Compsci 101
List and String Operations, For loop

Susan Rodger
January 27, 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?

“The 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 due today 11:30pm
  • Also REFLECT Form due same time
  • Get one grace day, but no consulting hours on Friday

• Exam on Tuesday!, Feb 1

PFTD

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

Exam 1

Read all rules posted in Announcement in Sakai

• This is your own work, no collaboration
• No book, No notes, only Exam 1 Python Ref Sheet

• Do not search for answers on the internet
• Do not type in code where it can be compiled and run
  • Do not use Pycharm, textbook code boxes, Python tutor or any other place the code can be run
• Do not talk to anyone about the exam during the exam, and until it is handed back!

Exam 1 Logistics

• Take on Tues. Feb 1 between 8am and 11pm
• You pick the start time
  • Must start by 9:30pm
• You get 1 hour 30 min
  • Longer if you have accommodations
• Once you start, your timer starts and you must finish in 1 hour, 30 minutes
• You cannot pause the timer
Exam 1 Logistics (2)

- Go to Gradescope to start
- Click on Exam 1 to start
- Gradescope saves answers as you type them in
  - Type 4 spaces to indent code
- Disconnected? Just log back in to Gradescope
- Question? Post a private post on Ed Discussion

- We do not have lecture on Feb. 1, Just take exam

Don’t go to Gradescope site until you are ready to start!

You click it, you have started!

We do not restart it!

Compare assign with integers, strings and lists – 1

```
>>> x = 6
>>> y = x
>>> x = 3
>>> m = "pink"
>>> n = m
>>> m = "red"
>>> a = ["pig", "cow", "dog"]
>>> b = a
>>> a[-1] = "ant"
```

Compare assign with integers, strings and lists – 2

```
>>> x = 6
>>> y = x
>>> x = 3
>>> m = "pink"
>>> n = m
>>> m = "red"
>>> a = ["pig", "cow", "dog"]
>>> b = a
>>> a[-1] = "ant"
```
Compare assign with integers, strings and lists – 3

```
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
Global frame
x 6
y 6
```

y gets a copy of the value of x

Compare assign with integers, strings and lists – 4

```
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
Global frame
x 3
y 6
```

x gets a new value

Compare assign with integers, strings and lists – 5

```
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
Global frame
x 3
y 6
```

n gets a copy of the value of m

Compare assign with integers, strings and lists – 6

```
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
Global frame
x 3
y 6
```

n gets a copy of the value of m
Compare assign with integers, strings and lists – 7

- **Python 3.6 (known limitations)**
- Frames
  - Global frame
    - \( x \): 6
    - \( y \): 3
    - \( m \): "pink"
    - \( n \): "red"
    - \( a \): ["pig", "cow", "dog"]
  - \( b \): \( a \)
- Objects
  - \( m \): "red"
  - \( a[0] \): "pig"
  - \( a[1] \): "cow"
  - \( a[2] \): "dog"

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

**m gets a new value**

1/27/22 Compsci 101, Spring 2022 17

Compare assign with integers, strings and lists – 8

- **Python 3.6 (known limitations)**
- Frames
  - Global frame
    - \( x \): 6
    - \( y \): 3
    - \( m \): "pink"
    - \( n \): "red"
    - \( a \): ["pig", "cow", "dog"]
  - \( b \): \( a \)
- Objects
  - \( m \): "red"
  - \( a[0] \): "pig"
  - \( a[1] \): "cow"
  - \( a[2] \): "dog"

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

**b gets a copy of the value of a**

1/27/22 Compsci 101, Spring 2022 18

Compare assign with integers, strings and lists – 9

- **Python 3.6 (known limitations)**
- Frames
  - Global frame
    - \( x \): 6
    - \( y \): 3
    - \( m \): "pink"
    - \( n \): "red"
    - \( a \): ["pig", "cow", "dog"]
  - \( b \): \( a \)
- Objects
  - \( m \): "red"
  - \( a[0] \): "pig"
  - \( a[1] \): "cow"
  - \( a[2] \): "dog"

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

**'dog' changed to 'ant'**

1/27/22 Compsci 101, Spring 2022 19

Compare assign with integers, strings and lists - 10

- **Python 3.6 (known limitations)**
- Frames
  - Global frame
    - \( x \): 6
    - \( y \): 3
    - \( m \): "pink"
    - \( n \): "red"
    - \( a \): ["pig", "cow", "dog"]
  - \( b \): \( a \)
- Objects
  - \( m \): "red"
  - \( a[0] \): "pig"
  - \( a[1] \): "cow"
  - \( a[2] \): "dog"

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

**Changing list a also changes list b**

As they are the same list!

