On your way in...

Nothing to pick-up!

(Although, Lab 2 grades are now available:
  cd ~/cs134/lab2
  git pull
  more GradeSheet.txt
 )
Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Tuples-

Spring 2019
Computer Science Colloquium Today!
(and pretty much every Friday)

• 2:35pm (today only: TCL 202)

• Hal Daume III
(UMD, MSR)

ARTIFICIAL INTELLIGENCE

NATURAL LANGUAGE PROCESSING
Is anyone taking BIOL 102 or MATH 150?

Raise your hand.
Spaceless Strings

- How do we go through a string, letter by letter, and return a new string without any spaces?

  - i.e. “Hello! My name is Iris (I like dogs and cats)”
  - → “Hello!MynameisIris(Ilikedogsandcats)”
Spaceless Strings

- def spacelessStr(s):
  
  """ Removes spaces from Strings"""
  
  result = ""
  
  for letter in s:
    
    if letter != ' ':
      result += letter
  
  return result

What if we want to return a String with everything but the letters? What do we need to change?
TODAY’S LESSON

Tuples

(like lists, but immutable)
A Tale of Two Mutabilities...

- \( l = ['\text{dog}', '\text{cat}', '\text{mouse}', '\text{cheese}'] \)
- \( l[1] = \text{dizzy} \)
- print(\( l \))
- ['\text{dog}', '\text{dizzy}', '\text{mouse}', '\text{cheese}']

- \( t = (\text{dog}, \text{cat}, \text{mouse}, \text{cheese}) \)
- \( t[1] = \text{dizzy} \)
- TypeError: 'tuple' object does not support item assignment

Lists are mutable (i.e., changeable)
Tuples are immutable.
A Tale of Two Mutabilities...

- \( l = [ 'd', 'c', 'm'] \)
- \( l.append('cheese') \)
- \( print(l) \)
  
  \[ [ 'd', 'c', 'm', 'cheese'] \]

- \( t = ( 'd', 'c', 'm') \)
- \( t = t + ( 'cheese', ) \)
- \( print(t) \)
  
  \( ('d', 'c', 'm', 'cheese') \)

You can’t modify tuples, but you can replace them!
A One Element Tuple

• \( s = ( 'cheese' ) \)
  • type(s)
  • <class 'str'>

• \( t = ( 'cheese', ) \)
  • type(t)
  • <class 'tuple'>

• \( u = 'cheese', \)
  • type(u)
  • <class 'tuple'>

What happened here?

What happened here?
Zipping

• \( p = 'dcm' \)
• \( n = [5, 1, 0] \)
• for pair in zip(p, n):
  ▪ print(pair)

• Will print:
  ▪ ('d', 5)
  ▪ ('c', 1)
  ▪ ('m', 0)

What happened here?
Zipping

• `zip('abcdefg','123')`

• Will generate the following tuples:
  - `(a, 1)`
  - `(b, 2)`
  - `(c, 3)`

What happened here?
Zipping

• Turning a zip object into a list:

  ```python
  list(zip('abcdefg','123'))
  # [(‘a’,’1’),(‘b’,’2’),(‘c’,’3’)]
  ```

• For loops and tuples:

  ```python
  t = list(zip('abcdefg','123'))
  for letter, num in t:
      print(num, letter)
  # 1 a
  # 2 b
  # 3 c
  ```
TODAY’S LESSON

More Lists

(lists of lists)
Appending/Extending

- \( l = [5, 16, 18] \)
- \( m = ['\text{pixel}', '\text{pup}'] \)
- \( l.append(m) \)

\[
\begin{align*}
l & = \[5, 16, 18, ['\text{pixel}', '\text{pup}']\] \\
\end{align*}
\]

What happened there?

- \( l = [5, 16, 18] \)
- \( m = ['\text{pixel}', '\text{pup}'] \)
- \( l.extend(m) \)

\[
\begin{align*}
l & = [5, 16, 18, '\text{pixel}', '\text{pup}'] \\
\end{align*}
\]

What happened here?
Other List Functions

• $x = ['new']*3 + ['old']*8$
• $x$
  ▪ ['new', 'new', 'new', 'old', 'old', 'old', 'old', 'old', 'old', 'old', 'old', 'old']

What happened here?

