Prelim 1 Review
Fall 2017
Exam Info

• Prelim 1: 7:30–9:00PM, Thursday, October 12th
  ▪ Last name A – J in Uris G01
  ▪ Last name K – Z in Statler Auditorium
  ▪ SDS Students will get an e-mail

• To help you study:
  ▪ Study guides, review slides are online
  ▪ Solutions to Assignment 2 are in CMS

• Arrive early! Helps reducing stress
Grading

• We will announce *approximate* letter grades
  ▪ We adjust letter grades based on all exams
  ▪ But no hard guidelines (e.g. mean = grade X)
  ▪ May adjust borderline grades again at final grades

• Use this to determine whether you want to drop
  ▪ **Drop deadline** is next week, October 17th
  ▪ **Goal:** Have everyone graded by end of Saturday
  ▪ Will definitely notify you if you made less than C
What is on the Exam?

• **Five** Questions out of Six Topics:
  - String slicing functions (A1)
  - Call frames and the call stack (A2)
  - Functions on mutable objects (A3)
  - Testing and debugging (Lab 3 & 6, Lec. 11)
  - Lists and For-Loops (Lab 7)
  - Short Answer (Terminology)

• + 2 pts for writing your name and net-id
What is on the Exam?

- **String slicing functions (A1)**
  - Will be given a function specification
  - Implement it using string methods, slicing
- **Call frames and the call stack (A2)**
- **Functions on mutable objects (A3)**
- **Testing and debugging (Lab 3 & 6, Lecture 11)**
- **Lists and For-Loops (Lab 7)**
- **Short Answer (Terminology)**
def make_netid(name,n):
    """Returns: a netid for name with suffix n
    Netid is either two letters and a number (if the student has no
    middle name) or three letters and a number (if the student has
    a middle name). Letters in netid are lowercase.
    Example: make_netid('Walker McMillan White',2) is 'wmw2'
    Example: make_netid('Walker White',4) is 'ww4'
    Parameter name: the student name
    Precondition: name is a string either with format '<first-name>
    <last-name>' or '<first-name> <middle-name> <last-name>'
    Parameter n: the netid suffix
    Precondition: n > 0 is an int."""
# Useful String Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.find(s1)</td>
<td>Returns first position of s1 in s; -1 if not there.</td>
</tr>
<tr>
<td>s.rfind(s1)</td>
<td>Returns LAST position of s1 in s; -1 if not there.</td>
</tr>
<tr>
<td>s.lower()</td>
<td>Returns copy of s with all letters lower case</td>
</tr>
<tr>
<td>s.upper()</td>
<td>Returns copy of s with all letters upper case</td>
</tr>
</tbody>
</table>

- We will give you any methods you need
- But you must know how to slice strings!
```python
def make_netid(name, n):
    
    """Returns: a netid for name with suffix n.""
    
    name = name.lower() # switch to lower case
    fpos = name.find(' ') # find first space
    first = name[:fpos]
    last = name[fpos+1:]
    mpos = last.find(' ') # see if there is another space

    if mpos == -1:
        return first[0] + last[0] + str(n) # remember, n is not a string
    else:
        middle = last[:mpos]
        last = last[mpos+1:]
        return first[0] + middle[0] + last[0] + str(n)
```
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
  - Very similar to A2 (see solution in CMS)
  - May have to draw a full call stack
  - See lectures 4 and 9 (slide typos corrected)
- Functions on mutable objects (A3)
- Testing and debugging (Lab 3 & 6, Lecture 11)
- Lists and For-Loops (Lab 7)
- Short Answer (Terminology)
Call Stack Example

- Given functions to right
  - Function `fname()` is not important for problem
  - Use the numbers given
- Execute the call:
  `lname_first('John Doe')`
- Draw **entire** call stack when helper function `lname` completes line 1
  - Draw nothing else

```python
def lname_first(s):
    """Precondition: s in the form
    <first-name> <last-name>""
    first = fname(s)
    last = lname(s)
    return last + ',' + first

def lname(s):
    """Prec: see last_name_first""
    end = s.find(' ')
    return s[end+1:]
```

10/11/17  Prelim 1 Review
Call Stack Example: `lname_first('John Doe')`

```
def lname_first(s):
    """Precondition: s in the form <first-name> <last-name>"""
    first = fname(s)
    last = lname(s)
    return last + ',' + first

def lname(s):
    """Prec: see last_name_first"
    end = s.find(' ')
    return s[end+1:]
```
Call Stack Example: `lname_first('John Doe')`

```python
def lname_first(s):
    """Precondition: s in the form <first-name> <last-name>""
    first = fname(s)
    last = lname(s)
    return last + ', ' + first

def lname(s):
    """Prec: see last_name_first""
    end = s.find(' ')
    return s[end+1:]
```

10/11/17

1. `end = s.find(' ')`
2. `return s[end+1:]`

Line 1 is complete. Counter is next line.

Line 2 is not complete.

