Compsci 101
List and String Operations, For loop

Susan Rodger
September 20, 2022

G is for …

- **Google**
  - How to find the answer to everything
- **Global Variable**
  - Accessible everywhere, typically do not do
- **GIGO**
  - Garbage In, Garbage Out
- **Git**
  - Working Together or Solo

Sir Tim Berners-Lee

- Invented World Wide Web
  - Turing award 2016
- HTTP vs. TCP/IP
  - Just protocols?

“This Web as I envisaged it, we have not seen it yet. The future is still so much bigger than the past.”

“We need diversity of thought in the world to face the new challenges.”

Did you sign up for compsci@duke.edu mailing list?

- **Mailing list to get the CompSci weekly newsletter**
  - Events, research and job opportunities
- **To add yourself:**
  - Go to lists.duke.edu
  - Authenticate and then add compsci@duke.edu
- **Sample item:**
  - Duke Women in Tech looking for new members and to get our mailing lists. Fill out this form: https://tinyurl.com/witspring22
Announcements

• Assignment 1 Faces
  • Assign 1 QZ due today 11:30pm (no grace day!)
  • Program due Thursday (has one grace day)
  • Also REFLECT Form due same time
  • Remember, no consulting hours on Friday
• APT-2 out today, due Sept 29
• Lab 3 Friday
  • Do prelab 3 before attending!
• Exam 1 on Tuesday, Sept 27

Lecture for Thursday Sept 22

• Lecture will be asynchronous
  • There is no in-person lecture
• Watch the videos (there are 4), do WOTOs
• Complete it before going to lab
• WOTOs will turn off by Friday night 11:30pm
• Prof. Rodger is out of town Tues-Friday, no office hours today or the rest of the week
• We have fantastic UTAs in consulting hours! And there is always Ed Discussion!

PFTD

• Exam 1
• Lists continued
• String methods and more
• For Loops

Exam 1 – Sept 27, 2022

• All lecture/reading topics through Tues. Sept 20
  • Topics on Sept 20 at simpler level
    • Loop over list, loop over characters in a string
• Understand/Study
  • Reading, lectures
  • Assignment 1, APT-1, (APT-2 helpful, not required)
  • Labs 0-3
  • Very Important! Practice writing code on paper
• Logistics:
  • In person, in lecture
Exam 1 – Sept 27, 2022 (cont)

- What you should be able to do
  - Read/trace code
  - Determine output of code segment
  - Write small code segments/function
- Look at old test questions
  - We will look at some in Lab 3
- Exam 1 is your own work!
  - Do not consult with anyone else.
  - Closed book, no notes, no paper, no calculators
  - See Exam 1 Reference sheet (will be on exam)

Nested Lists

- Lists are heterogenous, therefore!
  - `lst = [1, 'a', [2, 'b']]` is valid
  - `len(lst) == 3`
    - `[2, 'b']` is one element in list `lst`

  ```python
  lst[2][1]
  [2, 'b'][1] == 'b'
  ```

- How to index?
  - `[...] all the way down
  - `lst[2][1]` returns ‘b’
Nested Lists with Python Tutor

 Mutating Lists

- `lt = ['Hello', 'world']`
  - Change to: ['Hello', 'Ashley']

- Two ways: 1. Build new list or 2. modify list
  1. Concatenation: `lt = [lt[0]] + ['Ashley']`
  2. Index: `lt[1] = 'Ashley'`

- How change 'b' in `lt = [1, 'a', [2, 'b']]`?
  - `lt[2][1] = 'c'`

Mutating Lists code

```python
lst1 = ['Hello', 'world']
print(lst1)
lst2 = [lst1[0]] + ['Ashley']
print(lst2)
print(lst1)
lst1[1] = 'Ashley'
print(lst1)
lst3 = [1, 'a', [2, 'b']]
print(lst3)
lst3[2][1] = 'c'
print(lst3)
```