• $x.count('new')$
  ▪ 3
• $x.index('old')$
  ▪ 3
List Comprehensions

```python
words = [line.strip() for line in open('/usr/share/dict/words')]
```

- Is the same as:

```python
words = []
for line in open('/usr/share/dict/words'):
    words.append(line.strip())
```
List Comprehensions

\[ w_4 = \left[ \text{word for word in words if } \text{len(word)} == 4 \right] \]

• Is the same as:

\[
\cdot w_4 = [ ]
\cdot \text{for word in words:}
  \cdot \quad \text{if len(word) == 4:}
    \quad \cdot w_4.\text{append(word)}
\]
List Comprehensions

• Can use nested for loops, too

• Find compound words made up of 2 four-letter words

```python
ww8 = [w1+w2 for w1 in w4 for w2 in w4 if w1+w2 in words]
```

5000 X 5000 combinations of words
See if it’s in a list of several 100,000
...this will take awhile...
Algorithms
How do we rotate a character by 1?

- $a \rightarrow b$, $b \rightarrow c$, ..., $y \rightarrow z$, $z \rightarrow a$, etc

Any ideas?
ord(c) and chr(n)

- ord('a')
  - 97
- ord('z')
  - 122
- ord('A')
  - 65
- ord(';')
  - 59

- chr(97)
  - 'a'
- chr(122)
  - 'z'
- chr(65)
  - 'A'
- chr(59)
  - ';'
# ASCII Values

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Oct</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>40</td>
<td>100</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>140</td>
<td>`</td>
</tr>
<tr>
<td>65</td>
<td>41</td>
<td>101</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>141</td>
<td>a</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>102</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>142</td>
<td>b</td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>103</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>143</td>
<td>c</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>104</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>144</td>
<td>d</td>
</tr>
<tr>
<td>69</td>
<td>45</td>
<td>105</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>145</td>
<td>e</td>
</tr>
<tr>
<td>70</td>
<td>46</td>
<td>106</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>146</td>
<td>f</td>
</tr>
<tr>
<td>71</td>
<td>47</td>
<td>107</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>147</td>
<td>g</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>110</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>150</td>
<td>h</td>
</tr>
<tr>
<td>73</td>
<td>49</td>
<td>111</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>151</td>
<td>i</td>
</tr>
<tr>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>152</td>
<td>j</td>
</tr>
<tr>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>153</td>
<td>k</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>154</td>
<td>l</td>
</tr>
<tr>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>155</td>
<td>m</td>
</tr>
<tr>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>156</td>
<td>n</td>
</tr>
<tr>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>157</td>
<td>o</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>120</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>160</td>
<td>p</td>
</tr>
<tr>
<td>81</td>
<td>51</td>
<td>121</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>161</td>
<td>q</td>
</tr>
<tr>
<td>82</td>
<td>52</td>
<td>122</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>162</td>
<td>r</td>
</tr>
<tr>
<td>83</td>
<td>53</td>
<td>123</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>163</td>
<td>s</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>124</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>164</td>
<td>t</td>
</tr>
<tr>
<td>85</td>
<td>55</td>
<td>125</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>165</td>
<td>u</td>
</tr>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Oct</td>
<td>Char</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 1</td>
<td>1</td>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2 2</td>
<td>2</td>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3 3</td>
<td>3</td>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4 4</td>
<td>4</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5 5</td>
<td>5</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6 6</td>
<td>6</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7 7</td>
<td>7</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8 8</td>
<td>8</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9 9</td>
<td>9</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A 12</td>
<td>11</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B 13</td>
<td>14</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C 14</td>
<td>17</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D 15</td>
<td>20</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E 16</td>
<td>23</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F 17</td>
<td>26</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10 20</td>
<td>32</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>11 21</td>
<td>37</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12 22</td>
<td>40</td>
<td>z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>13 23</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>14 24</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15 25</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>16 26</td>
<td>54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>17 27</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18 30</td>
<td>61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>19 31</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1A 32</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1B 33</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1C 34</td>
<td>76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D 35</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1E 36</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1F 37</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rotating Letters

• `chr(ord(‘a’) + 1)`
  ▪ ‘b’
• `chr(ord(‘z’) + 1)`
  ▪ ‘{’

What happened here?

