = ?
```

Similar structure to list comprehension

List Comprehension examples

For loop:

```
def inflate(lst, value=1):
    result = []
    for i in lst:
        if type(i) in [int, float, complex]:
            result.append(value + i)
        else:
            result.append(i)
    return result
```

```
def evenout(lst):
    lst = lst[:]
    for i,n in enumerate(lst):
        if isinstance(n,int):
            if n % 2:
                lst[i] = 1 + n
    return lst
```

List Comprehension:

```
def lcinflate(lst, value=1):
    return [x + value if type(x) in [int, float, complex] else x for x in lst]
```

```
def lcevenout(lst):
    return [n+1 if (isinstance(n,int) and n % 2) else n for n in lst]
```

Example from practice midterm

As list comprehension:

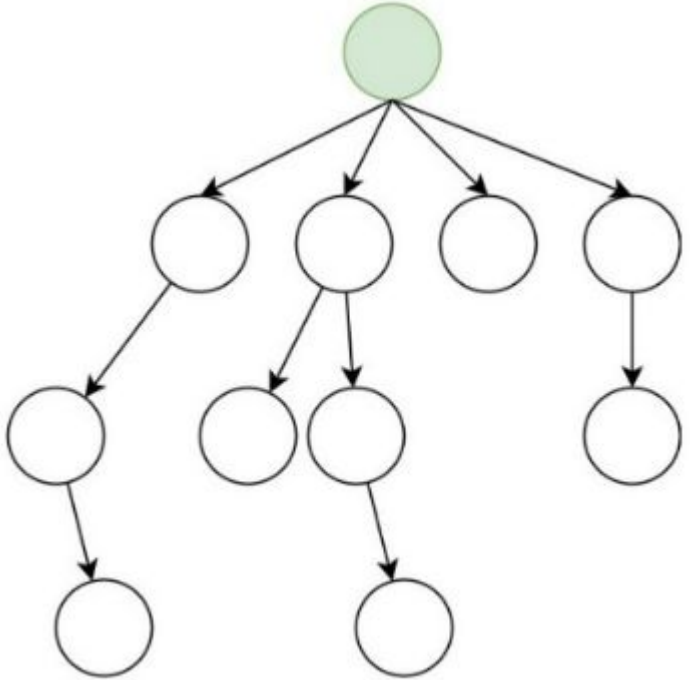
```
>>> roundup(2,[1,2,3,4,5,6]) => [2, 2, 4, 4, 6, 6]
>>> roundup(2,[2,4,6,8]) => [2, 4, 6, 8]
>>> roundup(3,[1,2,3,4,5,6]) => [3, 3, 3, 6, 6, 6]
>>> roundup(4,[1,2,3,4,5,6]) => [4, 4, 4, 4, 8, 8]
>>> roundup(4,[-1,-2,-3,-4,-5,-6]) => [0, 0, 0, -4, -4, -4]
```

Solution

```
def lcroundup(base, lst):
    return [n + (base - (n % base)) if n % base != 0 else n for n in lst]
```

Trees (Recursion)

- Exploring a tree recursively
  - Consider base cases (where are you)
  - What do you do at a leaf?
  - Then explore the kids
- Getting the depth of the tree
  - Max depth at a node is the max depth of a child + 1



Example from practice midterm

```
changetree(1,2,[1,2,3,1,4,5]) => [1, 1, 3, 1, 4, 5]
changetree(1,2,[1,2,[1,2]]) => [1, 1, [1, 1]]
changetree(1,2,[2,4,[2,4,[[[2,4]]]]]) => [1, 4, [1, 4, [[[1, 4]]]]]
changetree('tulip','rose',['a','rose',['is','a',['rose']],['is','a','rose']])
=> ['a', 'tulip', ['is', 'a', ['tulip']], ['is', 'a', 'tulip']]
```

Answer

```
def changetree(new, old, tree):
    if not tree:
        return tree
    if tree == old:
        return new
    if type(tree) is not list:
        return tree
    result = []
    for i in tree:
        result.append(changetree(new, old, i))
    return result
```

Base cases

Explore the children

Also consider what happens to the value returned from recursion. Do we return it? Do we add to a list? Some other calculation? Depends on the problem

Here we add to a result list because we want to keep track of all values in the tree and preserve the tree structure

How can I get better at regex?

1) Study regex patterns! Here's a useful cheat sheet to study symbols, ranges, and groupings: https://cheatography.com/davechild/cheat-sheets/regular-expressions/

2) Practice regex in python! Get familiar with using the re module here: https://zoo.cs.yale.edu/classes/cs200/lectures/RegExp.html

General tips:

- Practice grouping regex symbols and thinking about expressions that would fit those patterns

Example 1 from practice midterm

```
'^[^aeiou]*$'
```

Which of the following does the above pattern match?

Strings:

1. 'aaa'
2. 'bbb'
3. 'abc'
4. '123'
5. '123...456'
6. 'abc...def'
7. 'a.b.c.d'
8. ' a b c '
9. 'AAA'

Answer 1

```
'^[^aeiou]*$'
```

Which of the following does the above pattern match?