Mutating Lists code

```python
lst1 = ['Hello', 'world']
print(lst1)
lst2 = [lst1[0]] + ['Ashley']
print(lst2)
print(lst1)
lst1[1] = 'Ashley'
print(lst1)
lst3 = [1, 'a', [2, 'b']]
print(lst3)
lst3[2][1] = 'c'
print(lst3)
```

OUTPUT:

```
['Hello', 'world']
<class 'list'>
[2, 'b']
b
```
```python
# Mutating Lists code

lst1 = ['Hello', 'world']
print(lst1)
lst2 = [lst1[0]] + ['Ashley']
print(lst2)
print(lst1)
lst1[1] = 'Ashley'
print(lst1)
lst3 = [1, 'a', [2, 'b']] 
print(lst3)
lst3[2][1] = 'c'
print(lst3)

# Output

['Hello', 'world']
['Hello', 'Ashley']
['Hello', 'world']
['Hello', 'Ashley']
[1, 'a', [2, 'b']] 
[1, 'a', [2, 'c']]
```
Mutating Lists code

```python
lst1 = ['Hello', 'world']
print(lst1)
lst2 = [lst1[0]] + ['Ashley']
print(lst2)
print(lst1)
lst1[1] = 'Ashley'
print(lst1)
lst3 = [1, 'a', [2, 'b']]
print(lst3)
lst3[2][1] = 'c'
print(lst3)
```

OUTPUT:

```
[1, 'a', [2, 'b']]
[1, 'a', [2, 'c']]
```

Immutable built-in Types

- **In python string, int, float, boolean - Immutable**
  - Once created cannot change
  - These are still objects in Python3!!
- **PythonTutor gets this wrong**
  - Everything should be in Objects area
- **Objects don’t change**
  - Value associated with variable changes

```python
val = 0
bee = val
val = val + 20
```

val is 0

bee is 0
Immutable built-in Types

- In python string, int, float, boolean - Immutable
  - Once created cannot change
  - These are still objects in Python3!!
- PythonTutor gets this wrong
  - Everything should be in Objects area
- Objects don’t change
  - Value associated with variable changes

```
val = 0
bee = val
val = val + 20
```

val is 20
bee is 0

```
val = "apple"
bee = val
val = val + "sauce"
```

val = "apple"
bee = val
val = val + "sauce"

val is "apple"
bee is "apple"
Immutable built-in Types

- In python string, int, float, boolean - Immutable
  - Once created cannot change
  - These are still objects in Python3!!
- PythonTutor gets this wrong
  - Everything should be in Objects area
- Objects don’t change
  - Value associated with variable changes

```
val = "apple"
bee = val
val = val + "sauce"

val is "applesauce"
bee is "apple"
```

Compare assign with integers, strings and lists – 1

```
Python 3.6
(known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"

Edit this code
```

```
y gets a copy of the value of x
```

Compare assign with integers, strings and lists – 2

```
Python 3.6
(known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"

Edit this code
```

Compare assign with integers, strings and lists – 3

```
Python 3.6
(known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"

Edit this code
```

```
y gets a copy of the value of x
```

Compare assign with integers, strings and lists – 4

```
Python 3.6 (known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"
```

```
Frames
Objects
Global frame
x = 3
y = 6
```

```
x gets a new value
```

Compare assign with integers, strings and lists – 5

```
Python 3.6 (known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"
```

```
Frames
Objects
Global frame
x = 3
y = 6
m = "pink"
```

```
m gets a new value
```

Compare assign with integers, strings and lists – 6

```
Python 3.6 (known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"
```

```
Frames
Objects
Global frame
x = 3
m = "pink"
```

```
n gets a copy of the value of m
```

Compare assign with integers, strings and lists – 7

```
Python 3.6 (known limitations)
1 x = 6
2 y = x
3 x = 3
4 m = "pink"
5 n = m
6 m = "red"
7 a = ["pig", "cow", "dog"]
8 b = a
9 a[-1] = "ant"
```

```
Frames
Objects
Global frame
x = 3
m = "pink"
```

```
m gets a new value
```
Compare assign with integers, strings and lists – 8

1. x = 6
2. y = x
3. x = 3
4. m = "pink"
5. n = m
6. m = "red"
7. a = ["pig", "cow", "dog"]
8. b = a
9. a[-1] = "ant"

Edit this code
- line that just executed
- next line to execute

Frames
- Global frame
  - x 3
  - y 6
  - m "red"
  - n "pink"
  - a

Objects
- list
  - ["pig", "cow", "dog"]

Compare assign with integers, strings and lists – 9

B gets a copy of the value of a

a’s value is the address of its list, the address is copied!

a and b refer to the same list!