No variable last.
Example with a Mutable Object

def cycle_left(p):
    """Cycle coords left
    Precondition: p a point""
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

• May get a function on a mutable object
  >>> p = Point3(1.0,2.0,3.0)
  >>> cycle_left(p)

• You are not expected to come up w/ the “folder”
  - Will provide it for you
  - You just track changes

• Diagram all steps
Example with a Mutable Object

```python
def cycle_left(p):
    """Cycle coords left
    Precondition: p a point""
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0,2.0,3.0)
>>> cycle_left(p)
```

10/11/17 Prelim 1 Review
Example with a Mutable Object

```python
def cycle_left(p):
    """Cycle coords left
    Precondition: p a point"
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0, 2.0, 3.0)

>>> cycle_left(p)  # Function Call
```

10/11/17  Prelim 1 Review
Example with a Mutable Object

def cycle_left(p):
    """Cycle coords left
    Precondition: p a point""
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0,2.0,3.0)
>>> cycle_left(p)  # Function Call
Example with a Mutable Object

def cycle_left(p):
    """Cycle coords left
    Precondition: p a point"
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0,2.0,3.0)

>>> cycle_left(p)  # Function Call
Example with a Mutable Object

def cycle_left(p):
    """Cycle coords left
    \textbf{Precondition}: p a point""
    1. temp = p.x
    2. p.x = p.y
    3. p.y = p.z
    4. p.z = temp

>>> p = Point3(1.0, 2.0, 3.0)
>>> cycle_left(p)  # Function Call
def cycle_left(p):
    """Cycle coords left
    Precondition: p a point"
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0,2.0,3.0)

>>> cycle_left(p)  # Function Call
Example with a Mutable Object

```python
def cycle_left(p):
    """Cycle coords left
    Precondition: p a point""
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point3(1.0,2.0,3.0)
>>> cycle_left(p)
```

>>> p = Point3(1.0,2.0,3.0)
>>> cycle_left(p)
What is on the Exam?

• String slicing functions (A1)
• Call frames and the call stack (A2)
• Functions on mutable objects (A3)
  § Given an object type (e.g. class)
  § Attributes will have invariants
  § Write a function respecting invariants
• Testing and debugging (Lab 3 & 6, Lecture 11)
• Lists and For-Loops (Lab 7)
• Short Answer (Terminology)
Example from Assignment 3

• Class: RGB
  ▪ Constructor function: RGB(r,g,b)
  ▪ Remember constructor is just a function that gives us back a mutable object of that type
  ▪ Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>green</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>blue</td>
<td>int, within range 0..255</td>
</tr>
</tbody>
</table>
def lighten(rgb):

"""Lighten each attribute by 10%
Attributes get lighter when they increase.

Parameter rgb: the color to lighten
Precondition: rgb an RGB object"""

pass # implement me
Function that Modifies Object

def lighten(rgb):
    """Lighten each attribute by 10%""
    red = rgb.red  # puts red attribute in local var
    red = 1.1*red  # increase by 10%
    red = int(round(red,0))  # convert to closest int
    rgb.red = min(255,red)  # cannot go over 255
    # Do the others in one line
    rgb.green = min(255,int(round(1.1*rgb.green,0))))
    rgb.blue = min(255,int(round(1.1*rgb.blue,0))))
Another Example

• Class: Length
  - Constructor function: Length(ft,in)
  - Remember constructor is just a function that gives us back a mutable object of that type
  - Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>feet</td>
<td>int, non-negative, = 12 in</td>
</tr>
<tr>
<td>inches</td>
<td>int, within range 0..11</td>
</tr>
</tbody>
</table>
Function that Does Not Modify Object

def difference(len1, len2):

    """**Returns**: Difference between len1 and len2

    Result is returned in inches

    **Parameter** len1: the first length
    **Precondition**: len1 is a length object longer than len2

    **Parameter** len2: the second length
    **Precondition**: len2 is a length object shorter than len1"

    pass  # implement me
def difference(len1, len2):

    """**Returns**: Difference between len1 and len2
    Result is returned in inches

    **Parameter** len1: the first length

    **Parameter** len2: the second length

    **Precondition**: len2 is a length object shorter than len1"

    feetdif = (len1.feet - len2.feet) * 12
    inchdif = len1.inches - len2.inches  # may be negative

    return feetdif + inchdif
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
- Functions on mutable objects (A3)
- Testing and debugging (Lab 3 & 6, Lecture 11)
  - Coming up with test cases
  - Tracing program flow
  - Understanding assert statements
- Lists and For-Loops (Lab 7)
- Short Answer (Terminology)
def pigify(w):

    """Returns: copy of w converted to Pig Latin
'y' is a vowel if it is not the first letter
If word begins with a vowel, append 'hay'
If word starts with 'q', assume followed by 'u';
move 'qu' to the end, and append 'ay'
If word begins with a consonant, move all
consonants up to first vowel to end and add 'ay'
Parameter w: the word to translate
Precondition: w contains only (lowercase) letters"""
def pigify(w):
    """Returns: copy of w converted to Pig Latin""
    ...