```
Strings:
   1. 'aaa'
   2. 'bbb'
   3. 'abc'
   4. '123'
   5. '123...456'
   6. 'abc...def'
   7. 'a.b.c.d'
   8. ' a b c '
   9. 'AAA'
```

2, 4, 5, 9

Example 2 from practice midterm

```
'^\d\s?'
```

Which of the following does the above pattern match?

Strings:

1. 'aaa'
2. 'bbb'
3. 'abc'
4. '123'
5. '123...456'
6. 'abc...def'
7. 'a.b.c.d'
8. ' a b c '
9. 'AAA'

Answer 2 from practice midterm

```
'^\d\s?'
```

Which of the following does the above pattern match?

```
Strings:
   1. 'aaa'
   2. 'bbb'
   3. 'abc'
   4. '123'
   5. '123...456'
   6. 'abc...def'
   7. 'a.b.c.d'
   8. ' a b c '
   9. 'AAA'
```

4, 5

Object Oriented Programming

Resources:

- (1) Review lecture notes on OOP concepts! https://zoo.cs.yale.edu/classes/cs200/lectures/Oop.html
- (2) There are also plenty of external resources if you want some more practice or clarification on certain OOP concepts
  - e.g. https://www.socratica.com/lesson/classes-and-objects

Constructor and class variables

- All students have the same school
- Each individual student has their own properties (e.x. name)

```
class Student:
    # Class variable
    school_name = 'ABC School '

    # constructor
    def __init__(self, name, roll_no):
        self.name = name
        self.roll_no = roll_no
```

Example from practice midterm

Define a class employee and associated methods that has the following behavior.

```
e1 = employee("John", 30000)
e2 = employee("Mary", 40000)
e3 = employee("Jane", 50000)
e4 = employee("Hannah", 60000)

e1.add_supervisor(e3)
e2.add_supervisor(e3)
e3.add_supervisor(e4)

employee.members => [employee('John', 30000), employee('Mary',
40000), employee('Jane', 50000), employee('Hannah', 60000)]

employee.highest_paid() => ('Hannah', 60000)

e4.all_reports() =>
employee('Jane', 50000)
employee('John', 30000)
employee('Mary', 40000)
```

Answer 1

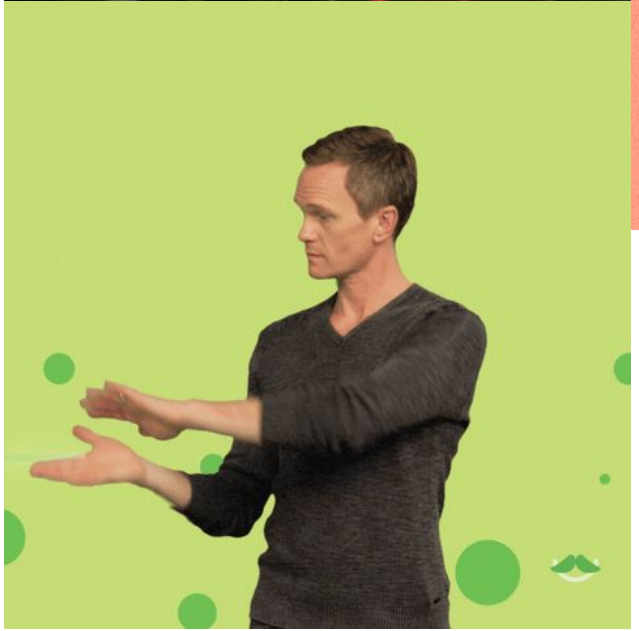
- Members is for all employees not just one instance
- -__init__() is the constructor for one instance of a class
- __repr__() is a printable representation of object
- @staticmethod is for class not a single object
  - E.x. highest paid in whole class can't be for just one employee object, but add supervisor adds a supervisor to one particular employee

```
class employee:
    members = []

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        employee.members.append(self)
        self.supervisor = ''
        self.reports = []

    def __repr__(self):
      return "employee({}, {})".format(repr(self.name), repr(self.salary))

    def add_supervisor(self, item):
        # should check to see if already has a supervisor
        self.supervisor = item
        item.reports.append(self)


    def all_reports(self):
        # should check for cycles
        for r in self.reports:
            print (r)

       r.all_reports()

@staticmethod
def highest_paid():
  max = 0
  maxname = ''
  for e in employee.members:
        if e.salary > max:
        max = e.salary
        maxname = e.name
  return (maxname, max)
```


You can do it!
GIPHY


YOU CAN
DO IT !


YOU




motivational penguin


YOU'RE DOING A GREAT JOB!