List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```

'dog' changed to 'ant'

Changing list a also changes list b
As they are the same list!
List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```

Frames

<table>
<thead>
<tr>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
</tr>
<tr>
<td>lst1</td>
</tr>
<tr>
<td>lst2</td>
</tr>
<tr>
<td>lst3</td>
</tr>
</tbody>
</table>

Objects

- 0: "a"
- 1: "b"
- 2: 1
- 3: 2

List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
lst1[-1] = "SUN"
```

Frames

<table>
<thead>
<tr>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
</tr>
<tr>
<td>lst1</td>
</tr>
<tr>
<td>lst2</td>
</tr>
<tr>
<td>lst3</td>
</tr>
</tbody>
</table>

Objects

- 0: "a"
- 1: "b"
- 2: 1
- 3: 2

- "SUN"
List Concatenation Steps

1. Calculate the length of the new list
2. Create list of that length
3. Copy values from first list
4. Copy values from second list
5. Assign the variable to the new list

Concatenation:

length, create, copy, copy, assign

```
1  lst0 = [1,2]
2  lst1 = [3, 4, 5]
3  lst2 = lst0 + lst1
```
Concatenation: Makes new List

1. `lst0 = [1, 2]`
2. `tmp = lst0`
3. `lst0 = lst0 + [4]`

What will Python Tutor Display? How many lists will there be?

Concatenation: Makes new List

1. `lst0 = [1, 2]`
2. `lst1 = [3, 4, 5]`
3. `lst2 = lst0 + lst1`

What will Python Tutor Display? How many lists will there be?
Concatenation: Makes new List

1. lst0 = [1, 2]
2. tmp = lst0
3. lst0 = lst0 + [4]

Concatenation: length, create, copy, copy, assign

• How is the inner list copied?

1. lst0 = [1, ['b', 3.0]]
2. lst1 = [4]
3. lst2 = lst0 + lst1

What will Python Tutor Display? How many copies of ['b', 3.0] will be present?

Concatenation: length, create, copy, copy, assign

1. Calculate length
2. Create new list
3. Copy left list

This is a shallow copy!
Don’t copy inner lists

Concatenation: length, create, copy, copy, copy

1. Calculate length
2. Create new list
3. Copy left list
4. Copy right list
5. Assign lst2
List Mutation: `.append(...)`

- `.append()` – list function that adds element to end of list
  - Mutates list to left of “.”
  - “.” – call function to the right of the dot on the thing to the left of the dot (LEFT . RIGHT)

```python
x = [6, 2, 4]
x.append(3)
x.append([5,2])
```

What will Python Tutor Display? One or two lists?

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```

Same list!

```python
x = [6, 2, 4]
x.append(3)
x.append([5,2])
```

```
frame
lst0 | list
| 0 | 1 | 2 | 3 |
```

```python
What will Python Tutor Display? One or two lists?
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```
List Mutation: `.append(…)`

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```

Same list! No new list

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
```

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append([5, 6])
4 lst0.append([5, 6])
```

Same list! No new list

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append(4)
4 lst0.append([5, 6])
```

```
1 lst0 = [1, 2, 3]
2 tmp = lst0
3 lst0.append([5, 6])
4 list
```
Anatomy of a `for` loop

```
for VARIABLE in SEQUENCE:
    CODE_BLOCK
```

- Think of as:
  - “For each element in the SEQUENCE put it in the VARIABLE and execute the CODE_BLOCK.”
- Also called: *Iterate* over the sequence
- What type(s) are sequences?
  - Strings, Lists
- Will VARIABLE likely be in CODE_BLOCK?

Example for loop with a list

```
1 lst = [5, 3, 2]
2 sum = 0
3 for num in lst:
4     sum = sum + num
5 print(sum)
```

- What does this for loop do?
- What is first value of `num`?
- What is final value of `num`?
Example for loop with a list

What does this for loop do?