- Test Cases (Determined by the rules):
  - are => arehay (Starts with vowel)
  - quiet => ietquay (Starts with qu)
  - ship => ipshay (Starts with consonant(s))
  - bzzz => bzzzay (All consonants)
  - yield => ieldyay (y as consonant)
  - byline => ylinebay (y as vowel)
def replace_first(word, a, b):

    """Returns: a copy with FIRST instance of a replaced by b""

    Example: replace_first('crane','a','o') returns 'crone'
    Example: replace_first('poll','l','o') returns 'pool'

    Parameter word: The string to copy and replace
    Precondition: word is a string

    Parameter a: The substring to find in word
    Precondition: a is a valid substring of word

    Parameter b: The substring to use in place of a
    Precondition: b is a string"""
Debugging Example

```python
def replace_first(word, a, b):
    
    """Returns: a copy with FIRST a replaced by b""
    
pos = word.rfind(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before + b + after
    print(result)
    return result
```

```output
>>> replace_first('poll', 'l', 'o')
3
pol
polo
'polo'
```

```output
>>> replace_first('askew', 'sk', 'ch')
1
a
kehr
'kehr'
```

Identify the bug(s) in this function.
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""
    pos = word.rfind(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before+b+after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3 Unexpected!
pol

polo
'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
ke
achkew
'achkew'
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""

    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before + b + after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol

polo

'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew

'achkew'
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""
    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before+b+after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol

>>> replace_first('askew', 'sk', 'ch')
1
a

kew Unexpected!

achkew

'achkew'
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""

    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+len(a):]
    print(after)
    result = before + b + after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol

polo

'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew

ackkew

'ackkew'
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
- Functions on mutable objects (A3)
- Testing and debugging (Lab 3 & 6, Lecture 11)
- Lists and For-Loops (Lab 7)
  - Given a function specification
  - Implement it using a for-loop
  - Challenge is how to use accumulators
- Short Answer (Terminology)
# Useful List Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.index(a)</td>
<td>Returns first position of a in x; error if not there</td>
</tr>
<tr>
<td>x.append(a)</td>
<td>Modify x to add element a to the end</td>
</tr>
<tr>
<td>x.insert(a,k)</td>
<td>Modify x to put a at position k (and move rest to right)</td>
</tr>
<tr>
<td>x.remove(a)</td>
<td>Modify x to remove first occurrence of a</td>
</tr>
<tr>
<td>x.sort()</td>
<td>Modify x so that elements are in sorted order</td>
</tr>
</tbody>
</table>

- We will give you any methods you need
- But you must know how to slice lists!
For-Loop in a Fruitful Function

```python
def replace(thelist, a, b):
    """Returns: COPY of thelist with all occurrences of a replaced by b
    Example: replace([1,2,3,1], 1, 4) = [4,2,3,4].
    Parameter thelist: list to copy
    Precondition: thelist is a list of ints
    Parameter a: the value to remove
    Precondition: a is an int
    Parameter b: the value to insert
    Precondition: b is an int """
    return [] # Stub return. IMPLEMENT ME
```

For-Loop in a Fruitful Function

```python
def replace(thelist, a, b):
    """Returns: COPY of thelist with all occurrences of a replaced by b
    Example: replace([1,2,3,1], 1, 4) = [4,2,3,4]."

    result = [] # Accumulator
    for x in thelist:
        if x == a:
            result.append(b)
        else:
            result.append(x)
    return result
```

10/11/17 Prelim 1 Review
def pairswap(seq):
    """MODIFIES thelist, swapping each two elements with each other

Example: if a = [0,2,4,5], pairswap(a) makes a into [2,0,5,4]
        if a = [1,2], pairswap(a) turns a into [2,1]

Parameter thelist: list to modify
Precondition: thelist is a list with an even number of elements."""
    pass # implement me
For-Loop in a Procedure

```python
def pairswap(thelist):
    
    """MODIFIES thelist, swapping each two elements with each other

    Example: if a = [0,2,4,5], pairswap(a) makes a into [2,0,5,4]
    if a = [1,2], pairswap(a) turns a into [2,1]

    Precondition: thelist is a list with an even number of elements."""

    for k in range(len(thelist)):
        if k % 2 == 0:
            tmp = thelist[k]  # Store old value
            thelist[k] = thelist[k+1]  # Get next value
        else:
            thelist[k] = tmp  # Value stored in previous step
```
What is on the Exam?

• String slicing functions (A1)
• Call frames and the call stack (A2)
• Functions on mutable objects (A3)
• Testing and debugging (Lab 3 & 6, Lecture 10)
• Lists and For-Loops (Lab 7)
• Short Answer (Terminology)
  ▪ See the study guide
  ▪ Look at the lecture slides
  ▪ Read relevant book chapters

In that order
Any More Questions?
Good Luck!