How do we wrap around back to ‘a’ after ‘z’?

\[
\begin{array}{cccccc}
\text{v} & \text{w} & \text{x} & \text{y} & \text{z} & \text{118} & \text{119} & \text{120} & \text{121} & \text{122} & \text{123}
\end{array}
\]
Rotating Letters

\[
\begin{array}{ccccccccc}
a & \ldots & v & w & x & y & z & \{ \\
97 & \ldots & 118 & 119 & 120 & 121 & 122 & 123 \\
\end{array}
\]

• What we want is for ASCII 123 to wrap around back to 97...

• Let’s solve a simpler problem:

\[
\begin{array}{ccccccccc}
a & \ldots & v & w & x & y & z & \{ \\
0 & \ldots & 21 & 22 & 23 & 24 & 25 & 26 \\
\end{array}
\]

• Any number over 25 should wrap back around to 0, 1, 2, etc.

• What might we use to do that?
Rotating Numbers

a  b  ...  x  y  z  {  |  }

0  1  ...  23  24  25  26  27  28

• Fill in the blank:
  - For ‘{’ → ‘a’:  \(<\text{character-ascii}>\) <operator> <number> = <rotated-ascii>
    o 26 <operator> <number> = 0
    o 26 <operator> <num-letters-in-alphabet> = 0
    o 26 ___ 26 = 0
    o 26 % 26 = 0
  - For ‘|’ → ‘b’:  27 <operator> <num-letters-in-alphabet> = 1
    o 27 % 26 = 1
  - For ‘}’ → ‘c’:  28 % 26 = 2
Rotating Letters

\[
\begin{array}{ccccccc}
\text{a} & \ldots & \text{v} & \text{w} & \text{x} & \text{y} & \text{z} \\
0 & \ldots & 21 & 22 & 23 & 24 & 25 & 26
\end{array}
\]

• Our formula is: \(<\text{rotated-ascii}> = \text{<character-ascii>} \mod 26\)

• But that’s for 0+, how do we convert to 97+?

\[
\begin{array}{ccccccc}
\text{a} & \ldots & \text{v} & \text{w} & \text{x} & \text{y} & \text{z} \\
97 & \ldots & 118 & 119 & 120 & 121 & 122 & 123
\end{array}
\]

• Subtract 97 to adjust to ‘a’ starting at 0:
  \[<\text{rotated-ascii}> = (\text{<character-ascii>} - 97) \mod 26\]

• And then add 97 back so we can convert to real ASCII
  \[<\text{rotated-ascii}> = 97 + (\text{<character-ascii>} - 97) \mod 26\]
Rotating Numbers

- \(<\text{rotated-ascii}> = 97 + (<\text{character-ascii}>-97) \% 26\)
- Generalizes to:
  - \(\text{rotAscii} = \text{ord}(\text{‘a’}) + (\text{ord}(\text{givChar})-\text{ord}(\text{‘a’})) \% 26\)
- Convert from ASCII to character:
  - \(\text{rotChar} = \text{chr}(\text{ord}(\text{‘a’}) + (\text{ord}(\text{givChar})-\text{ord}(\text{‘a’})) \% 26\)
- ...Are we missing anything?
Rotating Numbers

• We need to rotate by ‘n’:
  ▪ rotChar = chr(ord(‘a’) + (((ord(givChar)-ord(‘a’)) + n) % 26))

• ...Are we missing anything?

• What about...

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>...</td>
<td>y</td>
<td>z</td>
<td>[</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>66</td>
<td>...</td>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td></td>
</tr>
</tbody>
</table>

• rotChar = chr(ord(‘A’) + (((ord(givChar)-ord(‘A’)) + n) % 26))
Rotating Letters

• result = ‘’
• for c in s:
  ▪ if c.islower():
    ○ c = chr(ord('a')+((ord(c)-ord('a'))+n)%26))
  ▪ elif c.isupper():
    ○ c = chr(ord('A')+((ord(c)-ord('A'))+n)%26))
  ▪ result += c
• return result

See crypt.py in examples: rot('xyz',1)
Stdin

< for stdin

• `python3 crypt.py < crypt.py > whatever`
  ▪ Encrypts stdin into whatever file
  ▪ Rotates by 13 (by default, see program)

• `python3 crypt.py < whatever > second.py`
  ▪ Rotates ‘whatever’ by 13, stores in second.py
  ▪ What happens when you rotate by 13 twice?

| to run a second command

• `python3 crypt.py < crypt.py | python3 crypt.py`
  ▪ Rotates by 13, then rotates by 13 again

> for pushing output to a file
QUESTIONS?
Leftover Slides