```python
lst = [5, 3, 2]
sum = 0
for num in lst:
    sum = sum + num
print(sum)
```

What is first value of `num`?
5

What is final value of `num`?
2

Trace through for loop – 1

Trace through for loop – 2

Trace through for loop – 3
Trace through for loop – 4

1. \( \text{lst} = [5, 3, 2] \)
2. \( \text{sum} = 0 \)
3. \( \text{for} \ \text{num in lst:} \)
   4. \( \text{sum} = \text{sum} + \text{num} \)
5. \( \text{print(sum)} \)

Trace through for loop – 5

1. \( \text{lst} = [5, 3, 2] \)
2. \( \text{sum} = 0 \)
3. \( \text{for} \ \text{num in lst:} \)
   4. \( \text{sum} = \text{sum} + \text{num} \)
5. \( \text{print(sum)} \)

Trace through for loop – 6

1. \( \text{lst} = [5, 3, 2] \)
2. \( \text{sum} = 0 \)
3. \( \text{for} \ \text{num in lst:} \)
   4. \( \text{sum} = \text{sum} + \text{num} \)
5. \( \text{print(sum)} \)

Trace through for loop – 7

1. \( \text{lst} = [5, 3, 2] \)
2. \( \text{sum} = 0 \)
3. \( \text{for} \ \text{num in lst:} \)
   4. \( \text{sum} = \text{sum} + \text{num} \)
5. \( \text{print(sum)} \)
Trace through for loop – 8

1. `lst = [5, 3, 2]`
2. `sum = 0`
3. `for num in lst:`
   4. `sum = sum + num`
4. `print(sum)`

Frames

Objects

num gets third value in list

Global frame

<table>
<thead>
<tr>
<th>lst</th>
<th>num</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

9/20/22
CompSci 101, Fall 2022

Trace through for loop – 9

1. `lst = [5, 3, 2]`
2. `sum = 0`
3. `for num in lst:`
   4. `sum = sum + num`
4. `print(sum)`

Frames

Objects

Add num to sum

Global frame

<table>
<thead>
<tr>
<th>lst</th>
<th>sum</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

9/20/22
CompSci 101, Fall 2022

Trace through for loop – 10

1. `lst = [5, 3, 2]`
2. `sum = 0`
3. `for num in lst:`
   4. `sum = sum + num`
4. `print(sum)`

Frames

Objects

No more values in lst

The for loop is done!

Global frame

<table>
<thead>
<tr>
<th>lst</th>
<th>sum</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

9/20/22
CompSci 101, Fall 2022

Trace through for loop – 11

1. `lst = [5, 3, 2]`
2. `sum = 0`
3. `for num in lst:`
   4. `sum = sum + num`
4. `print(sum)`

Frames

Objects

Print result

Global frame

<table>
<thead>
<tr>
<th>lst</th>
<th>sum</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

9/20/22
CompSci 101, Fall 2022
Example for loop with a string

• What does this for loop do?

```python
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

• What is first value of `ch`?

• What is final value of `ch`?

9/20/22 Compsci 101, Fall 2022 81

Example for loop with a string

• What does this for loop do?

```python
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

• What is first value of `ch`?
  
  ‘c’

• What is final value of `ch`?
  
  ‘t’

9/20/22 Compsci 101, Fall 2022 82

Trace through for loop – 1

```python
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Trace through for loop – 2

```python
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Frames

Global frame

```
word "cat"
```
Trace through for loop – 3

1. `word = 'cat'`
2. `for ch in word:`
3. `word = word + ch`
4. `print(word)`

Iterate over copy of word: ‘c’ ‘a’ ‘t’

Frames

- ch gets first character in word