1/27/22 Compsci 101, Spring 2022 20
List Cloning (or copying)

```python
lst1 = ['a', 'b', 1, 2]
lst2 = lst1
lst3 = lst1[:]
```
List Cloning (or copying)

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

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

Concatenation:

```
length, create, copy, copy, assign
```

Brand new list!
1. Calculate length
2. Create new list
3. Copy left list
4. Copy right list
5. Assign lst2

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

1. Calculate length
2. Create new list
3. Copy left list
4. Copy right list
5. Assign lst0

Concatenation: Makes new List

length, create, copy, copy, assign

- How is the inner list copied?

What will Python Tutor Display? How many copies of ['b', 3.0] will be present?
Concatenation:
length, create, copy, copy

• How is the inner list copied?

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

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

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])
```

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[0].append([5, 2])
x.append([5, 2])
```

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])
x[0].append([5, 2])
x.append([5, 2])
```

What will Python Tutor Display? One or two lists?
List Mutation: \texttt{.append(...)}

1. \texttt{lst0 = [1, 2, 3]}
2. \texttt{tmp = lst0}
3. \texttt{lst0.append(4)}

\begin{itemize}
\item \texttt{Same list! No new list}
\end{itemize}
List Mutation: `.append(...)`

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

Same list! No new list

**Anatomy of a `for` loop**

```python
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?

**WOTO-2 – Mutable and Append**

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`?

- What is final value of `num`?

Trace through for loop – 1

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

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 – 2

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

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

Trace through for loop – 4

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

Trace through for loop – 5

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

Trace through for loop – 6

```python
1 lst = [5, 3, 2]
2 sum = 0
3 for num in lst:
4     sum = sum + num
5 print(sum)
```
Trace through for loop – 7

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

Frames

- Global frame:
  - \texttt{lst}
  - \texttt{sum}
  - \texttt{num}

Objects

- \texttt{lst}: [5, 3, 2]
- \texttt{sum}: 0
- \texttt{num}: 3

Add num to sum

Trace through for loop – 8

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

Frames

- Global frame:
  - \texttt{lst}
  - \texttt{sum}
  - \texttt{num}

Objects

- \texttt{lst}: [5, 3, 2]
- \texttt{sum}: 8
- \texttt{num}: 2

num gets third value in list

Trace through for loop – 9

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

Frames

- Global frame:
  - \texttt{lst}
  - \texttt{sum}
  - \texttt{num}

Objects

- \texttt{lst}: [5, 3, 2]
- \texttt{sum}: 10
- \texttt{num}: 2

Add num to sum

Trace through for loop – 10

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

Frames

- Global frame:
  - \texttt{lst}
  - \texttt{sum}
  - \texttt{num}

Objects

- \texttt{lst}: [5, 3, 2]
- \texttt{sum}: 10
- \texttt{num}: 2

No more values in lst

The for loop is done!
Trace through for loop – 11

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

Example for loop with a string

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

- What does this for loop do?
- What is first value of `ch`?
- What is final value of `ch`?

Example for loop with a string

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

- What does this for loop do?
- What is first value of `ch`?
  - 'c'
- What is final value of `ch`?
  - 't'
Trace through for loop – 2

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

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’

Trace through for loop – 4

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’

Trace through for loop – 5

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

ch gets first character in word

Global frame
word | "cat"
ch | "c"

Add ch to end of word

Global frame
word | "catc"
ch | "c"

ch gets second character in word

Global frame
word | "catc"
ch | "a"
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

<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catca&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;t&quot;</td>
</tr>
</tbody>
</table>

Trace through for loop – 8

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;t&quot;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catca&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;t&quot;</td>
</tr>
</tbody>
</table>

No more characters in word to process

The for loop is done!
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(',')`

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'])`
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'])`
    
    `'one fish two fish'`

    `' + .join(['one','fish','two','fish'])`
    
    `'one+fish+two+fish'`

    `'ish .join(['f','w','d','end'])`
    
    `'fishwishdishend'`