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!

1/27/22 Compsci 101, Spring 2022
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

#### Python 3.6

*(known limitations)*

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x = 6</td>
</tr>
<tr>
<td>2</td>
<td>y = x</td>
</tr>
<tr>
<td>3</td>
<td>x = 3</td>
</tr>
<tr>
<td>4</td>
<td>m = &quot;pink&quot;</td>
</tr>
<tr>
<td>5</td>
<td>n = m</td>
</tr>
<tr>
<td>6</td>
<td>m = &quot;red&quot;</td>
</tr>
<tr>
<td>7</td>
<td>a = [&quot;pig&quot;, &quot;cow&quot;, &quot;dog&quot;]</td>
</tr>
<tr>
<td>8</td>
<td>b = a</td>
</tr>
<tr>
<td>9</td>
<td>a[-1] = &quot;ant&quot;</td>
</tr>
</tbody>
</table>

*Edit this code*

- ➔ line that just executed
- ➔ next line to execute
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"

Frames

Objects

Global frame

x 6
```

Edit this code

- line that just executed
- next line to execute
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"

Frames
Objects

Global frame

x  6
y  6

y gets a copy of the value of x
Compare assign with integers, strings and lists – 4

```python
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

Global frame
- x 3
- y 6

Objects

x gets a new value

Edit this code

- line that just executed
- next line to execute
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"
```

Edit this code

- line that just executed
- next line to execute
Compare assign with integers, strings and lists – 6

Python 3.6 (known limitations)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>x = 6</code></td>
</tr>
<tr>
<td>2</td>
<td><code>y = x</code></td>
</tr>
<tr>
<td>3</td>
<td><code>x = 3</code></td>
</tr>
<tr>
<td>4</td>
<td><code>m = &quot;pink&quot;</code></td>
</tr>
<tr>
<td>5</td>
<td><code>n = m</code></td>
</tr>
<tr>
<td>6</td>
<td><code>m = &quot;red&quot;</code></td>
</tr>
<tr>
<td>7</td>
<td><code>a = [&quot;pig&quot;, &quot;cow&quot;, &quot;dog&quot;]</code></td>
</tr>
<tr>
<td>8</td>
<td><code>b = a</code></td>
</tr>
<tr>
<td>9</td>
<td><code>a[-1] = &quot;ant&quot;</code></td>
</tr>
</tbody>
</table>

Frames

<table>
<thead>
<tr>
<th>Variable</th>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>6</td>
</tr>
<tr>
<td>m</td>
<td>&quot;pink&quot;</td>
</tr>
<tr>
<td>n</td>
<td>&quot;pink&quot;</td>
</tr>
</tbody>
</table>

Objects

n gets a copy of the value of m

Edit this code

- line that just executed
- next line to execute
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

<table>
<thead>
<tr>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>y</td>
</tr>
<tr>
<td>m</td>
</tr>
<tr>
<td>n</td>
</tr>
</tbody>
</table>

Objects

m gets a new value

Edit this code

- line that just executed
- next line to execute
Compare assign with integers, strings and lists – 8

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

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

Objects

list
0 "pig" 1 "cow" 2 "dog"

Edit this code

- line that just executed
- next line to execute
Compare assign with integers, strings and lists – 9

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

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

Objects

list
0   "pig"
1   "cow"
2   "dog"

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

b gets a copy of the value of a

a and b refer to the same list!
Compare assign with integers, strings and lists - 10

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"
```

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[:]
```
List Cloning (or copying)

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

Frames

Objects

Global frame
lst1

list

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;a&quot;</td>
<td>&quot;b&quot;</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
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[:]
```
List Cloning (or copying)

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

Global frame

<table>
<thead>
<tr>
<th>lst1</th>
<th>lst2</th>
<th>lst3</th>
</tr>
</thead>
</table>

List

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;a&quot;</td>
<td>&quot;b&quot;</td>
<td>1</td>
<td>&quot;SUN&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;a&quot;</td>
<td>&quot;b&quot;</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
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

1  \texttt{lst0} = [1, 2]
2  \texttt{lst1} = [3, 4, 5]
3  \texttt{lst2} = \texttt{lst0} + \texttt{lst1}
Concatenation:
length, create, copy, copy, assign

1. Calculate length
2. Create new list
3. Copy left list
Concatenation:
length, create, copy, copy, assign

1. \( \text{lst0} = [1,2] \)
2. \( \text{lst1} = [3,4,5] \)
3. \( \text{lst2} = \text{lst0} + \text{lst1} \)
Concatenation: Makes new List

```python
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

```python
1  lst0 = [1,2]
2  tmp = lst0
3  lst0 = lst0 + [4]
```
Concatenation: Makes new List

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

```
lst0 = [1, 2]
tmp = lst0
lst0 = lst0 + [4]
```
Concatenation: Makes new List

1. \texttt{lst0} = [1, 2]
2. \texttt{tmp} = \texttt{lst0}
3. \texttt{lst0} = \texttt{lst0} + [4]

4. Copy right list

5. Assign \texttt{lst0}

```
Global frame
\begin{array}{c}
\text{lst0} \\
\text{tmp}
\end{array}
```

```
Frames
\begin{array}{c}
\text{lst0} \\
\text{tmp}
\end{array}
```

```
Objects
\begin{array}{c}
\text{list} \\
0 & 1 & 2 \\
0 & 1 & 2 & 4
\end{array}
```