```
Global frame
<table>
<thead>
<tr>
<th>word</th>
<th>&quot;cat&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;c&quot;</td>
</tr>
</tbody>
</table>
```

Trace through for loop – 4

1. `word = 'cat'`
2. `for ch in word:`
3. `word = word + ch`
4. `print(word)`

Add ch to end of word

Frames

```
Global frame
<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catc&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;c&quot;</td>
</tr>
</tbody>
</table>
```

Trace through for loop – 5

1. `word = 'cat'`
2. `for ch in word:`
3. `word = word + ch`
4. `print(word)`

Iterate over what is left in copy of word: ‘a’ ‘t’

Frames

- ch gets second character in word

```
Global frame
<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catc&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;a&quot;</td>
</tr>
</tbody>
</table>
```

Trace through for loop – 6

1. `word = 'cat'`
2. `for ch in word:`
3. `word = word + ch`
4. `print(word)`

Add ch to end of word

Frames

```
Global frame
<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catca&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;a&quot;</td>
</tr>
</tbody>
</table>
```
Trace through for loop – 7

```
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Iterate over what is left in copy of word: ‘t’

Frames:
- Global frame:
  - word: "catca"
  - ch: "t"

ch gets third character in word

Trace through for loop – 8

```
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Frames:
- Global frame:
  - word: "catcat"
  - ch: "t"

Add ch to end of word

Trace through for loop – 9

```
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Iterate over what is left in copy of word:

Frames:
- Global frame:
  - word: "catcat"
  - ch: "t"

No more characters in word to process

The for loop is done!

Trace through for loop – 10

```
1 word = 'cat'
2 for ch in word:
3     word = word + ch
4 print(word)
```

Frames:
- Global frame:
  - word: "catcat"
  - ch: "t"

Print output (drag lower right corner to resize)
catcat
String’s split(...)  
- Strings have functions too!  
  - TYPE_STRING.FUNCTION(PARAMETERS)  
    - “.” means apply function to what is on the left  
      'one fish two fish'.split() returns a list  
    - What did it divide the string by?  
      - When no parameter, default whitespace  
      'one fish, two fish'.split(',')  
      ['one fish', ' two fish']  
  
String’s join(...)  
- TYPE_STRING.join(SEQ_OF_STRINGS)  
  - Opposite of .split()  
  - Creates string from sequence’s items separated by the string to the left of join  
    ' '.join(['one','fish','two','fish'])  
    '+'.join(['one','fish','two','fish'])  
    'ish'.join(['f','w','d','end'])  
    'one fish two fish'  
    'one+fish+two+fish'  
    'fishwishdishend'
### More Methods

<table>
<thead>
<tr>
<th>String</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.find(s)</code></td>
<td>index of first occurrence of s</td>
</tr>
<tr>
<td><code>.rfind(s)</code></td>
<td>index of last occurrence of s (from Right)</td>
</tr>
<tr>
<td><code>.upper()</code>/<code>.lower()</code></td>
<td>uppercase/lowercase version of string</td>
</tr>
<tr>
<td><code>.strip()</code></td>
<td>remove leading/trailing whitespace</td>
</tr>
<tr>
<td><code>.count(s)</code></td>
<td>number of times see s in string</td>
</tr>
<tr>
<td><code>.startswith(s)</code></td>
<td>bool of whether the string begins with s</td>
</tr>
<tr>
<td><code>.endswith(s)</code></td>
<td>bool of whether the string ends with s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sum(lst)</code></td>
<td>sum of the elements in lst</td>
</tr>
<tr>
<td><code>max(lst)</code></td>
<td>maximum value of lst</td>
</tr>
<tr>
<td><code>min(lst)</code></td>
<td>minimum value of lst</td>
</tr>
<tr>
<td><code>.append(elm)</code></td>
<td>Mutates the list by adding elm to the end of the list</td>
</tr>
<tr>
<td><code>.count(elm)</code></td>
<td>Number of times see elm in the list</td>
</tr>
</tbody>
</table>