**Exam Info**

- Prelim 1: 7:30–9:00PM, Thursday, October 4th
  - Last name **A – P** in Kennedy 1116
  - Last name **R – T** in Warren 131
  - Last name **U – Z** in Warren 231

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

- Arrive early! Helps reducing stress

- Grades released the same evening (if possible)
What is on the Exam?

• Five Questions (+2pts for name, netid):
  ▪ String slicing functions (A1)
  ▪ Call frames and the call stack (A2)
  ▪ Functions on mutable objects (A3)
  ▪ Testing and debugging (A1, Lab 3)
  ▪ Short Answer (Terminology)

• Roughly equal weight each
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 (A1, Lab 3)
• 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 'wmw4'
    Precondition: name is a string either with format '<first-name>
    <last-name>' or '<first-name> <middle-name> <last-name>';
    names are separated by spaces. 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!
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] + `n`  # remember, n is not a string
    else:
        middle = last[:mpos]
        last = last[mpos+1:]
        return first[0] + middle[0] + last[0] + `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 5 and 9 (slide typos corrected)
• Functions on mutable objects (A3)
• Testing and debugging (A1, Lab 3)
• 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
```

```python
def lname(s):
    """Prec: see last_name_first""
    end = s.find(' ')
    return s[end+1:]
```
Call Stack Example: \texttt{lname\_first('John Doe')}

\begin{itemize}
  \item \texttt{def lname\_first(s):}
    \begin{itemize}
      \item \texttt{"""\textbf{Precondition:} s in the form <first-name> <last-name>"""}
      \item \texttt{first = fname(s)}
      \item \texttt{last = lname(s)}
      \item \texttt{return last + ', ' + first}
    \end{itemize}
  \item \texttt{def lname(s):}
    \begin{itemize}
      \item \texttt{"""\textbf{Prec:} see last\_name\_first"""}
      \item \texttt{end = s.find( ' ')}
      \item \texttt{return s[end+1:]}\end{itemize}
\end{itemize}
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"

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

Omitting this is okay. Line 2 is not complete.

Line 1 is complete. Counter is next line.

"""Prec: see last_name_first"""
Example with a Mutable Object

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

>>> p = Point(1.0,2.0,3.0)
>>> shift(p)
```

- May get a function on a mutable object
  >>> p = Point(1.0,2.0,3.0)
  >>> shift(p)

- You are not expected to come up w/ the “folder”
  - Will provide it for you
  - You just track changes
Example with a Mutable Object

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

>>> p = Point(1.0, 2.0, 3.0)
>>> shift(p)
```

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

```python
def shift(p):
    '''Shift coords left
    Precondition: p a point'''
    temp = p.x
    p.x = p.y
    p.y = p.z
    p.z = temp

>>> p = Point(1.0, 2.0, 3.0)
>>> shift(p)
```

Function Call

```
43001122  p  43001122
x 1.0
y 2.0
z 3.0
shift

>>> p = Point(1.0, 2.0, 3.0)
>>> shift(p)  Function Call
```
Example with a Mutable Object

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

>>> p = Point(1.0,2.0,3.0)
>>> shift(p)
```

```
Function Call
```

```python
>>> p = Point(1.0,2.0,3.0)
>>> shift(p)
```

```
Function Call
```
Example with a Mutable Object

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

>>> p = Point(1.0,2.0,3.0)
>>> shift(p)  # Function Call
```

```plaintext
| 1 | temp = p.x |
| 2 | p.x = p.y |
| 3 | p.y = p.z |
| 4 | p.z = temp |
```
Example with a Mutable Object

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

>>> p = Point(1.0,2.0,3.0)
>>> shift(p)
```

Function Call: `shift(p)`
Example with a Mutable Object

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

>>> p = Point(1.0, 2.0, 3.0)
>>> shift(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 (A1, Lab 3)
• Short Answer (Terminology)
Example from Assignment 3

- **Type: 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>
Function that Modifies Object

```python
def lighten(rgb):
    """Lighten each attribute by 10%
    Attributes get ligher when they increase.
    Precondition: rgb an RGB object"

    pass  # implement me
```
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

• Type: 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>
def difference(len1, len2):
    """Returns: Difference between len1 and len2
    Result is returned in inches
    Precondition: len1 and len2 are length objects
    len1 is longer than len2""
    pass  # implement me
def difference(len1, len2):
    """Returns: Difference between len1 and len2
    Result is returned in inches
    Precondition: len1 and len2 are length objects
    len1 is longer than len2""
    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 (A1, Lab 3)
  - Coming up with test cases
  - Tracing program flow
  - Understanding asserts, try-except
- 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'
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 first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(2)?
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(2)?

'Starting first.'
'Starting second.'
'Starting third.'
'Caught at second'
'Ending second'
'Ending first'
Tracing Control Flow

```python
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'
```

What is the output of `first(0)`?
def first(x):
    print 'Starting first.'
    try:
        second(x)
    except:
        print 'Caught at first'
    print 'Ending first'

def second(x):
    print 'Starting second.'
    try:
        third(x)
    except:
        print 'Caught at second'
    print 'Ending second'

def third(x):
    print 'Starting third.'
    assert x < 1
    print 'Ending third.'

What is the output of first(0)?

'Starting first.'
'Starting second.'
'Starting third.'
'Ending third'
'Ending second'
'Ending first'
What is on the Exam?

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

In that order
Any More Questions?
Good Luck!