```
list
\begin{array}{c}
0 & 1 & 2 \\
0 & 1 & 2 & 4
\end{array}
```
Concatenation:
length, create, copy, copy, assign

• How is the inner list copied?

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

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

• How is the inner list copied?

1. Calculate length
2. Create new list
3. Copy left list
Concatenation:
length, create, copy, copy

• How is the inner list copied?

1. \( \text{lst0} = [1, ['b', 3.0]] \)
2. \( \text{lst1} = [4] \)
3. \( \text{lst2} = \text{lst0} + \text{lst1} \)

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: \texttt{.append(...)}

- \texttt{.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 (\texttt{LEFT.RIGHT})

\begin{align*}
x & = [6, 2, 4] \\
x & \text{.append}(3) \\
x & \text{.append( [5,2] )}
\end{align*}

\begin{align*}
x & \text{ is } [6, 2, 4] \\
x & \text{ is } [6, 2, 4, 3] \\
x & \text{ is } [6, 2, 4, 3, [5, 2] ]
\end{align*}
List Mutation: .append(…)

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

What will Python Tutor Display? One or two lists?
List Mutation: `.append(…)`

```
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)`
List Mutation: `.append(…)`

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

Same list! No new list
List Mutation: .append(…)

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

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

![Diagram of list mutation with frames and objects showing that the list is modified in place and no new list is created.

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?
Anatomy of a for loop

```python
for v in seq:
    CODE_BLOCK
    if not More elements in seq:
        v = next element in seq
        CODE_BLOCK
    Finished
```

Start

More elements in seq? Yes

No

Finished
Example for loop with a list

• What does this for loop do?

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

• 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

Adds the numbers in the list
Trace through for loop – 1

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

Global frame:
- `lst`:
  - 0: 5
  - 1: 3
  - 2: 2
- `sum`: 0

List:
- 0: 5
- 1: 3
- 2: 2
Trace through for loop – 4

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

- `num` gets the first value in `lst`.
Trace through for loop – 5

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

Add num to sum
Trace through for loop – 6

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

num gets second value in list
Trace through for loop – 7

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

Add num to sum
Trace through for loop – 8

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

num gets third value in list

- Global frame:
  - lst: 3
  - sum: 8
  - num: 2

- list:
  - 0: 5
  - 1: 3
  - 2: 2
Trace through for loop – 9

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

Add num to sum
Trace through for loop – 10

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

No more values in lst

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

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

Print output (drag lower right corner to resize)

```
10
```

Print result

<table>
<thead>
<tr>
<th>Global frame</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>lst</td>
<td>list</td>
</tr>
<tr>
<td>sum</td>
<td>5</td>
</tr>
<tr>
<td>num</td>
<td>0</td>
</tr>
</tbody>
</table>
Example for loop with a string

• What does this for loop do?

```python
word = 'cat'
for ch in word:
    word = word + ch
print(word)
```

• What is first value of `ch`?

• What is final value of `ch`?
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`
Trace through for loop – 1

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

```python
word = 'cat'
for ch in word:
    word = word + ch
print(word)
```
Trace through for loop – 3

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

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

Frames

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>

ch gets first character in word
Trace through for loop – 4

```python
word = 'cat'
for ch in word:
    word = word + ch
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)
```

ch gets second character in word

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

Frames

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

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

```
word = 'cat'
for ch in word:
    word = word + ch
print(word)
```

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

Frames

<table>
<thead>
<tr>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
</tr>
<tr>
<td>ch</td>
</tr>
</tbody>
</table>
Trace through for loop – 8

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

Frames

<table>
<thead>
<tr>
<th>Global frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
</tr>
<tr>
<td>ch</td>
</tr>
<tr>
<td>&quot;catcat&quot;</td>
</tr>
<tr>
<td>&quot;t&quot;</td>
</tr>
</tbody>
</table>

Add ch to end of word
Trace through for loop – 9

No more characters in word to process

The for loop is done!

Iterate over what is left in copy of word:

```python
word = 'cat'
for ch in word:
    word = word + ch
print(word)
```

Frames

Global frame

<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catcat&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;t&quot;</td>
</tr>
</tbody>
</table>
Trace through for loop – 10

```python
word = 'cat'
for ch in word:
    word = word + ch
print(word)
```

Print output (drag lower right corner to resize)
catcat

Frames

Objects

Global frame

<table>
<thead>
<tr>
<th>word</th>
<th>&quot;catcat&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>&quot;t&quot;</td>
</tr>
</tbody>
</table>
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 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  
  [ 'one', 'fish', 'two', 'fish' ]  
- 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’])
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`

  ```python
  ' '.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'
  ```
# More Methods

## String

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.find(s)</code></td>
<td>Index of first occurrence of <code>s</code></td>
</tr>
<tr>
<td><code>.rfind(s)</code></td>
<td>Index of last occurrence of <code>s</code> (from Right)</td>
</tr>
<tr>
<td><code>.upper()</code>/</td>
<td>Uppercase/lowercase version of string</td>
</tr>
<tr>
<td><code>.lower()</code></td>
<td></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 <code>s</code> in string</td>
</tr>
<tr>
<td><code>.startswith(s)</code></td>
<td>Bool of whether the string begins with <code>s</code></td>
</tr>
<tr>
<td><code>.endswith(s)</code></td>
<td>Bool of whether the string ends with <code>s</code></td>
</tr>
</tbody>
</table>

## List

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sum(lst)</code></td>
<td>Sum of the elements in <code>lst</code></td>
</tr>
<tr>
<td><code>max(lst)</code></td>
<td>Maximum value of <code>lst</code></td>
</tr>
<tr>
<td><code>min(lst)</code></td>
<td>Minimum value of <code>lst</code></td>
</tr>
<tr>
<td><code>.append(elm)</code></td>
<td>Mutates the list by adding <code>elm</code> to the end of the list</td>
</tr>
<tr>
<td><code>.count(elm)</code></td>
<td>Number of times see <code>elm</code> in the list</td>
</tr>
</tbody>
</table>
WOTO-3 – Split and Join